การพัฒนาโมดูลการตัดยอดสินค้าสำหรับกรอบงานออฟบิส

นายธิรวุฒิ ไชยชะอุ่ม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2559

DEVELOPMENT OF STOCK CUT-OFF MODULE FOR OFBIZ FRAMEWORK

Thirawuth Chaicha_um

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Degree of Master of Engineering in Computer Engineering

Suranaree University of Technology

Academic Year 2016

การพัฒนาโมดูลการตัดยอดสินค้าสำหรับกรอบงานออฟบิส

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

(รศ. ดร.กิตติศักดิ์ เกิดประสพ)
ประธานกรรมการ

(ผศ. ดร.พิชโยทัย มหัทธนาภิวัฒน์)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

(ผศ. ดร.ชาญวิทย์ แก้วกสิ)
กรรมการ

(ศ. ดร.สันติ แม้นศิริ)
รักษาการแทนรองอธิการบดีฝ่ายวิชาการ
และพัฒนาความเป็นสากล

(รศ. ร.อ. ดร.กนต์ธร ชำนิประศาสน์)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

ถิรวุฒิ ไชยชะอุ่ม : การพัฒนาโมดูลการตัดยอดสินค้าสำหรับกรอบงานออฟบิส
(DEVELOPMENT OF STOCK CUT-OFF MODULE FOR OFBIZ FRAMEWORK)
อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.พิชโยทัย มหัทธนาภิวัฒน์, 132 หน้า

งานวิจัยนี้ได้ศึกษาเกี่ยวกับการพัฒนากรอบงานโดยการพัฒนาโมดูลการตัดยอดสินค้าในกรอบงานออฟบิส ซึ่งเป็นส่วนการทำงานหนึ่งของระบบคลังสินค้าที่ยังมีส่วนที่สามารถพัฒนาเพิ่มเติมได้ โดยการเพิ่มรูปแบบการตัดยอดสินค้าในแบบต่าง ๆ ให้ได้เลือกใช้ คือ แบบเข้าก่อนออกก่อน (First In First Out), แบบหมดอายุก่อนออกก่อน (First Expire date First Out) และแบบเข้าทีหลังออกก่อน (Last In First Out) ซึ่งส่วนของการตัดยอดสินค้านี้จะช่วยให้ระบบคลังสินค้ามีความสมบูรณ์แบบมากขึ้น สามารถจัดการกับสินค้าได้ง่ายยิ่งขึ้น โดยการพัฒนานี้จะใช้ภาษาจาวาเป็นหลัก และเมื่อพัฒนาเสร็จจะทำการทดสอบว่าใช้งานได้จริงและไม่ทำให้ส่วนอื่น ๆ ของกรอบงานทำงานผิดปกติ

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนักศึกษา _____

ปีการศึกษา 2559 ลายมือชื่ออาจารย์ที่ปรึกษา _____

THIRAWUTH CHAICHA_UM : DEVELOPMENT OF STOCK CUT-OFF MODULE FOR OFBIZ FRAMEWORK. THESIS ADVISOR : ASST. PROF. PICHAYOTAI MAHATHANAPIWAT, Ph.D., 132 PP.

OFBIZ FRAMEWORK/WAREHOUSE SYSTEM/JAVA LANGUAGE

This research studies about the development of stock cut-off module for Ofbiz framework. The cut-off module is a part of warehouse system that can be extended using various ways of cut-off forms, for example FIFO (First In First Out), FEFO (First Expire date First Out) and LIFO (Last In First Out). The cut-off module will make the warehouse system more complete and handle easily. The research uses Java language for development and the results show that the cut-off module performs well while the rest of the framework operate normally.

School of Computer Engineering

Acadamic Year 2016

Student's Signature

Advisor's Signature

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงด้วยดี เนื่องจากได้รับความช่วยเหลืออย่างดียิ่ง ทั้งด้านวิชาการและด้านการดำเนินงานวิจัย จากบุคคลและกลุ่มบุคคลต่าง ๆ ได้แก่

ผู้ช่วยศาสตราจารย์ ดร.พิชโยทัย มหัทธนาภิวัฒน์ อาจารย์ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้โอกาสทางการศึกษา ให้คำแนะนำ ช่วยแก้ไขปัญหา และให้กำลังใจแก่ผู้วิจัยมาโดยตลอด รวมทั้งช่วยตรวจทานและแก้ไขวิทยานิพนธ์เล่มนี้จนเสร็จสมบูรณ์

รองศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ หัวหน้าสาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ที่ให้โอกาสทางการศึกษา และให้คำแนะนำด้านการศึกษา การปรับตัวในการเรียนระดับมหาบัณฑิตศึกษา และทุนการศึกษาให้กับผู้วิจัย

ผู้ช่วยศาสตราจารย์ ดร.ชาญวิทย์ แก้วกสิ อาจารย์ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ที่ให้คำแนะนำด้านการวิจัย แนะนำแนวทางในการศึกษาค้นคว้าให้กับผู้วิจัย

ขอขอบคุณ อาจารย์ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ทุกท่าน ที่ได้ให้ความรู้และให้กำลังใจในการศึกษามาโดยตลอด

สุดท้ายนี้ ขอขอบคุณ บิดา มารดา ที่เคารพรักยิ่ง ที่คอยแนะนำ ให้กำลังใจ และสนับสนุนผู้วิจัยมาโดยตลอด จนทำให้ประสบความสำเร็จในชีวิต

ถิรวุฒิ ไชยชะอุ่ม

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของปัญหาการวิจัย

ในปัจจุบันเทคโนโลยีด้านซอฟต์แวร์ได้เข้ามามีบทบาทสำคัญทั้งในชีวิตประจำวันและด้านอื่น ๆ ของมนุษย์ จึงมีการผลักดันให้มีการพัฒนาซอฟต์แวร์ใหม่ ๆ ขึ้นมาจำนวนมากเพื่อตอบสนองความต้องการของมนุษย์ การพัฒนาซอฟต์แวร์ในแต่ละครั้งจะเริ่มจาก การเก็บความต้องการกับลูกค้า วิเคราะห์ความต้องการที่เก็บรวบรวมมา นำความต้องการที่วิเคราะห์เรียบร้อยแล้ว มาเขียนซอร์สโค้ดพัฒนาเป็นซอฟต์แวร์ให้ได้ตรงตามความต้องการที่เก็บมาได้

กรอบงาน (Framework) คือ โครงสร้างพื้นฐานของซอฟต์แวร์ที่ถูกพัฒนาขึ้นเพื่อใช้เป็นตัวช่วยในการพัฒนาซอฟต์แวร์ต่าง ๆ ซึ่งประกอบด้วยเทคนิคต่าง ๆ ที่จำเป็นสำหรับซอฟต์แวร์นั้น ๆ และเทคนิคใหม่ๆที่ใช้ในการแก้ไขปัญหาที่มักจะเกิดขึ้นในซอฟต์แวร์นั้น ๆ โดยการนำตัวกรอบงาน มาเพิ่มเติมชุดคำสั่งบางส่วนเพื่อให้ทำงานได้ตามที่ต้องการ ซึ่งจะช่วยให้ผู้พัฒนาซอฟต์แวร์สามารถพัฒนาซอฟต์แวร์ได้อย่างรวดเร็วและมีประสิทธิภาพมากขึ้น

ระบบการจัดการคลังสินค้าเป็นระบบที่ใช้ในการบริหารจัดการคลังสินค้า ซึ่งเป็นตัวแปรที่สำคัญในการทำธุรกิจ คือ ถ้าสินค้าที่ต้องใช้ในการผลิตขาด อาจทำให้การผลิตหยุดและผลิตสินค้าไม่ทัน ถ้ามีสินค้าค้างในคลังมาก ๆ นาน ๆ การหมุนเวียนของสินค้าน้อย การทำกำไรก็จะน้อยลงตามไปด้วย การนำระบบคอมพิวเตอร์มาจัดการคลังสินค้ายังใช้ได้น้อย ไม่เต็มประสิทธิภาพ เนื่องจากระบบคลังสินค้ามีความซับซ้อนและมีเงื่อนไขมากมายในการนำคอมพิวเตอร์มาจัดการคลังสินค้า

กรอบงานออฟบิส (Ofbiz Framework) เป็นกรอบงานแบบเปิดเผยต้นรหัส (Open Source) มีความยืดหยุ่นในการใช้งาน มีคลาสและเมธอดที่ใช้ในการพัฒนาซอฟต์แวร์ต่าง ๆ รวมไปถึงซอฟต์แวร์ของคลังสินค้า ทำให้สามารถพัฒนาระบบคลังสินค้าได้อย่างรวดเร็ว แต่ก็ยังมีส่วนที่ยังไม่สมบูรณ์ เช่น โมดูลของการตัดยอดสินค้า เป็นต้น ซึ่งเป็นส่วนที่มีความสำคัญของระบบ

ในงานวิจัยนี้ได้เล็งเห็นถึงปัญหาดังกล่าวจึงได้ทำการพัฒนาตัวกรอบงานออฟบิสให้มีโมดูลที่สามารถตัดยอดสินค้าในแบบต่าง ๆ ได้ ซึ่งในงานวิจัยนี้จะนำเสนอเทคนิคและวิธีต่าง ๆ เพื่อให้สามารถทำงานตามที่ต้องการได้อย่างมีประสิทธิภาพ

1.2 วัตถุประสงค์ของการวิจัย

1) เพื่อศึกษาเกี่ยวกับการพัฒนากรอบงาน
2) เพื่อศึกษาการทำงานของกรอบงานออฟบิส
3) เพื่อสร้างโมดูลของการตัดยอดสินค้าสำหรับกรอบงานออฟบิส

1.3 ข้อตกลงเบื้องต้น

1) ใช้ภาษาจาวาเป็นหลักในการพัฒนา
2) การพัฒนาทำบนระบบปฏิบัติการวินโดวส์
3) การพัฒนาจะพัฒนาเฉพาะส่วนของเมธอดที่สร้างขึ้นมาใหม่และเมธอดที่เกี่ยวข้องไม่รวมถึงส่วนของอินเตอร์เฟสที่ใช้ในการตอบสนองและเมธอดที่มีอยู่เดิมและเมธอดที่ไม่เกี่ยวข้อง

1.4 ขอบเขตของการวิจัย

1) ใช้ภาษาจาวาในการสร้างเมธอดและคลาสต่าง ๆ
2) การพัฒนาจะครอบคลุมในส่วนของการตัดยอดสินค้าในแบบต่าง ๆ คือ ตัดตามวันหมดอายุ, ตัดแบบ First In First Out และ ตัดแบบ Last In First Out

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้โมดูลการตัดยอดสินค้าในแบบต่าง ๆ
2) ได้เรียนรู้การทำงานของกรอบงานออฟบิส
3) ได้เรียนรู้การพัฒนากรอบงาน

บทที่ 2

ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง โดยมีรายละเอียดของระบบคลังสินค้า กรอบงานเชิงวัตถุ ภาษาจาวา และงานวิจัยที่เกี่ยวข้อง

2.1 ระบบคลังสินค้า

2.1.1 ความหมาย

“คำว่า “คลัง” หมายถึง สถานที่สำหรับเก็บของเป็นจำนวนมากๆ ดังนั้น “คลังสินค้า” (Warehouse) จึงหมายถึง สถานที่สำหรับเก็บสินค้าจำนวนมากๆ และ “การคลังสินค้า” (Warehousing) หมายถึง การเก็บรักษาสินค้านั่นเอง การคลังสินค้าเป็นหน้าที่หนึ่งของระบบการจัดจำหน่าย ทำการเก็บรักษาสินค้าในช่วงเวลาที่สินค้าได้ผลิตเสร็จแล้วและรอการจำหน่าย สินค้าดังกล่าวอาจจะเป็นสินค้าที่จะเป็นวัตถุดิบสำหรับกระบวนการผลิตในขั้นตอนต่อไปหรือเป็นสินค้าสำเร็จรูปที่จะนำไปใช้บริโภค ดังนั้นสินค้าคงคลัง (Inventory) ที่จัดเก็บในคลังสินค้าจึงจำแนกได้เป็น 2 ประเภท คือ วัตถุดิบและสินค้าสำเร็จรูปและอาจจะมีสินค้าที่ยังอยู่ในระหว่างการผลิตไม่เสร็จเก็บอยู่ในคลังสินค้าด้วยก็ได้แต่จะมีจำนวนน้อย” (คำนาย อภิปรัชญาสกุล, 2550)

คำว่า “คลังสินค้า” และคำอื่น ๆ ที่มีความหมายใกล้เคียงกัน หรือเกี่ยวเนื่องกับคำว่า “คลังสินค้า” นั้น ได้มีคำจำกัดความให้ไว้ในที่ต่าง ๆ หลายแห่ง แม้จะมีความแตกต่างกันในเรื่องการใช้ถ้อยคำ และเน้นถึงความหมายโดยเฉพาะสำหรับคลังสินค้าแต่ละประเภทที่มีวัตถุประสงค์ในการประกอบกิจการแตกต่างกันออกไปก็ตาม แต่ความหายเหล่านั้นสอดคล้องต้องกันในประเด็นสำคัญที่จะบอกให้ทราบว่า คลังสินค้าคืออะไรและคลังสินค้าทำหน้าที่อย่างไร ยกตัวอย่างดังต่อไปนี้

- **ตาม TM. 743-200, Storage and Materials Handing.** (Tompkins & Smith, 1998) ได้ให้คำนิยามไว้ ดังนี้

1. คลังสินค้า (Warehouse) หมายถึง พื้นที่ภายในอาคาร ซึ่งออกแบบขึ้นเพื่อใช้ในการเก็บรักษาพัสดุ และสร้างขึ้นโดยให้มีหลังคาและฝาผนังที่สมบูรณ์ทั้งด้านข้างและด้านหัวท้ายของอาคาร

2. การคลังสินค้า (Warehousing) หมายถึง การปฏิบัติทางกายภาพเกี่ยวกับการรับ การเก็บรักษา และการจ่ายพัสดุ
3. การเก็บรักษา (Storage) หมายถึง การเก็บเอาไว้ หรือการจัดวางทรัพย์สินในคลังสินค้าในโรงเก็บหรือในพื้นที่เก็บรักษากลางแจ้ง
4. การจัดเก็บ (Storing) หมายถึง การจัดวางพัสดุอย่างเป็นระเบียบในการเก็บรักษา

- **ตามเงื่อนไขควบคุมคลังสินค้า พ.ศ. 2526** (คำนาย อภิปรัชญาสกุล, 2550) ซึ่งกระทรวงพาณิชย์ออกใช้บังคับเพื่อการควบคุมคลังสินค้าสาธารณะ ได้ให้นิยามศัพท์ ซึ่งเป็นความหมายโดยเฉพาะสำหรับคลังสินค้าสาธารณะ ซึ่งได้รับอนุญาตให้ประกอบกิจการคลังสินค้าตามกฎหมาย และดำเนินงานเป็นธุรกิจเอกเทศในลักษณะอุตสาหกรรมบริการไว้ดังนี้
    1. **คลังสินค้า** หมายถึง อาคารที่มีโครงสร้างมั่นคงแข็งแรงผนังทำด้วยอิฐหรือคอนกรีตบล็อก หรือคอนกรีตเสริมเหล็ก หรือ วัสดุที่มีความมั่นคงแข็งแรง ทนทานหลังคาต้องมุงด้วยกระเบื้อง หรือสังกะสี หรือวัสดุที่มีความแข็งแรงทนทาน พื้นต้องทำด้วยคอนกรีตเสริมเหล็กสามารถรับน้ำหนักได้ไม่น้อยกว่าสามสิบเมตริกตันต่อหนึ่งตารางเมตร
    2. **การคลังสินค้า** หมายถึง การรับทำการเก็บรักษาสินค้า หรือการรับทำการเก็บรักษาสินค้าและให้บริการเกี่ยวกับสินค้านั้น เพื่อบำเหน็จเป็นทางค้าปกติ ไม่ว่าบำเหน็จนั้นจะเป็นเงินค่าตอบแทนหรือประโยชน์อื่นใด
- **กระบวนการ** (Process) (คำนาย อภิปรัชญาสกุล, 2550) คือ กิจการ หรือหน้าที่ในการจัดการที่เป็นขั้นตอนตามลำดับ ซึ่งผู้บริหารจะต้องกระทำ ซึ่งได้แก่
    1. **การวางแผน** (Planning) เป็นเรื่องเกี่ยวกับการกำหนดวัตถุประสงค์ นโยบาย โครงการ และวิธีดำเนินการ เพื่อให้บรรลุผลตามวัตถุประสงค์ นโยบายและโครงการที่กำหนดไว้
    2. **การจัดองค์การ** (Organizing) เป็นเรื่องของของการจัดโครงสร้างและบทบาทต่าง ๆ โดยพิจารณากำหนดกิจกรรมต่าง ที่จำเป็นสำหรับการบรรลุเป้าหมายขององค์การ

3. **การจัดคนเข้าทำงาน** (Staffing) เป็นเรื่องการจัดการเกี่ยวกับตัวบุคคล และการจัดให้บุคคลเข้าทำงาน ในตำแหน่งที่กำหนดไว้ในโครงสร้างขององค์การ
4. **การสั่งการ** (Directing) เป็นเรื่องของการให้แนวทางและการบังคับบัญชา โดยเฉพาะ การสั่งการ หมายถึง การออกคำสั่งการปฏิบัติที่ถูกต้องเหมาะสม เพื่อให้การดำเนินการเป็นไปตามแผนที่วางไว้
5. **การประสานงาน** (Coordination) เป็นเรื่องของผู้บริหารจะต้องใช้มาตรการที่เหมาะสม เพื่อให้กิจกรรมกลุ่มย่อย ที่เป็นองค์ประกอบขององค์การได้ร่วมกันดำเนินงานต่อไปได้ อย่างสอดคล้อง
6. **การควบคุม** (Controlling) เป็นเรื่องเกี่ยวกับการกำหนดแนวดำเนินงานตามที่วางแผนไว้ เพื่อให้นำมาปฏิบัติอย่างมีประสิทธิผล ซึ่งเป็นลำดับขั้นตอนสุดท้ายของการจัดการ

- **การบูรณาการ** (Integration) คือ การสร้างความสมดุล และความสอดคล้องต่อเนื่องกันระหว่างทรัพยากรต่าง ๆ เพื่อความสำเร็จขององค์การ
- **ทรัพยากร** (Resource) (คำนาย อภิปรัชญาสกุล, 2550) คือ ทรัพยากรในการจัดการคลังสินค้า ได้แก่

1. **ทรัพยากรเนื้อที่** คือ ที่ดินและอาคารทั้งหมด ที่ใช้ในการเก็บรักษาสินค้า และที่ใช้สำหรับงานอื่น ๆ เพื่อความสำเร็จขององค์การ
2. **ทรัพยากรมนุษย์** คือ บุคลากร และกำลังคนในหน่วยงานต่าง ๆ ของกิจการคลังสินค้า ตลอดจนทักษะที่จำเป็นสำหรับการดำเนินงานคลังสินค้า
3. **ทรัพยากรเครื่องมือ** คือ อุปกรณ์ เครื่องมือยกขน และอุปกรณ์ทั้งปวงที่จำเป็นต้องใช้ในการประกอบกิจการคลังสินค้า
4. **ทรัพยากรทุน** คือ การหาเงินทุน และการใช้จ่ายเงินทุนในการประกอบกิจการ
5. **ทรัพยากรเวลา** คือ ทรัพยากรที่มีอยู่อย่างจำกัดในการดำเนินงานคลังสินค้า ทั้งนี้รวมถึงกำหนดเวลาและระยะเวลาในการทำงานในแต่ละกิจกรรม

- **ประสิทธิผล** (Effectiveness) (คำนาย อภิปรัชญาสกุล, 2550) หมายถึง ความสามารถขององค์การที่จะบริหารงานให้บรรลุวัตถุประสงค์ได้เป็นอย่างดี ในช่วงเวลาที่กำหนด
- **วัตถุประสงค์** (Objective) (คำนาย อภิปรัชญาสกุล, 2550) หมายถึง ผลประโยชน์สุดท้ายที่กำหนดไว้สำหรับการประกอบกิจการคลังสินค้าแต่ละประเภท โดยเหตุผลเบื้องต้นในการดำเนินงานของทุกองค์การ คือความอยู่รอดขององค์การ ดังนั้นคลังสินค้าแต่ละประเภทจะต้องมีเป้าหมายในการดำเนินงาน หรือมีวัตถุประสงค์ที่ผู้บริหารกิจการคลังสินค้า จะต้องดำเนินการให้บรรลุผลสำเร็จ

2.1.2 วิวัฒนาการของการคลังสินค้า

"การจัดเก็บรักษาสินค้าเป็นกิจกรรมที่สำคัญอย่างหนึ่งทางเศรษฐกิจ ในสมัยก่อนแต่ละครอบครัวต่างช่วยเหลือตัวเอง ในการแสวงหาอาหาร และปัจจัยสี่ เพื่อเก็บรักษาไว้สำหรับใช้ในยามจำเป็น เช่น ในสมัยโบราณมีการเก็บรักษาอาหารไว้ในห้องเก็บของใต้ดินเพื่อป้องกันมิให้อาหารเน่าเปื่อยได้ ต่อมาเมื่อระบบการขนส่งคล่องตัวขึ้น การเก็บรักษาสินค้าจึงได้พัฒนาการเก็บในครัวเรือนไปเป็นการเก็บรักษาของผู้ค้าปลีก ผู้ค้าส่ง และผู้ผลิต จากสภาพทางการตลาดได้ชี้ให้เห็นว่าคลังสินค้าได้เริ่มกำหนดขึ้นมาเป็นหน่วยงานเก็บรักษาสินค้าสนับสนุนกระบวนการตลาด เพื่อให้บรรลุเป้าหมายการเก็บรักษาสินค้าในการตอบสนองอุปสงค์ในตลาดที่เกิดขึ้นอย่างไม่แน่นอน ดังนั้นคลังสินค้าจึงเป็นสถานที่สำหรับเก็บรักษาสินค้าไว้จนกว่าจะมีการจำหน่ายออกไปเพื่อสนองความต้องการของผู้ใช้" (คำนาย อภิปรัชญาสกุล, 2550)

2.1.3 ลักษณะและความสำคัญของการคลังสินค้า

คลังสินค้าทำหน้าที่ในการเก็บสินค้าระหว่างกระบวนการจัดส่ง ซึ่งสินค้าที่เก็บไว้สามารถแบ่งได้หลายประเภท ได้แก่ วัตถุดิบ (Material) ชิ้นส่วน (Components) ชิ้นส่วนต่าง ๆ (Parts) สินค้าสำเร็จรูป (Finished Goods) ในบางกระบวนการผลิตคำว่า "สินค้า" จะนับรวมไปถึงงานระหว่างผลิต (Work in Process) ตลอดจนสินค้าที่ต้องการทิ้ง (Disposed) และวัสดุที่นำมาใช้ใหม่ (Recycle) แม้ว่าสัดส่วนของสินค้าประเภทนี้จะมีไม่มากก็ตามวัตถุประสงค์ของการใช้คลังสินค้ามีหลายประเภทดังต่อไปนี้ (คำนาย อภิปรัชญาสกุล, 2550)

1. เพื่อให้เกิดการประหยัดในการขนส่ง
2. เพื่อให้เกิดการประหยัดในการผลิต

3. เพื่อต้องการลดการสั่งซื้อจำนวนมากหรือส่วนลดจากการสั่งซื้อล่วงหน้า
4. เพื่อเป็นแหล่งของวัตถุดิบ ส่วนประกอบ และชิ้นส่วนที่ใช้ในการผลิต
5. เพื่อสนับสนุนนโยบายการให้บริการลูกค้า
6. เพื่อให้สามารถรองรับการเปลี่ยนแปลงของสภาวะทางการตลาด เช่น ความต้องการสินค้าที่ผันผวน ความต้องการสินค้าแบบฤดูกาล หรือสภาวะการแข่งขันที่สูง
7. เพื่อลดเวลานำ (Lead time) ของการสั่งซื้อสินค้า
8. เพื่อสนับสนุนระบบการผลิตแบบทันเวลาพอดี (JIT : Just in Time) ของผู้ขายปัจจัยการผลิตและลูกค้า
9. เพื่อให้สามารถขนส่งสินค้าได้หลายประเภท
10. เพื่อใช้เป็นที่เก็บสินค้าชั่วคราวสำหรับสินค้าที่ต้องทิ้งหรือที่ต้องนำไปผลิตใหม่

2.1.4 ปัจจัยในการพิจารณาในการจัดการคลังสินค้า

ในการกำหนดนโยบายที่เกี่ยวกับคลังสินค้าเพื่อให้สอดคล้องกับวัตถุประสงค์ดังกล่าวข้างต้น มีปัจจัยที่ใช้พิจารณาดังนี้ (คำนาย อภิปรัชญาสกุล, 2550)

1. ประเภทของอุตสาหกรรม ปรัชญาของธุรกิจ ความเพียงพอของเงินลงทุน
2. ลักษณะของสินค้า ขนาดสินค้า สินค้าที่เป็นฤดูกาล ความเน่าเสียง่ายของสินค้า การทดแทนกันได้ของสินค้า และความเสื่อมของสินค้า
3. สภาวะทางเศรษฐกิจ สภาวะของการแข่งขัน
4. กระบวนการผลิตที่ใช้ การใช้ระบบผลิตแบบทันเวลาพอดี

2.1.5 ประโยชน์ของการจัดการคลังสินค้า

การจัดการนี้สามารถใช้งานได้ทั้งในด้านการเป็นแหล่งอุปทาน (Physical supply) และการกระจายสินค้า (Physical distribution) ดังนี้ (คำนาย อภิปรัชญาสกุล, 2550)

1. คลังสินค้าช่วยสนับสนุนการผลิต (Manufacturing support) โดยคลังสินค้าทำหน้าที่ในการรวบรวมวัตถุดิบในการผลิตชิ้นส่วนและส่วนประกอบต่าง ๆ จากผู้ขายปัจจัยการผลิต เพื่อส่งป้อนให้กับโรงงานเพื่อผลิตเป็นสินค้าสำเร็จรูปต่อไป

2. คลังสินค้าเป็นที่ผสมผลิตภัณฑ์ (Mixing warehouse) ในกรณีที่มีการผลิตสินค้าจากโรงงานหลายแห่ง โดยอยู่ในรูปของคลังสินค้ากลาง (Central warehouse) จะทำหน้าที่รวบรวมสินค้าสำเร็จรูปจากโรงงานต่าง ๆ ไว้ในที่เดียวกัน (Mixing warehouse) เพื่อส่งมอบให้ลูกค้าตามต้องการขึ้นอยู่กับลูกค้าแต่ละรายว่าต้องการสินค้าจากโรงงานใดบ้าง
3. คลังสินค้าเป็นที่รวบรวมสินค้า (Consolidation warehouse) ในกรณีที่ลูกค้าต้องการซื้อสินค้าจำนวนมากจากโรงงานหลายแห่ง คลังสินค้าจะช่วยรวบรวมสินค้าจากหลายแหล่งเพื่อจัดเป็นขนส่งขนาดใหญ่หรือทำให้เต็มเที่ยวซึ่งช่วยประหยัดค่าขนส่ง
4. คลังสินค้าใช้ในการแบ่งแยกสินค้าให้มีขนาดเล็กลง (Break Bulk warehouse) ในกรณีที่การขนส่งจากผู้ผลิตมีหีบห่อหรือพาเลทขนาดใหญ่ คลังสินค้าจะเป็นแหล่งที่ช่วยในการแบ่งแยกสินค้าให้มีขนาดเล็กลงเพื่อส่งมอบกับลูกค้ารายย่อยต่อไป

2.1.6 ประเภทของคลังสินค้า

สามารถจำแนก คลังสินค้าตามลักษณะของจุดมุ่งหมายได้เป็น 3 กลุ่มใหญ่ๆ ดังต่อไปนี้

- กลุ่มคลังสินค้าสาธารณะ (Public warehouse)
- กลุ่มคลังสินค้าส่วนบุคคล (Private warehouse)
- กลุ่มคลังเก็บวัสดุ (Material warehouse)

2.1.6.1 กลุ่มคลังสินค้าสาธารณะ

คลังสินค้าสาธารณะเป็นกิจการทางธุรกิจที่เป็นเอกเทศเฉพาะ ไม่ใช้เป็นส่วนของกิจการหลักอย่างอื่นและเป็นกิจการแขนงหนึ่งของอุตสาหกรรมบริการ ซึ่งจัดให้มีสิ่งอำนวยความสะดวกในการเก็บรักษาสินค้าและให้บริการรับจัดเก็บรักษาสินค้ารวมทั้งให้บริการต่าง ๆ เกี่ยวกับสินค้า เพื่อบำเหน็จตอบแทนเป็นทางการค้าปกติของกิจการ หรือคลังสินค้า อาจเป็นคลังสินค้าที่จัดตั้งขึ้นในรูปของบริษัทจำกัดมีวัตถุประสงค์ในการดำเนินงานเกี่ยวกับการให้บริการรับฝากสินค้าและบริการอื่น ๆ ที่เกี่ยวกับคลังสินค้า โดยเฉพาะ ทั้งนี้เพื่อหวังค่าตอบแทนจากการให้บริการนั้น ในประเทศไทยการจัดตั้งคลังสินค้าสาธารณะจะต้องได้รับอนุญาตจาก

รัฐมนตรีว่าการกระทรวงพาณิชย์ และประกอบกิจการภายใต้ เงื่อนไขการควบคุมของกระทรวงพาณิชย์ด้วย

ตามหลังสากลคลังสินค้าสาธารณะสามารถจำแนกตามลักษณะการใช้งานออกเป็น 6 รูปแบบ ดังนั้น

1. คลังสินค้าทั่วไป (General merchandise warehouse) เป็นคลังสินค้าที่ออกแบบสำหรับเก็บรักษา สินค้าชนิดใดก็ได้ของผู้ผลิต ผู้ขายส่งและผู้ขายปลีก
2. คลังสินค้าห้องเย็น (Refrigerated or cold-storage warehouse) คือ คลังสินค้าที่มีอุปกรณ์ทำความเย็น สามารถควบคุมอุณหภูมิให้อยู่ในระดับที่ต้องการได้
3. คลังสินค้าทัณฑ์บน (Bonded warehouse) คือ คลังสินค้าที่ตั้งขึ้นสำหรับเก็บรักษาสินค้าบางชนิดซึ่งต้องรอการดำเนินการตามระเบียบของศุลกากร
4. คลังสินค้าเฟอร์นิเจอร์และเครื่องใช้ในครัวเรือน (Houses hold-goods and furniture warehouse) คือ คลังสินค้าที่ทำหน้าที่เก็บรักษาทรัพย์สินส่วนบุคคลมากกว่าเก็บรักษาสินค้าและมักจะเป็นการเก็บรักษาชั่วคราวและอาจจะเป็นในพื้นที่โล่ง
5. คลังสินค้าสำหรับพืชผลเฉพาะอย่าง (Special-commodity warehouse) คือ คลังสินค้าที่ใช้สำหรับเก็บรักษาผลิตผลทางเกษตรกรรม
6. คลังสินค้าสำหรับสินค้าที่มีลักษณะเป็นกองใหญ่ (Bulk storage warehouse) คือ คลังสินค้าที่ทำการเก็บรักษาสินค้าที่มีจำนวนมากเป็นกองใหญ่

2.1.6.2 กลุ่มคลังสินค้าส่วนบุคคล

คลังสินค้าส่วนบุคคลเป็นกิจการคลังสินค้าที่อำนวยประโยชน์แก่กิจการหลักอย่างอื่นไม่ใช่กิจการคลังสินค้าที่เป็นธุรกิจเฉพาะเช่นเดียวกับคลังสินค้าสาธารณะ จุดหมายในการจัดตั้งและประกอบกิจการของคลังสินค้าส่วนบุคคล คือ การเก็บรักษาสินค้าเพื่อสนับสนุนวัตถุประสงค์ของกิจการอันเป็นธุรกิจหลักที่เป็นเจ้าของสินค้านั้น ซึ่งผู้ประกอบธุรกิจที่เป็นหลักนั้นอาจ เป็นบริษัทเอกชน องค์การรัฐบาลหรือสหกรณ์ สิ่งอำนวยความสะดวกในการเก็บรักษาอาจเป็นอาคารคลังแบบหนึ่งแบบใดหรือเป็นเพียง พื้นที่เก็บรักษาที่อยู่ในอาคารเดียวกันกับกิจการอันเป็น

ธุรกิจหลักของบริษัทหรือองค์การนั้นเองก็ได้ ทั้งนี้ขึ้นอยู่กับลักษณะของสินค้าที่เก็บรักษาและปริมาณที่จำเป็นต้องเก็บรักษาไว้ในขณะใดขณะหนึ่ง

คลังสินค้าเอกชน คือ คลังสินค้าที่กิจการเป็นเจ้าของเองหรือเช่ามีวัตถุประสงค์ในการเก็บรักษาวัตถุดิบและสินค้าสำเร็จรูปของผู้ผลิตหรือผู้จำหน่ายโดยเป็นเจ้าของคลังสินค้า และไม่มีการรับฝากสินค้าจากผู้อื่น คลังสินค้าประเภทนี้จะเป็นองค์ประกอบส่วนหนึ่งที่สนับสนุนการประกอบธุรกิจอื่นของกิจการสามารถอำนวยประโยชน์แก่กิจการหลายๆประการดังนี้

1. การออกแบบ เจ้าของกิจการสามารถออกแบบคลังสินค้าเพื่อนสนองความต้องการพิเศษของตนเองได้
2. การควบคุม กิจการสามารถมอบหมายความรับผิดชอบและควบคุมการดำเนินงานทุก ๆ ด้านด้วยตนเอง ตลอดจนสามารถกำหนดเป้าหมายและนโยบายให้สอดคล้องกับการผลิตและการขายได้
3. ค่าใช้จ่าย ค่าใช้จ่ายในการดำเนินงานโดยทั่วไปของคลังสินค้าทั้งสองประเภทจะคล้ายคลึงกัน ยกเว้นค่าใช้จ่ายที่เกี่ยวกับการขายและโฆษณาที่คลังสินค้าเอกชนจะไม่มี ดังนั้นค่าใช้จ่ายดำเนินงานต่อหน่วยของคลังสินค้าเอกชน จึงมักต่ำกว่าและค่อนข้างคงที่กว่าคลังสินค้าสาธารณะ
4. การติดต่อกับลูกค้า กิจการที่มีคลังสินค้าของตนเองย่อมสามารถติดต่อกับลูกค้าและตลาดได้สะดวกกว่าและเมื่อมีปัญหาเกิดขึ้นกับลูกค้าย่อมจะตรวจสอบและแก้ไขได้อย่างรวดเร็ว
5. การใช้ประโยชน์ร่วมกัน คลังสินค้าเอกชนนอกจากทำหน้าที่เก็บรักษาสินค้าของกิจการแล้ว ยังอาจทำหน้าที่เป็นหน่วยงานซื้อหรือขายระดับท้องถิ่นของกิจการด้วย เพราะมีสถานที่ บุคลากร อุปกรณ์และเครื่องมือเครื่องใช้ที่จะใช้ประโยชน์ร่วมกันได้

2.1.6.3 กลุ่มคลังเก็บพัสดุ

คำว่า “พัสดุ” หมายถึง สิ่งของที่มีไว้เพื่อใช้งาน หรือที่มีความมุ่งหมายที่จะนำไปใช้งาน ซึ่งต่างกับคำว่า “สินค้า” ซึ่งเป็นสิ่งของที่มีไว้ขายหรือมุ่งหมายที่จะนำไปเพื่อขายเป็นค้าหากำไรในธุรกิจ วัสดุอาจเป็นได้ทั้งสิ่งที่หมดสภาพไปทันทีในการใช้หรือเป็นของใช้ หรือเครื่องใช้ที่คงรูปเดิมอยู่ต่อไปในการใช้งาน หรืออาจเป็นอุปกรณ์ที่นำไปประกอบรวมในการผลิต

หรือซ่อมก็ได้ การจัดการวัสดุ หมายถึง กระบวนการพัสดุแก่องค์การตามความต้องการ เพื่อให้สามารถดำเนินกิจการได้สำเร็จตามวัตถุประสงค์ของ องค์การนั้นผู้จัดการวัสดุขององค์การผู้ผลิตทำหน้าที่เกี่ยวกับการจัดหาพัสดุ ซึ่งนำไปแปลงสภาพเป็นผลผลิตสำเร็จรูปขององค์การนั้น ส่วนผู้จัดการวัสดุขององค์การบริการ เช่น องค์การรัฐบาล สายการบิน โรงพยาบาล ธนาคาร และโรงแรม เป็นต้น ไม่ได้จัดซื้อวัตถุที่จะไปแปรรูปเป็นผลิตภัณฑ์ เพื่อขายแต่จัดให้มีพัสดุที่จำเป็นต้องใช้ในการปฏิบัติงาน

คลังเก็บวัสดุเป็นการอำนวยประโยชน์ของการจัดการวัตถุ ทำหน้าที่เก็บรักษาวัสดุ เพื่อจ่ายสนองความต้องการของกิจการหรือองค์การที่เป็นเจ้าของคลังเก็บวัสดุนั้นในการผลิตหรือการบริการ แล้วแต่กรณี คลังเก็บวัสดุมีลักษณะเช่นเดียวกันกับคลังสินค้าส่วนบุคคล แต่ต่างกันในเรื่องสิ่งที่เก็บรักษานั้นไม่ใช่สินค้าสำหรับขายแต่เป็นพัสดุสำหรับสนองความต้องการในการใช้สำหรับการปฏิบัติ งานภายในของกิจการนั้น ๆ

2.2 เทคนิคการตัดยอดสินค้า

มีอยู่หลายแบบ เราจะพูดถึง 3 แบบ คือ FIFO (First in First out), FEFO (First Expire date First Out) และ LIFO (Last In First Out)

2.2.1 FIFO (First In First Out) (Donald, 2004) หรือ เข้าก่อนออกก่อน คือ การจ่ายสินค้าตามหลักที่ว่า มาก่อนใช้ก่อน หรือ มาก่อนเอาออกก่อน ซึ่งเป็นวิธีที่นิยมใช้กันในคลังสินค้าเพื่อลดความเสียหายที่เกิดจากการเก็บของไว้เป็นเวลานาน เพราะสินค้าบางประเภทไม่มีวันหมดอายุที่แน่นอน แต่ถ้าเก็บไว้นานอาจทำให้ตัวสินค้าเสียหายได้

ตัวอย่างการคิดการตัดยอดแบบ FIFO

วันที่ 1/1/2560 มีสินค้า A อยู่ในคลังจำนวน 100 ชิ้น กำหนดให้เป็น A1

วันที่ 3/1/2560 มีการซื้อสินค้า A เข้าคลังเพิ่มอีกจำนวน 50 ชิ้น กำหนดให้เป็น A2 ทำให้ตอนนี้มีสินค้า A ในคลังเท่ากับจำนวน 150 ชิ้น

วันที่ 5/1/2560 มีการขายสินค้า A ได้จำนวน 70 ชิ้น ทำให้เหลือสินค้า A ในคลังเท่ากับจำนวน 80 ชิ้น โดยเป็น A1 จำนวน 30 ชิ้น และ A2 จำนวน 50 ชิ้น

วันที่ 10/1/2560 มีการขายสินค้า A ได้จำนวน 50 ชิ้น ทำให้เหลือสินค้า A ในคลังเท่ากับ 30 ชิ้น โดยเป็น A2 จำนวน 30 ชิ้น

ตารางที่ 2.1 ตารางแสดงตัวอย่างการตัดยอดสินค้าแบบ FIFO

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
1/1/2560			A1 = 100
3/1/2560	A = 50		A1 = 100 A2 = 50
5/1/2560		A = 70	A1 = 100-70 =30 A2 = 50
10/1/2560		A = 50	A1 = 30-30 = 0 A2 = 50-20 = 30

จากตารางที่ 2.1 จะพบว่า สินค้าจะถูกนำออกตามลำดับ โดยสินค้าที่เข้ามาในคลังก่อน จะถูกนำออกขายก่อน เมื่อหมดจึงจะนำสินค้าในลำดับถัดไปออกมาขายเรื่อย ๆ ตามลำดับ

2.2.2 FEFO (First Expire date First Out) (Donald, 2004) หรือ หมดอายุก่อนออกก่อน คือ การจ่ายสินค้าตามวันหมดอายุ เป็นวิธีที่นิยมใช้กันในร้านขายของ เพื่อลดความเสียหายที่จะเกิดจากการหมดอายุ

ตัวอย่างการคิดการตัดยอดแบบ FEFO

วันที่ 1/1/2560 มีสินค้า A เป็นสินค้าที่หมดอายุในวันที่ 1/2/2560 อยู่ในคลังจำนวน 100 ชิ้น กำหนดให้เป็น A1

วันที่ 2/1/2560 มีการซื้อสินค้า A หมดอายุในวันที่ 1/2/2560 เข้าคลังเพิ่มอีกจำนวน 50 ชิ้น จึงให้เป็น A1 เหมือนกันกับสินค้าของวันที่ 1/1/2560 เพราะหมดอายุพร้อมกัน ทำให้ตอนนี้มีสินค้า A ในคลังเท่ากับจำนวน 150 ชิ้น

วันที่ 3/1/2560 มีการซื้อสินค้า A หมดอายุในวันที่ 3/2/2560 เข้าคลังเพิ่มอีกจำนวน 50 ชิ้น กำหนดให้เป็น A2 เพราะหมดอายุหลัง A1 ทำให้ตอนนี้มีสินค้า A ในคลังเท่ากับจำนวน 200 ชิ้น โดยเป็น A1 จำนวน 150 ชิ้น และ A2 จำนวน 50 ชิ้น

วันที่ 5/1/2560 มีการขายสินค้า A ได้จำนวน 100 ชิ้น ทำให้เหลือสินค้า A ในคลังเท่ากับจำนวน 100 ชิ้น โดยเป็น A1 จำนวน 50 ชิ้น และ A2 จำนวน 50 ชิ้น

วันที่ 10/1/2560 มีรับสินค้า A ของคลังในเครือที่ค้างคลังมาช่วยขายจำนวน 50 ชิ้น แต่เป็นสินค้าที่หมดอายุวันที่ 25/1/2560 จึงต้องขายสินค้าที่รับมาใหม่นี้ก่อน กำหนดให้เป็น A3 และทำให้

ตอนนี้มีสินค้า A ในคลังเท่ากับจำนวน 150 ชิ้น โดยเป็น A1 จำนวน 50 ชิ้น A2 จำนวน 50 ชิ้น และ A3 จำนวน 50 ชิ้น

วันที่ 15/1/2560 มีการขายสินค้า A ได้จำนวน 120 ชิ้น ทำให้เหลือสินค้า A ในคลังเท่ากับ 30 ชิ้น โดยเป็น A2 จำนวน 30 ชิ้น

ตารางที่ 2.2 ตารางแสดงตัวอย่างการตัดยอดสินค้าแบบ FEFO

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
1/1/2560			A1(1/2/2560) = 100
2/1/2560	A(1/2/2560) = 50		A1(1/2/2560) = 100 + 50 = 150
3/1/2560	A(3/2/2560) = 50		A1(1/2/2560) = 150 A2(3/2/2560) = 50
5/1/2560		A = 100	A1(1/2/2560) = 150-100 = 50 A2(3/2/2560) = 50
10/1/2560	A(25/1/2560) = 50		A1(1/2/2560) = 50 A2(3/2/2560) = 50 A3(25/1/2560) = 50
15/1/2560		A = 120	A1(1/2/2560) = 50-50 = 0 A2(3/2/2560) = 50-20 = 30 A3(25/1/2560) = 50-50 = 0

จากตารางที่ 2.2 จะพบว่า แม้ว่าจะมีสินค้าที่เข้ามาทีหลัง แต่สินค้าชิ้นนั้น หมดอายุก่อนชิ้นอื่น ๆ ก็ต้องขายสินค้าชิ้นนั้นออกไปก่อน เพื่อไม่ให้เกิดปัญหาสินค้าหมดอายุก่อนจำหน่าย

2.2.3 LIFO (Last In First Out) (Donald, 2004) หรือ เข้าทีหลังออกก่อน คือ การจ่ายสินค้าโดยการนำสินค้าที่เข้าทีหลังออกมาใช้ก่อน ซึ่งปกติแล้วจะใช้กับ วัตถุดิบในการผลิต สินค้าที่มีอายุจำกัด หรือสินค้าเทคโนโลยีที่ล้าสมัยได้เร็ว เช่น คอมพิวเตอร์ โปรแกรมคอมพิวเตอร์ เป็นต้น

ตัวอย่างการคิดการตัดยอดแบบ LIFO

วันที่ 1/1/2560 มีมือถือ A อยู่ในคลังจำนวน 100 ชิ้น กำหนดให้เป็น A1

วันที่ 3/1/2560 มีการซื้อมือถือ A เข้าคลังเพิ่มอีกจำนวน 50 ชิ้น กำหนดให้เป็น A2 ทำให้ตอนนี้มีมือถือ A ในคลังจำนวน150 ชิ้น

วันที่ 5/1/2560 มีการขายมือถือ A ได้จำนวน 70 ชิ้น ทำให้เหลือมือถือ A ในคลังจำนวน 80 ชิ้น โดยเป็น A1 จำนวน 80 ชิ้น

วันที่ 10/1/2560 มีการขายมือถือ A ได้จำนวน 50 ชิ้น ทำให้เหลือมือถือ A ในคลังจำนวน 30 ชิ้น โดยเป็นจำนวน A1 = 30 ชิ้น

ตารางที่ 2.3 ตารางแสดงตัวอย่างการตัดยอดสินค้าแบบ LIFO

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
1/1/2560			A1 = 100
3/1/2560	A = 50		A1 = 100 A2 = 50
5/1/2560		A = 70	A1 = 100-20 = 80 A2 = 50-50 = 0
10/1/2560		A = 50	A1 = 80-50 = 30

จากตารางที่ 2.3 จะพบว่า รูปแบบการทำงานจะเหมือนกันกับ FIFO คือ ขายออกตามลำดับ แต่จะสลับกัน คือ ขายชิ้นที่เข้ามาทีหลังก่อน เมื่อหมดค่อยนำสินค้าในลำดับก่อนหน้ามาขายต่อไป

2.3 ภาษาจาวา

2.3.1 ความหมาย

ภาษาจาวา (Java) (สุดา เธียรมนตรี, 2556) เป็นภาษาโปรแกรมที่พัฒนาขึ้นโดยทีมวิจัยโครงการชื่อ Green ของบริษัท Sun Microsystems ซึ่งมีวัตถุประสงค์เพื่อพัฒนาภาษาที่ใช้สร้างโปรแกรม หรือระบบงานที่สนับสนุนการทำงานของแต่ละระบบงานย่อยๆ ทำให้จาวาถูกพัฒนาขึ้นบนคุณสมบัติดังนี้

- ง่ายต่อการเรียนและเข้าใจ เนื่องจากมีกลไกของภาษาไม่ซับซ้อน

- มีความคงทน (Robust) เนื่องจากมีการตรวจจับของผิดพลาด (Exception Handling) และมีกลไกในการคืนพื้นที่ในหน่วยความจำอัตโนมัติ (Garbage Collection)
- มีความปลอดภัยสูง
- ทำงานได้ในทุกระบบ คือ มีคุณลักษณะของจาวาแพลตฟอร์ม (Platform Independent)
- มีคลาสและอินเตอร์เฟสให้ใช้เป็นจำนวนมาก

จากคุณสมบัติดังกล่าวทำให้จาวาสนับสนุนการเขียนโปรแกรมเชิงวัตถุที่เน้นการแบ่งงานให้เป็นส่วนย่อย ๆ ที่มีความสัมพันธ์กันและเป็นอิสระจากส่วนย่อยของงานอื่น ๆ ซึ่งช่วยลดความซ้ำซ้อนและเวลาของการพัฒนางาน อีกทั้งเมื่อต้องการเปลี่ยนแปลงแก้ไขงาน จะไม่ส่งผลกระทบไปยังงานอื่น ๆ ทำให้ลดต้นทุนในการพัฒนาและบำรุงรักษา

2.3.2 หลักการทำงาน

การพัฒนาโปรแกรมด้วยภาษาจาวาจะเริ่มต้นจากการเขียนซอร์สโค้ดโปรแกรมจนได้ไฟล์ซอร์สโค้ดที่มีนามสกุลเป็น .java (สุดา เธียรมนตรี, 2556) ซึ่งก็คือ ไฟล์ซอร์สโค้ดที่เราได้เขียนขึ้นและเมื่อรันโปรแกรมซอร์สโค้ดดังกล่าวจะถูกคอมไพล์เป็นจาวาไบต์โค้ดซึ่งจะเก็บในรูปของไฟล์ .class

เมื่อโปรแกรมทำงานบนคอมพิวเตอร์หรืออุปกรณ์ใด ๆ ก็ตาม จาวาไบต์โค้ดจะถูกตีความด้วย Java Virtual Machine (JVM) ให้เป็นภาษาเครื่องเฉพาะอุปกรณ์ชนิดนั้น ๆ เพื่อให้โปรแกรมสามารถทำงานบนอุปกรณ์นั้น ๆ ได้ ซึ่งจะต้องติดตั้ง Java Runtime Environment (JRE) ก่อนเสมอ

2.3.3 โปรแกรมเชิงวัตถุ

ภาษาจาวาเป็นภาษาเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming : OOP) ที่แบ่งขอบเขตของงานออกเป็นส่วนย่อย ๆ ที่เป็นอิสระต่อกัน โดยมองส่วนต่าง ๆ เป็นวัตถุหรือออบเจ็กต์ที่ไม่ขึ้นต่อกัน แต่มีการทำงานร่วมกัน ซึ่งวัตถุในมุมมองของ OOP ไม่จำเป็นต้องเป็นสิ่งที่จับต้องได้แต่ให้มีอยู่จริง (สุดา เธียรมนตรี, 2556)

โดยคุณสมบัติหลักของแนวคิดการพัฒนาโปรแกรมเชิงวัตถุมีดังนี้

- นำกลับมาใช้ใหม่ (Reuse) บนสภาพแวดล้อมของระบบเดิม หรือระบบอื่น ๆ ได้

- ประหยัดเวลาในการพัฒนา (Rapid Delivery)
- ใช้งานง่าย (User Friendly) ผ่านส่วนติดต่อกับผู้ใช้ซึ่งเป็นที่ยอมรับของผู้ใช้
- ดูแลรักษาได้ง่าย (More Maintainable)
- มีคุณภาพสูง (Greater Quality System) ในเชิงความถูกต้องและความรวดเร็วในการประมวลผล

2.3.4 รูปแบบการเขียนโปรแกรมภาษาจาวา

ภาษาจาวามีรูปแบบการเขียนโปรแกรมที่ประกอบด้วยคลาสอย่างน้อยหนึ่งคลาสที่มีเมธอด main() ซึ่งเปรียบเสมือนจุดเริ่มต้นของโปรแกรม โดยมีรูปแบบการเขียนโปรแกรมดังนี้

```
public class JavaApplication11 {
    public static void main(String[] args) {
        System.out.println("Hello");
    }
}
```

รูปที่ 2.1 รูปแบบการเขียนโปรแกรมภาษาจาวา

จากรูปที่ 2.1 สามารถอธิบายได้ดังนี้

- เป็นการสร้างหรือนิยามคลาสขึ้นในชื่อ JavaApplication11 ชื่อไฟล์จึงต้องเป็นชื่อ JavaApplication11.java
- คลาส JavaApplication11 มีเมธอด main() เมื่อเรียกทำงานโปรแกรมนี้ JRE จะค้นหาเมธอดที่ชื่อ main() เพื่อเริ่มประมวลผลตามชุดคำสั่งที่เขียนไว้
- เมธอด main() จะต้องมีคีย์เวิร์ด public, static, void รวมไปถึงพารามิเตอร์ที่มีอาร์เรย์ของข้อมูลประเภทข้อความ (String)
- ชุดคำสั่งที่ต้องการให้เกิดการประมวลผลจะเขียนไว้ภายในเครื่องหมายวงเล็บ { } ของเมธอด main()
- ทุกคำสั่งต้องจบด้วยเครื่องหมาย ; เสมอ

2.3.5 Class และ Object

หากจะอธิบาย OOP แบบง่ายๆ ให้คิดถึงการสร้างบ้านที่ต้องเขียนพิมพ์เขียวก่อนสร้างเสมอ ซึ่งแบบพิมพ์เขียว 1 แบบสามารถสร้างบ้านเท่าไหร่ก็ได้ไม่มีจำกัด

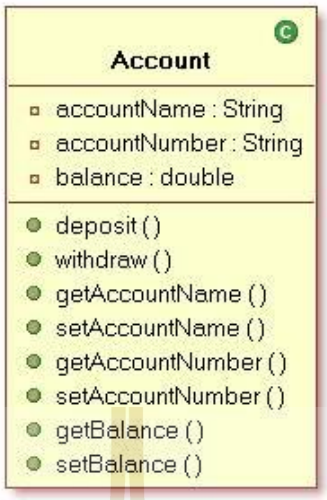
จากแนวคิดนี้เมื่อเปรียบเทียบกับแนวคิด OOP จะได้ว่า แบบพิมพ์เขียว คือ คลาส (Class) บ้านแต่ละหลังที่สร้างจากพิมพ์เขียว คือ ออบเจ็กต์ (Object) นั่นหมายถึง เราออกแบบสร้างคลาสเพียงครั้งเดียว สามารถนำคลาสไปสร้างออบเจ็กต์เพื่อใช้งานได้ไม่จำกัด

และหากเปรียบบ้าน 1 หลังเป็นโปรแกรม 1 ตัว ก็จะได้ว่าบ้านหนึ่งหลังประกอบด้วยประตู หน้าต่าง ห้องนอน ฯลฯ เช่นเดียวกับโปรแกรมหนึ่งตัวที่ประกอบด้วยออบเจ็กต์หลายๆตัว โดยออบเจ็กต์แต่ละตัวจะมีหน้าที่แตกต่างกันออกไป (ออบเจ็กต์แต่ละตัวก็เปรียบได้กับประตู หน้าต่าง ห้องนอน ฯลฯ)

ดังนั้น คลาส (Class) ก็คือ สิ่งที่ใช้อธิบายลักษณะและความสามารถของออบเจ็กต์ เปรียบได้กับแม่แบบของออบเจ็กต์ โดยสามารถสร้างออบเจ็กต์ให้มีคุณสมบัติเหมือนแม่แบบได้กี่ครั้งก็ได้

ออบเจ็กต์ (Object) ก็คือ สิ่งต่าง ๆ ที่ปรากฏอยู่รอบตัว ซึ่งมีคุณลักษณะและความสามารถในการทำงาน เช่น คน, รถยนต์ เป็นต้น

โดยสิ่งที่ใช้อธิบายลักษณะของออบเจ็กต์จะถูกเรียกว่า แอตทริบิวต์ (Attribute) และสิ่งที่ใช้อธิบายการทำงานของออบเจ็กต์จะถูกเรียกว่า เมธอด (Method) (สุดา เธียรมนตรี, 2556)



รูปที่ 2.2 ตัวอย่างแสดงส่วนประกอบของคลาส

```
public class Account {
   private String accountName;
   private String accountNumber;
   private double balance;

   public void deposit (double amount) {
      balance = balance + amount;
   }

   public void withdraw (double amount) {
      balance = balance - amount;
   }

   public String getAccountName () {
      return accountName;
   }

   public void setAccountName (String name) {
      accountName = name;
   }

   public String getAccountNumber () {
      return accountNumber;
   }

   public void setAccountNumber (String number) {
      accountNumber = number;
   }

   public double getBalance () {
      return balance;
   }

   public void setBalance (double amount) {
      balance = amount;
   }
}
```

รูปที่ 2.3 ตัวอย่างแสดงซอร์สโค้ดของคลาสในภาษาจาวา

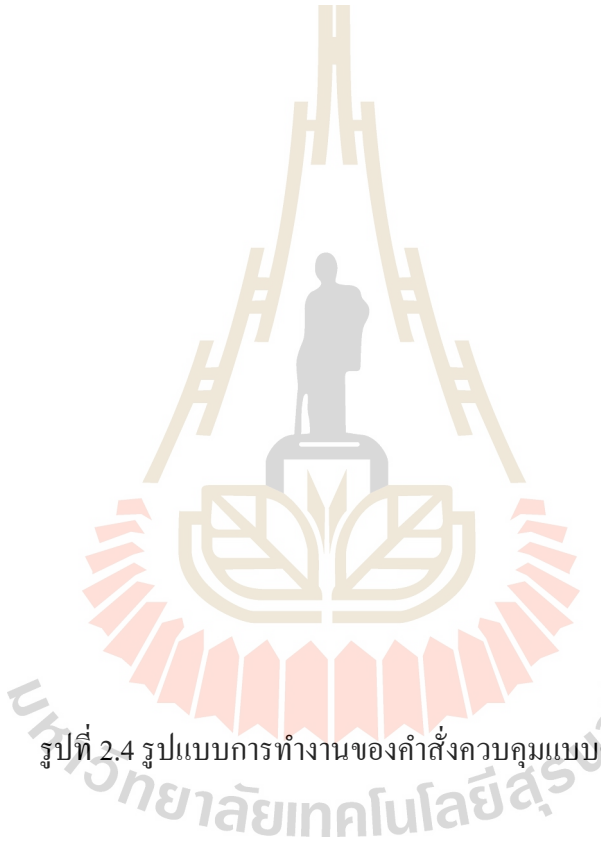
จากรูปที่ 2.2 มีคลาสที่ชื่อว่า Account มีแอตทริบิวต์ชื่อ accoundName, accoundNumber และ balance และมีเมธอดชื่อ deposit, withdraw, getAccoundName, setAccoundName, getAccoundNumber, setAccoundNumber, getBalance และ setBalance ซึ่งสามารถอธิบายในรูปแบบภาษาจาวาได้ดังรูปที่ 2.3

2.3.6 คำสั่งควบคุม

คำสั่งควบคุมทิศทางการทำงานของโปรแกรม (Control Statement) (Jeff, 2011; สุดา เธียรมนตรี, 2556) เราสามารถจำแนกได้เป็น 3 รูปแบบ คือ

2.3.6.1 คำสั่งควบคุมทิศทางการทำงานของโปรแกรมแบบตามลำดับ (Sequence Control Statement)

คำสั่งควบคุมทิศทางการทำงานของโปรแกรมแบบตามลำดับ (Sequence Control Statement) (Jeff, 2011; สุดา เธียรมนตรี, 2556) มีรูปแบบการทำงานที่ทุกคำสั่งจะต้องทำงานตามลำดับ โดยไม่ข้ามคำสั่ง แต่ละคำสั่งถูกเรียกใช้เพียงครั้งเดียว โดยมีรูปแบบการทำงานดังรูปที่ 2.4



รูปที่ 2.4 รูปแบบการทำงานของคำสั่งควบคุมแบบตามลำดับ

```
1
2   package comeng.sut;
3   import java.util.Scanner;
4   public class Test {
5       public static void main(String[] args) {
6           Scanner input = new Scanner(System.in);
7           System.out.print("Enter a number : ");
8           int num = input.nextInt();
9           System.out.print("Enter a number : ");
10          int num2 = input.nextInt();
11          int sum = num+num2;
12          System.out.println("Sum = " + sum);
13
14
15      }
16  }
```

รูปที่ 2.5 ตัวอย่างการใช้คำสั่งควบคุมแบบตามลำดับ

จากรูปที่ 2.5 เป็นตัวอย่างการใช้คำสั่งควบคุมแบบตามลำดับเพื่อใช้คำนวณค่าของการบวกเลข 2 จำนวน โดยมีการทำงานดังนี้

บรรทัดที่ 6 ประกาศตัวแปร input จากคลาส Scanner

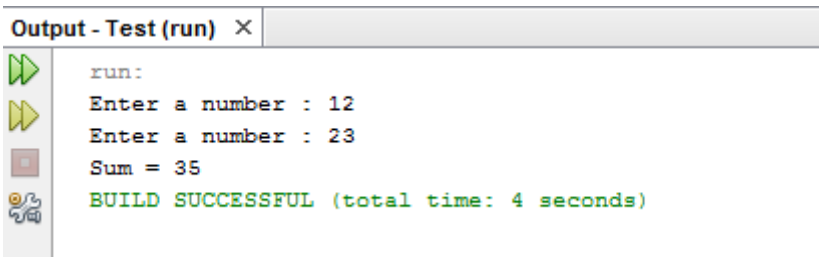
บรรทัดที่ 8 รับค่าตัวเลขตัวที่ 1 เป็นชนิด Integer โดยใช้เมธอด nextInt() เก็บค่าไว้ที่ตัวแปร num

บรรทัดที่ 10 รับค่าตัวเลขตัวที่ 2 เป็นชนิด Integer โดยใช้เมธอด nextInt() เก็บค่าไว้ที่ตัวแปร num2

บรรทัดที่ 11 คำนวณผลบวกของตัวเลข num และ num2 แล้วเก็บค่าไว้ที่ตัวแปร sum ที่เป็นชนิด Integer

บรรทัดที่ 12 แสดงผลของการบวกเลขด้วยเมธอด println()

จากการทำงาน จะเห็นได้ว่า โปรแกรมทำงานแบบตามลำดับโดยไม่มีการข้ามคำสั่งใดเลย เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังรูปที่ 2.6



รูปที่ 2.6 ตัวอย่างผลลัพธ์

2.3.6.2 คำสั่งควบคุมแบบมีทางเลือก (Selection Control Statement)

คำสั่งควบคุมแบบมีทางเลือก (Selection Control Statement) (Jeff, 2011; สุดา เธียรมนตรี, 2556) มีรูปแบบการทำงานที่มีการตรวจสอบเงื่อนไขของคำสั่งการทำงาน เพื่อใช้ในการเลือกทิศทางของการทำงาน มี 4 คำสั่ง คือ if , if…else , if…else if , switch

คำสั่ง if

คำสั่ง if (สุดา เธียรมนตรี, 2556) เป็นคำสั่งที่ใช้ในการควบคุมให้โปรแกรมตัดสินใจว่าจะทำงานหรือไม่ทำงานในคำสั่งนั้น ๆ โดยตรวจสอบจากเงื่อนไขที่กำหนดว่าเป็นจริงหรือเท็จ ถ้าเป็นจริงจะทำคำสั่งนั้น ๆ ถ้าเป็นเท็จจะไม่ทำคำสั่งนั้น ๆ โดยมีรูปแบบการทำงานดังรูปที่ 2.7

รูปที่ 2.7 รูปแบบการทำงานของคำสั่ง if

กำหนดให้

Boolean Expression	เป็นเงื่อนไขทางตรรกศาสตร์ มีผลการตรวจสอบเป็นจริง (true) หรือ เท็จ (false)
Statement	เป็นชุดคำสั่งที่จะทำงานเมื่อเงื่อนไขมีผลการตรวจสอบเป็นจริง

```
1
2   package comeng.sut;
3   import java.util.Scanner;
4   public class Test {
5       public static void main(String[] args) {
6           Scanner input = new Scanner(System.in);
7           System.out.print("Enter a number : ");
8           int num = input.nextInt();
9           if(num%2==0){
10              System.out.println(num + " is Even Number.");
11          }
12          System.out.println("End");
13      }
14  }
15
```

รูปที่ 2.8 ตัวอย่างการใช้คำสั่ง if

จากรูปที่ 2.78 เป็นตัวอย่างการใช้คำสั่ง if เพื่อตรวจสอบว่าเลขที่รับเข้ามาเป็นเลขคู่หรือไม่ โดยมีการทำงานดังนี้

บรรทัดที่ 6 ประกาศตัวแปร input จากคลาส Scanner

บรรทัดที่ 8 รับค่าตัวเลขเป็นชนิด Integer โดยใช้เมธอด nextInt() เก็บค่าไว้ที่ตัวแปร num

บรรทัดที่ 9 ตรวจสอบค่าที่รับเข้ามาว่าเป็นเลขคู่หรือไม่ โดยการนำตรวจสอบว่าผลของการนำเลขนั้นมาหารแบบเอาเศษจากการหาร กับ 2 นั้น มีค่าเท่ากับ 0 หรือไม่

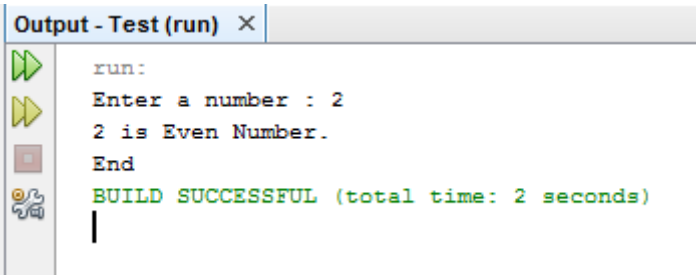
ถ้าเป็นจริง จะทำงานที่บรรทัดที่ 10

ถ้าเป็นเท็จ จะข้ามการทำงานในบรรทัดที่ 10 แล้วไปทำงานที่บรรทัดที่ 12

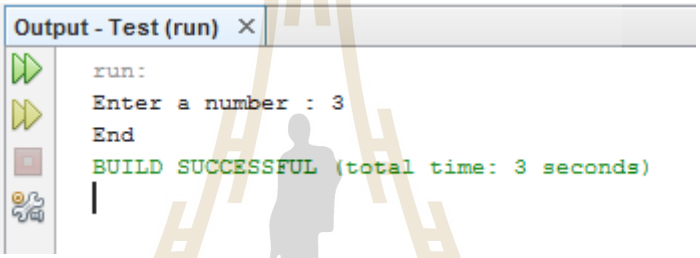
บรรทัดที่ 10 แสดงผว่าตัวเลขนั้นเป็นเลขคู่ด้วยเมธอด println()

บรรทัดที่ 12 แสดงข้อความว่า End เพื่อจบการทำงานด้วยเมธอด println()

จากการทำงาน จะเห็นได้ว่า ชุดคำสั่งในบรรทัดที่ 10 จะทำงานก็ต่อเมื่อตรวจสอบเงื่อนไขที่กำหนดว่าเป็นจริงเท่านั้น เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังรูปที่ 2.9 และ รูปที่ 2.10



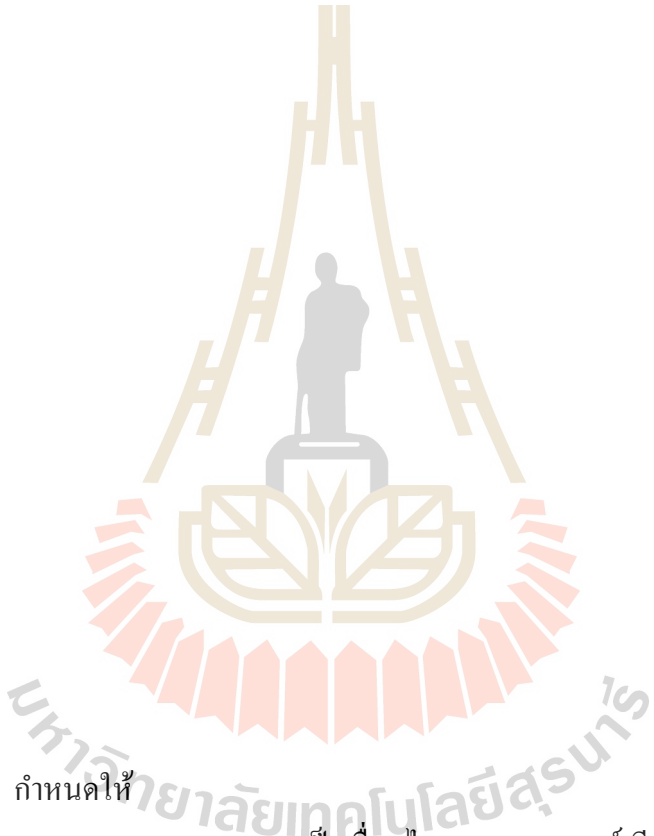
รูปที่ 2.9 ตัวอย่างผลลัพธ์เมื่อเงื่อนไขเป็นจริง



รูปที่ 2.10 ตัวอย่างผลลัพธ์เมื่อเงื่อนไขเป็นเท็จ

คำสั่ง if…else

คำสั่ง if…else (สุดา เธียรมนตรี, 2556) เป็นคำสั่งที่ใช้ควบคุมให้โปรแกรมเลือกทำงานในชุดคำสั่งใดคำสั่งหนึ่งจาก 2 ชุดคำสั่ง โดยตรวจสอบจากเงื่อนไขที่กำหนดว่าเป็นจริงหรือเท็จ ถ้าเป็นจริงจะทำคำสั่งภายใต้คำสั่ง if ถ้าเป็นเท็จจะทำคำสั่งภายใต้คำสั่ง else โดยมีรูปแบบการทำงานดังรูปที่ 2.11



กำหนดให้

Boolean Expression	เป็นเงื่อนไขทางตรรกศาสตร์ มีผลการตรวจสอบเป็นจริง (true) หรือ เท็จ (false)
Statement 1	เป็นชุดคำสั่งที่จะทำงานเมื่อเงื่อนไขมีผลการตรวจสอบเป็นจริง
Statement 2	เป็นชุดคำสั่งที่จะทำงานเมื่อเงื่อนไขมีผลการตรวจสอบเป็นเท็จ

```
1
2   package comeng.sut;
3   import java.util.Scanner;
4   public class Test {
5       public static void main(String[] args) {
6           Scanner input = new Scanner(System.in);
7           System.out.print("Enter a number : ");
8           int num = input.nextInt();
9           if(num%2==0){
10              System.out.println(num + " is Even Number.");
11          }else{
12              System.out.println(num + " is Odd Number.");
13          }
14          System.out.println("End");
15      }
16  }
```

รูปที่ 2.12 ตัวอย่างการใช้คำสั่ง if...else

จากรูปที่ 2.12 เป็นตัวอย่างการใช้คำสั่ง if…else ซึ่งเป็นโปรแกรมที่เพิ่มเติมจากโปรแกรมในรูปที่ 2.8 โดยเพิ่มชุดคำสั่งสำหรับการทำงานเมื่อตรวจสอบตัวเลขแล้วว่าไม่ใช่เลขคู่ (เงื่อนไขเป็นเท็จ) โดยการแทรก else และชุดคำสั่งลงไปในบรรทัดที่ 11-13 และเมื่อรันโปรแกรมและรับตัวเลขที่เป็นเลขคี่ที่ทำให้เงื่อนไขเป็นเท็จจะได้ผลลัพธ์ดังรูปที่ 2.13

```
Output - Test (run)
run:
Enter a number : 3
3 is Odd Number.
End
BUILD SUCCESSFUL (total time: 3 seconds)
```

รูปที่ 2.13 ตัวอย่างผลลัพธ์การใช้คำสั่ง if...else เมื่อเงื่อนไขเป็นเท็จ

คำสั่ง if…else if

คำสั่ง if…else if (สุดา เธียรมนตรี, 2556)เป็นคำสั่งที่ใช้ควบคุมให้โปรแกรมเลือกทำงานชุดคำสั่งจากหลายชุดคำสั่ง และในแต่ละชุดคำสั่งจะมีเงื่อนไขของแต่ละชุดคำสั่ง โดยถ้า

ตรวจสอบเงื่อนไขว่าเป็นจริงในชุดคำสั่งใดจะทำชุดคำสั่งนั้น ๆ โดยไม่สนใจในชุดคำสั่งอื่น ๆ และถ้าตรวจสอบแล้วเป็นเท็จในทุกเงื่อนไข โปรแกรมจะทำงานชุดคำสั่งภายใต้ชุดคำสั่ง else โดยมีรูปแบบการทำงานดังรูปที่ 2.14

กำหนดให้

Boolean Expression 1	เป็นเงื่อนไขทางตรรกศาสตร์ที่ 1 มีผลการตรวจสอบเป็นจริง (true) หรือ เท็จ (false)
Boolean Expression 2	เป็นเงื่อนไขทางตรรกศาสตร์ที่ 1 มีผลการตรวจสอบเป็นจริง (true) หรือ เท็จ (false)
Statement 1	เป็นชุดคำสั่งที่จะทำงานเมื่อเงื่อนไขที่ 1 มีผลการตรวจสอบเป็นจริง
Statement 2	เป็นชุดคำสั่งที่จะทำงานเมื่อเงื่อนไขที่ 2 มีผลการตรวจสอบเป็นจริง
Statement 3	เป็นชุดคำสั่งที่จะทำงานเมื่อเงื่อนไขที่ 1 และ 2 มีผลการตรวจสอบเป็นเท็จ

```
3  import java.util.Scanner;
4  public class Test {
5      public static void main(String[] args) {
6          Scanner input = new Scanner(System.in);
7          System.out.print("Enter a first number : ");
8          int num = input.nextInt();
9          System.out.print("Enter a second number : ");
10         int num2 = input.nextInt();
11         if(num>num2){
12             System.out.println(num + " is more than"+ num2);
13         }else if(num<num2){
14             System.out.println(num + " is less than " + num2);
15         }else{
16             System.out.println(num + " is equal than " + num2);
17         }
18         System.out.println("End");
19     }
20 }
```

รูปที่ 2.15 ตัวอย่างการใช้คำสั่ง if...else if

จากรูปที่ 2.15 เป็นตัวอย่างการใช้คำสั่ง if…else if เพื่อเปรียบเทียบค่าของเลขที่รับมา 2 จำนวน โดยมีการทำงานดังนี้

บรรทัดที่ 6	ประกาศตัวแปร input จากคลาส Scanner
บรรทัดที่ 8	รับค่าตัวเลขตัวที่ 1 เป็นชนิด Integer โดยใช้เมธอด nextInt() เก็บค่าไว้ที่ตัวแปร num

บรรทัดที่ 10 รับค่าตัวเลขตัวที่ 2 เป็นชนิด Integer โดยใช้เมธอด nextInt() เก็บค่าไว้ที่ตัวแปร num2

บรรทัดที่ 11 ตรวจสอบเงื่อนไขว่าเลขที่เก็บไว้ใน num มากกว่าเลขที่เก็บไว้ใน num2 หรือไม่
ถ้าเป็นจริง ให้ทำคำสั่งในบรรทัดที่ 12
ถ้าเป็นเท็จ ให้ข้ามบรรทัดที่ 12 ไปตรวจสอบเงื่อนไขในบรรทัดที่ 13

บรรทัดที่ 12 แสดงผลลัพธ์ว่า num มากกว่า num2 ด้วยเมธอด println()และข้ามไปทำบรรทัดที่ 18

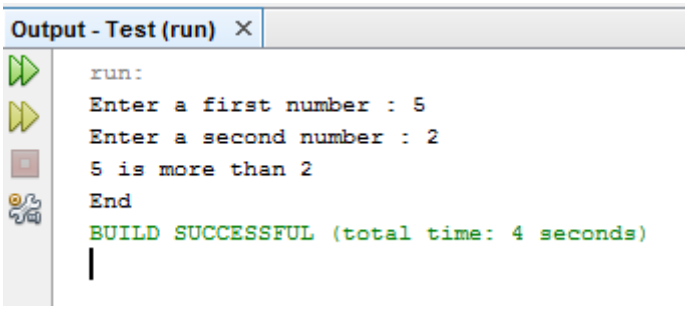
บรรทัดที่ 13 ตรวจสอบเงื่อนไขว่าเลขที่เก็บไว้ใน num น้อยกว่าเลขที่เก็บไว้ใน num2 หรือไม่
ถ้าเป็นจริง ให้ทำคำสั่งในบรรทัดที่ 14
ถ้าเป็นเท็จ ให้ข้ามบรรทัดที่ 14 ไปทำชุดคำสั่งในบรรรทัดที่ 16

บรรทัดที่ 14 แสดงผลลัพธ์ว่า num น้อยกว่า num2 ด้วยเมธอด println() และข้ามไปทำบรรทัดที่ 18

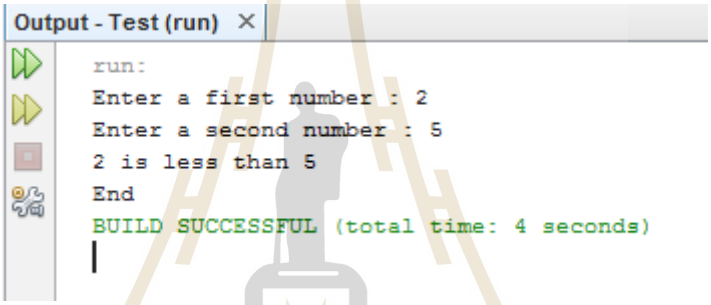
บรรทัดที่ 16 แสดงผลลัพธ์ว่า num เท่ากับ num2 ด้วยเมธอด println()

บรรทัดที่ 18 แสดงข้อความว่า End เพื่อจบการทำงาน ด้วยเมธอด println()

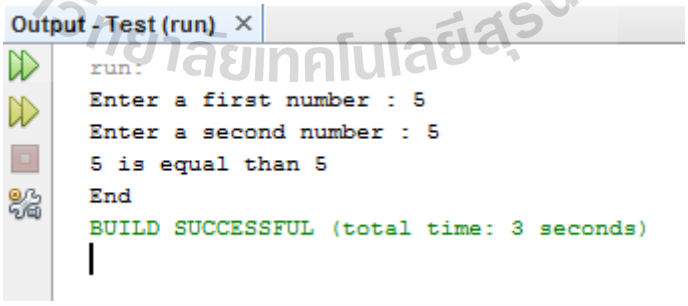
จากการทำงาน จะเห็นได้ว่า ชุดคำสั่งในบรรทัดที่ 12 จะทำงานก็ต่อเมื่อตรวจสอบเงื่อนไขที่กำหนดไว้ในบรรทัดที่ 11 ว่าเป็นจริงเท่านั้น และชุดคำสั่งในบรรทัดที่ 14 จะทำงานก็ต่อเมื่อตรวจสอบเงื่อนไขที่กำหนดไว้ในบรรทัดที่ 13 ว่าเป็นจริงเท่านั้น และถ้าเงื่อนไขทั้งหมดเป็นเท็จจึงจะสามารถทำชุดคำสั่งในบรรทัดที่ 16 ได้ และเมื่อทำชุดคำสั่งในแต่ละจุดเสร็จแล้วจะข้ามไปทำชุดคำสั่งในบรรทัดที่ 18 ทันทีโดยไม่สนใจชุดคำสั่งอื่น ๆ ที่อยู่นอกเหนือจากขอบเขตเงื่อนไข และเมื่อรันโปรแกรมจะได้ผลลัพธ์ดังรูปที่ 2.16 ถึง รูปที่ 2.18



รูปที่ 2.16 ตัวอย่างผลลัพธ์การใช้คำสั่ง if...else if เมื่อเงื่อนไขที่ 1 เป็นจริง



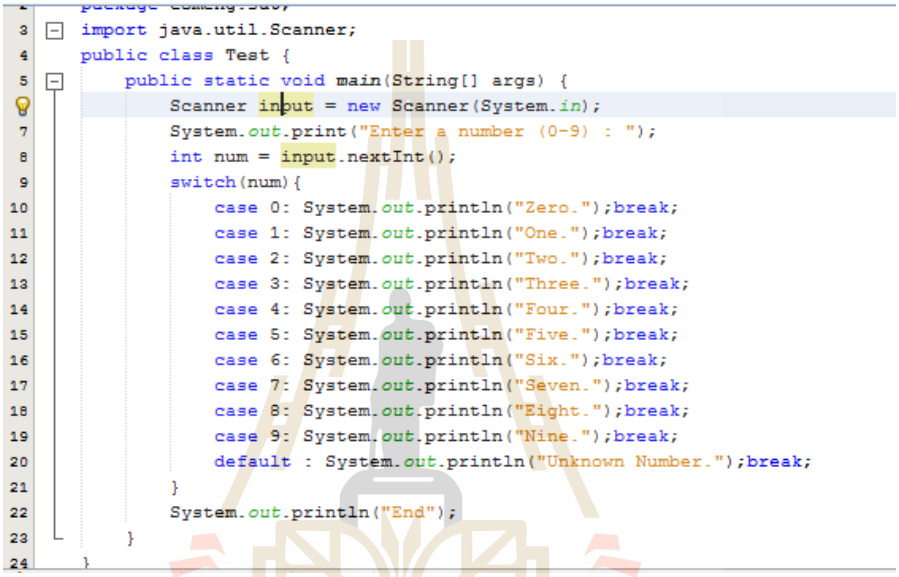
รูปที่ 2.17 ตัวอย่างผลลัพธ์การใช้คำสั่ง if...else if เมื่อเงื่อนไขที่ 2 เป็นจริง



รูปที่ 2.18 ตัวอย่างผลลัพธ์การใช้คำสั่ง if...else if เมื่อทุกเงื่อนไขเป็นเท็จ

คำสั่ง switch

คำสั่ง switch (Jeff, 2011; สุดา เธียรมนตรี, 2556) เป็นคำสั่งที่มีการทำงานและหน้าที่เหมือนกันกับ if...else if แตกต่างกันที่วิธีเขียน มีขอบเขตของเงื่อนไขที่ใช้ในการตรวจสอบว่า ค่าของสิ่งที่นำมาตรวจสอบต้องเป็นเลขจำนวนเต็มหรือตัวอักขระเท่านั้น ไม่สามารถใช้เลขทศนิยมหรือข้อความได้



```
3  import java.util.Scanner;
4  public class Test {
5      public static void main(String[] args) {
6          Scanner input = new Scanner(System.in);
7          System.out.print("Enter a number (0-9) : ");
8          int num = input.nextInt();
9          switch(num){
10             case 0: System.out.println("Zero.");break;
11             case 1: System.out.println("One.");break;
12             case 2: System.out.println("Two.");break;
13             case 3: System.out.println("Three.");break;
14             case 4: System.out.println("Four.");break;
15             case 5: System.out.println("Five.");break;
16             case 6: System.out.println("Six.");break;
17             case 7: System.out.println("Seven.");break;
18             case 8: System.out.println("Eight.");break;
19             case 9: System.out.println("Nine.");break;
20             default : System.out.println("Unknown Number.");break;
21         }
22         System.out.println("End");
23     }
24 }
```

รูปที่ 2.19 ตัวอย่างการใช้คำสั่ง switch

จากรูปที่ 2.19 เป็นตัวอย่างการใช้คำสั่ง switch เพื่อแสดงค่าภาษาอังกฤษของเลขที่รับมา โดยมีการทำงานดังนี้

บรรทัดที่ 6 ประกาศตัวแปร input จากคลาส Scanner

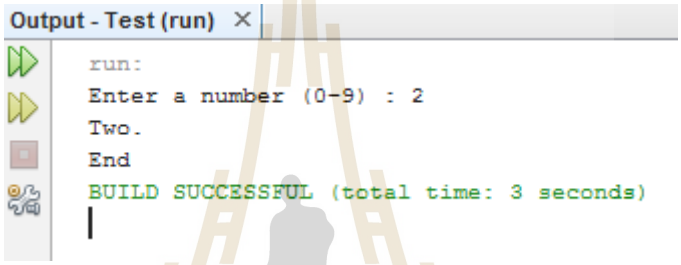
บรรทัดที่ 8 รับค่าตัวเลขเป็นชนิด Integer โดยใช้เมธอด nextInt() เก็บค่าไว้ที่ตัวแปร num

บรรทัดที่ 9-21 ใช้คำสั่ง switch เพื่อตรวจสอบว่า เลขที่รับเข้ามานั้น ตรงกับเงื่อนไขใน case ใด เพื่อทำงานตามชุดคำสั่งภายใต้ case นั้น ๆ เมื่อทำงานเสร็จจะข้ามชุดคำสั่งของ case อื่น ๆ เพื่อไปทำชุดคำสั่งในบรรทัดที่ 22 หากไม่ตรงกับ case ใดเลย โปรแกรมจะ

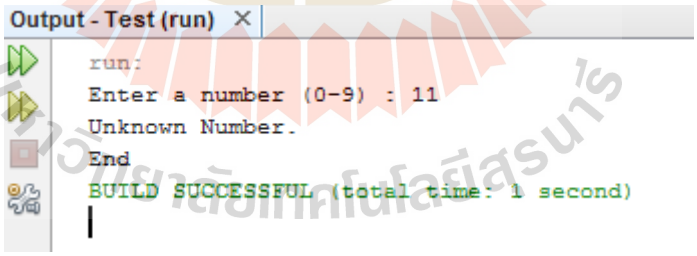
ทำชุดคำสั่งที่อยู่ภายใต้ default แล้วจึงข้ามไปทำชุดคำสั่งในบรรทัดที่ 22 ต่อไป

บรรทัดที่ 22 แสดงข้อความว่า End เพื่อจบการทำงาน ด้วยเมธอด println()

จากการทำงาน จะเห็นได้ว่า ในแต่ละชุดคำสั่งจะมีเงื่อนไขในการทำงานแต่ต่างกันออกไป ถ้าตรวจสอบแล้วพบว่าค่าที่รับมานั้นตรงกับ case ใด ก็จะทำชุดคำสั่งของ case นั้นและข้ามการทำงานของ case อื่น ๆ ไปทำในบรรทัดที่ 22 ทันที เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังรูปที่ 2.20 และ รูปที่ 2.21



รูปที่ 2.20 ตัวอย่างผลลัพธ์การใช้คำสั่ง switch เมื่อค่าที่รับตรงกับเงื่อนไขของ case



รูปที่ 2.21 ตัวอย่างผลลัพธ์การใช้คำสั่ง switch เมื่อค่าที่รับไม่ตรงกับเงื่อนไขของ case

2.3.6.3 คำสั่งควบคุมแบบทำซ้ำ (Iteration Control Statement)

คำสั่งควบคุมแบบทำซ้ำ (Iteration Control Statement) (Jeff, 2011; เธียรมนตรี สุดา , 2556) มีรูปแบบการทำงานที่มีการวนลูปทำชุดคำสั่งเดิม ๆ ซ้ำ ๆ จนกว่าเงื่อนไขจะเป็นเท็จจึงจะหยุดทำงานแล้วออกจากลูป มี 3 ชุดคำสั่ง คือ คำสั่ง while , do…while และ for

คำสั่ง while

คำสั่ง while (Jeff, 2011; สุดา เธียรมนตรี, 2556)เป็นคำสั่งควบคุมการทำงานแบบทำซ้ำ โดยมีการตรวจสอบเงื่อนไขว่าเป็นจริงก่อนทำชุดคำสั่งที่อยู่ภายใต้ลูป while ทุกครั้ง เมื่อทำงานเสร็จก็จะกลับไปตรวจสอบเงื่อนไขอีกครั้ง วนไปเรื่อย ๆ จนกว่าเงื่อนไขจะเป็นเท็จจึงจะหยุดทำงานแล้วออกจากลูป มีรูปแบบการทำงานดังรูปที่ 2.22

กำหนดให้

Boolean Expression	เป็นเงื่อนไขทางตรรกศาสตร์ มีผลการตรวจสอบเป็นจริง (true) หรือ เท็จ (false) ที่ตรวจสอบก่อนการทำงานเสมอ
Statements	เป็นชุดคำสั่งที่ทำงานเมื่อเงื่อนไขเป็นจริง

```
1
2   package comeng.sut;
3   import java.util.Scanner;
4   public class Test {
5       public static void main(String[] args) {
6           Scanner input = new Scanner(System.in);
7           System.out.print("Enter a number (0-9): ");
8           int num = input.nextInt();
9           while(num<0||num>9){
10              System.out.println("Error!");
11              System.out.print("Enter a number (0-9): ");
12              num = input.nextInt();
13          }
14          System.out.println("Success");
15          System.out.println("End");
16      }
17  }
```

รูปที่ 2.23 ตัวอย่างการใช้คำสั่ง while

จากรูปที่ 2.23 เป็นตัวอย่างการใช้คำสั่ง while เพื่อตรวจสอบว่าค่าที่รับมาอยู่ในช่วง 0-9 หรือไม่ โดยมีการทำงานดังนี้

บรรทัดที่ 6 ประกาศตัวแปร input จากคลาส Scanner

บรรทัดที่ 8 รับค่าตัวเลขเป็นชนิด Integer โดยใช้เมธอด nextInt() เก็บค่าไว้ที่ตัวแปร num

บรรทัดที่ 9-13 ตรวจสอบเงื่อนไขว่า ค่าที่รับมาเก็บในตัวแปร num นั้นมีค่า น้อยกว่า 0 หรือ มากกว่า 9 หรือไม่

ถ้าเป็นจริง จะทำชุดคำสั่งภายใต้ลูป while ในบรรทัดที่ 10-12

ถ้าเป็นเท็จ จะข้ามชุดคำสั่งในบรรทัดที่ 10-12 และเริ่มทำงานในบรรทัดที่ 14

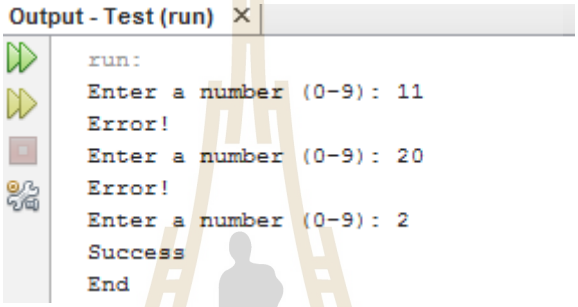
บรรทัดที่ 10 แสดงข้อความว่า Error! ด้วยเมธอด println()

บรรทัดที่ 11-12 รับค่าตัวเลขเป็นชนิด Integer โดยใช้เมธอด nextInt() เก็บค่าไว้ที่ตัวแปร num แล้ววนลูปไปตรวจสอบเงื่อนไขที่บรรทัดที่ 9

บรรทัดที่ 14 แสดงข้อความว่า Success ด้วยเมธอด println()

บรรทัดที่ 15 แสดงข้อความว่า End เพื่อจบการทำงาน ด้วยเมธอด println()

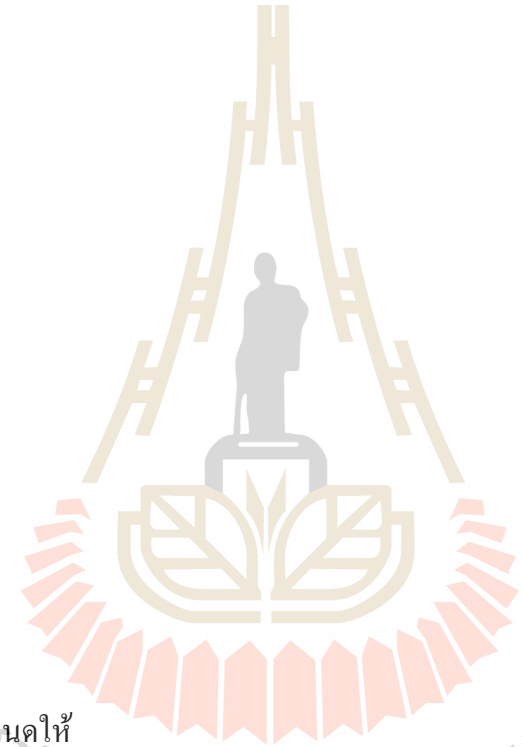
จากการทำงาน จะเห็นได้ว่า เมื่อรับค่ามาและนำไปตรวจสอบเงื่อนไขของลูป while ถ้าเป็นจริงจะเข้าไปทำชุดคำสั่งที่อยู่ภายใต้ลูป while เพื่อรับค่าใหม่แล้วกลับไปตรวจสอบเงื่อนไขอีกครั้ง และวนไปเรื่อย ๆ จนกว่าเงื่อนไขจะเป็นเท็จจึงจะหยุดทำงานแล้วออกจากลูป เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังรูปที่ 2.24



รูปที่ 2.24 ตัวอย่างผลลัพธ์การใช้คำสั่ง while

คำสั่ง do…while

คำสั่ง do…while (Jeff, 2011)เป็นคำสั่งควบคุมการทำงานแบบทำซ้ำโดยโปรแกรมจะทำงานอย่างน้อย 1 รอบ ก่อนตรวจสอบเงื่อนไขว่าเป็นจริงหรือเท็จ ถ้าเป็นจริงจะวนลูปเพื่อนทำชุดคำสั่งภายใต้คำสั่ง do และตรวจสอบเงื่อนไขทุกครั้ง และวนทำชุดคำสั่งไปเรื่อย ๆ จนกว่าเงื่อนไขจะเป็นเท็จจึงจะหลุดลูป มีรูปแบบการทำงานดังรูปที่ 2.25



กำหนดให้

Boolean Expression	เป็นเงื่อนไขทางตรรกศาสตร์ มีผลการตรวจสอบเป็นจริง (true) หรือ เท็จ (false) ที่ตรวจสอบก่อนการทำงานเสมอ
Statements	เป็นชุดคำสั่งที่ทำงานอย่างน้อย 1 ครั้งก่อนตรวจสอบเงื่อนไขและเมื่อเงื่อนไขเป็นจริง

```
1
2   package comeng.sut;
3   import java.util.Scanner;
4   public class Test {
5       public static void main(String[] args) {
6           Scanner input = new Scanner(System.in);
7           int num;
8           do{
9               System.out.print("Enter a number (0-9): ");
10              num = input.nextInt();
11          }while(num<0||num>9);
12          System.out.println("Success");
13          System.out.println("End");
14      }
15  }
```

รูปที่ 2.26 ตัวอย่างการใช้คำสั่ง do…while

จากรูปที่ 2.26 เป็นตัวอย่างการใช้คำสั่ง do…while เพื่อตรวจสอบว่าค่าที่รับมาอยู่ในช่วง 0-9 หรือไม่ โดยมีการทำงานดังนี้

บรรทัดที่ 6 ประกาศตัวแปร input จากคลาส Scanner

บรรทัดที่ 7 ประกาศตัวแปร num เป็นชนิดตัวเลข

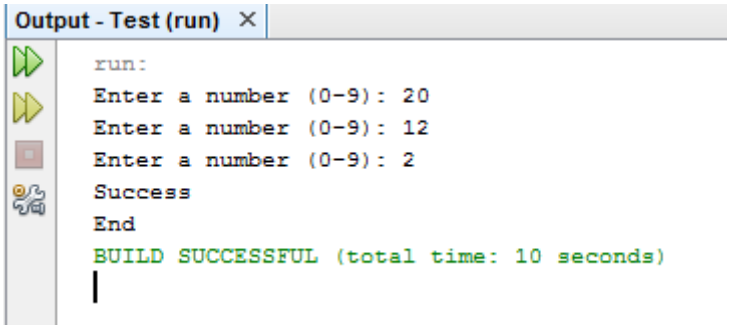
บรรทัดที่ 8-11 ทำชุดคำสั่งในบรรทัดที่ 9-10 ก่อน 1 รอบ แล้วทำการตรวจสอบเงื่อนไขตามบรรทัดที่ 11 ว่าค่าที่รับมาไม่อยู่ในช่วง 0-9 หรือไม่ ถ้าเป็นจริง วนลูปเพื่อทำชุดคำสั่งในบรรทัดที่ 9-10 ถ้าเป็นเท็จ โปรแกรมจะหลุดลูปไปที่บรรทัดที่ 12

บรรทัดที่ 9-10 รับค่าตัวเลขเป็นชนิด Integer โดยใช้เมธอด nextInt() เก็บค่าไว้ที่ตัวแปร num

บรรทัดที่ 12 แสดงข้อความว่า Success ด้วยเมธอด println()

บรรทัดที่ 13 แสดงข้อความว่า End เพื่อจบการทำงาน ด้วยเมธอด println()

จากการทำงาน จะเห็นได้ว่า เมื่อเข้าในลูป do…while โปรแกรมจะทำการรับค่าก่อน 1 รอบ แล้วตรวจสอบเงื่อนไขว่า ค่าที่รับมานั้นไม่ได้อยู่ในช่วง 0-9 ถ้าจริงจะวนลูปเพื่อรับค่าใหม่ ถ้าเท็จโปรแกรมจะหลุดลูป เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังรูปที่ 2.27



รูปที่ 2.27 ตัวอย่างผลลัพธ์การใช้คำสั่ง do...while

คำสั่ง for

คำสั่ง for (Jeff, 2011)เป็นคำสั่งควบคุมการทำงานแบบทำซ้ำ โดยมีการตรวจสอบเงื่อนไขก่อนการทำงานทุกครั้ง เมื่อตรวจสอบเงื่อนไขว่าเป็นจริงจะมีการทำงานชุดคำสั่งภายในลูป และจำมีการเพิ่มหรือลดค่าของตัวแปรควบคุมตามคำสั่งที่กำหนดไว้ทุกครั้งที่ทำงานชุดคำสั่งเสร็จ และเมื่อตรวจสอบเงื่อนไขว่าเป็นเท็จจึงจะหลุดจากลูป มีรูปแบบการทำงานดังรูปที่ 2.28

มหาวิทยาลัยเทคโนโลยีสุรนารี

กำหนดให้

Control_Variable	ตัวแปรควบคุมจำนวนรอบ เป็นชนิดตัวเลข
Value	ค่าเริ่มต้นที่กำหนดให้ตัวแปรควบคุม
Boolean Expression	เป็นเงื่อนไขทางตรรกศาสตร์ มีผลการตรวจสอบเป็นจริง (true) หรือ เท็จ (false) ที่ตรวจสอบก่อนการทำงานเสมอ
Statements	เป็นชุดคำสั่งที่ทำงานเมื่อเงื่อนไขเป็นจริง
Increment or decrement	คำสั่งสำหรับเพิ่มค่าหรือลดค่าของตัวแปรควบคุม

```
1
2   package comeng.sut;
3   import java.util.Scanner;
4   public class Test {
5       public static void main(String[] args) {
6           Scanner input = new Scanner(System.in);
7           System.out.print("Enter a number : ");
8           int num = input.nextInt();
9           int sum=0;
10          for(int i=0;i<=num;i++){
11              sum+=i;
12          }
13          System.out.println("Sum = " + sum);
14          System.out.println("End");
15      }
16  }
```

รูปที่ 2.29 ตัวอย่างการใช้คำสั่ง for

จากรูปที่ 2.29 เป็นตัวอย่างการใช้คำสั่ง for เพื่อบวกเลขตั้งแต่ 0 ถึงเลขที่รับมา โดยมีการทำงานดังนี้

บรรทัดที่ 6 ประกาศตัวแปร input จากคลาส Scanner

บรรทัดที่ 8 รับค่าตัวเลขเป็นชนิด Integer โดยใช้เมธอด nextInt() เก็บค่าไว้ที่ตัวแปร num

บรรทัดที่ 9 ประกาศตัวแปร sum เป็นชนิดตัวเลขที่มีค่าเริ่มต้นเท่ากับ 0

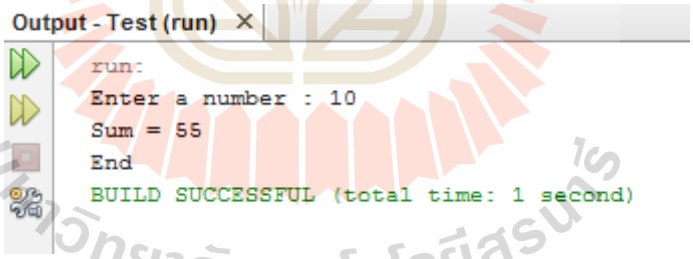
บรรทัดที่ 10-12 ประกาศตัวแปร i เป็นชนิดตัวเลขมีค่าเริ่มต้นเป็น 0 เพื่อใช้เป็นตัวแปรควบคุม มีเงื่อนไขว่า ตัวแปรควบคุมมีค่าน้อยกว่าหรือเท่ากับค่าที่รับมาหรือไม่
ถ้าเป็นจริง จะทำชุดคำสั่งในบรรทัดที่ 11
ถ้าเป็นเท็จ จะหลุดลูป
เมื่อทำงานชุดคำสั่งในบรรทัดที่ 11 เสร็จ ตัวแปรควบคุมจะเพิ่มค่าขึ้น 1 และทำการตรวจสอบเงื่อนไขอีกครั้ง

บรรทัดที่ 11 บวกค่า sum ด้วย i แล้วเก็บผลลัพธ์ไว้ที่ sum

บรรทัดที่ 13 แสดงข้อความว่า Sum = ตามด้วยค่าที่เก็บไว้ในตัวแปร sum ด้วยเมธอด println()

บรรทัดที่ 14 แสดงข้อความว่า End เพื่อจบการทำงาน ด้วยเมธอด println()

จากการทำงาน จะเห็นได้ว่า เมื่อเข้าลูป for จะมีการสร้างตัวแปรควบคุมและนำตัวแปรควบคุมนั้นมาตรวจสอบเงื่อนไขตามที่กำหนด ถ้าจริงจะทำงานชุดคำสั่งภายใต้ลูป for และทำการเพิ่มค่าตัวแปรควบคุม และนำไปตรวจสอบอีกครั้ง ทำวนไปเรื่อย ๆ จนกว่าเงื่อนไขจะเป็นเท็จจึงจะหลุดลูป เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังรูปที่ 2.30



รูปที่ 2.30 ตัวอย่างผลลัพธ์การใช้คำสั่ง for

2.3.7 Collection

2.3.7.1 List

List (Jeff, 2011; สุดา เธียรมนตรี, 2556) เป็นอินเตอร์เฟสที่มีการสืบทอดข้อมูลและวัตถุเก็บแบบเรียงลำดับเป็นรายการและมีลำดับ สามารถเข้าถึงข้อมูลได้ผ่านเลขระบุตำแหน่ง (index) ซึ่งเริ่มจากศูนย์ และมีค่าของสมาชิกเป็นค่าว่าง (null) ได้ มี class ที่สืบทอด คือ LinkedList , ArrayList , Vector

```
3  import java.util.*;
4  public class Test {
5      public static void main(String[] args) {
6          List l = new LinkedList();
7          l.add(1);
8          l.add(2);
9          l.add(3);
10         l.add(null);
11         l.add(2);
12         System.out.println("Size = " + l.size());
13         for(int i=0;i<l.size();i++){
14             System.out.print(l.get(i) + " ");
15         }
16         System.out.println("");
17         System.out.println("Index 1 = " + l.get(1));
18         l.add(1,null);
19         System.out.println("Index 1 = " + l.get(1));
           System.out.println("Index of 2 = " + l.indexOf(2));
21         l.remove(2);
22         l.remove(1);
23         System.out.println("Size = " + l.size());
24         System.out.println("Index of 2 = " + l.indexOf(2));
25     }
26 }
```

รูปที่ 2.31 ตัวอย่างการใช้ List

จากรูปที่ 2.31 เป็นตัวอย่างการใช้ List โดยมีการทำงานดังนี้

บรรทัดที่ 6 ประกาศตัวแปร l เป็นตัวแปรแบบ LinkedList

บรรทัดที่ 7-11 ทำการเพิ่มข้อมูลลงใน List ด้วยเมธอด add()

บรรทัดที่ 12 แสดง size ของ List ด้วยเมธอด size()

บรรทัดที่ 13-15 ใช้ลูป for ที่มีตัวแปรควบคุมชื่อ i เริ่มต้นที่ 0 มีการตรวจสอบเงื่อนไขว่าตัวแปรควบคุมมีค่าน้อยกว่า size ของ List หรือไม่
ถ้าเป็นจริง จะทำชุดคำสั่งในบรรทัดที่ 14
ถ้าเป็นเท็จ จะข้ามบรรทัดที่ 14 ไปเริ่มทำงานที่บรรทัดที่ 16

บรรทัดที่ 14 แสดงข้อมูลที่อยู่ใน List ในตำแหน่งที่ i ด้วยเมธอด get()

บรรทัดที่ 16 ขึ้นบรรทัดใหม่

บรรทัดที่ 17 แสดงข้อมูลใน list ตามตำแหน่งที่กำหนดด้วยเมธอด get()

บรรทัดที่ 18 เพิ่มข้อมูลลงใน List โดยกำหนดตำแหน่งที่อยู่ของข้อมูล ด้วยเมธอด add()

บรรทัดที่ 19 แสดงข้อมูลใน list ตามตำแหน่งที่กำหนดด้วยเมธอด get()

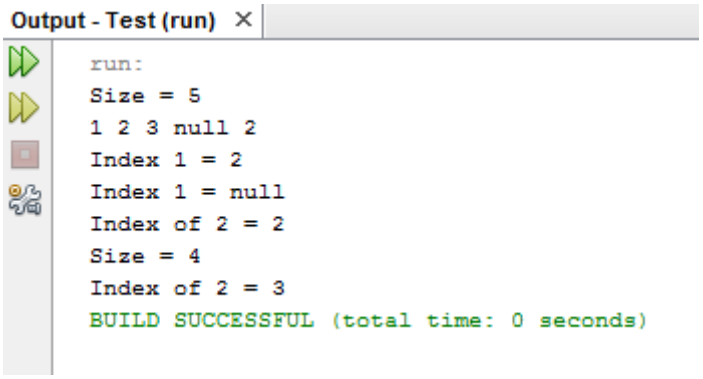
บรรทัดที่ 20 แสดงตำแหน่งแรกใน List ของเลข 2 ด้วยเมธอด indexOf()

บรรทัดที่ 21-22 ลบข้อมูลออกจาก List ด้วยเมธอด remove()

บรรทัดที่ 23 แสดง size ของ List หลังจากลบข้อมูลด้วยเมธอด size()

บรรทัดที่ 24 แสดงตำแหน่งแรกใน List ของเลข 2 หลังจากลบข้อมูลด้วยเมธอด indexOf()

จากการทำงาน จะเห็นได้ว่า การเพิ่มข้อมูลด้วยเมธอด add() สามารถทำได้สองแบบ คือ เพิ่มเข้าไปต่อท้ายข้อมูลล่าสุดที่มีอยู่ใน List ในบรรทัดที่ 7-11 และ แทรกลงไปในตำแหน่งที่ต้องการในบรรทัดที่ 18 สามารถเรียกข้อมูลออกมาใช้ได้ผ่านเมธอด get() โดยใส่ตำแหน่งที่ต้องการเรียกลงไปในวงเล็บในบรรทัดที่ 14 , 17 , 19 สามารถหาตำแหน่งแรกของข้อมูลที่ต้องการได้ผ่านเมธอด indexOf() โดยใส่ข้อมูลที่ต้องการค้นหาลงไปในวงเล็บในบรรทัดที่ 20 และ 24 สามารถดู size ของ List ได้ผ่านเมธอด size() ในบรรทัดที่ 12 และ 23 และสามารถลบข้อมูลออกจาก List ได้ผ่านเมธอด remove() โดยใส่ตำแหน่งที่ต้องการเรียกลงไปในวงเล็บในบรรทัด 21-22 เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังรูปที่ 2.32



รูปที่ 2.32 ตัวอย่างผลลัพธ์การใช้งาน List

2.3.7.1 Set

Set (Jeff, 2011; สุดา เธียรมนตรี, 2556) เป็นอินเตอร์เฟสที่มีการสืบทอดข้อมูลและวัตถุเก็บแบบไม่ซ้ำกันและจะถูกจัดเรียกลำดับข้อมูลโดยอัตโนมัติ ไม่มีลำดับของสมาชิก คลาสที่สืบทอด คือ HashSet

```
3  import java.util.*;
4  public class Test {
5      public static void main(String[] args) {
6          Set a = new HashSet();
7          Set b = new HashSet();
8
9          a.add(1);a.add(3);
10         a.add(2); a.add(1);
11         a.add(4);
12
13         b.add(3);b.add(1);
14         Set c = new HashSet();
15         System.out.println("Set A = " + a);
16         System.out.println("Set B = " + b);
17         System.out.println("C = " + c);
18         c.addAll(a);
19         System.out.println("After addAll C = " + c);
20         c.retainAll(b);
21         System.out.println("A intersect B = " + c);
22         c.removeAll(c);
23         System.out.println("After removeAll C = " + c);
24         c.addAll(a);c.addAll(b);
25         System.out.println("A Union B = " + c);
26         c.removeAll(c);c.addAll(a);
27         c.removeAll(b);
28         System.out.println("A difference B = " + c);
29     }
30 }
```

รูปที่ 2.33 ตัวอย่างการใช้ Set

จากรูปที่ 2.33 เป็นตัวอย่างการใช้ Set โดยมีการทำงานดังนี้

บรรทัดที่ 6 ประกาศตัวแปร a เป็นตัวแปรแบบ HashSet

บรรทัดที่ 7 ประกาศตัวแปร b เป็นตัวแปรแบบ HashSet

บรรทัดที่ 9-11 เพิ่มข้อมูลลงใน Set a ด้วยเมธอด add()

บรรทัดที่ 13 เพิ่มข้อมูลลงใน Set b ด้วยเมธอด add()

บรรทัดที่ 14 ประกาศตัวแปร c เป็นตัวแปรแบบ HashSet

บรรทัดที่ 15-17 แสดงข้อมูลที่อยู่ใน Set a , b , c ด้วยเมธอด println()

บรรทัดที่ 18 ทำการ Union Set c กับ Set a ด้วยเมธอด addAll()

บรรทัดที่ 19 แสดงข้อมูลที่อยู่ใน Set c หลังจาก Union ด้วยเมธอด println()

บรรทัดที่ 20 ทำการ Intersect Set c กับ Set b ด้วยเมธอด retainAll()

บรรทัดที่ 21 แสดงข้อมูลที่อยู่ใน Set c หลังจาก Intersect ด้วยเมธอด println()

บรรทัดที่ 22 ทำการ Difference Set c กับตัวมันเอง เพื่อเคลียร์ค่าทั้งหมดใน Set ด้วยเมธอด removeAll()

บรรทัดที่ 23 แสดงข้อมูลที่อยู่ใน Set c หลังจาก Difference ด้วยเมธอด println()

บรรทัดที่ 24 ทำการ Union Set c กับ Set a และ Union Set c กับ Set b ด้วยเมธอด addAll()

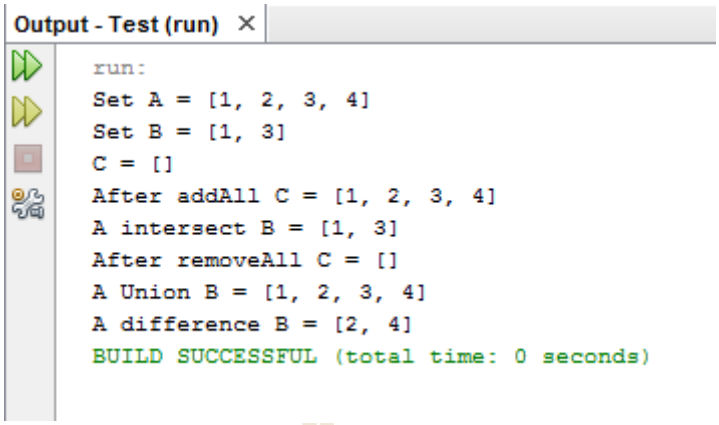
บรรทัดที่ 25 แสดงข้อมูลที่อยู่ใน Set c หลังจาก Union ด้วยเมธอด println()

บรรทัดที่ 26 ทำการ Difference Set c กับตัวมันเอง เพื่อเคลียร์ค่าทั้งหมดใน Set ด้วยเมธอด removeAll() และ Union Set c กับ Set a ด้วยเมธอด addAll()

บรรทัดที่ 27 ทำการ Difference Set c กับ Set b ด้วยเมธอด removeAll()

บรรทัดที่ 28 แสดงข้อมูลที่อยู่ใน Set c หลังจาก Difference ด้วยเมธอด println()

จากการทำงาน จะเห็นได้ว่า สามารถเพิ่มข้อมูลลงใน Set ได้ 2 แบบ คือ เพิ่มทีละตัวผ่านเมธอด add() ในบรรทัดที่ 9-13 และเพิ่มเข้าไปทั้ง set ผ่านเมธอด addAll() ซึ่งเป็นคำสั่งที่ใช้สำหรับ Union Set ในบรรทัดที่ 18 , 24 และ 26 และสามารถลบข้อมูลทั้งหมดใน set ได้ผ่านคำสั่ง removeAll() ซึ่งเป็นคำสั่งที่ใช้สำหรับ Difference Set ในบรรทัดที่ 22 , 26 และ 27 และ Intersect Set ได้ผ่าน retainAll() ในบรรทัดที่ 20 และเนื่องจาก Set ไม่มีการอ้างอิงตำแหน่ง ทำให้สามารถแสดงข้อมูลทั้งหมดใน Set ได้ผ่านเมธอด println() ได้เลย เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังรูปที่ 2.34



รูปที่ 2.34 ตัวอย่างผลลัพธ์การใช้งาน Set

2.4 กรอบงานเชิงวัตถุ (Object-Oriented Application Framework)

2.4.1 ความหมาย

กรอบงานเชิงวัตถุ (Timothy & Robert, 2001; กิตติ ภักดีวัฒนะกุล และ พานิชกุล พนิดา, 2550) คือ เทคโนโลยีที่ถูกสร้างขึ้นเพื่อทำให้ลดค่าใช้จ่ายและเพิ่มประสิทธิภาพในกระบวนการพัฒนาซอฟต์แวร์เกิดขึ้นเป็นรูปธรรม กรอบงานนั้นคือโปรแกรมประยุกต์ที่ไม่สมบูรณ์ถูกสร้างด้วยส่วนต่าง ๆ อันประกอบไปด้วยรูปแบบเชิงซอฟต์แวร์ คลังรหัสคำสั่งเชิงคลาส และส่วนประกอบเชิงซอฟต์แวร์ เพื่อสนับสนุนการนำกลับมาใช้ใหม่ในการพัฒนาโปรแกรมที่เฉพาะเจาะจงสำหรับลูกค้า โดยประโยชน์หลักที่ได้จากการใช้กรอบงานเชิงวัตถุ ประกอบไปด้วย สภาพเป็นส่วนจำเพาะ ความสามารถในการนำกลับมาใช้ใหม่ (Reusability) ความสามารถในด้านการเพิ่มความสามารถ (Extensibility) และ Inversion of Control

“กรอบงานเชิงวัตถุเพิ่มสภาพเป็นส่วนจำเพาะโดยการห่อหุ้ม (Encapsulation) รายละเอียดของส่วนโปรแกรมให้อยู่เบื้องหลังส่วนต่อประสานที่ชัดเจน สภาพเป็นส่วนจำเพาะของกรอบงานนั้นสามารถช่วยเพิ่มประสิทธิภาพของซอฟต์แวร์โดยการจำกัดผลกระทบอันเกิดเนื่องมาจากการเปลี่ยนแปลงของการออกแบบและการพัฒนา ซึ่งการเปลี่ยนแปลงที่เกิดขึ้นนี้ทำให้เกิดความยากลำบาก ในการดูแลซอฟต์แวร์ที่ได้รับการพัฒนาเสร็จสิ้นแล้ว” (Ghezzi, Jazayeri, & Mandrioli, 2002)

ส่วนต่อประสานที่ชัดเจนซึ่งได้รับจัดเตรียมไว้โดยกรอบงานเชิงวัตถุนั้นเพิ่มความสามารถในการนำกลับมาใช้ใหม่โดยการกำหนดส่วนโปรแกรมที่รองรับการสร้างโปรแกรม

ประยุกต์ขึ้นใหม่ ซึ่งความสามารถในการนำกลับมาใช้ใหม่นี้ ทำให้ประสบการณ์หรือความรู้ของนักพัฒนาโปรแกรมประยุกต์นั้นไม่สูญเสียไปโดยไร้ประโยชน์ ทำให้หลีกเลี่ยงการสร้างหรือแก้ไขปัญหาเพื่อตอบสนองต่อความต้องการเชิงซอฟต์แวร์ (Software Requirement) จากเดิมที่ได้รับการแก้ไขแล้ว เช่นเดียวกันความสามารถในการนำกลับมาใช้ใหม่นั้น ทำให้นักพัฒนาโปรแกรมพัฒนาโปรแกรมได้อย่างมีประสิทธิภาพ เพิ่มอัตราการผลิต อีกทั้งยังก่อให้เกิดความน่าเชื่อถือต่อซอฟต์แวร์ที่ได้รับการพัฒนาอีกด้วย กรอบงานเพิ่มความสามารถในด้านการเพิ่มความสามารถ (Extensibility) มากขึ้นโดยอาศัยหลักการของภาวะพหุสัณฐาน การสืบทอดและการนิยาม ฮุคซึ่งรับรองการขยายหรือเชื่อมต่อผ่านส่วนต่อประกอบที่ถูกกำหนดไว้อย่างชัดเจนจึงทำให้กรอบงานยังคงมีความเสถียรและทำงานได้อย่างมีประสิทธิภาพภายใต้กาลเวลาและความต้องการที่เปลี่ยนแปลงไป

กรอบงานเชิงวัตถุนั้นถูกออกแบบและกำหนดคุณสมบัติต่าง ๆ ด้วย “Inversion of Control” ซึ่งเป็นสถาปัตยกรรมที่มีกลไกในการสนับสนุนให้โปรแกรมเฉพาะที่ถูกพัฒนาภายใต้กรอบงาน ถูกกำหนดค่าได้อย่างพลวัตในขณะดำเนินงาน (Run-time) ด้วยเหตุนี้จึงทำให้กรอบงานเชิงวัตถุที่ถูกสร้างด้วยกลไก Inversion of Control มีความยืดหยุ่น และมีความสามารถในด้านการเพิ่มความสามารถ (Extensibility) สูง

2.4.2 การจำแนกกรอบงานสำหรับโปรแกรมประยุกต์

กรอบงานเชิงวัตถุนั้นสามารถจำแนกออกเป็นประเภทต่าง ๆ โดยขึ้นอยู่กับเกณฑ์ของการจำแนกซึ่งโดยทั่วไปแล้วประกอบไปด้วย (Ghezzi et al., 2002; Timothy & Robert, 2001)

จำแนกจากจุดประสงค์ในการสร้าง

1. System Infrastructure Framework เป็นกรอบงานที่ถูกพัฒนาขึ้นสำหรับสนับสนุนโปรแกรมประยุกต์ในด้านโครงสร้างพื้นฐาน (Infrastructure) ของโปรแกรมนั้น ๆ ตัวอย่างเช่น Operating System, Communication Framework, Language Processing Tools และกรอบงานที่เกี่ยวกับส่วนติดต่อกับผู้ใช้ (User Interface) ซึ่งกรอบงานเหล่านี้ถูกสร้างให้เป็นองค์ประกอบภายในโปรแกรมประยุกต์จึงไม่ถูกขายให้กับลูกค้าโดยตรง
2. Middleware Integration Framework เป็นกรอบงานที่ถูกสร้างขึ้นเพื่อเชื่อมการทำงาน Distribute Component ในโปรแกรมประยุกต์ กรอบงานประเภทนี้ช่วยเพิ่มความสามารถในด้านการเพิ่มความสามารถ (Extensibility) และ

ความสามารถในการนำกลับมาใช้ใหม่ของโครงสร้างพื้นฐานของกรอบงานให้ทำงานกับ Distributed Environment ได้ดีและราบรื่นมากยิ่งขึ้น

3. Enterprise Application Framework กรอบงานประเภทนี้มักถูกออกแบบเพื่อสนับสนุนการพัฒนาโปรแกรมประยุกต์ที่ใหญ่และมีความซับซ้อนอย่างโปรแกรมประยุกต์ที่ใช้ในด้านการสื่อสาร วิทยาศาสตร์ อุตสาหกรรมการผลิต และวิศวกรรมด้านการเงิน เมื่อเปรียบเทียบแล้ว Enterprise Application Framework นั้นใช้ต้นทุนในการผลิตมากกว่า System Infrastructure Framework และ Middleware Integration Framework แต่อย่างไรก็ตามกรอบงานประเภทนี้ให้ผลกลับคืนที่ดีกว่า เนื่องจากสนับสนุนการทำงานต่อผู้ใช้ปลายทางโดยตรง

นอกจากการจำแนกประเภทของกรอบงานข้างต้นแล้วยังสามารถจำแนกกรอบงานโดยพิจารณาจากเทคนิคที่ใช้ในการสืบทอดกรอบงานได้แก่ White Box Framework และ Black Box Framework (Timothy & Robert, 2001)

จำแนกจากเทคนิคที่ใช้ในการสร้างกรอบงาน

1. White Box Framework นั้นอาศัยแนวคิดเชิงวัตถุอย่างการสืบทอด (Inheritance) และ Dynamics Binding เพื่อให้เพิ่มความสามารถของ กรอบงานจาก Functionality เดิมที่มีอยู่ในกรอบงานนั้นถูกนำกลับมาใช้ และเพิ่มเติมโดยการ (1) สืบทอดจากคลาส (Classes) ที่ถูกเตรียมไว้ในกรอบ งาน (2) ซ้อนทับโดยใช้ฮุคที่กรอบงานนั้น ๆ เตรียมไว้
2. Black Box Framework เป็นกรอบงานที่สนับสนุนความสามารถในด้านการเพิ่มความสามารถ (Extensibility) โดยการกำหนดส่วนต่อประสาน (Interface) สำหรับการนำส่วนโปรแกรมเชิงวัตถุที่ถูกพัฒนาขึ้นใหม่ต่อ (Plug) เข้าสู่กรอบงาน ความสามารถที่มีอยู่เดิมของ กรอบงานนั้นถูกนำกลับมาใช้ใหม่โดย (1) การกำหนดส่วนโปรแกรมตามส่วนต่อประสาน (Interface) และ (2) รวมส่วนโปรแกรมเหล่านั้นเข้ากับกรอบงานโดยใช้รูปแบบเชิงซอฟต์แวร์อย่าง Strategy และ Functor

ในการพัฒนาโปรแกรมประยุกต์ภายใต้กรอบงานแบบ White box นั้นนักพัฒนาโปรแกรมประยุกต์จำเป็นต้องทราบโครงสร้างทางสถาปัตยกรรมของกรอบงานประเภทนี้เป็น อย่างดี แต่ในทางกลับกันการพัฒนาโปรแกรมประยุกต์โดยใช้กรอบงานแบบ Black Box นั้นไม่จำเป็นต้องทราบโครงสร้างทางสถาปัตยกรรมของกรอบงาน เพียงแต่ต้องทราบวิธีการเชื่อมต่อกับ

ส่วนต่อประสานที่กำหนดไว้อย่างดีแล้วเท่านั้น (Allen H. Dutoit & Bruegge, 2004; Timothy C. Lethbridge & Laganiere, 2005)

2.4.3 ข้อดีและข้อด้อยของกรอบงานสำหรับโปรแกรมประยุกต์

ในการพัฒนากรอบงานเชิงวัตถุนั้นถูกสร้างด้วยการนำแนวคิดและเทคโนโลยีหลากหลายมารวมเข้าด้วยกันได้แก่ รูปแบบเชิงซอฟต์แวร์ คลังรหัสคำสั่งเชิงคลาสและส่วนประกอบเชิงซอฟต์แวร์ จึงทำให้กรอบงานเชิงวัตถุนั้นเพิ่มอัตราการผลิดซอฟต์แวร์และลดขั้นตอนการพัฒนา อย่างไรก็ตามด้วยเหตุผลเดียวกัน การที่กรอบงานเชิงวัตถุนั้นประกอบไปด้วยหลายแนวคิดและองค์ประกอบ ทำให้เกิดความซับซ้อน ความพยายามในการนำกรอบงานเชิงวัตถุขนาดใหญ่มาใช้นั้นจึงมักล้มเหลว เว้นแต่มีกระบวนการจัดการในด้านขั้นตอนการพัฒนา ระยะเวลาในการศึกษาการใช้กรอบงาน การจัดการองค์ประกอบที่ถูกนำมารวมกันในกรอบงาน การบำรุงรักษา การตรวจสอบความถูกต้อง การจัดการความผิดพลาด สมรรถภาพและความไม่มีมาตรฐานของการสร้างกรอบงาน (วุฒิพล และ พิชโยทัย, 2551; กิตติ ภักดีวัฒนะกุล และ พานิชกุล พนิดา, 2550)

1. การพัฒนาซอฟต์แวร์ที่มีความซับซ้อนให้มีคุณภาพสูงภายใต้กรอบงานที่มีความซับซ้อนนั้นเป็นสิ่งที่ทำได้ยากเป็นอย่างยิ่ง จำเป็นต้องใช้นักพัฒนา ที่มีความเชี่ยวชาญสูง
2. การพัฒนาซอฟต์แวร์ภายใต้กรอบงานเชิงวัตถุนั้นจำเป็นต้องเสียเวลาส่วนหนึ่งในการเรียนรู้วิธีการใช้กรอบงานนั้น ๆ ตัวอย่างเช่นการเรียนรู้การใช้งานกรอบงานสำหรับซอฟต์แวร์ในเชิงกราฟฟิคอย่าง MFC หรือ MacApp ใช้เวลาโดยทั่วไปประมาณ 6 – 12 เดือนทั้งนี้ทั้งนั้นขึ้นอยู่กับประสบการณ์ของนักพัฒนาด้วย เว้นแต่การศึกษากรอบงานนั้นสามารถนำมาใช้ในการพัฒนาซอฟต์แวร์ได้หลายโครงการในคราวเดียวซึ่งย่อมถือเป็นการคุ้มค่าต่อต้นทุนและเวลา
3. การใช้กรอบงานหลายกรอบงานทำงานร่วมกัน (Integration of Multiple Framework) นั้นทำให้กระบวนการพัฒนาซอฟต์แวร์นั้นดีขึ้น (ตัวอย่างเช่นการทำงานร่วมกันของกรอบงานเกี่ยวกับ GUI, Communication และ Database) อย่างไรก็ตามกรอบงานนั้นถูกออกแบบในเริ่มแรกให้ทำงานร่วมกันภายในเป็นหลัก ไม่ได้ถูกออกแบบให้ทำงานร่วมกับส่วนโปรแกรมภายนอกที่อยู่

ภายในกรอบงานอื่นจึงทำให้เกิดปัญหาเกี่ยวกับ Concurrency/Distribution Architecture และ Event Dispatching Model

4. ความต้องการในเชิงซอฟต์แวร์ของโปรกรมแกรมประยุกต์โดยทั่วไปจะถูกเปลี่ยนแปลงอยู่บ่อย ๆ เช่นเดียวกัน ความต้องการหลักของกรอบงานย่อมถูกเปลี่ยนได้เช่นกัน จึงทำให้เกิดการปรับปรุงเปลี่ยนแปลงกรอบงาน
5. แม้ว่าการออกแบบกรอบงานเป็นส่วนย่อย (Module) นั้นทำให้จำกัดขอบเขตของผลกระทบอันเกิดจากข้อบกพร่องของซอฟต์แวร์ การตรวจสอบและการแก้ไขข้อบกพร่องที่เกิดขึ้น แต่ยังคงมีข้อเสียอื่น ๆ อีกด้วย ได้แก่ ส่วนโปรแกรมที่ได้รับการออกแบบให้มีความยืดหยุ่นสูง (Generic Component) นั้นยากในการตรวจสอบในเชิงนามธรรม เนื่องจากส่วนโปรแกรมเหล่านี้ ได้รับการออกแบบโดยปราศจากรายละเอียด ส่วนโปรแกรมเหล่านี้จะทำงาน ได้ต่อเมื่อมีการสร้าง ส่วนโปรแกรมที่เป็นรูปธรรม (Concrete Component) แล้วเท่านั้น ขณะที่การออกแบบ Generic Component นั้นก่อให้เกิดความ ยืดหยุ่นสูงแต่ทำให้เกิดความยากและซับซ้อนในการทดสอบในระดับ โปรแกรมย่อย (Module Testing) เนื่องมาจากส่วนโปรแกรมไม่สามารถ ทำงานได้ด้วยคำสั่งที่เกิดขึ้นจริงจากคลาสย่อยของ Generic Component นอกจากนี้แล้วปัญหายังเกิดกับกรอบงานที่ใช้แนวคิดของ Inversion of Control ใน การออกแบบกรอบงาน ซึ่งเป็นแนวคิดที่กลับการไหลของการควบคุม โปรแกรมทำให้ยากในการแก้ไขข้อผิดและยากต่อการเข้าใจพฤติกรรมของ กรอบงานนั้น

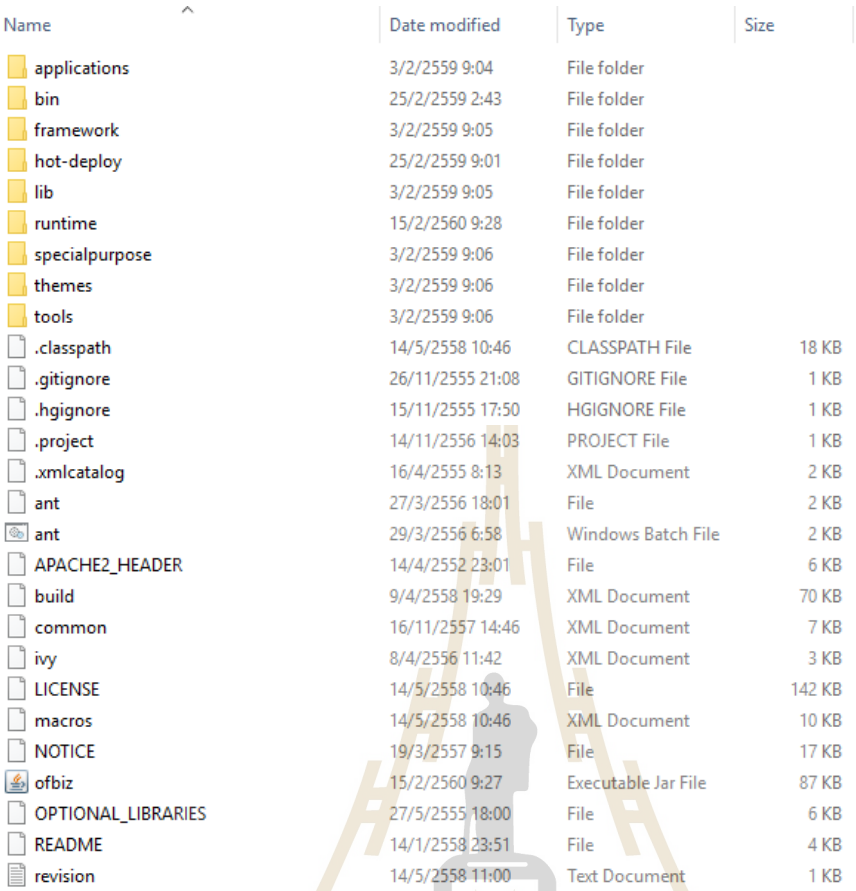
2.5 กรอบงานออฟบิส

2.5.1 ประวัติและความเป็นมา

กรอบงานออฟบิส (Apache Ofbiz : The Apache Open For Business Project) เป็นโครงการของอะแพชี (Apache) (David E. Jones, 2015) ที่เป็นแบบเปิดเผยต้นรหัส (Open Source) ที่เป็น ERP (Enterprise Resource Planning) ที่สร้างขึ้นเพื่อพัฒนาซอฟต์แวร์สำหรับองค์กรที่มีความยืดหยุ่นมากกว่ากรอบงานแบบเปิดเผยต้นรหัสตัวอื่น ๆ ไม่ว่าจะเป็นในเรื่องของ การติดตั้ง การใช้งาน การปรับแต่งแก้ไข การดูแลรักษาระบบ ฯลฯ และเนื่องจากความยืดหยุ่นและความมีประสิทธิภาพของออฟบิสทำให้ได้รับการยอมรับจากผู้ใช้งานทั่วโลก และปัจจุบันยังเป็น TLP (Top Level Project) ของอาปาเช่อีกด้วย มีโลโก้ของโปรเจคดังรูปที่ 2.35

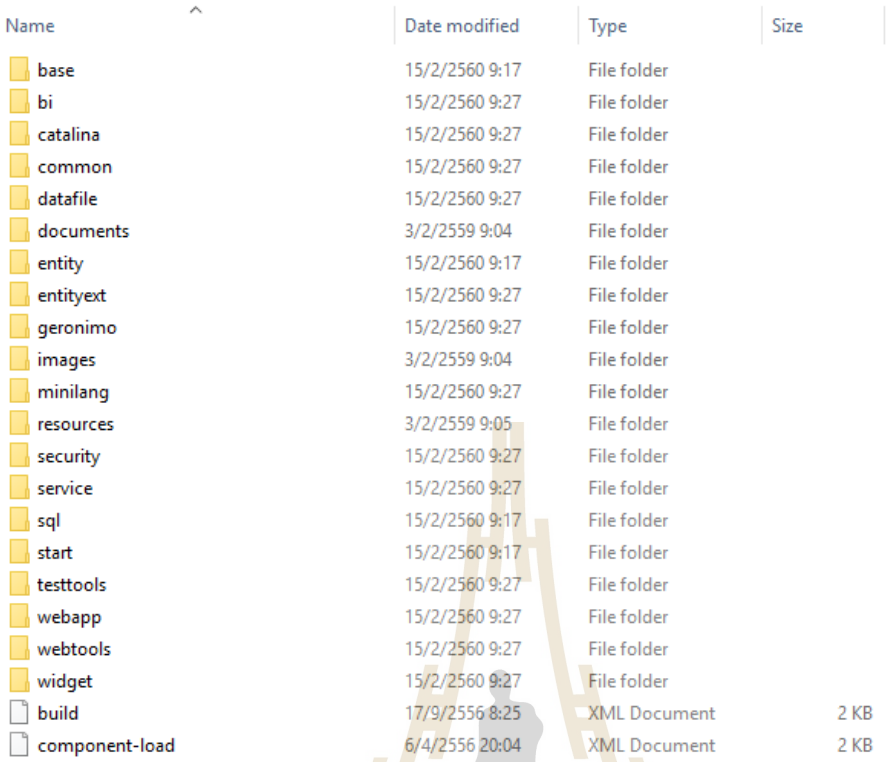
รูปที่ 2.35 โลโก้ของ Ofbiz

กรอบงานออฟบิสไม่ได้มีแค่อำนวยความสะดวกในการสร้างซอฟต์แวร์ แต่ยังมีเครื่องมือมากมายที่ผู้ใช้งานต้องการในระบบ ERP และอื่น ๆ อีกมากมาย ซึ่งประกอบด้วย Application Component(ERP, CRM, CMS, E- Commerce, POS) , Application Development Framework(UI Engine, Service Engine, Workflow Engine, Entity Engine) และ Data Model ผู้พัฒนาสามารถนำไปพัฒนาเพิ่มเติมความสามารถของออฟบิสโดยอาศัยฟังก์ชันพื้นฐานของออฟบิสได้ หรืออาจจะพัฒนาโดยใช้ Application Development Framework ก็สามารถสร้าง Application ที่มีประสิทธิภาพได้ ซึ่งก็ขึ้นอยู่กับผู้ใช้งานว่าจะเลือกใช้แค่บ้างส่วนหรือทั้งหมดที่มีให้ ซึ่งมันง่าย สะดวก และรวดเร็วในการพัฒนา เมื่อเทียบกับการเลือกใช้ซอฟต์แวร์อื่น ๆ ที่ต้องเสียเงินและเวลาในการสร้างสิ่งเหล่านี้ขึ้นมาใหม่ เพราะกรอบงานออฟบิสได้เตรียมเครื่องมือในการพัฒนาสิ่งต่าง ๆ ไว้อย่างครบถ้วน โดยมีทั้ง Widget สำหรับสร้าง User Interface ส่วนของ Service Engine และส่วนของ Entity Engine ดังตัวอย่างในรูปที่ 2.36 ถึง รูปที่ 2.39



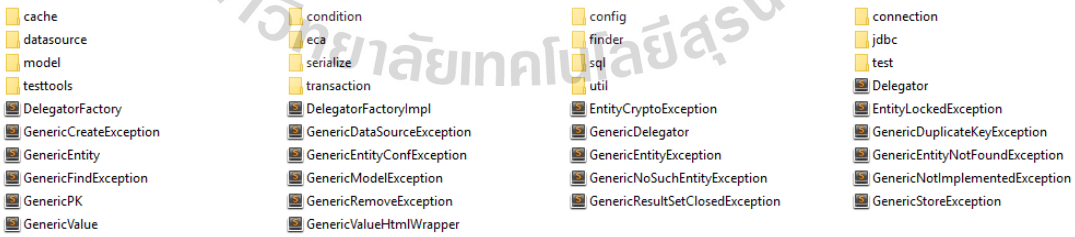
Name	Date modified	Type	Size
applications	3/2/2559 9:04	File folder	
bin	25/2/2559 2:43	File folder	
framework	3/2/2559 9:05	File folder	
hot-deploy	25/2/2559 9:01	File folder	
lib	3/2/2559 9:05	File folder	
runtime	15/2/2560 9:28	File folder	
specialpurpose	3/2/2559 9:06	File folder	
themes	3/2/2559 9:06	File folder	
tools	3/2/2559 9:06	File folder	
.classpath	14/5/2558 10:46	CLASSPATH File	18 KB
.gitignore	26/11/2555 21:08	GITIGNORE File	1 KB
.hgignore	15/11/2555 17:50	HGIGNORE File	1 KB
.project	14/11/2556 14:03	PROJECT File	1 KB
.xmlcatalog	16/4/2555 8:13	XML Document	2 KB
ant	27/3/2556 18:01	File	2 KB
ant	29/3/2556 6:58	Windows Batch File	2 KB
APACHE2_HEADER	14/4/2552 23:01	File	6 KB
build	9/4/2558 19:29	XML Document	70 KB
common	16/11/2557 14:46	XML Document	7 KB
ivy	8/4/2556 11:42	XML Document	3 KB
LICENSE	14/5/2558 10:46	File	142 KB
macros	14/5/2558 10:46	XML Document	10 KB
NOTICE	19/3/2557 9:15	File	17 KB
ofbiz	15/2/2560 9:27	Executable Jar File	87 KB
OPTIONAL_LIBRARIES	27/5/2555 18:00	File	6 KB
README	14/1/2558 23:51	File	4 KB
revision	14/5/2558 11:00	Text Document	1 KB

รูปที่ 2.36 ภายในโฟลเดอร์ของ Ofbiz

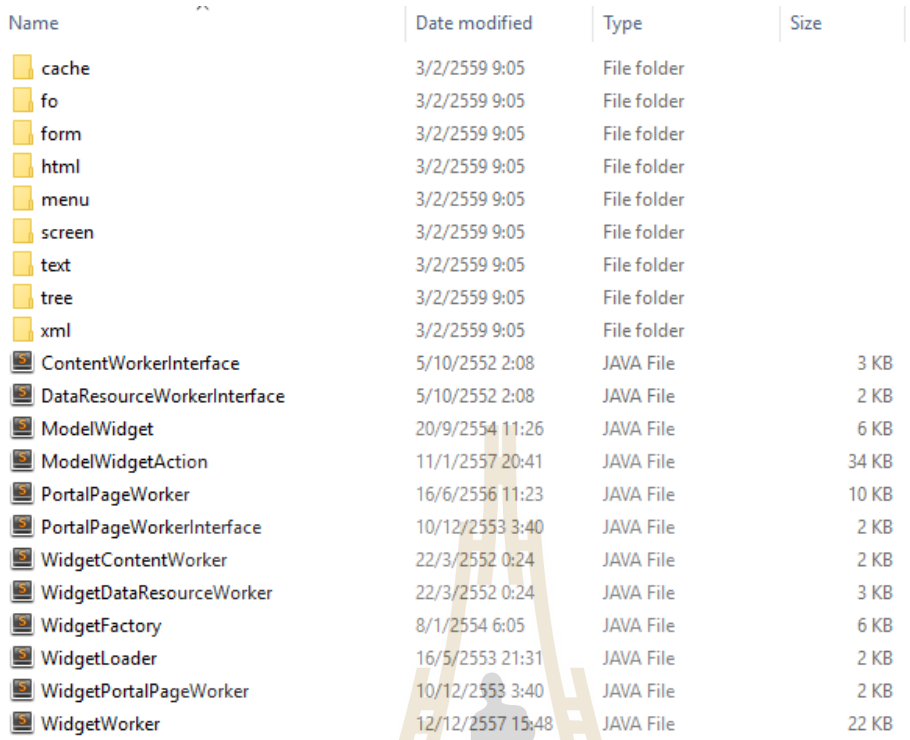


Name	Date modified	Type	Size
base	15/2/2560 9:17	File folder	
bi	15/2/2560 9:27	File folder	
catalina	15/2/2560 9:27	File folder	
common	15/2/2560 9:27	File folder	
datafile	15/2/2560 9:27	File folder	
documents	3/2/2559 9:04	File folder	
entity	15/2/2560 9:17	File folder	
entityext	15/2/2560 9:27	File folder	
geronimo	15/2/2560 9:27	File folder	
images	3/2/2559 9:04	File folder	
minilang	15/2/2560 9:27	File folder	
resources	3/2/2559 9:05	File folder	
security	15/2/2560 9:27	File folder	
service	15/2/2560 9:27	File folder	
sql	15/2/2560 9:17	File folder	
start	15/2/2560 9:17	File folder	
testtools	15/2/2560 9:27	File folder	
webapp	15/2/2560 9:27	File folder	
webtools	15/2/2560 9:27	File folder	
widget	15/2/2560 9:27	File folder	
build	17/9/2556 8:25	XML Document	2 KB
component-load	6/4/2556 20:04	XML Document	2 KB

รูปที่ 2.37 โฟลเดอร์ Framework



รูปที่ 2.38 โฟลเดอร์ Entity



Name	Date modified	Type	Size
cache	3/2/2559 9:05	File folder	
fo	3/2/2559 9:05	File folder	
form	3/2/2559 9:05	File folder	
html	3/2/2559 9:05	File folder	
menu	3/2/2559 9:05	File folder	
screen	3/2/2559 9:05	File folder	
text	3/2/2559 9:05	File folder	
tree	3/2/2559 9:05	File folder	
xml	3/2/2559 9:05	File folder	
ContentWorkerInterface	5/10/2552 2:08	JAVA File	3 KB
DataResourceWorkerInterface	5/10/2552 2:08	JAVA File	2 KB
ModelWidget	20/9/2554 11:26	JAVA File	6 KB
ModelWidgetAction	11/1/2557 20:41	JAVA File	34 KB
PortalPageWorker	16/6/2556 11:23	JAVA File	10 KB
PortalPageWorkerInterface	10/12/2553 3:40	JAVA File	2 KB
WidgetContentWorker	22/3/2552 0:24	JAVA File	2 KB
WidgetDataResourceWorker	22/3/2552 0:24	JAVA File	3 KB
WidgetFactory	8/1/2554 6:05	JAVA File	6 KB
WidgetLoader	16/5/2553 21:31	JAVA File	2 KB
WidgetPortalPageWorker	10/12/2553 3:40	JAVA File	2 KB
WidgetWorker	12/12/2557 15:48	JAVA File	22 KB

รูปที่ 2.39 โฟลเดอร์ Widget

กรอบงานออฟบิสมีฟังก์ชันการทำงานสำหรับสร้างซอฟต์แวร์ด้านต่าง ๆ เช่น การบัญชี, การบำรุงรักษาสินทรัพย์,แคตตาล็อกและการจัดการสินค้า,ระบบการจัดการคลังสินค้า (WMS), การผลิต, การประมวลผลการสั่งซื้อ, การจัดการสินค้าคงคลัง, ระบบจัดการเนื้อหา, ทรัพยากรมนุษย์, ผู้คนและกลุ่มบริหาร, การบริหารจัดการโครงการ เป็นต้น

```
 */
public interface Security {

    public Delegator getDelegator();

    public void setDelegator(Delegator delegator);

    /**
     * Uses userLoginSecurityGroupByUserLoginId cache to speed up the finding of the userLogin's security group list.
     *
     * @param userLoginId The userLoginId to find security groups by
     * @return An iterator made from the Collection either cached or retrieved from the database through the
     *             UserLoginSecurityGroup Delegator.
     */
    public Iterator<GenericValue> findUserLoginSecurityGroupByUserLoginId(String userLoginId);

    /**
     * Finds whether or not a SecurityGroupPermission row exists given a groupId and permission.
     * The groupId,permission pair is cached instead of the userLoginId,permission pair to keep the cache small and to
     * make it more changeable.
     *
     * @param groupId The ID of the group
     * @param permission The name of the permission
     * @return boolean specifying whether or not a SecurityGroupPermission row exists
     */
    public boolean securityGroupPermissionExists(String groupId, String permission);

    /**
     * Checks to see if the currently logged in userLogin has the passed permission.
     *
     * @param permission Name of the permission to check.
     * @param session The current HTTP session, contains the logged in userLogin as an attribute.
     * @return Returns true if the currently logged in userLogin has the specified permission, otherwise returns false.
     */
    public boolean hasPermission(String permission, HttpSession session);

    /**
     * Checks to see if the userLogin has the passed permission.
     *
     * @param permission Name of the permission to check.
     * @param userLogin The userLogin object for user to check against.
     * @return Returns true if the currently logged in userLogin has the specified permission, otherwise returns false.
     */
    public boolean hasPermission(String permission, GenericValue userLogin);

    /**
     * Like hasPermission above, except it has functionality specific to Entity permissions. Checks the entity for the
     * specified action, as well as for "_ADMIN" to allow for simplified general administration permission.
     *
```

รูปที่ 2.40 Interface Security

จากรูปที่ 2.40 เป็นตัวอย่างบางส่วนของอินเตอร์เฟส Security ซึ่งมีเมธอดที่จำเป็นสำหรับใช้ในการจัดการด้านความปลอดภัย การจำกัดสิทธิการเข้าถึงในส่วนต่าง ๆ เป็นต้น

```
48
49  public interface Delegator {
50
51      enum OperationType {INSERT, UPDATE, DELETE}
52
53      public void clearAllCacheLinesByDummyPK(Collection<GenericPK> dummyPKs);
54
55      public void clearAllCacheLinesByValue(Collection<GenericValue> values);
56
57      /**
58       * This method is a shortcut to completely clear all entity engine caches.
59       * For performance reasons this should not be called very often.
60       */
61      public void clearAllCaches();
62
63      public void clearAllCaches(boolean distribute);
64
65      /**
66       * Remove a CACHED Generic Entity from the cache by its primary key, does
67       * NOT check to see if the passed GenericPK is a complete primary key. Also
68       * tries to clear the corresponding all cache entry.
69       *
70       * @param primaryKey
71       *            The primary key to clear by.
72       */
73      public void clearCacheLine(GenericPK primaryKey);
74
75      public void clearCacheLine(GenericPK primaryKey, boolean distribute);
76
77      /**
78       * Remove a CACHED GenericValue from as many caches as it can. Automatically
79       * tries to remove entries from the all cache, the by primary key cache, and
80       * the by and cache. This is the ONLY method that tries to clear
81       * automatically from the by and cache.
82       *
83       * @param value
84       *            The GenericValue to clear by.
85       */
86      public void clearCacheLine(GenericValue value);
87
88      public void clearCacheLine(GenericValue value, boolean distribute);
89
90      /**
91       * Remove all CACHED Generic Entity (List) from the cache
92       *
93       * @param entityName
94       *            The Name of the Entity as defined in the entity XML file
```

รูปที่ 2.41 Interface Delegator

จากรูปที่ 2.41 เป็นตัวอย่างบางส่วนของอินเตอร์เฟส Delegator ซึ่งมีเมธอดที่จำเป็นสำหรับการจัดการกับ cache ในส่วนต่าง ๆ

กรอบงานออฟบิส มีวัตถุประสงค์หลัก (Howell, 2008) คือ ธุรกิจอีคอมเมิร์ซ ซึ่งมีเครื่องมือที่สามารถปรับแต่งได้ง่าย เช่น รูปแบบของระบบการจัดการคลังสินค้า (Warehouse Management System) ระบบบัญชีและการสั่งซื้อแบบเต็มรูปแบบและการตัดการสินค้า และยังมีส่วนหน้าเว็บไซต์สำหรับลูกค้าและระบบรถเข็น ซึ่งมีเครื่องมือและคุณสมบัติเทียบเท่ากับเว็บไซต์ที่

มีอยู่ในปัจจุบัน เช่น Amazon ออฟบิสรองรับหลายสกุลเงินและหลายภาษาซึ่งสามารถแสดงข้อความในภาษาต่าง ๆ ได้ตามที่ผู้ใช้ต้องการ

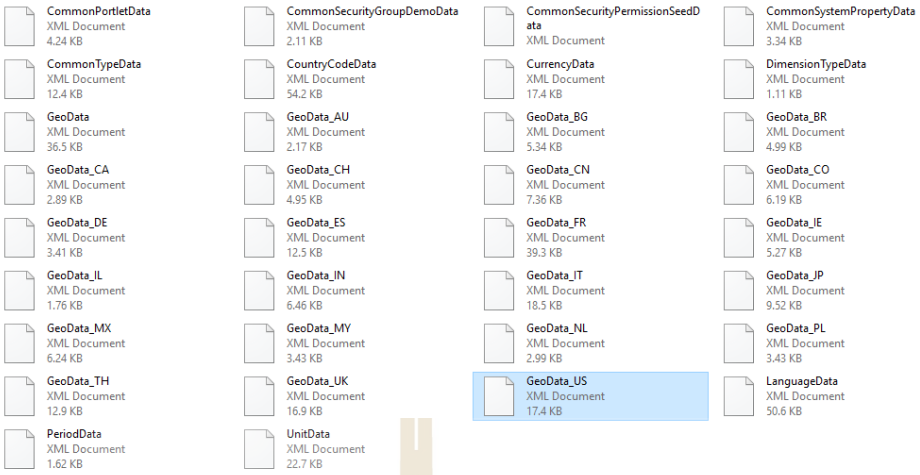
2.5.2 ส่วนประกอบที่สำคัญของกรอบงานออฟบิส

ทีมผู้พัฒนากรอบงานออฟบิสพยายามที่จะทำให้ออฟบิสมีคุณสมบัติที่ทำงานได้หลายๆรูปแบบ โดยได้พยายามหารูปแบบข้อมูลเริ่มต้นที่ดีเพื่อนำมาเป็นรากฐานของระบบ ทำการวิจัย ERP และระบบ CRM รวมถึงระบบทั่วไปเพื่อที่จะหาข้อมูลในรูปแบบต่าง ๆ และนำมาทำให้อยู่ในรูปแบบของข้อมูลทางกายภาพที่มีความยืดหยุ่นและให้องค์กรได้ทดลองใช้ รูปแบบของข้อมูลก็ได้ถูกปรับแต่งแก้ไขไปตามการใช้งานระบบในชีวิตประจำวันจนลงตัว เพราะความยืดหยุ่นจึงทำให้ง่ายต่อการปรับแต่งรูปแบบข้อมูล ซึ่งเป็นสิ่งจำเป็นต่อการพัฒนาระบบตามที่ผู้ใช้ต้องการ(David E. Jones, 2015)

ต่อไปเป็นตัวอย่างคร่าวๆของส่วนประกอบที่สำคัญของกรอบงานออฟบิส

1. **Common Data**

ข้อมูลทั่วไป (Common Data) ที่มีในกรอบงาน เช่น ขอบเขตทางภูมิศาสตร์ หน่วยวัด และอื่น ๆ เป็นสิ่งที่มีมาให้ตั้งแต่เริ่มติดตั้ง ซึ่งเป็นข้อมูลที่ถูกเปลี่ยนแปลงน้อยมาก เพราะเป็นข้อมูลที่ถูกบังคับใช้บนมาตรฐาน ISO และมาตรฐานอื่น ๆ ดังรูปที่ 2.42 โฟลเดอร์ common/data และ รูปที่ 2.43



รูปที่ 2.42 โฟลเดอร์ common/data

```
19  -->
20
21  <entity-engine-xml>
22      <Geo abbreviation="AA" geoCode="AA" geoId="AA" geoName="Armed Forces Americas" geoTypeId="STATE"/>
23      <Geo abbreviation="AE" geoCode="AE" geoId="AE" geoName="Armed Forces Europe" geoTypeId="STATE"/>
24      <Geo abbreviation="AP" geoCode="AP" geoId="AP" geoName="Armed Forces Pacific" geoTypeId="STATE"/>
25
26      <Geo abbreviation="AK" geoCode="AK" geoId="AK" geoName="Alaska" geoTypeId="STATE"/>
27      <Geo abbreviation="AL" geoCode="AL" geoId="AL" geoName="Alabama" geoTypeId="STATE"/>
28      <Geo abbreviation="AR" geoCode="AR" geoId="AR" geoName="Arkansas" geoTypeId="STATE"/>
29      <Geo abbreviation="AS" geoCode="AS" geoId="AS" geoName="American Samoa" geoTypeId="STATE"/>
30      <Geo abbreviation="AZ" geoCode="AZ" geoId="AZ" geoName="Arizona" geoTypeId="STATE"/>
31      <Geo abbreviation="CA" geoCode="CA" geoId="CA" geoName="California" geoTypeId="STATE"/>
32      <Geo abbreviation="CO" geoCode="CO" geoId="CO" geoName="Colorado" geoTypeId="STATE"/>
33      <Geo abbreviation="CT" geoCode="CT" geoId="CT" geoName="Connecticut" geoTypeId="STATE"/>
34      <Geo abbreviation="DC" geoCode="DC" geoId="DC" geoName="District of Columbia" geoTypeId="STATE"/>
35      <Geo abbreviation="DE" geoCode="DE" geoId="DE" geoName="Delaware" geoTypeId="STATE"/>
36      <Geo abbreviation="FL" geoCode="FL" geoId="FL" geoName="Florida" geoTypeId="STATE"/>
37      <Geo abbreviation="FM" geoCode="FM" geoId="FM" geoName="Federated States of Micronesia" geoTypeId="STATE"/>
38      <Geo abbreviation="GA" geoCode="GA" geoId="GA" geoName="Georgia" geoTypeId="STATE"/>
39      <Geo abbreviation="GU" geoCode="GU" geoId="GU" geoName="Guam" geoTypeId="STATE"/>
40      <Geo abbreviation="HI" geoCode="HI" geoId="HI" geoName="Hawaii" geoTypeId="STATE"/>
41      <Geo abbreviation="IA" geoCode="IA" geoId="IA" geoName="Iowa" geoTypeId="STATE"/>
42      <Geo abbreviation="ID" geoCode="ID" geoId="ID" geoName="Idaho" geoTypeId="STATE"/>
43      <Geo abbreviation="IL" geoCode="IL" geoId="IL" geoName="Illinois" geoTypeId="STATE"/>
44      <Geo abbreviation="IN" geoCode="IN" geoId="IN" geoName="Indiana" geoTypeId="STATE"/>
45      <Geo abbreviation="KS" geoCode="KS" geoId="KS" geoName="Kansas" geoTypeId="STATE"/>
46      <Geo abbreviation="KY" geoCode="KY" geoId="KY" geoName="Kentucky" geoTypeId="STATE"/>
47      <Geo abbreviation="LA" geoCode="LA" geoId="LA" geoName="Louisiana" geoTypeId="STATE"/>
48      <Geo abbreviation="MA" geoCode="MA" geoId="MA" geoName="Massachusetts" geoTypeId="STATE"/>
49      <Geo abbreviation="MD" geoCode="MD" geoId="MD" geoName="Maryland" geoTypeId="STATE"/>
50      <Geo abbreviation="ME" geoCode="ME" geoId="ME" geoName="Maine" geoTypeId="STATE"/>
51      <Geo abbreviation="MH" geoCode="MH" geoId="MH" geoName="Marshall Islands" geoTypeId="STATE"/>
52      <Geo abbreviation="MI" geoCode="MI" geoId="MI" geoName="Michigan" geoTypeId="STATE"/>
53      <Geo abbreviation="MN" geoCode="MN" geoId="MN" geoName="Minnesota" geoTypeId="STATE"/>
54      <Geo abbreviation="MP" geoCode="MP" geoId="MP" geoName="Northern Mariana Islands" geoTypeId="STATE"/>
55      <Geo abbreviation="MO" geoCode="MO" geoId="MO" geoName="Missouri" geoTypeId="STATE"/>
56      <Geo abbreviation="MS" geoCode="MS" geoId="MS" geoName="Mississippi" geoTypeId="STATE"/>
57      <Geo abbreviation="MT" geoCode="MT" geoId="MT" geoName="Montana" geoTypeId="STATE"/>
58      <Geo abbreviation="NC" geoCode="NC" geoId="NC" geoName="North Carolina" geoTypeId="STATE"/>
59      <Geo abbreviation="ND" geoCode="ND" geoId="ND" geoName="North Dakota" geoTypeId="STATE"/>
60      <Geo abbreviation="NE" geoCode="NE" geoId="NE" geoName="Nebraska" geoTypeId="STATE"/>
61      <Geo abbreviation="NH" geoCode="NH" geoId="NH" geoName="New Hampshire" geoTypeId="STATE"/>
62      <Geo abbreviation="NJ" geoCode="NJ" geoId="NJ" geoName="New Jersey" geoTypeId="STATE"/>
63      <Geo abbreviation="NM" geoCode="NM" geoId="NM" geoName="New Mexico" geoTypeId="STATE"/>
64      <Geo abbreviation="NV" geoCode="NV" geoId="NV" geoName="Nevada" geoTypeId="STATE"/>
65      <Geo abbreviation="NY" geoCode="NY" geoId="NY" geoName="New York" geoTypeId="STATE"/>
66      <Geo abbreviation="OH" geoCode="OH" geoId="OH" geoName="Ohio" geoTypeId="STATE"/>
```

รูปที่ 2.43 GeoData_US.xml

2. Security

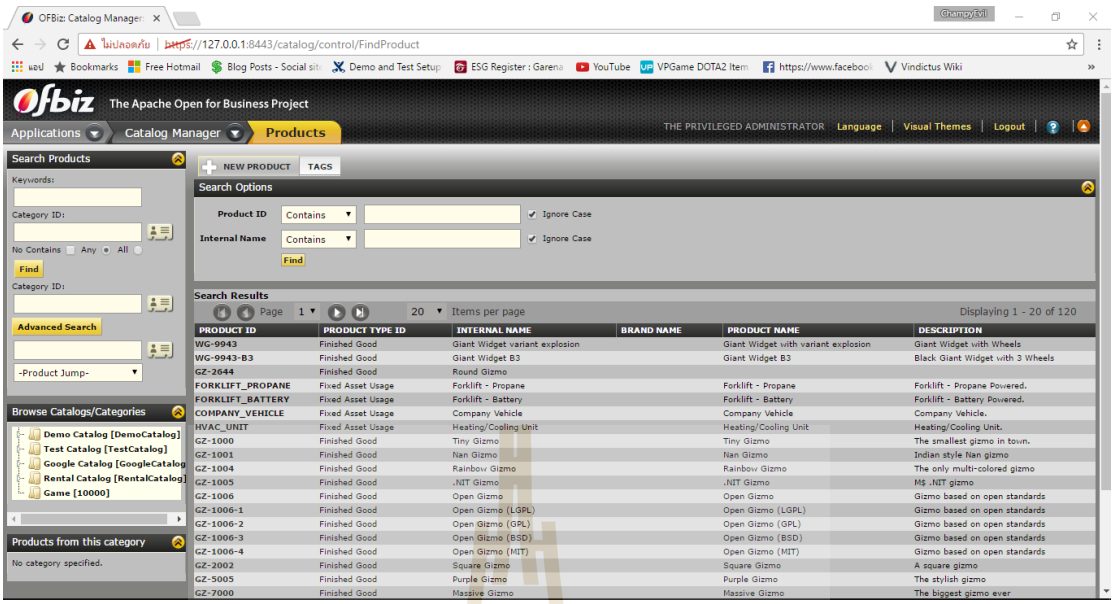
การรักษาความปลอดภัย (Security) ในกรอบงานนี้ สามารถใช้ควบคุมการเข้าถึงในส่วนต่าง ๆ ของระบบ และสามารถปรับแต่งเสริมให้มีความจำกัดและรัดกุมมากขึ้นเพื่อให้มันเฉพาะเจาะจงมากขึ้นดังรูปที่ 2.40

3. Party

การจัดกลุ่ม (Party) สามารถจัดได้ทั้งบุคคลและสิ่งของ ซึ่งประเภทข้อมูลที่เกี่ยวข้องกับส่วนนี้ เช่น ที่อยู่ไปรษณีย์, ที่อยู่อีเมล์ หรืออาจจะเป็นตำแหน่งหน้าที่หรือประเภทของบุคคล เช่น ลูกค้า, ผู้จัดจำหน่าย, พนักงาน, ผู้จัดการ ฯลฯ

4. Product

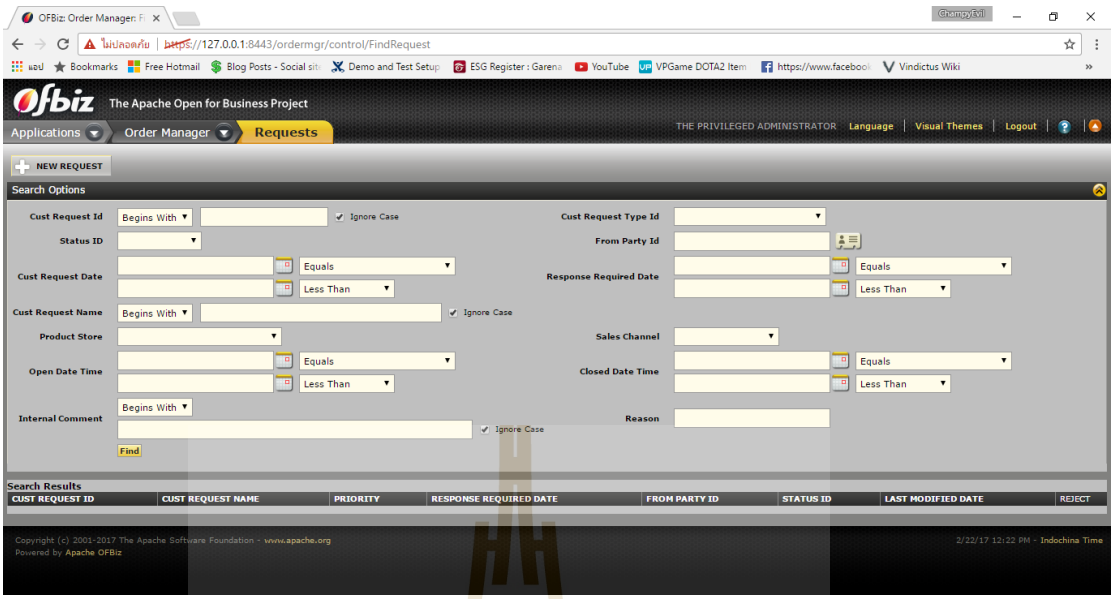
รายละเอียดสินค้า (Product) ใช้กันในหน่วยงานที่มีข้อมูลผลิตภัณฑ์ที่มีการขายหรือใช้ในหน่วยงาน เป็นสินค้า หรือ บริการ ที่มีหลายประเภท รวมถึงวัตถุดิบ อุปกรณ์ และ สินค้าสำเร็จได้ จัดหมวดหมู่สินค้าได้ โดย สินค้าหนึ่งอย่างเป็นได้หลายประเภทสามารถเชื่อมโยงข้าม แคตตาล็อกสินค้าได้ สามารถจัดการโปรโมชั่นที่แตกต่างกัน สามารถทำงานกับสินค้าชิ้นเดียวกันได้ มีความยืดหยุ่นในเรื่องของ คุณสมบัติของสินค้า ที่แตกต่างกันไป ตามหน่วยงานที่ใช้ จัดการราคาที่แตกต่างกัน ให้สามารถเชื่อมโยงกับสินค้าชิ้นเดียวกันได้ สามารถระบุสกุลเงินที่ต่างกันเพื่อใช้ในที่ ๆ ต่างกันและวันเวลาที่ต่างกันได้ สามารถกำหนดช่วงเวลาของราคาสินค้าได้ เพื่อเก็บประวัติการเปลี่ยนแปลงของราคาสินค้าดังรูปที่ 2.44



รูปที่ 2.44 หน้าต่างการทำงานส่วน Product

5. Order

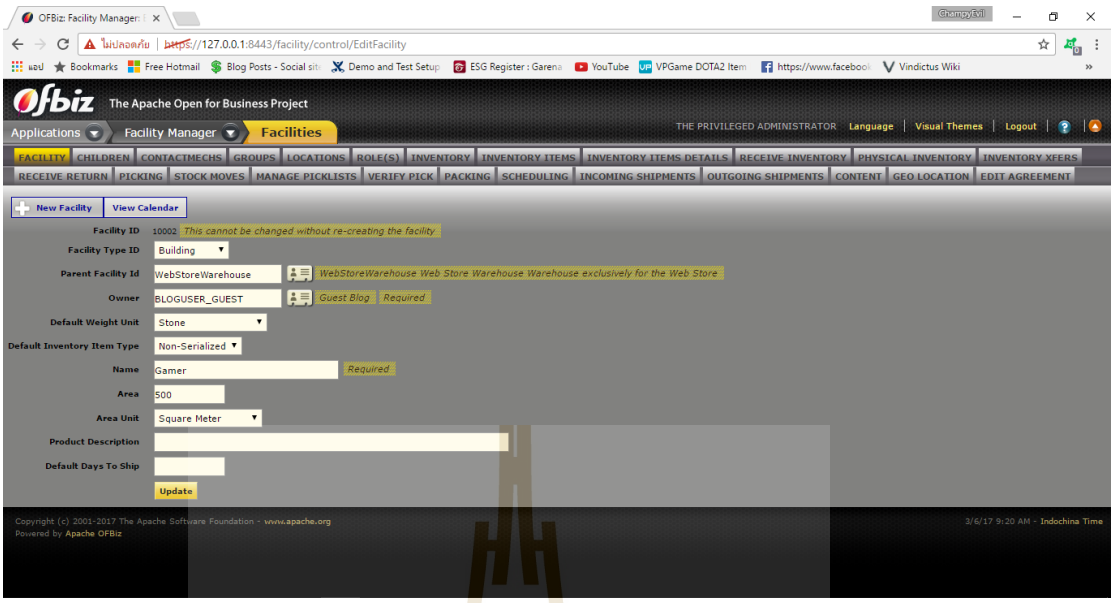
การสั่งซื้อสินค้า (Order) เป็นส่วนที่ใช้ในการจัดการกับการสั่งซื้อสินค้า โดยมีส่วนของใบสั่งซื้อที่สามารถแก้ไขได้ตามความต้องการของผู้ใช้ ไม่ว่าจะเป็นแบบฟอร์ม รายละเอียดต่าง ๆ รวมไปถึงสามารถปรับการคำนวณราคา,ภาษี ,ส่วนลด ให้ตรงตามที่ต้องการ และมีส่วนของการสร้างใบแจ้งหนี้ซึ่งจะมีรูปแบบมาตรฐานไว้ให้ ซึ่งเราสามารถปรับแต่งให้ตรงตามที่ต้องการได้ดังรูปที่ 2.45



รูปที่ 2.45 หน้าต่างการทำงานส่วน Order

6. Facility

สถานที่ (Facility) ซึ่งหมายถึง อาคาร หรือ สถานที่ทางกายภาพอื่น ๆ เช่น โกดัง, ร้านค้า และอื่น ๆ ซึ่งสามารถจัดกลุ่มได้ เช่น กลุ่มของร้านค้าในภูมิภาคเดียวกัน เป็นต้น รายการสินค้าคงคลังจะเชื่อมอยู่กับสิ่งอำนวยความสะดวกนี้ทำให้สามารถจัดการกับสินค้าคงคลังที่เชื่อมอยู่ได้ง่าย และสิ่งอำนวยความสะดวกยังเชื่อมอยู่กับองค์กรหรือกลุ่มคนที่เป็นคนจัดการด้วยดังรูปที่ 2.46



รูปที่ 2.46 หน้าต่างการทำงานส่วน Facility

7. Shipment

การจัดส่งสินค้า (Shipment) ใช้เพื่อติดตามการจัดส่งสินค้า ทั้งขาเข้าและขาออก รวมถึงการจัดการรายการของสินค้าคงคลัง ซึ่งการจัดส่งจะมีส่วนประกอบหลายอย่าง เช่น รายการสั่งซื้อที่มีจำนวนที่แน่นอน, ข้อมูลที่ใช้ติดตามการจัดส่ง ใบเสร็จรับเงิน, การจัดการหีบห่อสำหรับการจัดส่งของสินค้า เป็นต้น

นอกจากในส่วนของกรอบงานแล้ว ยังมีส่วนของตัวอย่างแอพพลิเคชัน ไว้สำหรับให้ผู้ใช้ได้ศึกษาการทำงานของออฟบิสผ่านทางแอพพลิเคชันที่มีให้ดังรูปที่ 2.44 ถึง รูปที่ 2.46 ซึ่งจากการทดลองใช้งานและศึกษาเบื้องต้นได้พบว่ามีการทำงานหลายส่วนที่ยังไม่สมบูรณ์และบางโมดูลที่ควรมีกลับไม่มี เช่น โมดูลของการตัดยอดสินค้า ซึ่งเป็นส่วนที่สำคัญในการจัดการการเข้าออกคลังของสินค้า ซึ่งผู้วิจัยเล็งเห็นความสำคัญในส่วนนี้ จึงเลือกโมดูลการตัดยอดสินค้าเป็นหัวข้อในการทำงานวิจัยครั้งนี้

2.6 งานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องกับการสร้างกรอบงานและการพัฒนากรอบกงานออฟบิส ผู้วิจัยได้ทำการศึกษาค้นคว้างานวิจัยที่มีความเกี่ยวข้องกับงานวิจัยที่จะทำโดยมีรายละเอียดโดยสรุปดังนี้

Laurent Bernardin (1999) (Bernardin, 1999) ได้นำเสนอการกรอบงานสำหรับการดำเนินการกระจายในการคำนวณสัญลักษณ์ที่พัฒนากรอบงานด้วยภาษาจาวา โดยใช้กรอบงานนี้คำนวณความต้องการทรัพยากรขนาดใหญ่และประมวลผลแบบขนานบนเครือข่ายอินเตอร์เน็ต ซึ่งใช้งานได้ดีในระดับหนึ่งแต่ยังมีข้อจำกัดบางประการ เช่น ความสมบูรณ์ของการทำงานคู่ขนาน, การจัดการกับข้อมูลขนาดใหญ่ที่ทำให้เครื่องช้า

Gun Ho Lee และ Junsu Jung (2007) (Lee & Jung, 2007) ได้ศึกษาและพัฒนากรอบงานสำหรับระบบธุรกิจบนเว็บไซต์ที่มีความซับซ้อนให้มีประสิทธิภาพมากขึ้น โดยใช้จาวาและเอกซ์เอ็มแอลในการพัฒนากรอบงานสำหรับเว็บไซต์แบบหลายชั้น (multi-tier) ซึ่งทำให้ใช้เวลาในการพัฒนาระบบน้อยลง

Rittammanart Nattanicha, Wongyued Wisut และ Dailey Matthew N (2008) (Rittammanart, Wongyued, & Dailey, 2008) ได้นำเสนอซอฟต์แวร์ประเภท ERP ที่เป็นแบบเปิดเผยแหล่งข้อมูล เพื่อให้องค์กรขนาดเล็กได้มีทางเลือกในการใช้งาน โดยได้ยกตัวอย่างกรอบงานมา 2 ตัว คือ JBoss และ Ofbiz เพื่อเปรียบเทียบประสิทธิภาพกัน ซึ่งก็ได้ผลสรุปว่า Ofbiz ดีกว่าเกือบทุกข้อและง่ายต่อการเรียนรู้ ง่ายต่อการพัฒนา และลดงบประมาณในการพัฒนา

Zhao, Longwenและ Liu, Jinyu (2011) (Zhao & Liu, 2011) ได้นำเสนอกรอบการทำงานแบบบูรณาการของ SaaS บนพื้นฐานของ Apache Ofbiz ที่มีมาตรฐานชัดเจนและมีความยืดหยุ่นในการทำงาน เพื่อเป็นทางเลือกในการพัฒนา SaaS

Carlos A. Jara, Francisco Esquembre และคณะ (2012) (Jara et al., 2012) ได้ศึกษาการใช้คอมพิวเตอร์ในการสร้างภาพสามมิติที่มีความซับซ้อน และอธิบายกรอบงานของการสร้างภาพสามมิติที่ใช้ภาษาจาวาที่อิมพลีเมนต์ในไลบรารี่ Open Source Physics (OSP) และพัฒนากรอบงานนี้เพื่อให้มีประสิทธิภาพมากขึ้น คือ มีการตอบสนองต่อการสร้างภาพสามมิติได้ดีขึ้น แก้ไขปัญหาเชิงตัวเลขของสมการเชิงอนุพันธ์ของภาพสองมิติและสามมิติ

Yu Liu, Qinghua Guo และ Yuan Tian (2012) (Liu, Guo, & Tian, 2012) ได้นำเสนอกรอบงานที่ใช้ในการจัดการหมู่ในการทำนายการกระจายทางภูมิศาสตร์ที่พัฒนาโดยใช้ C++ และอยู่บนพื้นฐานของ OOP เพื่อให้มีความแม่นยำมากขึ้น ซึ่งกรอบงานที่ปรับปรุงแล้วมีความละเอียดในการจำแนกหมวดหมู่ความและมีความแม่นยำมากขึ้น และในอนาคตได้มีการวางแผนที่จะปรับปรุงให้มีปลั๊กอินและไลบรารี่ที่คล้ายกับไพธอน

จากการศึกษางานวิจัยที่เกี่ยวข้องพบว่า มีการใช้ภาษาจาวาและภาษาอื่น ๆ ในการพัฒนากรอบงาน เช่น C++ ซึ่งเป็นภาษาที่มีใช้เทคนิคการเขียนเชิงวัตถุที่มีคุณสมบัติของคลาสและออบเจ็กต์ที่สามารถนำกลับมาใช้ใหม่ได้ง่ายจึงเหมาะกับการนำมาเขียนกรอบงาน และมีการใช้กรอบงานออฟบิสเป็นตัวพื้นฐานในการศึกษา ซึ่งมีทั้งนำกรอบงานมาเพื่อแสดงให้เห็นถึงความยืดหยุ่นของกรอบงานในการใช้งานเพื่อใช้เป็นตัวเลือกในการพัฒนา และนำมาเปรียบเทียบกับกรอบงานอีกตัวเพื่อให้เห็นว่าตัวกรอบงานออฟบิสทำงานได้ดีกว่าและยืดหยุ่นกว่า สาระสำคัญในงานวิจัยนี้เมื่อเปรียบเทียบกับงานวิจัยอื่นสรุปได้ดังตารางที่ 2.4

ตารางที่ 2.4 สรุปเปรียบเทียบงานวิจัยที่เกี่ยวข้องกับการพัฒนากรอบงานของระบบคลังสินค้า

กระบวนการทำงาน	งานวิจัยที่เกี่ยวข้อง						
	ก	ข	ค	ง	จ	ฉ	ช*
ภาษาที่ใช้ในการพัฒนา							
Java	✓	✓	✓	✓	✓		✓
C++						✓	
XML		✓					
ระบบที่ใช้ในการศึกษา							
การคำนวณสัญลักษณ์และประมวลผลแบบขนาน	✓						
ระบบธุรกิจบนเว็บไซต์		✓					
ERP			✓				
SaaS				✓			
การสร้างภาพสามมิติ					✓		
การทำนายการกระจายทางภูมิศาสตร์						✓	
การจัดการสินค้าคงคลัง							✓
เกณฑ์การประเมินประสิทธิภาพ							
เป็นทางเลือกในการเลือกใช้งาน	✓		✓	✓			
ลดงบประมาณในการพัฒนา			✓				
ลดระยะเวลาในการพัฒนา		✓	✓	✓			
ปรับปรุงให้มีประสิทธิภาพดีขึ้น	✓		✓		✓	✓	
ได้โมดูลใหม่					✓		✓

หมายเหตุ งานวิจัยที่เกี่ยวข้อง ประกอบด้วย

ก แทนงานวิจัยของ Laurent Bernardin (1999)

ข แทนงานวิจัยของ Gun Ho Lee และ Junsu Jung (2007)

ค แทนงานวิจัยของ Rittammanart Nattanicha และ คณะ (2008)

ง แทนงานวิจัยของ Zhao, Longwenและ Liu, Jinyu (2011)

จ แทนงานวิจัยของ Carlos A. Jara, Francisco Esquembre และคณะ (2012)

ฉ แทนงานวิจัยของ Yu Liu, Qinghua Guo และ Yuan Tian (2012)

ช* แทนงานวิจัยของ การพัฒนาโมดูลการตัดยอดสินค้าสำหรับกรอบงานออฟบิส (งานวิจัยของวิทยานิพนธ์ฉบับนี้)

บทที่ 3
วิธีดำเนินการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อศึกษาการพัฒนาโมดูลการตัดยอดสินค้าในแบบต่าง ๆ สำหรับกรอบงานออฟบิสโดยใช้ภาษาจาวาเป็นหลัก โดยมีรายละเอียดดังนี้

3.1 เครื่องมือที่ใช้ในการวิจัย

เครื่องมือที่ใช้ในงานวิจัยนี้ ประกอบด้วยฮาร์ดแวร์และซอฟต์แวร์ ดังนี้

1) เครื่องคอมพิวเตอร์ โดยมีรายละเอียดดังนี้

- หน่วยประมวลผลกลาง : Intel® Core i7
- หน่วยความจำสำรอง : 1024 GB
- หน่วยความจำหลัก : 8 GB
- อุปกรณ์เสริมอื่น ๆ เช่น เมาส์ แป้นพิมพ์ เป็นต้น

2) ระบบปฏิบัติการและโปรแกรมประยุกต์สำหรับการพัฒนากรอบงานสำหรับการจัดการคลังสินค้า ประกอบไปด้วย

- ระบบปฏิบัติการ : Windows 10 Pro 64-bit Operating System
- เครื่องมือในการพัฒนาโปรแกรม : Sumlime Text 2
- กรอบงานที่ใช้ในการศึกษา : Apache Ofbiz 13.07.02
- ภาษาที่ใช้ในการพัฒนา : Java JDK 1.7.0_13

3.2 การพัฒนาโมดูลการตัดยอดสินค้า

โมดูลการตัดยอดสินค้าเป็นโมดูลสำคัญโมดูลหนึ่ง ที่ช่วยในการจัดการการเข้าออกคลังของสินค้า เพื่อลดปริมาณของสินค้าคงคลัง ช่วยจัดการลำดับการจำหน่ายสินค้า เพื่อลดปัญหาการหมดอายุของสินค้าที่อยู่ในคลัง โดยจะแบ่งการพัฒนาออกเป็น 3 แบบ คือ โมดูลสำหรับสินค้าแบบเข้าก่อนออกก่อน (First In First Out : FIFO) , โมดูลสำหรับสินค้าแบบเข้าหลังออกก่อน (Last In First Out : LIFO) และโมดูลสำหรับสินค้าแบบหมดอายุก่อนออกก่อน (First Expire date First Out)

3.2.1 โมดูลสำหรับสินค้าแบบเข้าก่อนออกก่อน (FIFO)

โมดูลสำหรับสินค้าแบบเข้าก่อนออกก่อน (First In First Out : FIFO) เป็นโมดูลที่ใช้สำหรับจัดการสินค้าในคลังโดยมีเงื่อนไขว่า ถ้าสินค้าชิ้นไหนเข้ามาก่อนจะถูกนำจัดจำหน่ายก่อน

เป็นการจัดการสินค้าที่พื้นฐานที่สุด ป้องกันการอยู่ในคลังนานเกินไปของสินค้า ซึ่งอาจทำให้สินค้าบางชนิดชำรุดเสียหายก่อนจัดจำหน่าย ซึ่งเป็นสาเหตุที่ทำให้เกิดการขาดทุนได้

สินค้าที่จัดอยู่ในส่วนที่ต้องจัดการแบบเข้าก่อนออกก่อนมีอยู่หลากหลายชนิด ซึ่งในงานวิจัยนี้ จะกำหนดให้สินค้าที่มีชนิดเป็นวัสดุอุปกรณ์ เช่น ไม้อัด ตะปู ปูนปลาสเตอร์ ฯลฯ เป็นสินค้าที่จะถูกจัดการด้วยโมดูลแบบเข้าก่อนออกก่อน โดยโมดูลการจัดการสินค้าแบบเข้าก่อนออกก่อนนั้น มีการออกแบบการทำงาน คือ

กำหนดให้รายการของรหัสสินค้าที่ส่งมาเป็น listA และกำหนดให้ prodIn เป็นตัวแปรแบบลำดับของตัวเลขเพื่อใช้เก็บลำดับของสินค้าที่มีวันนำเข้าที่ถูกบันทึกก่อนชิ้นอื่น ๆ แล้วนำ listA ไปตรวจสอบ เมื่อมีการบันทึกสินค้าหรืออัพเดตจำนวนสินค้า จะเรียกใช้โมดูลและส่งลิสของสินค้าชิ้นดังกล่าวในระบบทั้งหมดไปที่โมดูล โดยมีลำดับการทำงานดังนี้

```
int prodIn;
for(int i=0; i<listA.size();i++){
    prodIn = 1;
        for(int j=0; j<listA.size();j++){
                if(listA.get(i).getFromDate()>listA.get(j).getFromDate()){
                    prodIn++;
                }
        }
        listA.get(i).setIndex(prodIn);
}
```

จากซอร์สโค้ดข้างต้น อธิบายได้ คือ ประกาศตัวแปรชื่อ prodIn เป็นตัวแปรชนิดตัวเลข จากนั้นใช้ for loop ทำการการวนลูปตามจำนวนของสินค้าในลิสที่ส่งมา โดยใช้ตัวแปร i ในการควบคุมจำนวนรอบ กำหนดให้ prodIn มีค่าเป็น 1 ใช้ for loop วนลูปตามจำนวนสินค้าในลิสที่ส่งมาอีกครั้ง โดยใช้ตัวแปร j ในการควบคุมจำนวนรอบ แล้วตรวจสอบว่า วันนำเข้าสินค้าของสินค้าตัวที่ i มีค่ามากกว่า (เข้ามาทีหลัง) วันนำเข้าสินค้าของสินค้าตัวที่ j หรือไม่ ถ้าใช่ ให้เพิ่มค่าของ prodIn ขึ้นไป 1 วนแบบนี้ไปเรื่อย ๆ จนครบรอบของลูป j จึงนำค่า prodIn ที่ได้ไปใช้ในการกำหนดค่า Index ของสินค้าตัวที่ i แล้วเพิ่มค่า i เพื่อตรวจสอบสินค้าชิ้นต่อไป

เมื่อนำซอร์สโค้ดที่ได้ มาเขียนจริงและปรับแต่งให้เข้ากับกรอบงานออฟบิสจะได้ดังรูปที่ 3.1 โดยใช้ชื่อว่า fifoCutOff รับค่าลิสที่ชื่อว่า prod โดยเขียนไว้ในกรอบงานในส่วนของ EntityUtil.java ซึ่งมีเมธอดที่ใช้สำหรับจัดการข้อมูลที่บันทึกลงในฐานข้อมูล

```
542     /* getFifoCutOff is Method for cut-off items that have fromdate before order
543     and have type of Material*/
544     public static void fifoCutOff(List<GenericValue> prod){
545         int prodIn;
546         for(int i=0;i<prod.size();i++){
547             prodIn=1;
548                 for (int j =0; j< prod.size();j++) {
549                     if(prod.get(i).getFromDate()>prod.get(j).getFromDate())
550                         prodIn++;
551                 }
552                 prod.get(i).setIndex(prodIn);
553         }
554     }
```

รูปที่ 3.1 ซอร์สโค้ดสำหรับการทำงานแบบ FIFO

3.2.2 โมดูลสำหรับสินค้าแบบเข้าหลังออกก่อน (LIFO)

โมดูลสำหรับสินค้าแบบเข้าหลังออกก่อน (Last In First Out : LIFO) เป็นโมดูลที่ใช้สำหรับจัดการสินค้าในคลังโดยมีเงื่อนไขว่า ถ้าสินค้าชิ้นไหนเข้ามาล่าสุดหรือหลังสุดจะถูกนำจัดจำหน่ายก่อน เป็นการจัดการสินค้าที่จะใช้กับการจัดการสินค้าที่เป็นอุปกรณ์อิเล็กทรอนิกส์ ซึ่งเป็นสินค้าที่มีอัตราความล้าสมัยสูง มีสินค้าที่คุณสมบัติคล้ายกันแต่ดีกว่าเข้ามาอยู่เรื่อย ๆ ทำให้ต้องหาวิธีการจัดการเพื่อให้ขาดทุนน้อยที่สุด ซึ่งการจัดการแบบเข้าหลังออกก่อนก็เป็นวิธีหนึ่งที่ถูกเลือกใช้ในการจัดการส่วนนี้

สินค้าที่จัดอยู่ในส่วนที่ต้องจัดการแบบเข้าหลังออกก่อนมีอยู่หลากหลายชนิด ซึ่งในงานวิจัยนี้ จะกำหนดให้สินค้าที่มีชนิดเป็นอุปกรณ์อิเล็กทรอนิกส์ เช่น มือถือ แลปทอป เครื่องคอมพิวเตอร์ ฯลฯ เป็นสินค้าที่จะถูกจัดการด้วยโมดูลแบบเข้าหลังออกก่อน โดยโมดูลการจัดการสินค้าแบบเข้าหลังออกก่อนนั้น มีการออกแบบการทำงาน คือ

กำหนดให้รายการของรหัสสินค้าที่ส่งมาเป็น listB และกำหนดให้ prodIn เป็นตัวแปรแบบลำดับของตัวเลขเพื่อใช้เก็บลำดับของสินค้าที่มีวันนำเข้าที่ถูกบันทึกหลังชิ้นอื่น ๆ แล้วนำ listB ไปตรวจสอบ เมื่อมีการบันทึกสินค้าหรืออัพเดตจำนวนสินค้า จะเรียกใช้โมดูลและส่งลิสของสินค้าชิ้นดังกล่าวในระบบทั้งหมดไปที่โมดูล โดยมีลำดับการทำงานดังนี้

```
int prodIn;
for(int i=0; i<listB.size();i++){
    prodIn = 1;
        for(int j=0; j<listB.size();j++){
                if(listB.get(i).getFromDate()<listB.get(j).getFromDate()){
                    prodIn++;
```

```
                }
            }
            listB.get(i).setIndex(prodIn);
        }
```

จากซอร์สโค้ดข้างต้น อธิบายได้ คือ ประกาศตัวแปรชื่อ prodIn เป็นตัวแปรชนิดตัวเลข จากนั้นใช้ for loop ทำการการวนลูปตามจำนวนของสินค้าในลิสที่ส่งมา โดยใช้ตัวแปร i ในการควบคุมจำนวนรอบ กำหนดให้ prodIn มีค่าเป็น 1 ใช้ for loop วนลูปตามจำนวนสินค้าในลิสที่ส่งมาอีกครั้ง โดยใช้ตัวแปร j ในการควบคุมจำนวนรอบ แล้วตรวจสอบว่า วันนำเข้าสินค้าของสินค้าตัวที่ i มีค่าน้อยกว่า (เข้ามาก่อน) วันนำเข้าสินค้าของสินค้าตัวที่ j หรือไม่ ถ้าใช่ ให้เพิ่มค่าของ prodIn ขึ้นไป 1 วนแบบนี้ไปเรื่อย ๆ จนครบรอบของลูป j จึงนำค่า prodIn ที่ได้ไปใช้ในการกำหนดค่า Index ของสินค้าตัวที่ i แล้วเพิ่มค่า i เพื่อตรวจสอบสินค้าชิ้นต่อไป

เมื่อนำซอร์สโค้ดที่ได้ มาเขียนจริงและปรับแต่งให้เข้ากับกรอบงานออฟบิสจะได้ดังรูปที่ 3.2 โดยใช้ชื่อว่า lifoCutOff รับค่าลิสที่ชื่อว่า prod โดยเขียนไว้ในกรอบงานในส่วนของ EntityUtil.java ซึ่งมีเมธอดที่ใช้สำหรับจัดการข้อมูลที่บันทึกลงในฐานข้อมูล

```
568     /* getlifoCutOff is Method for cut-off items that have fromdate after order
569      and have type of Electronic */
570     public static void lifoCutOff(List<GenericValue> prod){
571         int prodIn;
572         for(int i=0;i<prod.size();i++){
573             prodIn=1;
574                 for (int j =0; j< prod.size();j++) {
575                     if(prod.get(i).getFromDate()<prod.get(j).getFromDate())
576                         prodIn++;
577                 }
578                 prod.get(i).setIndex(prodIn);
579         }
580     }
```

รูปที่ 3.2 ซอร์สโค้ดสำหรับการทำงานแบบ LIFO

3.2.3 โมดูลสำหรับสินค้าแบบหมดอายุก่อนออกก่อน (FEFO)

โมดูลสำหรับสินค้าแบบหมดอายุก่อนออกก่อน (First Expire date First Out : FEFO) เป็นโมดูลที่ใช้สำหรับจัดการสินค้าในคลังโดยมีเงื่อนไขว่า ถ้าสินค้าชิ้นไหนหมดอายุก่อนจะถูกนำจัดจำหน่ายก่อน เป็นการจัดการสินค้าที่พื้นฐานอีกรูปแบบหนึ่ง ที่ใช้สำหรับสินค้าที่มีวัน

หมดอายุที่แน่นอน ป้องกันการหมดอายุก่อนจำหน่ายสินค้า ซึ่งเป็นอีกสาเหตุที่ทำให้เกิดการขาดทุนได้

สินค้าที่จัดอยู่ในส่วนที่ต้องจัดการแบบหมดอายุก่อนออกก่อนมีอยู่หลากหลายชนิด ซึ่งในงานวิจัยนี้ จะกำหนดให้สินค้าที่มีชนิดเป็นอาหาร ซึ่งมีวันหมดอายุที่ชัดเจน เช่น มะเขือเทศ บะหมี่ กระเทียม ฯลฯ เป็นสินค้าที่จะถูกจัดการด้วยโมดูลแบบหมดอายุก่อนออกก่อน โดยโมดูลการจัดการสินค้าแบบเข้าก่อนออกก่อนนั้น มีการออกแบบการทำงาน คือ

กำหนดให้รายการของรหัสสินค้าที่ส่งมาเป็น listC และกำหนดให้ prodIn เป็นตัวแปรแบบลำดับของตัวเลขเพื่อใช้เก็บลำดับของสินค้าที่มีวันหมดอายุที่หมดก่อนชิ้นอื่น ๆ แล้วนำ listC ไปตรวจสอบ เมื่อมีการบันทึกสินค้าหรืออัพเดตจำนวนสินค้า จะเรียกใช้โมดูลและส่งลิสของสินค้าชิ้นดังกล่าวในระบบทั้งหมดไปที่โมดูล โดยมีลำดับการทำงานดังนี้

```
int prodIn;
for(int i=0; i<listC.size();i++){
    prodIn = 1;
        for(int j=0; j<listC.size();j++){
                if(listC.get(i).getExpDate()>listC.get(j).getExpDate()){
                    prodIn++;
                }
        }
        listC.get(i).setIndex(prodIn);
}
```

จากซอร์สโค้ดข้างต้น อธิบายได้ คือ ประกาศตัวแปรชื่อ prodIn เป็นตัวแปรชนิดตัวเลข จากนั้นใช้ for loop ทำการการวนลูปตามจำนวนของสินค้าในลิสที่ส่งมา โดยใช้ตัวแปร i ในการควบคุมจำนวนรอบ กำหนดให้ prodIn มีค่าเป็น 1 ใช้ for loop วนลูปตามจำนวนสินค้าในลิสที่ส่งมาอีกครั้ง โดยใช้ตัวแปร j ในการควบคุมจำนวนรอบ แล้วตรวจสอบว่า วันหมดอายุของสินค้าตัวที่ i มีค่ามากกว่า (หมดอายุทีหลัง) วันหมดอายุของสินค้าตัวที่ j หรือไม่ ถ้าใช่ ให้เพิ่มค่าของ prodIn ขึ้นไป 1 วนแบบนี้ไปเรื่อย ๆ จนครบรอบของลูป j จึงนำค่า prodIn ที่ได้ไปใช้ในการกำหนดค่า Index ของสินค้าตัวที่ i แล้วเพิ่มค่า I เพื่อตรวจสอบสินค้าชิ้นต่อไป

เมื่อนำซอร์สโค้ดที่ได้ มาเขียนจริงและปรับแต่งให้เข้ากับกรอบงานออฟบิสจะได้ดังรูปที่ 3.3 โดยใช้ชื่อว่า fefoCutOff รับค่าลิสที่ชื่อว่า prod โดยเขียนไว้ในกรอบงานในส่วนของ EntityUtil.java ซึ่งมีเมธอดที่ใช้สำหรับจัดการข้อมูลที่บันทึกลงในฐานข้อมูล

```
555     /* getFefoCutOff is Method for cut-off items that have expdate before order
556     and have type of Food */
557     public static void fefoCutOff(List<GenericValue> prod){
558         int prodIn;
559         for(int i=0;i<prod.size();i++){
560             prodIn=1;
561                 for (int j =0; j< prod.size();j++) {
562                     if(prod.get(i).getExpDate()>prod.get(j).getExpDate())
563                         prodIn++;
564                 }
565                 prod.get(i).setIndex(prodIn);
566         }
567     }
```

รูปที่ 3.3 ซอร์สโค้ดสำหรับการทำงานแบบ FEFO

หลังจากที่ได้เขียนซอร์สโค้ดจากการออกแบบลงในกรอบงานออฟบิสแล้ว ก็ได้ทำการเรียกใช้เมธอดที่ได้สร้างขึ้น เพื่อใช้ในการทดสอบต่อไป ซึ่งหลักการทำงานคือ เมื่อมีการอัพเดตข้อมูลสินค้า หรือเพิ่มสินค้าเข้าไปในคลัง จะเรียกใช้เมธอด 1 ใน 3 เมธอด ตามเงื่อนไขที่กำหนด

```
920             Map<String, Object> m = prodFindProduct(productId);
921             List<T> productsCutOff = new ArrayList<values>(m.values());
922             if(productsCutOff.get(0).getType() == "MATERIAL"){
923                 EntityUtil.fifoCutOff(productsCutOff);
924             }else if(productsCutOff.get(0).getType() == "ELECTRONIC"){
925                 EntityUtil.lifoCutOff(productsCutOff);
926             }else if(productsCutOff.get(0).getType() == "FOOD"){
927                 EntityUtil.fefoCutOff(productsCutOff);
928             }
```

รูปที่ 3.4 ซอร์สโค้ดสำหรับเรียกใช้เมธอดที่สร้างขึ้น

จากรูปที่ 3.4 เป็นซอร์สโค้ดที่ใช้เรียกเมธอดที่สร้างขึ้น โดยมีการทำงาน คือ สร้างตัวแปรชื่อว่า m เป็นตัวแปรแบบ Map เพื่อรับค่าที่ส่งกลับมาจากเมธอด prodFindProduct ที่มีการคืนค่าเป็น Map โดยเมธอด prodFindProduct นี้ เป็นเมธอดที่ใช้ในการค้นหาสินค้าที่เราต้องการ โดยใช้รหัสสินค้าที่เราส่งไป ถัดมาสร้างตัวแปรชื่อว่า productsCutOff เป็นตัวแปรแบบลิส เพื่อทำการแปรค่าของ Map ที่ได้จากการค้นหามาเป็นลิสที่เราต้องการ โดยเก็บเฉพาะส่วนของ Value ที่เป็น Object ของสิ่งที่ค้นหามาเท่านั้น เมื่อได้ลิสที่ต้องการแล้ว ถัดไปจะเป็นการกำหนดเงื่อนไขการใช้เมธอดทั้ง 3 เมธอด โดยมีเงื่อนไขว่า

- ถ้าค่า productTypeID ของสินค้าในลิสนั้นมีค่าเป็น MATERIAL (วัสดุอุปกรณ์) จะส่งลิสนั้นไปจัด Index ตามแบบการจัดการสินค้าแบบเข้าก่อนออกก่อนด้วยเมธอด fifoCutOff
- ถ้าค่า productTypeID ของสินค้าในลิสนั้นมีค่าเป็น ELECTRONIC (อิเล็กทรอนิกส์) จะส่งลิสนั้นไปจัด Index ตามแบบการจัดการสินค้าแบบเข้าหลังออกก่อนด้วยเมธอด lifoCutOff
- ถ้าค่า productTypeID ของสินค้าในลิสนั้นมีค่าเป็น FOOD (อาหาร) จะส่งลิสนั้นไปจัด Index ตามแบบการจัดการสินค้าแบบหมดอายุก่อนออกก่อนด้วยเมธอด fefoCutOff

3.3 การปรับปรุงเขตข้อมูลของกรอบงาน

เพื่อให้เมธอดสามารถทำงานได้ตามที่วางแผนไว้ ผู้วิจัยจึงได้ทำการ ปรับเปลี่ยนและเพิ่มเติมเขตข้อมูลของกรอบงานบางส่วน เพื่อให้สอดคล้องกับการทำงาน

โดยส่วนที่เพิ่มเติมเข้าไป คือ ProductCutOff เป็นตารางฐานข้อมูลอีกหนึ่งตาราง สร้างขึ้นเพื่อใช้ในการแยกการจัดเก็บสินค้าว่า สินค้าชิ้นนี้ มีการนำเข้ากี่ครั้ง จัดลำดับการนำออกจำหน่ายได้ ดังรูปที่ 3.5

```
2701    <entity entity-name="ProductCutOff"
2702            package-name="org.ofbiz.product.product"
2703            title="Product Cut-Off Entity">
2704      <field name="productCutOffId" type="id-ne"></field>
2705      <field name="productId" type="id"></field>
2706      <field name="productTypeId" type="id"></field>
2707      <field name="fromDate" type="date-time"></field>
2708      <field name="expDate" type="date-time"></field>
2709      <field name="total" type="numeric"></field>
2710      <field name="index" type="numeric"></field>
2711      <field name="internalName" type="description"></field>
2712      <field name="brandName" type="name"></field>
2713      <field name="comments" type="comment"></field>
2714      <field name="description" type="description"></field>
2715      <prim-key field="productCutOffId"/>
2716      <relation type="one" fk-name="PROD_ID" rel-entity-name="Product">
2717        <key-map field-name="productId"/>
2718      </relation>
2719    </entity>
```

รูปที่ 3.5 ซอร์สโค้ดการสร้างตาราง ProductCutOff

ส่วนที่เพิ่มเติม คือ เขตข้อมูล Total หรือจำนวนสินค้าที่นำเข้า โดยเพิ่มไว้สองจุด คือ ในส่วนของตาราง Product สำหรับเก็บจำนวนทั้งหมดของสินค้า และในส่วนของการแสดงผลหน้า Product เพื่อแสดงให้เห็นว่า ตอนนี้มีจำนวนสินค้าชิ้นนั้น ๆ อยู่เท่าไร ดังรูปที่ 3.6 รูปที่ 3.7

```
2720    <entity entity-name="Product"
2721            package-name="org.ofbiz.product.product"
2722            title="Product Entity">
2723      <field name="productId" type="id-ne"></field>
2724      <field name="productTypeId" type="id"></field>
2725      <field name="primaryProductCategoryId" type="id"><description>The primary
2726        category ; it should be one of the productCategoryId already setup
2727        in ProductCategoryMember</description></field>
2728      <field name="manufacturerPartyId" type="id"></field>
2729      <field name="facilityId" type="id"></field>
2730
2731      <field name="total" type="numeric"></field>
2732      <!-- <field name="index" type="numeric"></field> -->
2733      <field name="releaseDate" type="date-time"></field>
2734      <field name="supportDiscontinuationDate" type="date-time"></field>
2735      <field name="salesDiscontinuationDate" type="date-time"></field>
2736      <field name="salesDiscWhenNotAvail" type="indicator"></field>
2737      <field name="internalName" type="description"></field>
2738      <field name="brandName" type="name"></field>
```

รูปที่ 3.6 ซอร์สโค้ดที่เพิ่มเขตข้อมูล Total ในตาราง Product

```
33  <form name="ListProducts" list-name="listIt" target="" title="" type="list" paginate-target="FindProduct"
34      odd-row-style="alternate-row" default-table-style="basic-table hover-bar" header-row-style="header-row-2">
35      <actions>
36          <set field="entityName" value="Product"/>
37          <service service-name="performFind" result-map="result" result-map-list="listIt">
38              <field-map field-name="inputFields" from-field="requestParameters"/>
39              <field-map field-name="entityName" from-field="entityName"/>
40              <field-map field-name="orderBy" from-field="parameters.sortField"/>
41              <field-map field-name="viewIndex" from-field="viewIndex"/>
42              <field-map field-name="viewSize" from-field="viewSize"/>
43          </service>
44      </actions>
45      <field name="productId" sort-field="true">
46          <hyperlink also-hidden="false" description="${productId}" target="EditProduct">
47              <parameter param-name="productId"/>
48          </hyperlink>
49      </field>
50      <field name="productTypeId" sort-field="true"><display-entity entity-name="ProductType" description="${description}"/></field>
51      <field name="internalName" sort-field="true"><display/></field>
52      <field name="brandName" sort-field="true"><display/></field>
53      <field name="productName" sort-field="true"><display/></field>
54      <field name="total" sort-field="true"><display/></field>
55      <field name="description" sort-field="true"><display/></field>
56      <!-- <field name="Exp. Date" sort-field="true"><display/></field> -->
57  </form>
```

รูปที่ 3.7 ซอร์สโค้ดที่เพิ่มเขตข้อมูล Total ในส่วนแสดงผล Product

3.4 การออกแบบการทดสอบ

3.4.1 ข้อมูลที่ใช้ทดสอบ

ข้อมูลที่ใช้ในการทดสอบนั้น มีทั้งหมด 10 ชนิด คละกัน โดยทุกชนิดมีการนำสินค้าเข้า 30 วัน วันละ 10 ชิ้นต่อชนิด รวมเป็นชนิดละ 300 ชิ้น

PRODUCT ID ▼	PRODUCT TYPE ID	INTERNAL NAME	BRAND NAME	PRODUCT NAME	TOTAL
00001	Food	Salt	Salt		300
00002	Food	Garlic	Garlic		300
00003	Electronic	Mobile	iPhone		300
00004	Electronic	Mobile	Samsung		300
00005	Material	Ply Wood			300
00006	Material	Nail			300
00007	Material	Plaster			300
00008	Food	Noodle			300
00009	Food	Tomato			300
00010	Electronic	Notebook	Lenovo		300

รูปที่ 3.8 ข้อมูลที่ใช้ทดสอบทั้งหมด

จากรูปที่ 3.8 แสดงสินค้าทั้งหมดที่ใช้ในการทดสอบ มีทั้งหมด 10 ชนิด แบ่งเป็น Food 4 ชนิด Material 3 ชนิด และ Electronic 3 ชนิด แต่ละชนิดมีจำนวนรวม 300 ชิ้น

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
1	00001	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	10	1	Salt	Salt
11	00001	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	10	2	Salt	Salt
21	00001	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	10	3	Salt	Salt
31	00001	FOOD	2017-01-04 00:00:00.000	2017-01-04 00:00:00.000	10	4	Salt	Salt
41	00001	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Salt	Salt
51	00001	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Salt	Salt
61	00001	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Salt	Salt
71	00001	FOOD	2017-01-08 00:00:00.000	2017-02-08 00:00:00.000	10	8	Salt	Salt
81	00001	FOOD	2017-01-09 00:00:00.000	2017-02-09 00:00:00.000	10	9	Salt	Salt
91	00001	FOOD	2017-01-10 00:00:00.000	2017-02-10 00:00:00.000	10	10	Salt	Salt
101	00001	FOOD	2017-01-11 00:00:00.000	2017-02-11 00:00:00.000	10	11	Salt	Salt

รูปที่ 3.9 ตัวอย่างข้อมูลประเภท FOOD

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
3	00003	ELECTRONIC	2017-01-30 00:00:00.000		10	1	Mobile	iPhone
13	00003	ELECTRONIC	2017-01-29 00:00:00.000		10	2	Mobile	iPhone
23	00003	ELECTRONIC	2017-01-28 00:00:00.000		10	3	Mobile	iPhone
33	00003	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Mobile	iPhone
43	00003	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Mobile	iPhone
53	00003	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Mobile	iPhone
63	00003	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Mobile	iPhone
73	00003	ELECTRONIC	2017-01-23 00:00:00.000		10	8	Mobile	iPhone
83	00003	ELECTRONIC	2017-01-22 00:00:00.000		10	9	Mobile	iPhone
93	00003	ELECTRONIC	2017-01-21 00:00:00.000		10	10	Mobile	iPhone
103	00003	ELECTRONIC	2017-01-20 00:00:00.000		10	11	Mobile	iPhone
113	00003	ELECTRONIC	2017-01-19 00:00:00.000		10	12	Mobile	iPhone

รูปที่ 3.10 ตัวอย่างข้อมูลประเภท ELECTRONIC

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
6	00006	MATERIAL	2017-01-01 00:00:00.000		10	1	Nail	
16	00006	MATERIAL	2017-01-02 00:00:00.000		10	2	Nail	
26	00006	MATERIAL	2017-01-03 00:00:00.000		10	3	Nail	
36	00006	MATERIAL	2017-01-04 00:00:00.000		10	4	Nail	
46	00006	MATERIAL	2017-01-05 00:00:00.000		10	5	Nail	
56	00006	MATERIAL	2017-01-06 00:00:00.000		10	6	Nail	
66	00006	MATERIAL	2017-01-07 00:00:00.000		10	7	Nail	
76	00006	MATERIAL	2017-01-08 00:00:00.000		10	8	Nail	
86	00006	MATERIAL	2017-01-09 00:00:00.000		10	9	Nail	
96	00006	MATERIAL	2017-01-10 00:00:00.000		10	10	Nail	
106	00006	MATERIAL	2017-01-11 00:00:00.000		10	11	Nail	

รูปที่ 3.11 ตัวอย่างข้อมูลประเภท MATERIAL

จากรูปที่ 3.9 ถึง รูปที่ 3.11 เป็นตัวอย่างการบันทึกข้อมูลของสินค้าแต่ละชนิด โดยทำการบันทึกสินค้าแต่ละชนิดทั้งหมด 30 วัน วันละ 10 ชิ้น รวมเป็น 300 ชิ้น

3.4.2 แบบเข้าก่อนออกก่อน

การทดสอบการทำงานแบบเข้าก่อนออกก่อน จะใช้สินค้าทั้งหมด 2 ชนิด คือ รหัสสินค้า 00005 Ply Wood และ 00006 Nail ซึ่งเป็นสินค้าที่มีประเภทเป็น Material

3.4.2.1 ตัดยอดสินค้าไม่หมดใน 1 วัน

ตารางที่ 3.1 ตัดยอดแบบไม่หมดใน 1 วัน สำหรับ Ply Wood

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
1/3/2560			Ply Wood = 300 (Ply Wood 1 - Ply Wood 30 = 10)
3/3/2560		Ply Wood = 6	Ply Wood = 294 (Ply Wood 1 = 4 Ply Wood 2 - Ply Wood 30 = 10)
5/3/2560		Ply Wood = 3	Ply Wood = 291 (Ply Wood 1 = 1 Ply Wood 2 - Ply Wood 30 = 10)
10/3/2560		Ply Wood = 1	Ply Wood = 290 (Ply Wood 1 = 0 Ply Wood 2 - Ply Wood 30 = 10)
13/3/2560		Ply Wood = 5	Ply Wood = 285 (Ply Wood 1 = 0 Ply Wood 2 = 5 Ply Wood 3 - Ply Wood 30 = 10)

จากตารางที่ 3.1 เป็นตารางแสดงการตัดยอดแบบเข้าก่อนออกก่อน โดยตัดไม่หมดใน 1 วัน หรือหมดใน 1 วันพอดี ของสินค้ารหัส 00005 Ply Wood อธิบายได้ดังนี้

วันที่ 1/3/2560 มี Ply Wood อยู่ในคลัง 300 ชิ้น โดยแบ่งเป็น Ply Wood ที่นำเข้าติดต่อกัน 30 วัน ตั้งแต่ Index 1-30 (Ply Wood 1 – Ply Wood 30) Index ละ 10 ชิ้น

วันที่ 3/3/2560 ขาย Ply Wood ได้ 6 ชิ้น เหลือในคลัง 294 โดย 6 ชิ้นที่ขายออกไป จะเป็นสินค้าใน Index ที่ 1 ทำให้ Ply Wood 1 เหลือ 4 ชิ้น

วันที่ 5/3/2560 ขาย Ply Wood ได้ 3 ชิ้น เหลือในคลัง 291 โดย 3 ชิ้นที่ขายออกไป จะเป็นสินค้าใน Index ที่ 1 ทำให้ Ply Wood 1 เหลือ 1 ชิ้น

วันที่ 10/3/2560 ขาย Ply Wood ได้ 1 ชิ้น เหลือในคลัง 290 โดย 1 ชิ้นที่ขายออกไป จะเป็นสินค้าใน Index ที่ 1 ทำให้ Ply Wood 1 เหลือ 0 ชิ้น

วันที่ 13/3/2560 ขาย Ply Wood ได้ 5 ชิ้น เหลือในคลัง 285 โดย 5 ชิ้นที่ขายออกไป จะเป็นสินค้าใน Index ที่ 2 ทำให้ Ply Wood 2 เหลือ 5 ชิ้น

ตารางที่ 3.2 ตัดยอดแบบไม่หมดใน 1 วัน สำหรับ Nail

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
1/3/2560			Nail = 300 (Nail 1 - Nail 30 = 10)
3/3/2560		Nail = 6	Nail = 294 (Nail 1 = 4 Nail 2 - Nail 30 = 10)
5/3/2560		Nail = 3	Nail = 291 (Nail 1 = 1 Nail 2 - Nail 30 = 10)
10/3/2560		Nail = 1	Nail = 290 (Nail 1 = 0 Nail 2 - Nail 30 = 10)
13/3/2560		Nail = 5	Nail = 285 (Nail 1 = 0 Nail 2 = 5 Nail 3 - Nail 30 = 10)

จากตารางที่ 3.2 เป็นตารางแสดงการตัดยอดแบบเข้าก่อนออกก่อน โดยตัดไม่หมดใน 1 วัน หรือหมดใน 1 วันพอดี ของสินค้ารหัส 00006 Nail อธิบายได้ดังนี้

วันที่ 1/3/2560 มี Nail อยู่ในคลัง 300 ชิ้น โดยแบ่งเป็น Nail ที่นำเข้าติดต่อกัน 30 วัน ตั้งแต่ Index 1-30 (Nail 1 – Nail 30) Index ละ 10 ชิ้น

วันที่ 3/3/2560 ขาย Nail ได้ 6 ชิ้น เหลือในคลัง 294 โดย 6 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Nail 1 เหลือ 4 ชิ้น

วันที่ 5/3/2560 ขาย Nail ได้ 3 ชิ้น เหลือในคลัง 291 โดย 3 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Nail 1 เหลือ 1 ชิ้น

วันที่ 10/3/2560 ขาย Nail ได้ 1 ชิ้น เหลือในคลัง 290 โดย 1 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Nail 1 เหลือ 0 ชิ้น

วันที่ 13/3/2560 ขาย Nail ได้ 5 ชิ้น เหลือในคลัง 285 โดย 5 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 2 ทำให้ Nail 2 เหลือ 5 ชิ้น

3.4.2.2 ตัดยอดสินค้าเกิน 1 วัน

ตารางที่ 3.3 ตัดยอดแบบเกิน 1 วัน สำหรับ Ply Wood

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
14/3/2560			Ply Wood = 285 (Ply Wood 1 = 0 Ply Wood 2 = 5 Ply Wood 3 - Ply Wood 30 = 10)
16/3/2560		Ply Wood = 6	Ply Wood = 279 (Ply Wood 1 - 2= 0 Ply Wood 3 = 9 Ply Wood 4 - Ply Wood 30 = 10)
18/3/2560		Ply Wood = 10	Ply Wood = 269 (Ply Wood 1 - 3= 0 Ply Wood 4 = 9 Ply Wood 5 - Ply Wood 30 = 10)

ตารางที่ 3.4 ตัดยอดแบบเกิน 1 วัน สำหรับ Ply Wood (ต่อ)

20/3/2560		Ply Wood = 15	Ply Wood = 254 (Ply Wood 1 - 4= 0 Ply Wood 5 = 4 Ply Wood 6 - Ply Wood 30 = 10)

จากตารางที่ 3.3 เป็นตารางแสดงการตัดยอดแบบเข้าก่อนออกก่อน โดยตัดเกิน 1 วัน หรือหมดข้ามวันที่นำเข้า ของสินค้ารหัส 00005 Ply Wood อธิบายได้ดังนี้

วันที่ 14/3/2560 มี Ply Wood อยู่ในคลัง 285 ชิ้น โดยแบ่งเป็น Ply Wood ตั้งแต่ Index 1 เหลือ 0 Index 2 เหลือ 5 ตั้งแต่ 3-30 เหลือ 10 ชิ้น

วันที่ 16/3/2560 ขาย Ply Wood ได้ 6 ชิ้น เหลือในคลัง 279 โดย 6 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 2 ทั้งหมด 5 ชิ้น และ Index 3 ทั้งหมด 1 ชิ้น ทำให้ Ply Wood 2 เหลือ 0 ชิ้น และ Ply Wood 3 เหลือ 9 ชิ้น

วันที่ 18/3/2560 ขาย Ply Wood ได้ 10 ชิ้น เหลือในคลัง 269 โดย 10 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 3 ทั้งหมด 9 ชิ้น และ Index 4 ทั้งหมด 1 ชิ้น ทำให้ Ply Wood 3 เหลือ 0 ชิ้น และ Ply Wood 4 เหลือ 9 ชิ้น

วันที่ 20/3/2560 ขาย Ply Wood ได้ 15 ชิ้น เหลือในคลัง 254 โดย 15 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 4 ทั้งหมด 9 ชิ้น และ Index 5 ทั้งหมด 6 ชิ้น ทำให้ Ply Wood 4 เหลือ 0 ชิ้น และ Ply Wood 5 เหลือ 4 ชิ้น

ตารางที่ 3.5 ตัดยอดแบบเกิน 1 วัน สำหรับ Nail

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
14/3/2560			Nail = 285 (Nail 1 = 0 Nail 2 = 5 Nail 3 - Nail 30 = 10)

ตารางที่ 3.6 ตัดยอดแบบเกิน 1 วัน สำหรับ Nail (ต่อ)

16/3/2560		Nail = 6	Nail = 279 (Nail 1 - 2= 0 Nail 3 = 9 Nail 4 - Nail 30 = 10)
18/3/2560		Nail = 10	Nail = 269 (Nail 1 - 3= 0 Nail 4 = 9 Nail 5 - Nail 30 = 10)
20/3/2560		Nail = 15	Nail = 254 (Nail 1 - 4= 0 Nail 5 = 4 Nail 6 - Nail 30 = 10)

จากตารางที่ 3.3 เป็นตารางแสดงการตัดยอดแบบเข้าก่อนออกก่อน โดยตัดเกิน 1 วัน หรือหมดข้ามวันที่นำเข้า ของสินค้ารหัส 00006 Nail อธิบายได้ดังนี้

วันที่ 14/3/2560 มี Nail อยู่ในคลัง 285 ชิ้น โดยแบ่งเป็น Nail ตั้งแต่ Index 1 เหลือ 0 Index 2 เหลือ 5 ตั้งแต่ 3-30 เหลือ 10 ชิ้น

วันที่ 16/3/2560 ขาย Nail ได้ 6 ชิ้น เหลือในคลัง 279 โดย 6 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 2 ทั้งหมด 5 ชิ้น และ Index 3 ทั้งหมด 1 ชิ้น ทำให้ Nail 2 เหลือ 0 ชิ้น และ Nail 3 เหลือ 9 ชิ้น

วันที่ 18/3/2560 ขาย Nail ได้ 10 ชิ้น เหลือในคลัง 269 โดย 10 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 3 ทั้งหมด 9 ชิ้น และ Index 4 ทั้งหมด 1 ชิ้น ทำให้ Nail 3 เหลือ 0 ชิ้น และ Nail 4 เหลือ 9 ชิ้น

วันที่ 20/3/2560 ขาย Nail ได้ 15 ชิ้น เหลือในคลัง 254 โดย 15 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 4 ทั้งหมด 9 ชิ้น และ Index 5 ทั้งหมด 6 ชิ้น ทำให้ Nail 4 เหลือ 0 ชิ้น และ Nail 5 เหลือ 4 ชิ้น

3.4.3 แบบเข้าหลังออกก่อน

การทดสอบการทำงานแบบเข้าหลังออกก่อน จะใช้สินค้าทั้งหมด 2 ชนิด คือ รหัสสินค้า 00003 iPhone และ 00010 Lenovo ซึ่งเป็นสินค้าที่มีประเภทเป็น Electronic

3.4.3.1 ตัดยอดสินค้าไม่หมดใน 1 วัน

ตารางที่ 3.7 ตัดยอดแบบไม่หมดใน 1 วัน สำหรับ iPhone

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
1/3/2560			iPhone = 300 (iPhone 1 - iPhone 30 = 10)
3/3/2560		iPhone = 6	iPhone = 294 (iPhone 1 = 4 iPhone 2 - iPhone 30 = 10)
5/3/2560		iPhone = 3	iPhone = 291 (iPhone 1 = 1 iPhone 2 - iPhone 30 = 10)
10/3/2560		iPhone = 1	iPhone = 290 (iPhone 1 = 0 iPhone 2 - iPhone 30 = 10)
13/3/2560		iPhone = 5	iPhone = 285 (iPhone 1 = 0 iPhone 2 = 5 iPhone 3 - iPhone 30 = 10)

จากตารางที่ 3.7 เป็นตารางแสดงการตัดยอดแบบเข้าหลังออกก่อน โดยตัดไม่หมดใน 1 วัน หรือหมดใน 1 วันพอดี ของสินค้ารหัส 00003 iPhone อธิบายได้ดังนี้

วันที่ 1/3/2560 มี iPhone อยู่ในคลัง 300 ชิ้น โดยแบ่งเป็น iPhone ที่นำเข้าติดต่อกัน 30 วัน ตั้งแต่ Index 1-30 (iPhone 1 – iPhone 30) Index ละ 10 ชิ้น

วันที่ 3/3/2560 ขาย iPhone ได้ 6 ชิ้น เหลือในคลัง 294 โดย 6 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ iPhone 1 เหลือ 4 ชิ้น

วันที่ 5/3/2560 ขาย iPhone ได้ 3 ชิ้น เหลือในคลัง 291 โดย 3 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ iPhone 1 เหลือ 1 ชิ้น

วันที่ 10/3/2560 ขาย iPhone ได้ 1 ชิ้น เหลือในคลัง 290 โดย 1 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ iPhone 1 เหลือ 0 ชิ้น

วันที่ 13/3/2560 ขาย iPhone ได้ 5 ชิ้น เหลือในคลัง 285 โดย 5 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 2 ทำให้ iPhone 2 เหลือ 5 ชิ้น

ตารางที่ 3.8 ตัดยอดแบบไม่หมดใน 1 วัน สำหรับ Lenovo

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
1/3/2560			Lenovo = 300 (Lenovo 1 - Lenovo 30 = 10)
3/3/2560		Lenovo = 6	Lenovo = 294 (Lenovo 1 = 4 Lenovo 2 - Lenovo 30 = 10)
5/3/2560		Lenovo = 3	Lenovo = 291 (Lenovo 1 = 1 Lenovo 2 - Lenovo 30 = 10)
10/3/2560		Lenovo = 1	Lenovo = 290 (Lenovo 1 = 0 Lenovo 2 - Lenovo 30 = 10)
13/3/2560		Lenovo = 5	Lenovo = 285 (Lenovo 1 = 0 Lenovo 2 = 5 Lenovo 3 - Lenovo 30 = 10)

จากตารางที่ 3.8 เป็นตารางแสดงการตัดยอดแบบเข้าหลังออกก่อน โดยตัดไม่หมดใน 1 วัน หรือหมดใน 1 วันพอดี ของสินค้ารหัส 00010 Lenovo อธิบายได้ดังนี้

วันที่ 1/3/2560 มี Lenovo อยู่ในคลัง 300 ชิ้น โดยแบ่งเป็น Lenovo ที่นำเข้าติดต่อกัน 30 วัน ตั้งแต่ Index 1-30 (Lenovo 1 – Lenovo 30) Index ละ 10 ชิ้น

วันที่ 3/3/2560 ขาย Lenovo ได้ 6 ชิ้น เหลือในคลัง 294 โดย 6 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Lenovo 1 เหลือ 4 ชิ้น

วันที่ 5/3/2560 ขาย Lenovo ได้ 3 ชิ้น เหลือในคลัง 291 โดย 3 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Lenovo 1 เหลือ 1 ชิ้น

วันที่ 10/3/2560 ขาย Lenovo ได้ 1 ชิ้น เหลือในคลัง 290 โดย 1 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Lenovo 1 เหลือ 0 ชิ้น

วันที่ 13/3/2560 ขาย Lenovo ได้ 5 ชิ้น เหลือในคลัง 285 โดย 5 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 2 ทำให้ Lenovo 2 เหลือ 5 ชิ้น

3.4.3.2 ตัดยอดสินค้าเกิน 1 วัน

ตารางที่ 3.9 ตัดยอดแบบเกิน 1 วัน สำหรับ iPhone

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
14/3/2560			iPhone = 285 (iPhone 1 = 0 iPhone 2 = 5 iPhone 3 - iPhone 30 = 10)
16/3/2560		iPhone = 6	iPhone = 279 (iPhone 1 - 2= 0 iPhone 3 = 9 iPhone 4 - iPhone 30 = 10)

ตารางที่ 3.10 ตัดยอดแบบเกิน 1 วัน สำหรับ iPhone (ต่อ)

18/3/2560		iPhone = 10	iPhone = 269 (iPhone 1 - 3= 0 iPhone 4 = 9 iPhone 5 - iPhone 30 = 10)
20/3/2560		iPhone = 15	iPhone = 254 (iPhone 1 - 4= 0 iPhone 5 = 4 iPhone 6 - iPhone 30 = 10)

จากตารางที่ 3.9 เป็นตารางแสดงการตัดยอดแบบเข้าหลังออกก่อน โดยตัดเกิน 1 วัน หรือหมดข้ามวันที่นำเข้า ของสินค้ารหัส 00003 iPhone อธิบายได้ดังนี้

วันที่ 14/3/2560 มี iPhone อยู่ในคลัง 285 ชิ้น โดยแบ่งเป็น iPhone ตั้งแต่ Index 1 เหลือ 0 Index 2 เหลือ 5 ตั้งแต่ 3-30 เหลือ 10 ชิ้น

วันที่ 16/3/2560 ขาย iPhone ได้ 6 ชิ้น เหลือในคลัง 279 โดย 6 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 2 ทั้งหมด 5 ชิ้น และ Index 3 ทั้งหมด 1 ชิ้น ทำให้ iPhone 2 เหลือ 0 ชิ้น และ iPhone 3 เหลือ 9 ชิ้น

วันที่ 18/3/2560 ขาย iPhone ได้ 10 ชิ้น เหลือในคลัง 269 โดย 10 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 3 ทั้งหมด 9 ชิ้น และ Index 4 ทั้งหมด 1 ชิ้น ทำให้ iPhone 3 เหลือ 0 ชิ้น และ iPhone 4 เหลือ 9 ชิ้น

วันที่ 20/3/2560 ขาย iPhone ได้ 15 ชิ้น เหลือในคลัง 254 โดย 15 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 4 ทั้งหมด 9 ชิ้น และ Index 5 ทั้งหมด 6 ชิ้น ทำให้ iPhone 4 เหลือ 0 ชิ้น และ iPhone 5 เหลือ 4 ชิ้น

ตารางที่ 3.11 ตัดยอดแบบเกิน 1 วัน สำหรับ Lenovo

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
14/3/2560			Lenovo = 285 (Lenovo 1 = 0 Lenovo 2 = 5 Lenovo 3 - Lenovo 30 = 10)
16/3/2560		Lenovo = 6	Lenovo = 279 (Lenovo 1 - 2= 0 Lenovo 3 = 9 Lenovo 4 - Lenovo 30 = 10)
18/3/2560		Lenovo = 10	Lenovo = 269 (Lenovo 1 - 3= 0 Lenovo 4 = 9 Lenovo 5 - Lenovo 30 = 10)
20/3/2560		Lenovo = 15	Lenovo = 254 (Lenovo 1 - 4= 0 Lenovo 5 = 4 Lenovo 6 - Lenovo 30 = 10)

จากตารางที่ 3.11 เป็นตารางแสดงการตัดยอดแบบเข้าก่อนออกก่อน โดยตัดเกิน 1 วัน หรือหมดข้ามวันที่นำเข้า ของสินค้ารหัส 00010 Lenovo อธิบายได้ดังนี้

วันที่ 14/3/2560 มี Lenovo อยู่ในคลัง 285 ชิ้น โดยแบ่งเป็น Lenovo ตั้งแต่ Index 1 เหลือ 0 Index 2 เหลือ 5 ตั้งแต่ 3-30 เหลือ 10 ชิ้น

วันที่ 16/3/2560 ขาย Lenovo ได้ 6 ชิ้น เหลือในคลัง 279 โดย 6 ชิ้นที่ขายออกไป จะเป็นสินค้าใน Index 2 ทั้งหมด 5 ชิ้น และ Index 3 ทั้งหมด 1 ชิ้น ทำให้ Lenovo 2 เหลือ 0 ชิ้น และ Lenovo 3 เหลือ 9 ชิ้น

วันที่ 18/3/2560 ขาย Lenovo ได้ 10 ชิ้น เหลือในคลัง 269 โดย 10 ชิ้นที่ขายออกไป จะเป็นสินค้าใน Index 3 ทั้งหมด 9 ชิ้น และ Index 4 ทั้งหมด 1 ชิ้น ทำให้ Lenovo 3 เหลือ 0 ชิ้น และ Lenovo 4 เหลือ 9 ชิ้น

วันที่ 20/3/2560 ขาย Lenovo ได้ 15 ชิ้น เหลือในคลัง 254 โดย 15 ชิ้นที่ขายออกไป จะเป็นสินค้าใน Index 4 ทั้งหมด 9 ชิ้น และ Index 5 ทั้งหมด 6 ชิ้น ทำให้ Lenovo 4 เหลือ 0 ชิ้น และ Lenovo 5 เหลือ 4 ชิ้น

3.4.4 แบบหมดอายุก่อนออกก่อน

การทดสอบการทำงานแบบหมดอายุก่อนออกก่อน จะใช้สินค้าทั้งหมด 2 ชนิด คือ รหัสสินค้า 00001 Salt และ 00002 Garlic ซึ่งเป็นสินค้าที่มีประเภทเป็น Food

3.4.4.1 ตัดยอดสินค้าไม่หมดใน 1 วัน

ตารางที่ 3.12 ตัดยอดแบบไม่หมดใน 1 วัน สำหรับ Salt

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
1/3/2560			Salt = 300 (Salt 1 - Salt 30 = 10)
3/3/2560		Salt = 6	Salt = 294 (Salt 1 = 4 Salt 2 - Salt 30 = 10)
5/3/2560		Salt = 3	Salt = 291 (Salt 1 = 1 Salt 2 - Salt 30 = 10)
10/3/2560		Salt = 1	Salt = 290 (Salt 1 = 0 Salt 2 - Salt 29 = 10)
13/3/2560		Salt = 5	Salt = 285 (Salt 1 = 0 Salt 2 = 5 Salt 3 - Salt 30 = 10)

จากตารางที่ 3.12 เป็นตารางแสดงการตัดยอดแบบหมดอายุออกก่อน โดยตัดไม่หมดใน 1 วัน หรือหมดใน 1 วันพอดี ของสินค้ารหัส 00001 Salt อธิบายได้ดังนี้

วันที่ 1/3/2560 มี Salt อยู่ในคลัง 300 ชิ้น โดยแบ่งเป็น Salt ที่นำเข้าติดต่อกัน 30 วัน ตั้งแต่ Index 1-30 (Salt 1 – Salt 30) Index ละ 10 ชิ้น

วันที่ 3/3/2560 ขาย Salt ได้ 6 ชิ้น เหลือในคลัง 294 โดย 6 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Salt 1 เหลือ 4 ชิ้น

วันที่ 5/3/2560 ขาย Salt ได้ 3 ชิ้น เหลือในคลัง 291 โดย 3 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Salt 1 เหลือ 1 ชิ้น

วันที่ 10/3/2560 ขาย Salt ได้ 1 ชิ้น เหลือในคลัง 290 โดย 1 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Salt 1 เหลือ 0 ชิ้น

วันที่ 13/3/2560 ขาย Salt ได้ 5 ชิ้น เหลือในคลัง 285 โดย 5 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 2 ทำให้ Salt 2 เหลือ 5 ชิ้น

ตารางที่ 3.13 ตัดยอดแบบไม่หมดใน 1 วัน สำหรับ Garlic

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
1/3/2560			Garlic = 300 (Garlic 1 - Garlic 30 = 10)
3/3/2560		Garlic = 6	Garlic = 294 (Garlic 1 = 4 Garlic 2 - Garlic 30 = 10)
5/3/2560		Garlic = 3	Garlic = 291 (Garlic 1 = 1 Garlic 2 - Garlic 30 = 10)
10/3/2560		Garlic = 1	Garlic = 290 (Garlic 1 = 0 Garlic 2 - Garlic 30 = 10)

ตารางที่ 3.14 ตัดยอดแบบไม่หมดใน 1 วัน สำหรับ Garlic (ต่อ)

13/3/2560		Garlic = 5	Garlic = 285 (Garlic 1 = 0 Garlic 2 = 5 Garlic 3 - Garlic 30 = 10)

จากตารางที่ 3.13 เป็นตารางแสดงการตัดยอดแบบหมดอายุก่อนออกก่อน โดยตัดไม่หมดใน 1 วัน หรือหมดใน 1 วันพอดี ของสินค้ารหัส 00002 Garlic อธิบายได้ดังนี้

วันที่ 1/3/2560 มี Garlic อยู่ในคลัง 300 ชิ้น โดยแบ่งเป็น Garlic ที่นำเข้าติดต่อกัน 30 วัน ตั้งแต่ Index 1-30 (Garlic 1 – Garlic 30) Index ละ 10 ชิ้น

วันที่ 3/3/2560 ขาย Garlic ได้ 6 ชิ้น เหลือในคลัง 294 โดย 6 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Garlic 1 เหลือ 4 ชิ้น

วันที่ 5/3/2560 ขาย Garlic ได้ 3 ชิ้น เหลือในคลัง 291 โดย 3 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Garlic 1 เหลือ 1 ชิ้น

วันที่ 10/3/2560 ขาย Garlic ได้ 1 ชิ้น เหลือในคลัง 290 โดย 1 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 1 ทำให้ Garlic 1 เหลือ 0 ชิ้น

วันที่ 13/3/2560 ขาย Garlic ได้ 5 ชิ้น เหลือในคลัง 285 โดย 5 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index ที่ 2 ทำให้ Garlic 2 เหลือ 5 ชิ้น

3.4.4.2 ตัดยอดสินค้าเกิน 1 วัน

ตารางที่ 3.15 ตัดยอดแบบเกิน 1 วัน สำหรับ Salt

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
14/3/2560			Salt = 285 (Salt 1 = 0 Salt 2 = 5 Salt 3 - Salt 30 = 10)
16/3/2560		Salt = 6	Salt = 279 (Salt 1 - 2= 0 Salt 3 = 9 Salt 4 - Salt 30 = 10)
18/3/2560		Salt = 10	Salt = 269 (Salt 1 - 3= 0 Salt 4 = 9 Salt 5 - Salt 30 = 10)
20/3/2560		Salt = 15	Salt = 254 (Salt 1 - 4= 0 Salt 5 = 4 Salt 6 - Salt 30 = 10)

จากตารางที่ 3.15 เป็นตารางแสดงการตัดยอดแบบหมดอายุก่อนออกก่อน โดยตัดเกิน 1 วัน หรือหมดข้ามวันที่นำเข้า ของสินค้ารหัส 00001 Salt อธิบายได้ดังนี้

วันที่ 14/3/2560 มี Salt อยู่ในคลัง 285 ชิ้น โดยแบ่งเป็น Salt ตั้งแต่ Index 1 เหลือ 0 Index 2 เหลือ 5 ตั้งแต่ 3-30 เหลือ 10 ชิ้น

วันที่ 16/3/2560 ขาย Salt ได้ 6 ชิ้น เหลือในคลัง 279 โดย 6 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 2 ทั้งหมด 5 ชิ้น และ Index 3 ทั้งหมด 1 ชิ้น ทำให้ Salt 2 เหลือ 0 ชิ้น และ Salt 3 เหลือ 9 ชิ้น

วันที่ 18/3/2560 ขาย Salt ได้ 10 ชิ้น เหลือในคลัง 269 โดย 10 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 3 ทั้งหมด 9 ชิ้น และ Index 4 ทั้งหมด 1 ชิ้น ทำให้ Salt 3 เหลือ 0 ชิ้น และ Salt 4 เหลือ 9 ชิ้น

วันที่ 20/3/2560 ขาย Salt ได้ 15 ชิ้น เหลือในคลัง 254 โดย 15 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 4 ทั้งหมด 9 ชิ้น และ Index 5 ทั้งหมด 6 ชิ้น ทำให้ Salt 4 เหลือ 0 ชิ้น และ Salt 5 เหลือ 4 ชิ้น

ตารางที่ 3.16 ตัดยอดแบบเกิน 1 วัน สำหรับ Garlic

วันที่	ซื้อเข้า	ขายออก	คงเหลือ
14/3/2560			Garlic = 285 (Garlic 1 = 0 Garlic 2 = 5 Garlic 3 - Garlic 30 = 10)
16/3/2560		Garlic = 6	Garlic = 279 (Garlic 1 - 2= 0 Garlic 3 = 9 Garlic 4 - Garlic 30 = 10)
18/3/2560		Garlic = 10	Garlic = 269 (Garlic 1 - 3= 0 Garlic 4 = 9 Garlic 5 - Garlic 30 = 10)
20/3/2560		Garlic = 15	Garlic = 254 (Garlic 1 - 4= 0 Garlic 5 = 4 Garlic 6 - Garlic 30 = 10)

จากตารางที่ 3.17 เป็นตารางแสดงการตัดยอดแบบหมดอายุก่อนออกก่อน โดยตัดเกิน 1 วัน หรือหมดข้ามวันที่นำเข้า ของสินค้ารหัส 00002 Garlic อธิบายได้ดังนี้

วันที่ 14/3/2560 มี Garlic อยู่ในคลัง 285 ชิ้น โดยแบ่งเป็น Garlic ตั้งแต่ Index 1 เหลือ 0 Index 2 เหลือ 5 ตั้งแต่ 3-30 เหลือ 10 ชิ้น

วันที่ 16/3/2560 ขาย Garlic ได้ 6 ชิ้น เหลือในคลัง 279 โดย 6 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 2 ทั้งหมด 5 ชิ้น และ Index 3 ทั้งหมด 1 ชิ้น ทำให้ Garlic 2 เหลือ 0 ชิ้น และ Garlic 3 เหลือ 9 ชิ้น

วันที่ 18/3/2560 ขาย Garlic ได้ 10 ชิ้น เหลือในคลัง 269 โดย 10 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 3 ทั้งหมด 9 ชิ้น และ Index 4 ทั้งหมด 1 ชิ้น ทำให้ Garlic 3 เหลือ 0 ชิ้น และ Garlic 4 เหลือ 9 ชิ้น

วันที่ 20/3/2560 ขาย Garlic ได้ 15 ชิ้น เหลือในคลัง 254 โดย 15 ชิ้นที่ขายออกไปจะเป็นสินค้าใน Index 4 ทั้งหมด 9 ชิ้น และ Index 5 ทั้งหมด 6 ชิ้น ทำให้ Garlic 4 เหลือ 0 ชิ้น และ Garlic 5 เหลือ 4 ชิ้น

บทที่ 4
ผลการวิเคราะห์ข้อมูลและอภิปรายผล

4.1 บทนำ

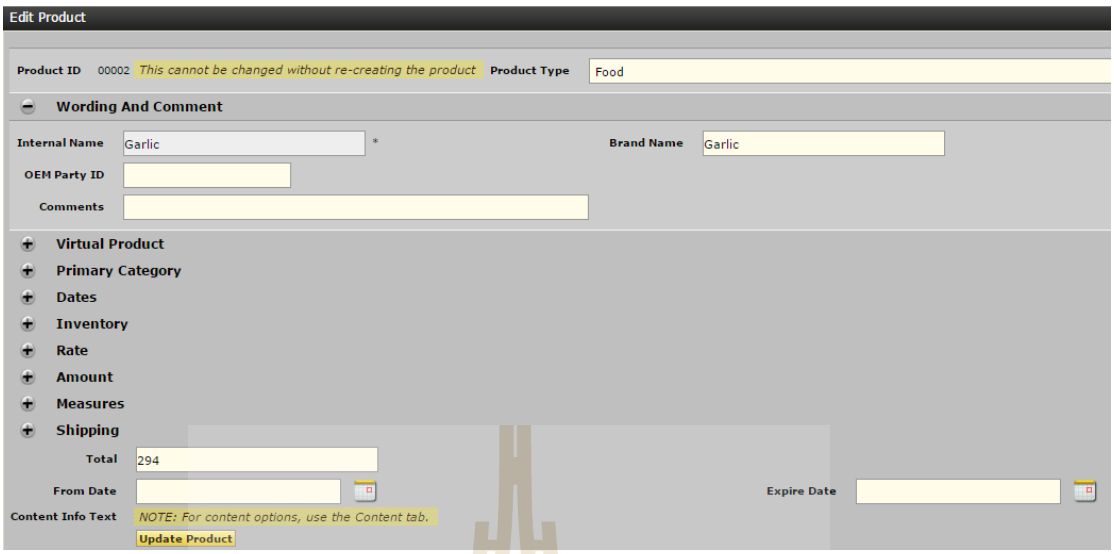
บทนี้จะนำเสนอผลของการทดลองหลังจากได้มีการพัฒนาโมดูลของการตัดยอดสินค้าเพิ่มเติมลงไปในกรอบงาน การเรียกใช้งานโมดูลพื้นฐานที่มีอยู่แล้ว เช่น การบันทึกข้อมูล การแก้ไขข้อมูล รวมไปถึงผลของการการเรียกใช้โมดูลการตัดยอดสินค้าที่ได้พัฒนาเพิ่มลงไปในกรอบงานด้วย

4.2 การเรียกใช้โมดูลการตัดยอดสินค้า

โมดูลการตัดยอดสินค้าเป็นโมดูลที่ได้ทำการพัฒนาขึ้นมาจึงต้องมีการทดสอบการทำงาน โดยหลังจากได้วางแผนการทดสอบไว้แล้ว ก็นำแผนที่วางไว้มาทำการทดสอบ

PRODUCT ID	PRODUCT TYPE ID	INTERNAL NAME	BRAND NAME	PRODUCT NAME	TOTAL
00001	Food	Salt	Salt		300
00002	Food	Garlic	Garlic		300
00003	Electronic	Mobile	iPhone		300
00004	Electronic	Mobile	Samsung		300
00005	Material	Ply Wood			300
00006	Material	Nail			300
00007	Material	Plaster			300
00008	Food	Noodle			300
00009	Food	Tomato			300
00010	Electronic	Notebook	Lenovo		300

รูปที่ 4.1 หน้าต่าง Product



รูปที่ 4.2 หน้าต่างการแก้ไขจำนวนตามรายการคงเหลือ

จากรูปที่ 4.1 เป็นหน้าต่าง Product ที่แสดงจำนวนคงเหลือทั้งหมดของสินค้าแต่ละชิ้น โดยจะทำการแก้ไขจำนวนคงเหลือทั้งหมดตามรายการที่นำออกตามแผนที่วางไว้ โดยหน้าต่างการแก้ไขจะเป็นดังรูปที่ 4.2

4.2.1 แบบเข้าก่อนออกก่อน

การทดสอบแบบเข้าก่อนออกก่อน ตามแผนที่วางไว้คือ ทดสอบแบบตัดไม่หมดใน 1 วัน และ แบบตัดเกินจำนวนของ 1 วัน โดยจำทดสอบกับสินค้ารหัส 00005 Ply Wood และ 00006 Nail ซึ่งได้ผลลัพธ์ของการทดสอบแบบตัดไม่หมดใน 1 วัน ดังรูปที่ 4.3 ถึง รูปที่ 4.12 และได้ผลลัพธ์ของการทดสอบแบบตัดเกินจำนวนของ 1 วัน ดังรูปที่ 4.13 ถึง รูปที่ 4.18

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
5	00005	MATERIAL	2017-01-01 00:00:00.000		10	1	Ply Wood
15	00005	MATERIAL	2017-01-02 00:00:00.000		10	2	Ply Wood
25	00005	MATERIAL	2017-01-03 00:00:00.000		10	3	Ply Wood
35	00005	MATERIAL	2017-01-04 00:00:00.000		10	4	Ply Wood
45	00005	MATERIAL	2017-01-05 00:00:00.000		10	5	Ply Wood
55	00005	MATERIAL	2017-01-06 00:00:00.000		10	6	Ply Wood
65	00005	MATERIAL	2017-01-07 00:00:00.000		10	7	Ply Wood
75	00005	MATERIAL	2017-01-08 00:00:00.000		10	8	Ply Wood
85	00005	MATERIAL	2017-01-09 00:00:00.000		10	9	Ply Wood
95	00005	MATERIAL	2017-01-10 00:00:00.000		10	10	Ply Wood
105	00005	MATERIAL	2017-01-11 00:00:00.000		10	11	Ply Wood
115	00005	MATERIAL	2017-01-12 00:00:00.000		10	12	Ply Wood

รูปที่ 4.3 ก่อนตัดยอด Ply Wood

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
5	00005	MATERIAL	2017-01-01 00:00:00.000		4	1	Ply Wood
15	00005	MATERIAL	2017-01-02 00:00:00.000		10	2	Ply Wood
25	00005	MATERIAL	2017-01-03 00:00:00.000		10	3	Ply Wood
35	00005	MATERIAL	2017-01-04 00:00:00.000		10	4	Ply Wood
45	00005	MATERIAL	2017-01-05 00:00:00.000		10	5	Ply Wood
55	00005	MATERIAL	2017-01-06 00:00:00.000		10	6	Ply Wood
65	00005	MATERIAL	2017-01-07 00:00:00.000		10	7	Ply Wood

รูปที่ 4.4 ตัดยอด Ply Wood ครั้งที่ 1

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
5	00005	MATERIAL	2017-01-01 00:00:00.000		1	1	Ply Wood
15	00005	MATERIAL	2017-01-02 00:00:00.000		10	2	Ply Wood
25	00005	MATERIAL	2017-01-03 00:00:00.000		10	3	Ply Wood
35	00005	MATERIAL	2017-01-04 00:00:00.000		10	4	Ply Wood
45	00005	MATERIAL	2017-01-05 00:00:00.000		10	5	Ply Wood
55	00005	MATERIAL	2017-01-06 00:00:00.000		10	6	Ply Wood
65	00005	MATERIAL	2017-01-07 00:00:00.000		10	7	Ply Wood

รูปที่ 4.5 ตัดยอด Ply Wood ครั้งที่ 2

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
5	00005	MATERIAL	2017-01-01 00:00:00.000		0	1	Ply Wood
15	00005	MATERIAL	2017-01-02 00:00:00.000		10	2	Ply Wood
25	00005	MATERIAL	2017-01-03 00:00:00.000		10	3	Ply Wood
35	00005	MATERIAL	2017-01-04 00:00:00.000		10	4	Ply Wood
45	00005	MATERIAL	2017-01-05 00:00:00.000		10	5	Ply Wood
55	00005	MATERIAL	2017-01-06 00:00:00.000		10	6	Ply Wood
65	00005	MATERIAL	2017-01-07 00:00:00.000		10	7	Ply Wood

รูปที่ 4.6 ตัดยอด Ply Wood ครั้งที่ 3

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
5	00005	MATERIAL	2017-01-01 00:00:00.000		0	1	Ply Wood
15	00005	MATERIAL	2017-01-02 00:00:00.000		5	2	Ply Wood
25	00005	MATERIAL	2017-01-03 00:00:00.000		10	3	Ply Wood
35	00005	MATERIAL	2017-01-04 00:00:00.000		10	4	Ply Wood
45	00005	MATERIAL	2017-01-05 00:00:00.000		10	5	Ply Wood
55	00005	MATERIAL	2017-01-06 00:00:00.000		10	6	Ply Wood
65	00005	MATERIAL	2017-01-07 00:00:00.000		10	7	Ply Wood

รูปที่ 4.7 ตัดยอด Ply Wood ครั้งที่ 4

จากรูปที่ 4.4 ถึง รูปที่ 4.7 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.1 โดยดึงสินค้า Ply Wood ออกตามลำดับ คือ ครั้งที่ 1 ดึงออก 6 ชิ้น ครั้งที่ 2 ดึงออก 3 ชิ้น ครั้งที่ 3 ดึงออก 1 ชิ้น และครั้งที่ 4 ดึงออก 5 ชิ้น

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
6	00006	MATERIAL	2017-01-01 00:00:00.000		10	1	Nail
16	00006	MATERIAL	2017-01-02 00:00:00.000		10	2	Nail
26	00006	MATERIAL	2017-01-03 00:00:00.000		10	3	Nail
36	00006	MATERIAL	2017-01-04 00:00:00.000		10	4	Nail
46	00006	MATERIAL	2017-01-05 00:00:00.000		10	5	Nail
56	00006	MATERIAL	2017-01-06 00:00:00.000		10	6	Nail
66	00006	MATERIAL	2017-01-07 00:00:00.000		10	7	Nail
76	00006	MATERIAL	2017-01-08 00:00:00.000		10	8	Nail
86	00006	MATERIAL	2017-01-09 00:00:00.000		10	9	Nail
96	00006	MATERIAL	2017-01-10 00:00:00.000		10	10	Nail
106	00006	MATERIAL	2017-01-11 00:00:00.000		10	11	Nail
116	00006	MATERIAL	2017-01-12 00:00:00.000		10	12	Nail

รูปที่ 4.8 ก่อนตัดยอด Nail

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
6	00006	MATERIAL	2017-01-01 00:00:00.000		4	1	Nail
16	00006	MATERIAL	2017-01-02 00:00:00.000		10	2	Nail
26	00006	MATERIAL	2017-01-03 00:00:00.000		10	3	Nail
36	00006	MATERIAL	2017-01-04 00:00:00.000		10	4	Nail
46	00006	MATERIAL	2017-01-05 00:00:00.000		10	5	Nail
56	00006	MATERIAL	2017-01-06 00:00:00.000		10	6	Nail
66	00006	MATERIAL	2017-01-07 00:00:00.000		10	7	Nail

รูปที่ 4.9 ตัดยอด Nail ครั้งที่ 1

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
6	00006	MATERIAL	2017-01-01 00:00:00.000		1	1	Nail
16	00006	MATERIAL	2017-01-02 00:00:00.000		10	2	Nail
26	00006	MATERIAL	2017-01-03 00:00:00.000		10	3	Nail
36	00006	MATERIAL	2017-01-04 00:00:00.000		10	4	Nail
46	00006	MATERIAL	2017-01-05 00:00:00.000		10	5	Nail
56	00006	MATERIAL	2017-01-06 00:00:00.000		10	6	Nail
66	00006	MATERIAL	2017-01-07 00:00:00.000		10	7	Nail

รูปที่ 4.10 ตัดยอด Nail ครั้งที่ 2

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
6	00006	MATERIAL	2017-01-01 00:00:00.000		0	1	Nail
16	00006	MATERIAL	2017-01-02 00:00:00.000		10	2	Nail
26	00006	MATERIAL	2017-01-03 00:00:00.000		10	3	Nail
36	00006	MATERIAL	2017-01-04 00:00:00.000		10	4	Nail
46	00006	MATERIAL	2017-01-05 00:00:00.000		10	5	Nail
56	00006	MATERIAL	2017-01-06 00:00:00.000		10	6	Nail
66	00006	MATERIAL	2017-01-07 00:00:00.000		10	7	Nail

รูปที่ 4.11 ตัดยอด Nail ครั้งที่ 3

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
6	00006	MATERIAL	2017-01-01 00:00:00.000		0	1	Nail
16	00006	MATERIAL	2017-01-02 00:00:00.000		5	2	Nail
26	00006	MATERIAL	2017-01-03 00:00:00.000		10	3	Nail
36	00006	MATERIAL	2017-01-04 00:00:00.000		10	4	Nail
46	00006	MATERIAL	2017-01-05 00:00:00.000		10	5	Nail
56	00006	MATERIAL	2017-01-06 00:00:00.000		10	6	Nail
66	00006	MATERIAL	2017-01-07 00:00:00.000		10	7	Nail

รูปที่ 4.12 ตัดยอด Nail ครั้งที่ 4

จากรูปที่ 4.9 ถึง รูปที่ 4.12 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.2 โดยดึงสินค้า Nail ออกตามลำดับ คือ ครั้งที่ 1 ดึงออก 6 ชิ้น ครั้งที่ 2 ดึงออก 3 ชิ้น ครั้งที่ 3 ดึงออก 1 ชิ้น และครั้งที่ 4 ดึงออก 5 ชิ้น

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
5	00005	MATERIAL	2017-01-01 00:00:00.000		0	1	Ply Wood
15	00005	MATERIAL	2017-01-02 00:00:00.000		0	2	Ply Wood
25	00005	MATERIAL	2017-01-03 00:00:00.000		9	3	Ply Wood
35	00005	MATERIAL	2017-01-04 00:00:00.000		10	4	Ply Wood
45	00005	MATERIAL	2017-01-05 00:00:00.000		10	5	Ply Wood
55	00005	MATERIAL	2017-01-06 00:00:00.000		10	6	Ply Wood
65	00005	MATERIAL	2017-01-07 00:00:00.000		10	7	Ply Wood

รูปที่ 4.13 ตัดยอด Ply Wood ครั้งที่ 5

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
5	00005	MATERIAL	2017-01-01 00:00:00.000		0	1	Ply Wood
15	00005	MATERIAL	2017-01-02 00:00:00.000		0	2	Ply Wood
25	00005	MATERIAL	2017-01-03 00:00:00.000		0	3	Ply Wood
35	00005	MATERIAL	2017-01-04 00:00:00.000		9	4	Ply Wood
45	00005	MATERIAL	2017-01-05 00:00:00.000		10	5	Ply Wood
55	00005	MATERIAL	2017-01-06 00:00:00.000		10	6	Ply Wood
65	00005	MATERIAL	2017-01-07 00:00:00.000		10	7	Ply Wood

รูปที่ 4.14 ตัดยอด Ply Wood ครั้งที่ 6

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
5	00005	MATERIAL	2017-01-01 00:00:00.000		0	1	Ply Wood
15	00005	MATERIAL	2017-01-02 00:00:00.000		0	2	Ply Wood
25	00005	MATERIAL	2017-01-03 00:00:00.000		0	3	Ply Wood
35	00005	MATERIAL	2017-01-04 00:00:00.000		0	4	Ply Wood
45	00005	MATERIAL	2017-01-05 00:00:00.000		4	5	Ply Wood
55	00005	MATERIAL	2017-01-06 00:00:00.000		10	6	Ply Wood
65	00005	MATERIAL	2017-01-07 00:00:00.000		10	7	Ply Wood

รูปที่ 4.15 ตัดยอด Ply Wood ครั้งที่ 7

จากรูปที่ 4.13 ถึง รูปที่ 4.15 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.3 โดยดึงสินค้า Ply Wood ออกตามลำดับ คือ ครั้งที่ 5 ดึงออก 6 ชิ้น ครั้งที่ 6 ดึงออก 10 ชิ้น ครั้งที่ 7 ดึงออก 15 ชิ้น

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
6	00006	MATERIAL	2017-01-01 00:00:00.000		0	1	Nail
16	00006	MATERIAL	2017-01-02 00:00:00.000		0	2	Nail
26	00006	MATERIAL	2017-01-03 00:00:00.000		9	3	Nail
36	00006	MATERIAL	2017-01-04 00:00:00.000		10	4	Nail
46	00006	MATERIAL	2017-01-05 00:00:00.000		10	5	Nail
56	00006	MATERIAL	2017-01-06 00:00:00.000		10	6	Nail
66	00006	MATERIAL	2017-01-07 00:00:00.000		10	7	Nail

รูปที่ 4.16 ตัดยอด Nail ครั้งที่ 5

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
6	00006	MATERIAL	2017-01-01 00:00:00.000		0	1	Nail
16	00006	MATERIAL	2017-01-02 00:00:00.000		0	2	Nail
26	00006	MATERIAL	2017-01-03 00:00:00.000		0	3	Nail
36	00006	MATERIAL	2017-01-04 00:00:00.000		9	4	Nail
46	00006	MATERIAL	2017-01-05 00:00:00.000		10	5	Nail
56	00006	MATERIAL	2017-01-06 00:00:00.000		10	6	Nail
66	00006	MATERIAL	2017-01-07 00:00:00.000		10	7	Nail

รูปที่ 4.17 ตัดยอด Nail ครั้งที่ 6

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME
6	00006	MATERIAL	2017-01-01 00:00:00.000		0	1	Nail
16	00006	MATERIAL	2017-01-02 00:00:00.000		0	2	Nail
26	00006	MATERIAL	2017-01-03 00:00:00.000		0	3	Nail
36	00006	MATERIAL	2017-01-04 00:00:00.000		0	4	Nail
46	00006	MATERIAL	2017-01-05 00:00:00.000		4	5	Nail
56	00006	MATERIAL	2017-01-06 00:00:00.000		10	6	Nail
66	00006	MATERIAL	2017-01-07 00:00:00.000		10	7	Nail

รูปที่ 4.18 ตัดยอด Nail ครั้งที่ 7

จากรูปที่ 4.16 ถึง รูปที่ 4.18 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.4 โดยดึงสินค้า Ply Wood ออกตามลำดับ คือ ครั้งที่ 5 ดึงออก 6 ชิ้น ครั้งที่ 6 ดึงออก 10 ชิ้น ครั้งที่ 7 ดึงออก 15 ชิ้น

4.2.2 แบบเข้าหลังออกก่อน

การทดสอบแบบเข้าหลังออกก่อน ตามแผนที่วางไว้คือ ทดสอบแบบตัดไม่หมดใน 1 วัน และ แบบตัดเกินจำนวนของ 1 วัน โดยจำทดสอบกับสินค้ารหัส 00003 iPhone และ 00010 Lenovo ซึ่งได้ผลลัพธ์ของการทดสอบแบบตัดไม่หมดใน 1 วัน ดังรูปที่ 4.19 ถึงรูปที่ 4.28 และ ได้ผลลัพธ์ของการทดสอบแบบตัดเกินจำนวนของ 1 วัน ดังรูปที่ 4.29 ถึง รูปที่ 4.34

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
3	00003	ELECTRONIC	2017-01-30 00:00:00.000		10	1	Mobile	iPhone
13	00003	ELECTRONIC	2017-01-29 00:00:00.000		10	2	Mobile	iPhone
23	00003	ELECTRONIC	2017-01-28 00:00:00.000		10	3	Mobile	iPhone
33	00003	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Mobile	iPhone
43	00003	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Mobile	iPhone
53	00003	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Mobile	iPhone
63	00003	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Mobile	iPhone
73	00003	ELECTRONIC	2017-01-23 00:00:00.000		10	8	Mobile	iPhone
83	00003	ELECTRONIC	2017-01-22 00:00:00.000		10	9	Mobile	iPhone
93	00003	ELECTRONIC	2017-01-21 00:00:00.000		10	10	Mobile	iPhone
103	00003	ELECTRONIC	2017-01-20 00:00:00.000		10	11	Mobile	iPhone
113	00003	ELECTRONIC	2017-01-19 00:00:00.000		10	12	Mobile	iPhone

รูปที่ 4.19 ก่อนตัดยอด iPhone

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
3	00003	ELECTRONIC	2017-01-30 00:00:00.000		4	1	Mobile	iPhone
13	00003	ELECTRONIC	2017-01-29 00:00:00.000		10	2	Mobile	iPhone
23	00003	ELECTRONIC	2017-01-28 00:00:00.000		10	3	Mobile	iPhone
33	00003	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Mobile	iPhone
43	00003	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Mobile	iPhone
53	00003	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Mobile	iPhone
63	00003	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Mobile	iPhone

รูปที่ 4.20 ตัดยอด iPhone ครั้งที่ 1

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
3	00003	ELECTRONIC	2017-01-30 00:00:00.000		1	1	Mobile	iPhone
13	00003	ELECTRONIC	2017-01-29 00:00:00.000		10	2	Mobile	iPhone
23	00003	ELECTRONIC	2017-01-28 00:00:00.000		10	3	Mobile	iPhone
33	00003	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Mobile	iPhone
43	00003	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Mobile	iPhone
53	00003	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Mobile	iPhone
63	00003	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Mobile	iPhone

รูปที่ 4.21 ตัดยอด iPhone ครั้งที่ 2

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
3	00003	ELECTRONIC	2017-01-30 00:00:00.000		0	1	Mobile	iPhone
13	00003	ELECTRONIC	2017-01-29 00:00:00.000		10	2	Mobile	iPhone
23	00003	ELECTRONIC	2017-01-28 00:00:00.000		10	3	Mobile	iPhone
33	00003	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Mobile	iPhone
43	00003	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Mobile	iPhone
53	00003	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Mobile	iPhone
63	00003	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Mobile	iPhone

รูปที่ 4.22 ตัดยอด iPhone ครั้งที่ 3

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
3	00003	ELECTRONIC	2017-01-30 00:00:00.000		0	1	Mobile	iPhone
13	00003	ELECTRONIC	2017-01-29 00:00:00.000		5	2	Mobile	iPhone
23	00003	ELECTRONIC	2017-01-28 00:00:00.000		10	3	Mobile	iPhone
33	00003	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Mobile	iPhone
43	00003	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Mobile	iPhone
53	00003	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Mobile	iPhone
63	00003	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Mobile	iPhone

รูปที่ 4.23 ตัดยอด iPhone ครั้งที่ 4

จากรูปที่ 4.20 ถึง รูปที่ 4.23 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.5 โดยดึงสินค้า iPhone ออกตามลำดับ คือ ครั้งที่ 1 ดึงออก 6 ชิ้น ครั้งที่ 2 ดึงออก 3 ชิ้น ครั้งที่ 3 ดึงออก 1 ชิ้น และครั้งที่ 4 ดึงออก 5 ชิ้น

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
10	00010	ELECTRONIC	2017-01-30 00:00:00.000		10	1	Notebook	Lenovo
20	00010	ELECTRONIC	2017-01-29 00:00:00.000		10	2	Notebook	Lenovo
30	00010	ELECTRONIC	2017-01-28 00:00:00.000		10	3	Notebook	Lenovo
40	00010	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Notebook	Lenovo
50	00010	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Notebook	Lenovo
60	00010	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Notebook	Lenovo
70	00010	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Notebook	Lenovo
80	00010	ELECTRONIC	2017-01-23 00:00:00.000		10	8	Notebook	Lenovo
90	00010	ELECTRONIC	2017-01-22 00:00:00.000		10	9	Notebook	Lenovo
100	00010	ELECTRONIC	2017-01-21 00:00:00.000		10	10	Notebook	Lenovo
110	00010	ELECTRONIC	2017-01-20 00:00:00.000		10	11	Notebook	Lenovo
120	00010	ELECTRONIC	2017-01-19 00:00:00.000		10	12	Notebook	Lenovo

รูปที่ 4.24 ก่อนตัดยอด Lenovo

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
10	00010	ELECTRONIC	2017-01-30 00:00:00.000	4	1	Notebook	Lenovo	
20	00010	ELECTRONIC	2017-01-29 00:00:00.000	10	2	Notebook	Lenovo	
30	00010	ELECTRONIC	2017-01-28 00:00:00.000	10	3	Notebook	Lenovo	
40	00010	ELECTRONIC	2017-01-27 00:00:00.000	10	4	Notebook	Lenovo	
50	00010	ELECTRONIC	2017-01-26 00:00:00.000	10	5	Notebook	Lenovo	
60	00010	ELECTRONIC	2017-01-25 00:00:00.000	10	6	Notebook	Lenovo	
70	00010	ELECTRONIC	2017-01-24 00:00:00.000	10	7	Notebook	Lenovo	

รูปที่ 4.25 ตัดยอด Lenovo ครั้งที่ 1

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
10	00010	ELECTRONIC	2017-01-30 00:00:00.000		1	1	Notebook	Lenovo
20	00010	ELECTRONIC	2017-01-29 00:00:00.000		10	2	Notebook	Lenovo
30	00010	ELECTRONIC	2017-01-28 00:00:00.000		10	3	Notebook	Lenovo
40	00010	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Notebook	Lenovo
50	00010	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Notebook	Lenovo
60	00010	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Notebook	Lenovo
70	00010	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Notebook	Lenovo

รูปที่ 4.26 ตัดยอด Lenovo ครั้งที่ 2

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
10	00010	ELECTRONIC	2017-01-30 00:00:00.000		0	1	Notebook	Lenovo
20	00010	ELECTRONIC	2017-01-29 00:00:00.000		10	2	Notebook	Lenovo
30	00010	ELECTRONIC	2017-01-28 00:00:00.000		10	3	Notebook	Lenovo
40	00010	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Notebook	Lenovo
50	00010	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Notebook	Lenovo
60	00010	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Notebook	Lenovo
70	00010	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Notebook	Lenovo

รูปที่ 4.27 ตัดยอด Lenovo ครั้งที่ 3

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
10	00010	ELECTRONIC	2017-01-30 00:00:00.000		0	1	Notebook	Lenovo
20	00010	ELECTRONIC	2017-01-29 00:00:00.000		5	2	Notebook	Lenovo
30	00010	ELECTRONIC	2017-01-28 00:00:00.000		10	3	Notebook	Lenovo
40	00010	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Notebook	Lenovo
50	00010	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Notebook	Lenovo
60	00010	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Notebook	Lenovo
70	00010	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Notebook	Lenovo

รูปที่ 4.28 ตัดยอด Lenovo ครั้งที่ 4

จากรูปที่ 4.25 ถึง รูปที่ 4.28 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.6 โดยดึงสินค้า Lenovo ออกตามลำดับ คือ ครั้งที่ 1 ดึงออก 6 ชิ้น ครั้งที่ 2 ดึงออก 3 ชิ้น ครั้งที่ 3 ดึงออก 1 ชิ้น และครั้งที่ 4 ดึงออก 5 ชิ้น

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
3	00003	ELECTRONIC	2017-01-30 00:00:00.000		0	1	Mobile	iPhone
13	00003	ELECTRONIC	2017-01-29 00:00:00.000		0	2	Mobile	iPhone
23	00003	ELECTRONIC	2017-01-28 00:00:00.000		9	3	Mobile	iPhone
33	00003	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Mobile	iPhone
43	00003	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Mobile	iPhone
53	00003	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Mobile	iPhone
63	00003	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Mobile	iPhone

รูปที่ 4.29 ตัดยอด iPhone ครั้งที่ 5

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
3	00003	ELECTRONIC	2017-01-30 00:00:00.000		0	1	Mobile	iPhone
13	00003	ELECTRONIC	2017-01-29 00:00:00.000		0	2	Mobile	iPhone
23	00003	ELECTRONIC	2017-01-28 00:00:00.000		0	3	Mobile	iPhone
33	00003	ELECTRONIC	2017-01-27 00:00:00.000		9	4	Mobile	iPhone
43	00003	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Mobile	iPhone
53	00003	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Mobile	iPhone
63	00003	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Mobile	iPhone

รูปที่ 4.30 ตัดยอด iPhone ครั้งที่ 6

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
3	00003	ELECTRONIC	2017-01-30 00:00:00.000		0	1	Mobile	iPhone
13	00003	ELECTRONIC	2017-01-29 00:00:00.000		0	2	Mobile	iPhone
23	00003	ELECTRONIC	2017-01-28 00:00:00.000		0	3	Mobile	iPhone
33	00003	ELECTRONIC	2017-01-27 00:00:00.000		0	4	Mobile	iPhone
43	00003	ELECTRONIC	2017-01-26 00:00:00.000		4	5	Mobile	iPhone
53	00003	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Mobile	iPhone
63	00003	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Mobile	iPhone

รูปที่ 4.31 ตัดยอด iPhone ครั้งที่ 7

จากรูปที่ 4.29 ถึง รูปที่ 4.31 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.7 โดยดึงสินค้า iPhone ออกตามลำดับ คือ ครั้งที่ 5 ดึงออก 6 ชิ้น ครั้งที่ 6 ดึงออก 10 ชิ้น ครั้งที่ 7 ดึงออก 15 ชิ้น

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
10	00010	ELECTRONIC	2017-01-30 00:00:00.000		0	1	Notebook	Lenovo
20	00010	ELECTRONIC	2017-01-29 00:00:00.000		0	2	Notebook	Lenovo
30	00010	ELECTRONIC	2017-01-28 00:00:00.000		9	3	Notebook	Lenovo
40	00010	ELECTRONIC	2017-01-27 00:00:00.000		10	4	Notebook	Lenovo
50	00010	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Notebook	Lenovo
60	00010	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Notebook	Lenovo
70	00010	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Notebook	Lenovo

รูปที่ 4.32 ตัดยอด Lenovo ครั้งที่ 5

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
10	00010	ELECTRONIC	2017-01-30 00:00:00.000		0	1	Notebook	Lenovo
20	00010	ELECTRONIC	2017-01-29 00:00:00.000		0	2	Notebook	Lenovo
30	00010	ELECTRONIC	2017-01-28 00:00:00.000		0	3	Notebook	Lenovo
40	00010	ELECTRONIC	2017-01-27 00:00:00.000		9	4	Notebook	Lenovo
50	00010	ELECTRONIC	2017-01-26 00:00:00.000		10	5	Notebook	Lenovo
60	00010	ELECTRONIC	2017-01-25 00:00:00.000		10	6	Notebook	Lenovo
70	00010	ELECTRONIC	2017-01-24 00:00:00.000		10	7	Notebook	Lenovo

รูปที่ 4.33 ตัดยอด Lenovo ครั้งที่ 6

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
10	00010	ELECTRONIC	2017-01-30 00:00:00.000	0	1	Notebook	Lenovo	
20	00010	ELECTRONIC	2017-01-29 00:00:00.000	0	2	Notebook	Lenovo	
30	00010	ELECTRONIC	2017-01-28 00:00:00.000	0	3	Notebook	Lenovo	
40	00010	ELECTRONIC	2017-01-27 00:00:00.000	0	4	Notebook	Lenovo	
50	00010	ELECTRONIC	2017-01-26 00:00:00.000	4	5	Notebook	Lenovo	
60	00010	ELECTRONIC	2017-01-25 00:00:00.000	10	6	Notebook	Lenovo	
70	00010	ELECTRONIC	2017-01-24 00:00:00.000	10	7	Notebook	Lenovo	

รูปที่ 4.34 ตัดยอด Lenovo ครั้งที่ 7

จากรูปที่ 4.32 ถึง รูปที่ 4.34 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.8 โดยดึงสินค้า Lenovo ออกตามลำดับ คือ ครั้งที่ 5 ดึงออก 6 ชิ้น ครั้งที่ 6 ดึงออก 10 ชิ้น ครั้งที่ 7 ดึงออก 15 ชิ้น

4.2.3 แบบหมดอายุก่อนออกก่อน

การทดสอบแบบหมดอายุก่อนออกก่อน ตามแผนที่วางไว้คือ ทดสอบแบบตัดไม่หมดใน 1 วัน และ แบบตัดเกินจำนวนของ 1 วัน โดยจำทดสอบกับสินค้ารหัส 00001 Salt และ 00002 Garlic ซึ่งได้ผลลัพธ์ของการทดสอบแบบตัดไม่หมดใน 1 วัน ดังรูปที่ 4.35 ถึง รูปที่ 4.44 และ ได้ผลลัพธ์ของการทดสอบแบบตัดเกินจำนวนของ 1 วัน ดังรูปที่ 4.45 ถึง รูปที่ 4.50

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
1	00001	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	10	1	Salt	Salt
11	00001	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	10	2	Salt	Salt
21	00001	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	10	3	Salt	Salt
31	00001	FOOD	2017-01-04 00:00:00.000	2017-01-04 00:00:00.000	10	4	Salt	Salt
41	00001	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Salt	Salt
51	00001	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Salt	Salt
61	00001	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Salt	Salt
71	00001	FOOD	2017-01-08 00:00:00.000	2017-02-08 00:00:00.000	10	8	Salt	Salt
81	00001	FOOD	2017-01-09 00:00:00.000	2017-02-09 00:00:00.000	10	9	Salt	Salt
91	00001	FOOD	2017-01-10 00:00:00.000	2017-02-10 00:00:00.000	10	10	Salt	Salt
101	00001	FOOD	2017-01-11 00:00:00.000	2017-02-11 00:00:00.000	10	11	Salt	Salt
111	00001	FOOD	2017-01-12 00:00:00.000	2017-02-12 00:00:00.000	10	12	Salt	Salt

รูปที่ 4.35 ก่อนตัดยอด Salt

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
1	00001	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	4	1	Salt	Salt
11	00001	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	10	2	Salt	Salt
21	00001	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	10	3	Salt	Salt
31	00001	FOOD	2017-01-04 00:00:00.000	2017-01-04 00:00:00.000	10	4	Salt	Salt
41	00001	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Salt	Salt
51	00001	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Salt	Salt
61	00001	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Salt	Salt

รูปที่ 4.36 ตัดยอด Salt ครั้งที่ 1

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
1	00001	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	1	1	Salt	Salt
11	00001	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	10	2	Salt	Salt
21	00001	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	10	3	Salt	Salt
31	00001	FOOD	2017-01-04 00:00:00.000	2017-01-04 00:00:00.000	10	4	Salt	Salt
41	00001	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Salt	Salt
51	00001	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Salt	Salt
61	00001	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Salt	Salt

รูปที่ 4.37 ตัดยอด Salt ครั้งที่ 2

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
1	00001	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	0	1	Salt	Salt
11	00001	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	10	2	Salt	Salt
21	00001	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	10	3	Salt	Salt
31	00001	FOOD	2017-01-04 00:00:00.000	2017-01-04 00:00:00.000	10	4	Salt	Salt
41	00001	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Salt	Salt
51	00001	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Salt	Salt
61	00001	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Salt	Salt

รูปที่ 4.38 ตัดยอด Salt ครั้งที่ 3

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
1	00001	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	0	1	Salt	Salt
11	00001	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	5	2	Salt	Salt
21	00001	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	10	3	Salt	Salt
31	00001	FOOD	2017-01-04 00:00:00.000	2017-01-04 00:00:00.000	10	4	Salt	Salt
41	00001	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Salt	Salt
51	00001	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Salt	Salt
61	00001	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Salt	Salt

รูปที่ 4.39 ตัดยอด Salt ครั้งที่ 4

จากรูปที่ 4.36 ถึง รูปที่ 4.39 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.9 โดยดึงสินค้า Salt ออกตามลำดับ คือ ครั้งที่ 1 ดึงออก 6 ชิ้น ครั้งที่ 2 ดึงออก 3 ชิ้น ครั้งที่ 3 ดึงออก 1 ชิ้น และครั้งที่ 4 ดึงออก 5 ชิ้น

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
2	00002	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	10	1	Garlic	Garlic
12	00002	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	10	2	Garlic	Garlic
22	00002	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	10	3	Garlic	Garlic
32	00002	FOOD	2017-01-04 00:00:00.000	2017-02-04 00:00:00.000	10	4	Garlic	Garlic
42	00002	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Garlic	Garlic
52	00002	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Garlic	Garlic
62	00002	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Garlic	Garlic
72	00002	FOOD	2017-01-08 00:00:00.000	2017-02-08 00:00:00.000	10	8	Garlic	Garlic
82	00002	FOOD	2017-01-09 00:00:00.000	2017-02-09 00:00:00.000	10	9	Garlic	Garlic
92	00002	FOOD	2017-01-10 00:00:00.000	2017-02-10 00:00:00.000	10	10	Garlic	Garlic
102	00002	FOOD	2017-01-11 00:00:00.000	2017-02-11 00:00:00.000	10	11	Garlic	Garlic
112	00002	FOOD	2017-01-12 00:00:00.000	2017-02-12 00:00:00.000	10	12	Garlic	Garlic

รูปที่ 4.40 ก่อนตัดยอด Garlic

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
2	00002	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	4	1	Garlic	Garlic
12	00002	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	10	2	Garlic	Garlic
22	00002	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	10	3	Garlic	Garlic
32	00002	FOOD	2017-01-04 00:00:00.000	2017-02-04 00:00:00.000	10	4	Garlic	Garlic
42	00002	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Garlic	Garlic
52	00002	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Garlic	Garlic
62	00002	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Garlic	Garlic

รูปที่ 4.41 ตัดยอด Garlic ครั้งที่ 1

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
2	00002	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	1	1	Garlic	Garlic
12	00002	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	10	2	Garlic	Garlic
22	00002	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	10	3	Garlic	Garlic
32	00002	FOOD	2017-01-04 00:00:00.000	2017-02-04 00:00:00.000	10	4	Garlic	Garlic
42	00002	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Garlic	Garlic
52	00002	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Garlic	Garlic
62	00002	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Garlic	Garlic

รูปที่ 4.42 ตัดยอด Garlic ครั้งที่ 2

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
2	00002	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	0	1	Garlic	Garlic
12	00002	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	10	2	Garlic	Garlic
22	00002	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	10	3	Garlic	Garlic
32	00002	FOOD	2017-01-04 00:00:00.000	2017-02-04 00:00:00.000	10	4	Garlic	Garlic
42	00002	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Garlic	Garlic
52	00002	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Garlic	Garlic
62	00002	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Garlic	Garlic

รูปที่ 4.43 ตัดยอด Garlic ครั้งที่ 3

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
2	00002	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	0	1	Garlic	Garlic
12	00002	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	5	2	Garlic	Garlic
22	00002	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	10	3	Garlic	Garlic
32	00002	FOOD	2017-01-04 00:00:00.000	2017-02-04 00:00:00.000	10	4	Garlic	Garlic
42	00002	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Garlic	Garlic
52	00002	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Garlic	Garlic
62	00002	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Garlic	Garlic

รูปที่ 4.44 ตัดยอด Garlic ครั้งที่ 4

จากรูปที่ 4.41 ถึง รูปที่ 4.44 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.10 โดยดึงสินค้า Garlic ออกตามลำดับ คือ ครั้งที่ 1 ดึงออก 6 ชิ้น ครั้งที่ 2 ดึงออก 3 ชิ้น ครั้งที่ 3 ดึงออก 1 ชิ้น และครั้งที่ 4 ดึงออก 5 ชิ้น

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
1	00001	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	0	1	Salt	Salt
11	00001	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	0	2	Salt	Salt
21	00001	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	9	3	Salt	Salt
31	00001	FOOD	2017-01-04 00:00:00.000	2017-01-04 00:00:00.000	10	4	Salt	Salt
41	00001	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Salt	Salt
51	00001	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Salt	Salt
61	00001	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Salt	Salt

รูปที่ 4.45 ตัดยอด Salt ครั้งที่ 5

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
1	00001	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	0	1	Salt	Salt
11	00001	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	0	2	Salt	Salt
21	00001	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	0	3	Salt	Salt
31	00001	FOOD	2017-01-04 00:00:00.000	2017-01-04 00:00:00.000	9	4	Salt	Salt
41	00001	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Salt	Salt
51	00001	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Salt	Salt
61	00001	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Salt	Salt

รูปที่ 4.46 ตัดยอด Salt ครั้งที่ 6

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
1	00001	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	0	1	Salt	Salt
11	00001	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	0	2	Salt	Salt
21	00001	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	0	3	Salt	Salt
31	00001	FOOD	2017-01-04 00:00:00.000	2017-01-04 00:00:00.000	0	4	Salt	Salt
41	00001	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	4	5	Salt	Salt
51	00001	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Salt	Salt
61	00001	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Salt	Salt

รูปที่ 4.47 ตัดยอด Salt ครั้งที่ 7

จากรูปที่ 4.45 ถึง รูปที่ 4.47 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.11 โดยดึงสินค้า Salt ออกตามลำดับ คือ ครั้งที่ 5 ดึงออก 6 ชิ้น ครั้งที่ 6 ดึงออก 10 ชิ้น ครั้งที่ 7 ดึงออก 15 ชิ้น

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
2	00002	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	0	1	Garlic	Garlic
12	00002	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	0	2	Garlic	Garlic
22	00002	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	9	3	Garlic	Garlic
32	00002	FOOD	2017-01-04 00:00:00.000	2017-02-04 00:00:00.000	10	4	Garlic	Garlic
42	00002	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Garlic	Garlic
52	00002	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Garlic	Garlic
62	00002	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Garlic	Garlic

รูปที่ 4.48 ตัดยอด Garlic ครั้งที่ 5

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
2	00002	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	0	1	Garlic	Garlic
12	00002	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	0	2	Garlic	Garlic
22	00002	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	0	3	Garlic	Garlic
32	00002	FOOD	2017-01-04 00:00:00.000	2017-02-04 00:00:00.000	9	4	Garlic	Garlic
42	00002	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	10	5	Garlic	Garlic
52	00002	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Garlic	Garlic
62	00002	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Garlic	Garlic

รูปที่ 4.49 ตัดยอด Garlic ครั้งที่ 6

PRODUCTCUTOFFID	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME
2	00002	FOOD	2017-01-01 00:00:00.000	2017-02-01 00:00:00.000	0	1	Garlic	Garlic
12	00002	FOOD	2017-01-02 00:00:00.000	2017-02-02 00:00:00.000	0	2	Garlic	Garlic
22	00002	FOOD	2017-01-03 00:00:00.000	2017-02-03 00:00:00.000	0	3	Garlic	Garlic
32	00002	FOOD	2017-01-04 00:00:00.000	2017-02-04 00:00:00.000	0	4	Garlic	Garlic
42	00002	FOOD	2017-01-05 00:00:00.000	2017-02-05 00:00:00.000	4	5	Garlic	Garlic
52	00002	FOOD	2017-01-06 00:00:00.000	2017-02-06 00:00:00.000	10	6	Garlic	Garlic
62	00002	FOOD	2017-01-07 00:00:00.000	2017-02-07 00:00:00.000	10	7	Garlic	Garlic

รูปที่ 4.50 ตัดยอด Garlic ครั้งที่ 7

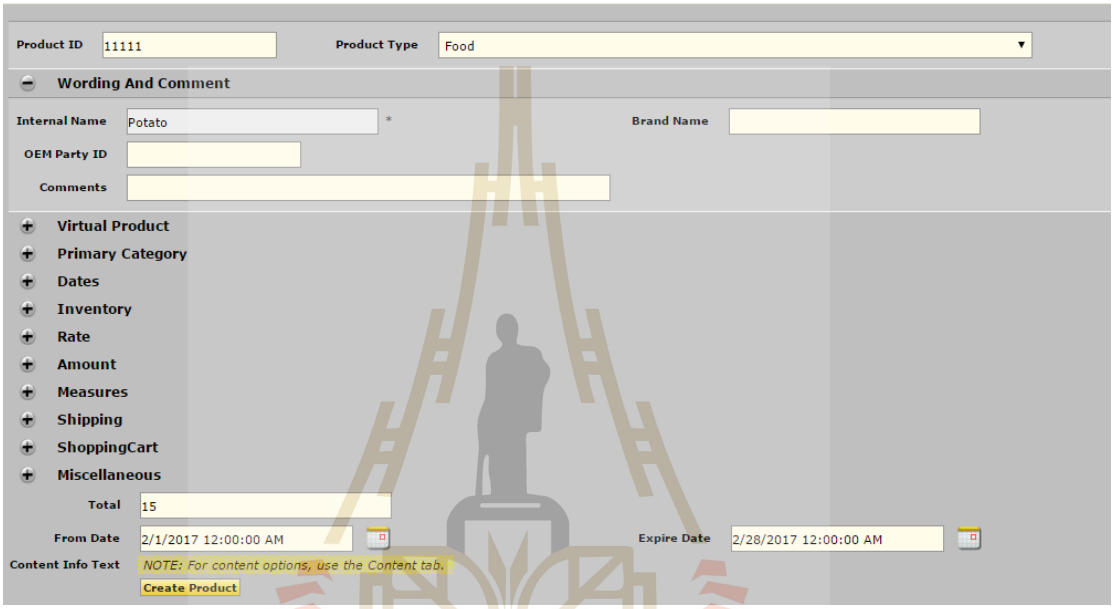
จากรูปที่ 4.48 ถึง รูปที่ 4.50 เป็นผลลัพธ์การทำงานตามการออกแบบในตารางที่ 3.12 โดยดึงสินค้า Garlic ออกตามลำดับ คือ ครั้งที่ 5 ดึงออก 6 ชิ้น ครั้งที่ 6 ดึงออก 10 ชิ้น ครั้งที่ 7 ดึงออก 15 ชิ้น

จากผลลัพธ์ที่ได้นั้น พบว่า โมดูลการตัดยอดที่พัฒนาขึ้นนั้น สามารถทำงานได้อย่างถูกต้องตามที่ได้ออกแบบไว้ และ ผลลัพธ์เป็นไปตามที่ได้วางแผนไว้

4.3 การเรียนใช้โมดูลพื้นฐานที่มีอยู่แล้วของกรอบงานออฟบิส

4.3.1 การบันทึกข้อมูล

โมดูลการบันทึกข้อมูล เป็นโมดูลพื้นฐานที่มีอยู่แล้วในกรอบงานออฟบิส โดยหลังจากที่เราได้มีการเพิ่มพื้นที่สำหรับเก็บข้อมูลในส่วนของ Index ก็ได้ทำการเพิ่มสินค้าใหม่เข้าไปในระบบเพื่อทดสอบการบันทึกข้อมูล



รูปที่ 4.51 หน้าต่างการเพิ่มสินค้าใหม่ไปในระบบ

จากรูปที่ 4.51 เป็นหน้าต่างของการเพิ่มสินค้าใหม่เข้าไปในระบบ เพื่อทดสอบว่าโมดูลการสร้างและบันทึกข้อมูลของกรอบงานยังใช้งานได้ปกติหรือไม่ ซึ่งเมื่อเรายืนยันการสร้างจะพบว่า สินค้าชิ้นดังกล่าวได้ถูกบันทึกลงในฐานข้อมูลเรียบร้อยแล้ว ดังรูปที่ 4.52 เป็นการยืนยันว่าโมดูลในส่วนของการสร้างและบันทึกข้อมูลยังใช้งานได้ตามปกติ

	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME	COMMENTS
VIEW DELETE	11111	FOOD	2017-02-01 00:00:00.000	2017-02-28 00:00:00.000	15	1	Potato		

รูปที่ 4.52 ผลลัพธ์ของการสร้างและบันทึกข้อมูล

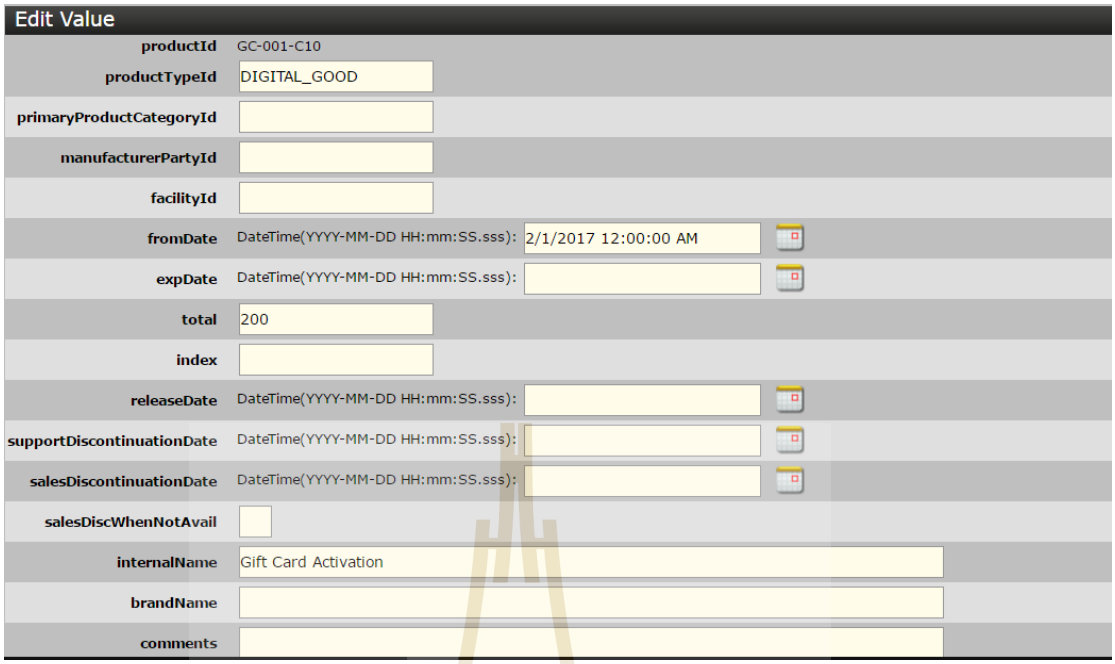
4.3.2 การแก้ไขข้อมูล

โมดูลการแก้ไขข้อมูล เป็นโมดูลที่ทำงานเหมือนกับการบันทึกข้อมูล โดยจะทำการแก้ไขข้อมูลของสินค้าที่มีอยู่แล้วในระบบเพื่อทดสอบการแก้ไขข้อมูล

	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME	COMMENTS
VIEW DELETE	GC-001-C10	DIGITAL_GOOD					Gift Card Activation		
VIEW DELETE	GC-001-C100	DIGITAL_GOOD					Gift Card Activation		
VIEW DELETE	GC-001-C25	DIGITAL_GOOD					Gift Card Activation		

รูปที่ 4.53 สินค้าก่อนทำการแก้ไข

จากรูปที่ 4.53 เป็นรายการสินค้าตัวอย่างก่อนทำการแก้ไข โดยจะทำการเพิ่มวันนำเข้าและจำนวนให้กับสินค้าตัวอย่างเพื่อตรวจสอบโมดูลการแก้ไขสินค้า ดังรูปที่ 4.54



รูปที่ 4.54 หน้าต่างการแก้ไขสินค้า

ซึ่งเมื่อเรายืนยันการแก้ไขจะพบว่า การแก้ไขถูกบันทึกตามที่ได้กรอกข้อมูล ดังเป็นการยืนยันว่าโมดูลในส่วนของการแก้ไขข้อมูลยังใช้งานได้ตามปกติ

	PRODUCTID	PRODUCTTYPEID	FROMDATE	EXPDATE	TOTAL	INDEX	INTERNALNAME	BRANDNAME	COMMENTS
VIEW DELETE	GC-001-C10	DIGITAL_GOOD	2017-02-01 00:00:00.000		200	1	Gift Card Activation		
VIEW DELETE	GC-001-C100	DIGITAL_GOOD	2017-02-01 00:00:00.000	2017-02-28 00:00:00.000	123	1	Gift Card Activation		
VIEW DELETE	GC-001-C25	DIGITAL_GOOD	2017-02-02 00:00:00.000		222	1	Gift Card Activation		

รูปที่ 4.55 สินค้าหลังทำการแก้ไข

จากการทดลองโมดูลการตัดยอดที่พัฒนาขึ้นใหม่ผ่านตัวทดลองใช้งานนั้นโดยใช้ตัวอย่างทดลองทั้งหมด 6 ชนิด ชนิดละ 2 เงื่อนไข คือ เงื่อนไขการดึงสินค้าไม่เกินจำนวนที่นำเข้าใน 1 วัน หรือหมดพอดี และ เงื่อนไขการดึงสินค้าเกินจำนวนที่นำเข้าใน 1 วัน พบว่า สินค้าที่ตัดแบบ FIFO คือ สินค้าประเภท Material คือ Ply Wood และ Nail สินค้าที่ตัดแบบ LIFO คือ สินค้าประเภท Electronic คือ iPhone และ Lenovo และ สินค้าที่ตัดแบบ FEFO คือ สินค้าประเภท Food คือ Salt และ Garlic นั้น สามารถดึงสินค้าตาม index ที่กำหนดไว้ได้ และ จากการทดลองการเรียกใช้โมดูลที่เกี่ยวข้องที่มีอยู่แล้วในกรอบงาน คือ ในส่วนของการบันทึกข้อมูล และ การแก้ไขข้อมูล นั้น พบว่า การบันทึกข้อมูลสามารถบันทึกข้อมูลได้ตามปกติ ทั้งข้อมูลแบบไม่มีวันที่นำเข้าหรือวันหมดอายุ และ แบบมีวันที่นำเข้าและหมดอายุ ในส่วนของการแก้ไขข้อมูล สามารถแก้ไขข้อมูลที่มีอยู่แล้วในฐานข้อมูลโดยการเพิ่มวันที่นำเข้าและวันหมดอายุได้ ซึ่งสรุปได้ว่า โมดูลใหม่ที่พัฒนาขึ้นนั้นสามารถทำงานได้ถูกต้อง และ ตามแผนที่วางไว้ และ ไม่กระทบต่อการทำงานของโมดูลที่เกี่ยวข้องที่มีอยู่แล้วในกรอบงาน และ ตัวทดลองยังสามารถทำงานได้ตามปกติ

บทที่ 5
บทสรุป

กรอบงานออฟบิสเป็นกรอบงานที่มีโมดูลพื้นฐานที่ใช้ในการสร้างซอฟต์แวร์ในการจัดการธุรกิจและการจัดการคลังสินค้าที่ดีและเหมาะสำหรับนำไปใช้พัฒนาให้กับธุรกิจหลายรูปแบบ การพัฒนาโมดูลในส่วนของการตัดยอดสินค้าทำให้ตัวออฟบิสมีประสิทธิภาพในด้านการจัดการสินค้ามากขึ้น โมดูลการตัดยอดที่ถูกพัฒนาขึ้นแบ่งเป็น 3 แบบ คือ FIFO , FEFO , LIFO และได้มีการทดสอบโมดูลผ่านทางตัวทดลองใช้งานที่มีให้ในออฟบิส สามารถสรุปได้ดังนี้

5.1 สรุปผลงานวิจัย

การทดสอบโมดูลผ่านทางตัวทดลองใช้งาน แบ่งออกเป็น 2 กรณี คือ ทดสอบโมดูลการตัดยอดที่พัฒนาขึ้นใหม่ และ ทดสอบโมดูลพื้นฐานที่เกี่ยวข้องที่มีอยู่แล้ว สรุปผลได้ดังนี้

5.1.1 โมดูลการตัดยอดที่พัฒนาขึ้นใหม่

จากการทดสอบโมดูลการตัดยอดที่พัฒนาขึ้นใหม่ โดยสร้างสินค้าที่ใช้ในการทดสอบขึ้น จำนวน 10 ชนิด ชนิดละ 300 ชิ้น โดยสร้างขึ้นในวันเวลาที่ต่างกันติดต่อกัน 30 วัน วันละ 10 ชิ้น แล้วทำการสั่งดึงสินค้าออกตามเงื่อนไขตามเงื่อนไข 2 เงื่อนไข คือ ดึงสินค้าไม่เกินจำนวนที่นำเข้าใน 1 วันหรือหมดพอดี และ ดึงสินค้าเกินจำนวนที่นำเข้าใน 1 วัน พบว่า โมดูลการตัดยอดที่พัฒนาขึ้นทั้ง 3 แบบนั้น สามารถทำงานได้ตามเงื่อนไขที่กำหนด สามารถตัดยอดได้ตามการออกแบบที่วางไว้ (ดังผลลัพธ์ในบทที่ 4)

5.1.2 โมดูลพื้นฐานที่เกี่ยวข้อง

จากการทดสอบโมดูลพื้นฐานที่เกี่ยวของ ที่มีอยู่แล้วในตัวออฟบิส เพื่อตรวจสอบว่า โมดูลการตัดยอดที่พัฒนาขึ้นมาใหม่นั้น เมื่อทำการเพิ่มเขตข้อมูลบางส่วนเพื่อใช้งานกับโมดูลการตัดยอดที่พัฒนาขึ้นแล้วใช้งานร่วมกับโมดูลอื่น ๆ ที่มีอยู่ จะไม่ทำให้โมดูลอื่น ๆ ทำงานผิดพลาดหรือทำงานไม่ได้ ซึ่งโมดูลพื้นฐานที่เกี่ยวข้องนั้นมี 2 ส่วน คือ ในส่วนของการบันทึกสินค้าใหม่ การแก้ไขข้อมูลสินค้าที่มีอยู่แล้ว จากการทดสอบนั้น พบว่า ในส่วนของการบันทึกสินค้าใหม่ สามารถบันทึกได้ตามปกติและถูกต้อง และ ในส่วนของการแก้ไขข้อมูลสินค้า เมื่อแก้ไขข้อมูลสินค้าที่มีอยู่แล้วในตัวทดลองใช้งาน สามารถแก้ไขข้อมูลและเพิ่มข้อมูลในส่วนที่เพิ่มขึ้นมาได้ตามปกติและถูกต้อง (ดังผลลัพธ์ในบทที่ 4)

จากผลการทดสอบของทั้ง 2 กรณีนั้น สามารถสรุปได้ว่า โมดูลการตัดยอดสินค้าที่พัฒนาขึ้นมาใหม่นั้น สามารถทำงานได้ตามที่วางแผนไว้ และ โมดูลพื้นฐานที่เกี่ยวข้องสามารถทำงานได้ตามปกติและถูกต้อง

5.2 ข้อเสนอแนะสำหรับงานวิจัยต่อไป

จากที่ได้ศึกษาและทดลองใช้กรอบงานออฟบิสผ่านทางตัวทดลองใช้งานนั้น ได้พบว่ากรอบงานออฟบิสเอง ยังมีบางส่วนที่ยังยังไม่สามารถทำงานได้อย่างถูกต้อง และ สามารถนำไปวิจัยพัฒนาต่อได้ในอนาคต เช่น ในส่วนของการคัดลอกสินค้าเพื่อแก้ไขข้อมูลในกรณีที่มีสินค้าประเภทเดียวกันเข้ามาเป็นจำนวนมากที่มีความแตกต่างกันเพียงเล็กน้อย เช่น ยี่ห้อ เพื่อบันทึกสินค้าใหม่ ซึ่งมีโมดูลไว้ใช้งาน แต่ยังไม่สามารถทำงานเนื่องจากข้อผิดพลาดในการคัดลอกข้อมูลที่ไม่สามารถบันทึกซ้ำได้แต่ก็ไม่สามารถแก้ไขข้อมูลในส่วนนั้นได้เช่นเดียวกัน

รายการอ้างอิง

กิตติ ภักดีวัฒนะกุล และ พานิชกุล พนิดา. (2550). **วิศวกรรมซอฟต์แวร์ : *Software Engineering* (Second ed.)**. กรุงเทพมหานคร : เคทีพี คอมพ์ แอนด์ คอนซัลท์.

คำนาย อภิปรัชญาสกุล. (2550). **การจัดการคลังสินค้า Warehouse Management**. กรุงเทพมหานคร : โฟกัสมีเดีย แอนด์พับลิชชิ่ง.

วุฒิพล หมัดเส็น และ พิชโยทัย มหัทธนาภิวัฒน์. (2551). กรอบงานการผูกความสัมพันธ์ข้อมูลเชิงวัตถุในชั้นการแสดงผลข้อมูลนามธรรมและข้อมูลเชิงสัมพันธ์. วิทยานิพนธ์ปริญญามหาบัณฑิต. สำนักวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี.

สุดา เธียรมนตรี. (2556). **คู่มือเรียนเขียนโปรแกรมภาษา *Java* (2 Ed.)**. นนทบุรี: ไอดีซี พรีเมียร์

Allen H. Dutoit, & Bruegge, B. (2004). **Object-Oriented Software Engineering *:* Using UML, Patterns, and Java**. New Jersey : Pearson.

Bernardin, L. (1999). A Java framework for massively distributed symbolic computing. **Mathematics and computers in simulation** *(*pp 151-160). (n.p.).

David E. Jones. (2015). **Apache Ofbiz Project Overview** [Online]. Available: https://ofbiz.apache.org/apache-ofbiz-project-overview.html

Donald W. (2004). **Inventory Control and Management**. New Jersey : Wiley.

Ghezzi, C., Jazayeri, M., & Mandrioli, D. (2002). **Fundamentals of Software Engineering**. New Jersey: Prentice Hall PTR.

Howell, R. (2008). **Apache OFBiz Development*:* The Beginner's Tutorial**. Birmingham: Packt.

Jara, C. A., Esquembre, F., Christian, W., Candelas, F. A., Torres, F., & Dormido, S. (2012). A new 3D visualization Java framework based on physics principles. **Computer Physics Communications** (pp 231-244). (n.p.).

Jeff F. (2011). **Beginning Java 7**.227-344. New York : Paul Manning

Lee, G. H., & Jung, J. (2007). Web framework with Java and XML in multi-tiers for productivity. **Future Generation Computer Systems**. (pp 263-268). (n.p.).

Liu, Y., Guo, Q., & Tian, Y. (2012). A software framework for classification models of geographical data. **Computers & Geosciences** (pp 47-56). (n.p.).

Rittammanart, N., Wongyued, W., & Dailey, M. N. (2008). ERP application development frameworks: Case study and evaluation. **5th International Conference on Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology** (pp 173-176). (n.p.).

Timothy C. Lethbridge, & Laganiere, R. (2001). **Object-Oriented Software Engineering Practical software development using UML and Java**. Boston: McGraw-Hill.

Timothy C. Lethbridge, & Laganiere, R. (2005). **Object-Oriented Software Engineering : Practical Software Development using UML and Java (Second ed.)**. Boston: McGraw-Hill.

Tompkins, J. A., & Smith, J. D. (1998). **The warehouse management handbook**. North Carolina: Tompkins press.

Zhao, L., & Liu, J. (2011). Component-Oriented SaaS Integration Framework Research Based on OFBiz. **2011 International Conference on Management and Service Science** (pp 1-3). (n.p.).

ภาคผนวก ก

บทความทางวิชาการที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างศึกษา

รายชื่อบทความที่ได้รับการตีพิมพ์เผยแพร่ในระหว่างศึกษา

ถิรวุฒิ ไชยชะอุ่ม, พิชโยทัย มหัทธนาภิวัฒน์. 2560. **การพัฒนาโมดูลการตัดยอดสินค้าสำหรับกรอบงานออฟบิส.** โครงการประชุมวิชาการระดับนานาชาติและระดับชาติด้านด้านเทคโนโลยีคอมพิวเตอร์และระบบสารสนเทศประยุกต์ และการประชุมวิชาการด้านบริหารธุรกิจ ครั้งที่ 11 (ACTIS & NCOBA), วิทยาลัยเซาธ์อิสท์บางกอก บางนา กรุงเทพมหานคร. 25 มกราคม 2560. หน้า 279-283

ประวัติผู้เขียน

นายถิรวุฒิ ไชยชะอุ่ม เกิดเมื่อวันที่ 21 เมษายน พ.ศ. 2535 เริ่มศึกษาชั้นประถมศึกษาที่โรงเรียนวรธรรมพิทยากาฬสินธุ์ (โรงเรียนเซนต์ยอแซฟกาฬสินธุ์ ในปัจจุบัน) ชั้นมัธยมศึกษาที่โรงเรียนกาฬสินธุ์พิทยาสรรพ์ และสำเร็จการศึกษาระดับชั้นปริญญาตรี สาขาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จังหวัดนครราชสีมา เมื่อปี พ.ศ. 2557

ปี พ.ศ. 2557 เข้าศึกษาต่อในระดับปริญญาโท สาขาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี โดยขณะศึกษาได้รับทุนการศึกษาสำหรับผู้มีศักยภาพเข้าศึกษาระดับบัณฑิตศึกษา และได้เป็นผู้ช่วยสอนในรายวิชาเทคโนโลยีเชิงวัตถุ (Object-Oriented Technology) และรายวิชาหัวข้อขั้นสูงในวิศวกรรมคอมพิวเตอร์ 1 (Advanced Topics in Computer Engineering I)

ผลงานวิจัย : ได้เสนอบทความเข้าร่วมในโครงการประชุมวิชาการระดับนานาชาติและระดับชาติด้านเทคโนโลยีคอมพิวเตอร์และระบบสารสนเทศประยุกต์ และการประชุมวิชาการด้านบริหารธุรกิจ ครั้งที่ 11 (ACTIS & NCOBA 2017) เรื่องการพัฒนาโมดูลการตัดยอดสินค้าสำหรับกรอบงานออฟบิส