รายงานสหกิจศึกษาฉบับสมบูรณ์

พัฒนาชุดทดสอบโปรแกรมโดยใช้ JUnit Framework

Create unit test suite with JUnit

นายฐปกร วันแก้ว

ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2559

รายงานสหกิจศึกษาฉบับสมบูรณ์

พัฒนาชุดทดสอบโปรแกรมโดยใช้ JUnit Framework
Create unit test suite with JUnit

นายฐูปกร วันแก้ว

ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2559

ชื่อโครงงานสหกิจศึกษา พัฒนาชุดทดสอบโปรแกรมโดยใช้ JUnit Framework
นักศึกษา นายฐปกร วันแก้ว
คณะ วิศวกรรมศาสตร์ **ภาควิชา** วิศวกรรมคอมพิวเตอร์
อาจารย์นิเทศ อาจารย์จิรศักดิ์ สิทธิกร, อาจารย์บัณฑิต พัสยา
ผู้นิเทศงาน นายเชษฐา จรัลวรวงศ์
สถานประกอบการ บริษัท เอ็นทีที เดต้า (ประเทศไทย) จำกัด สาขา 00001

บทคัดย่อ

โครงงานพัฒนาชุดทดสอบโปรแกรมโดยใช้ JUnit framework จัดทำขึ้นโดยมีวัตถุประสงค์คือใช้ JUnit framework ซึ่งเป็น Unit Testing framework ที่นิยมใช้กันในภาษา Java มาพัฒนาชุดทดสอบโปรแกรมเพื่อใช้ทดสอบการทำงานของโปรแกรม Web GUI version 2 ในชั้น Business Layer โดยโปรแกรม Web GUI version 2 นั้นเป็นโปรแกรมที่ถูกพัฒนาขึ้นมาใช้งานแทนโปรแกรม Web GUI เดิมซึ่งพัฒนามาในรูปแบบครอบจอการทำงาน ชุดทดสอบโปรแกรมนี้จะพัฒนาโดยใช้ Junit framework และได้มีการนำ Mockito framework ซึ่งเป็น library ของภาษา Java สำหรับจำลองข้อมูลเพื่อช่วยในการทำ Unit test รายงานเล่มนี้จะประกอบไปด้วยข้อมูลของเครื่องมือที่ใช้ในการพัฒนาชุดทดสอบโปรแกรม ขั้นตอนการพัฒนาชุดทดสอบโปรแกรม ผลของการใช้ชุดทดสอบมาทดสอบโปรแกรม Web GUI version 2 ตลอดจนเอกสารประกอบการทดสอบฟังก์ชันของการทำงานในกรณีต่างๆ นอกจากนี้ภายในรายงานเล่มนี้ยังมีทฤษฎีที่ใช้ในการพัฒนาชุดทดสอบโปรแกรมนี้ขึ้นมาอาทิเช่น Unit test, Automation Test, JUnit framework, Mockito framework เป็นต้น

คำสำคัญ : Unit test, Automate Test, Junit framework, Mockito framework

Project Title: Create unit test suite with JUnit
Student: Mr. Tapakorn Wankaew
Faculty: Engineering **Department:** Computer Engineering
Advisor: Mr. Jirasak Sittigorn, Mr. Bundit Pasaya
Mentor: Mr. Chetta Charanworawong
Company: NTT DATA (Thailand) Co., Ltd (Branch 00001)

ABSTRACT

Project development of program unit test suite using by JUnit framework. Prepared by the objective is to use Junit framework which was unit testing framework that is the most widely used in Java language to development unit test suite for test process of the program Web GUI version 2 in the business layer of Web GUI version 2, this program is a program that was developed using the primary Web GUI program, which was developed in to duplicate screen formed. This unit test suite is developed using JUnit framework and use Mockito framework which was library of Java language for mock data to help make the unit test suite. This report consists of the tools used to develop test programs, the process of developing a series of test programs, the results of the test program to test Web GUI version 2, as well as documents of the trials of function working in various cases. Also within this report has a theory based on the development of this program including Unit test, Automation Test, JUnit framework, Mockito framework.

Keywords : Unit test, Automate Test, Junit framework, Mockito framework

กิตติกรรมประกาศ

การจัดทำรายงานเล่มนี้ได้รับความอนุเคราะห์จากบริษัท เอ็นทีที เดต้า (ประเทศไทย) จำกัด ซึ่งประสบการณ์ที่ผู้จัดทำได้เรียนรู้จากบริษัทนั้นมีประโยชน์ และสามารถนำไปใช้ในการศึกษาต่อในอนาคตได้อย่างแน่นอน การที่ได้มาทำโครงการสหกิจที่บริษัทแห่งนี้ทำให้ได้เรียนรู้และได้ประสบการณ์ต่างๆมากมายที่ไม่สามารถหาได้จากการศึกษาในวิชาเรียนปกติ ผู้จัดทำอยากจะขอขอบคุณบริษัท เอ็นทีที เดต้า (ประเทศไทย) จำกัด เป็นอย่างมากที่ตอบรับให้เข้ามาเป็นส่วนหนึ่งของทีมพัฒนาโปรแกรม โดยเฉพาะอย่างยิ่งขอบคุณทุกคนในแผนก Software Processing Center ที่ดูแลและให้การช่วยเหลือผู้จัดทำมาตลอดระยะเวลาหกเดือนที่ผ่านมา ขอขอบคุณพี่ๆในทีม GUI ที่คอยเป็นพี่เลี้ยงให้ความรู้ คำแนะนำ ปรึกษาในเรื่องต่างๆ แก่ผู้จัดทำมาตลอด และให้โอกาสในการปฏิบัติงานจริง

ขอขอบคุณอาจารย์จิรศักดิ์ สิทธิกรและอาจารย์บัณฑิต พัสยาซึ่งเป็นอาจารย์ที่ปรึกษาของโครงการสหกิจศึกษานี้ ที่คอยให้คำแนะนำต่างๆ และช่วยการแก้ไขปัญหาที่เกิดจากการทำโครงการสหกิจศึกษาในภาคการศึกษานี้ ขอขอบคุณอาจารย์ประจำภาควิชาทุกท่านที่มอบความรู้ทั้งในภาคทฤษฎีและภาคปฏิบัติทำให้ผู้จัดทำสามารถนำมาประยุกต์ใช้ในการทำโครงการสหกิจศึกษาครั้งนี้ได้

ผู้จัดทำ
ฐปกร วันแก้ว

สารบัญ

หน้า

บทคัดย่อภาษาไทย I
บทคัดย่อภาษาอังกฤษ II
กิตติกรรมประกาศ III
สารบัญ IV
สารบัญตาราง VI
สารบัญภาพ VII
บทที่ 1 บทนำ 1
1.1 ที่มาและความสำคัญของโครงงาน 1
1.2 วัตถุประสงค์ของโครงงาน 1
1.3 ขอบเขตของโครงงาน 2
1.4 วิธีดำเนินการ 2
1.5 ประโยชน์ที่คาดว่าจะได้รับ 2
1.6 ตารางเวลาการทำโครงงาน 3
บทที่ 2 ทฤษฎีที่เกี่ยวข้อง 4
2.1 ทฤษฎีที่เกี่ยวข้อง 4
2.2 เครื่องมือที่ใช้ในการพัฒนา 22
บทที่ 3 ขั้นตอนการทำโครงงาน 24
3.1 ทำความเข้าใจการทำงานโปรแกรม Web GUI version 2 24
3.2 กำหนด test case ที่ต้องการทดสอบ 25
3.3 จัดทำเอกสารประกอบการทดสอบ 26
3.4 พัฒนาชุดทดสอบโปรแกรมตามที่ได้ออกแบบไว้ 26
3.5 นำชุดทดสอบโปรแกรมที่พัฒนามาทดสอบการทำงานของโปรแกรมและทำเอกสารบันทึกผลการทดสอบโปรแกรม 30
3.6 นำเสนอผลการพัฒนาชุดทดสอบโปรแกรม 30
บทที่ 4 ผลการดำเนินงาน 31
4.1 ผลของการพัฒนาชุดทดสอบโปรแกรม 31
4.2 ผลของการนำชุดทดสอบโปรแกรมไปใช้ทดสอบโปรแกรม Web GUI version 2 32
4.3 เอกสารประกอบการพัฒนาชุดทดสอบโปรแกรม 37

สารบัญ (ต่อ)

หน้า

บทที่ 5 สรุปผลการดำเนินงาน 38
5.1 สรุปผลการดำเนินการ 38
5.2 ปัญหาที่พบและแนวทางการแก้ไข 38
บรรณานุกรม 39

สารบัญตาราง

ตารางที่ **หน้า**

1.1 ตารางเวลาการทำโครงงาน 3
2.1 ข้อดี-ข้อเสีย Manual test 6
2.2 ข้อดี-ข้อเสีย Automate test 7

สารบัญภาพ

ภาพที่ **หน้า**

2.1 ประเภทของการทดสอบโปรแกรม 4
2.2 Unit Testing 5
2.3 source code ตัวอย่างการทดสอบ Unit test ภาพที่ 1 5
2.4 source code ตัวอย่างการทดสอบ Unit test ภาพที่ 2 6
2.5 โครงสร้างของ Unit test ที่ดี 8
2.6 สัญลักษณ์ของ JUnit version 5 9
2.7 source code แสดงการใช้ Parameterized ของ JUnit 10
2.8 source code แสดงการใช้ JUnitParams 11
2.9 source code แสดงการระบุข้อมูลที่ต้องการทดสอบ ไปยัง method 12
2.10 source code แสดงการระบุข้อมูลที่ต้องการทดสอบ ไปยัง class 12
2.11 source code แสดงการเขียน JUnit บน Eclipse 13
2.12 แสดงการสร้าง JUnit Test Case 14
2.13 แสดง dialog ในการสร้าง JUnit Test Case 15
2.14 source code เริ่มต้นหลังจากสร้าง JUnit Test Case 16
2.15 source code JUnit Test Case ที่แก้ไขแล้ว 16
2.16 ผลการรัน JUnit Test Case 17
2.17 source code ของคลาส MultiplyValue ที่แก้ไขแล้ว 17
2.18 ผลการรัน JUnit Test Case 18
2.19 สัญลักษณ์ของ Mockito 19
2.20 source code ของคลาส MyProfile 21
2.21 source code ของฟังก์ชัน main 21
2.22 console แสดงผลของการรันฟังก์ชัน main 22
2.23 สัญลักษณ์ของ Eclipse 22
2.24 สัญลักษณ์ของ Subversion 23
3.1 ภาพรวมของระบบ 24
3.2 ผังการทำงานของระบบ 25
3.3 source code แสดงตัวอย่างการใช้ Mockito 27
3.4 ผลการรันฟังก์ชัน testFunctionVolumeSquareBasedPyramid 28

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

3.5 source code แสดงการแก้ไขฟังก์ชัน areaSquare 29
3.6 ผลการรันฟังก์ชัน testFunctionVolumeSquareBasedPyramid 30
4.1 คลาสต่างๆที่ใช้ในการทดสอบฟังก์ชันการทำงานของโปรแกรม....31
4.2 คลาสต่างๆที่ใช้ในการทดสอบ Engine การทำงานของโปรแกรม....32
4.3 ผลการทดสอบการทำงานของคลาส TextField 32
4.4 ผลการทดสอบการทำงานของคลาส PanField....32
4.5 ผลการทดสอบการทำงานของคลาส NumericField 32
4.6 ผลการทดสอบการทำงานของคลาส DateField....33
4.7 ผลการทดสอบการทำงานของคลาส TextFieldParser 33
4.8 ผลการทดสอบการทำงานของคลาส PanFieldParser....33
4.9 ผลการทดสอบการทำงานของคลาส NumericFieldParser 33
4.10 ผลการทดสอบการทำงานของคลาส DateFieldParser....33
4.11 ผลการทดสอบการทำงานของคลาส TextFieldValidator....33
4.12 ผลการทดสอบการทำงานของคลาส PanFieldValidator 34
4.13 ผลการทดสอบการทำงานของคลาส NumericFieldValidator....34
4.14 ผลการทดสอบการทำงานของคลาส DateFieldValidator....34
4.15 ผลการทดสอบการทำงานของคลาส OccurrenceTextFieldConverter 34
4.16 ผลการทดสอบการทำงานของคลาส OccurrencePanFieldConverter....34
4.17 ผลการทดสอบการทำงานของคลาส OccurrenceNumericFieldConverter 34
4.18 ผลการทดสอบการทำงานของคลาส OccurrenceDateFieldConverter....35
4.19 ผลการทดสอบการทำงานของคลาส JsonTextFieldConverter 35
4.20 ผลการทดสอบการทำงานของคลาส JsonPanFieldConverter 35
4.21 ผลการทดสอบการทำงานของคลาส JsonNumericFieldConverter....35
4.22 ผลการทดสอบการทำงานของคลาส JsonDateFieldConverter....35
4.23 ผลการทดสอบการทำงานของคลาส ApiSelector 35
4.24 ผลการทดสอบการทำงานของคลาส JsonParser....36
4.25 ผลการทดสอบการทำงานของคลาส DataValidatorInput....36
4.26 ผลการทดสอบการทำงานของคลาส OccurrenceBuilder 36

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

4.27 ผลการทดสอบการทำงานของคลาส OccurrenceParser 36
4.28 ผลการทดสอบการทำงานของคลาส DataValidatorOutput 36
4.29 ผลการทดสอบการทำงานของคลาส JsonBuilder 36
4.30 รายการเอกสารประกอบการพัฒนาชุดทดสอบโปรแกรม 37

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

เนื่องด้วย บริษัท เอ็นทีที เดต้า (ประเทศไทย) จำกัด ซึ่งเป็นบริษัทให้บริการธุรกิจเกี่ยวกับระบบบัตรเครดิตทั้งในประเทศและต่างประเทศ มีความต้องการพัฒนาโปรแกรม Web GUI ขึ้นมาใหม่แทนโปรแกรม Web GUI เวอร์ชันเดิมที่ถูกพัฒนามาในรูปแบบการเขียนเว็บครอบจอการทำงาน ซึ่งโปรแกรม Web GUI ที่พัฒนาขึ้นมาใหม่นี้จะพัฒนาภายใต้การทำงานแบบ Model View Controller (MVC) โดยที่เลือกพัฒนาในรูปแบบนี้เนื่องจากตัวโปรแกรมอาจจะมีการเพิ่มฟังก์ชันในการทำงานหรือมีการปรับปรุงแก้ไขรายละเอียดของการใช้งานในอนาคต รวมไปถึงเพื่อรองรับการใช้งานในหลายๆแพลตฟอร์ม เพราะในปัจจุบันการใช้งานเทคโนโลยีต่างๆจะสามารถใช้งานได้หลายช่องทาง อาทิเช่น การใช้งานผ่านเว็บบราวเซอร์บนระบบปฏิบัติการต่างๆของคอมพิวเตอร์ การใช้งานผ่านโทรศัพท์มือถือ การใช้งานผ่านแท็บเล็ต

เมื่อบริษัทมีความต้องการพัฒนาโปรแกรมตัวใหม่ขึ้นมาทำให้ต้องมีการทำ Unit test suite เพื่อทดสอบการทำงานในแต่ละส่วนของโปรแกรมควบคู่ไปด้วย โดยทางบริษัทได้เลือกให้พัฒนาชุดทดสอบโปรแกรมซึ่งเป็น Automate Test มาใช้แทนการทดสอบแบบ Manual Test ทำให้เกิดโครงงานนี้ขึ้นมา โดยได้เลือกใช้ Junit framework มาใช้ในการเขียน Automate Test และใช้ Mockito framework มาช่วยในการจำลองข้อมูล

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อเป็นเครื่องมือในการตรวจสอบการทำงานของแต่ละส่วนของโปรแกรม Web GUI version 2
2) เพื่อศึกษาการใช้ Unit test มาตรวจสอบการทำงานของแต่ละส่วนภายในระบบ
3) เพื่อศึกษาการพัฒนา Automate test
4) เพื่อศึกษาการใช้ JUnit framework
5) เพื่อศึกษาการใช้ Mockito framework
6) เรียนรู้และสามารถจัดทำเอกสารประกอบการทดสอบส่วนของการทำงานในกรณีต่างๆ
7) เพื่อนำผลของการทดสอบส่วนการทำงานต่างๆของโปรแกรมไปปรับปรุงแก้ไขโปรแกรมให้มีประสิทธิภาพมากขึ้น
8) เพื่อประหยัดเวลาในการตรวจสอบการทำงานของแต่ละส่วนของโปรแกรม Web GUI version 2
9) เพื่อลดภาระการทำงานของบุคลากร

1.3 ขอบเขตของโครงงาน

1) พัฒนาชุดทดสอบโปรแกรมขึ้นมาตรวจสอบการทำงานของโปรแกรม Web GUI version 2 ได้ครบทุกส่วนในชั้น Business Layer
2) จัดทำเอกสารประกอบการทดสอบส่วนของการทำงานในกรณีต่างๆ ตามที่ได้พัฒนาชุดทดสอบขึ้นมาได้อย่างถูกต้องและครบถ้วน
3) ทำการทดสอบส่วนของการทำงานแต่ละส่วนของโปรแกรมโดยใช้ชุดทดสอบโปรแกรมที่พัฒนาขึ้นมาเพื่อแจ้งผลการทดสอบส่งไปให้ทีมพัฒนาโปรแกรม Web GUI version 2 ปรับปรุงโปรแกรมให้ทำงานได้อย่างถูกต้องและมีประสิทธิภาพมากขึ้น
4) ชุดทดสอบโปรแกรมที่พัฒนาขึ้นมาสามารถนำมาใช้ทดสอบการทำงานของโปรแกรม Web GUI version 2 ที่พัฒนาขึ้นมาใหม่เท่านั้น ไม่สามารถนำไปทดสอบการทำงานของโปรแกรม Web GUI ตัวเก่าได้

1.4 วิธีดำเนินการ

1) ศึกษาการทำงานในส่วนต่างๆของโปรแกรม Web GUI version 2
2) กำหนดขอบเขตของการทดสอบที่จะนำมาทดสอบโปรแกรม Web GUI version 2
3) กำหนด Error code ที่จะเกิดขึ้นในแต่ละส่วนของโปรแกรม Web GUI version 2 กับทีมพัฒนาโปรแกรม เพื่อระบุว่าเมื่อมีการทดสอบด้วยกรณีต่างๆ โปรแกรมควรจะแสดงผลอย่างไรเมื่อโปรแกรมทำงานสำเร็จ หรือแสดง Error code อย่างไรเมื่อรันโปรแกรมแล้วเกิดข้อผิดพลาด
4) วิเคราะห์และออกแบบว่าในแต่ละส่วนของโปรแกรมจะสามารถทดสอบในกรณีใดได้บ้าง โดยจัดทำเป็นเอกสารขึ้นมา
5) พัฒนาชุดทดสอบโปรแกรมขึ้นมาตามเอกสารที่ได้ออกแบบไว้
6) นำชุดทดสอบโปรแกรมที่พัฒนาขึ้นมา มาทดสอบการทำงานแต่ละส่วนของโปรแกรม Web GUI version 2
7) ทำเอกสารบันทึกผลของการทดสอบโปรแกรม เพื่อส่งให้ทีมพัฒนาโปรแกรมนำไปปรับปรุงโปรแกรมให้ทำงานได้อย่างถูกต้องและมีประสิทธิภาพมากขึ้น
8) นำเสนอผลการพัฒนาชุดทดสอบโปรแกรม

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) พัฒนาชุดทดสอบโปรแกรมที่สามารถนำไปให้ทดสอบการทำงานของโปรแกรม Web GUI version 2 ได้จริง
2) ทางบริษัท เอ็นทีที เดต้า (ประเทศไทย) จำกัด สามารถนำชุดสดทอบโปรแกรมที่พัฒนาขึ้นมาไปใช้ทดสอบการทำงานของโปรแกรม Web GUI version 2 เมื่อมีการแก้ไขการทำงานหรือมีการเพิ่มฟังก์ชันการทำงาน ได้ทั้งในปัจจุบันและอนาคต
3) ได้ความรู้ในการทำ Unit test suite เพื่อทดสอบการทำงานของโปรแกรม
4) ได้ความรู้และประสบการณ์การจากการพัฒนา Automate Test

5) ได้ความรู้และประสบการณ์ในการใช้ JUnit framework
6) ได้ความรู้และประสบการณ์ในการใช้ Mockito framework
7) ได้รับประสบการณ์ทำงานจริงจากการไปร่วมในทีมพัฒนาโปรแกรมของบริษัทที่ไปทำสหกิจศึกษา ซึ่งเป็นประสบการณ์ที่หาไม่ได้จากการเรียนในชั้นเรียนปกติ

1.6 ตารางเวลาการทำโครงงาน

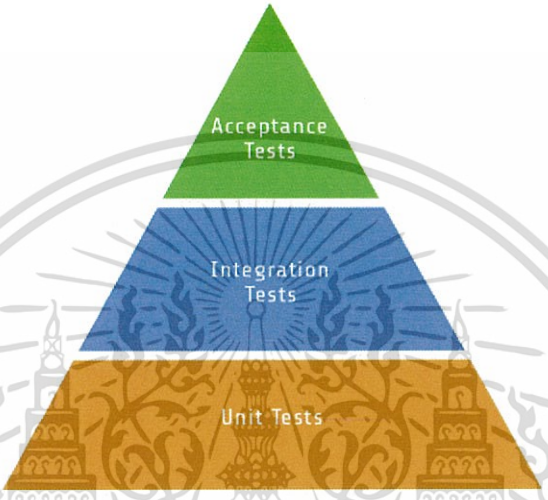
ตารางที่ 1.1 ตารางเวลาการทำโครงงาน

ลำดับ	หัวข้องาน	สิงหาคม				กันยายน				ตุลาคม				พฤศจิกายน			
1	ได้รับหัวข้อโครงงาน																
2	ได้รับคำอธิบายเนื้อหาของโครงงาน																
3	ทำความเข้าใจโปรแกรม Web GUI version 2																
4	กำหนด test case ที่ต้องการทดสอบระบบ																
5	พัฒนาชุดทดสอบโปรแกรม																
6	ทดสอบการทำงานของโปรแกรม																
7	จัดทำเอกสารและรูปเล่มโครงงาน																
8	นำเสนอโครงงาน																

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 การทดสอบโปรแกรม



ภาพที่ 2.1 ประเภทของการทดสอบโปรแกรม

ประเภทของการทดสอบโปรแกรมนั้นแบ่งได้เป็น 3 ประเภทใหญ่ๆ ได้แก่

- **Unit Test** เป็นการทดสอบการทำงานของแต่ละ Module โดยเป็นการทดสอบส่วนที่เล็กที่สุดของโปรแกรมว่าสามารถทำงานได้ถูกต้องหรือไม่
- **Integration Test** เป็นการทดสอบการทำงานร่วมกันของแต่ละ Module ว่าเมื่อนำหลายๆ Module มาทำงานร่วมกันแล้วจะสามารถทำงานได้ถูกต้องตามที่ได้ออกแบบไว้หรือไม่
- **Acceptance Test** เป็นการทดสอบการทำงานของระบบทั้งหมด ว่าสามารถทำงานได้ตามความต้องการของผู้ใช้งานหรือไม่ โดยปกติจะเป็นการทดสอบจากผู้ใช้งาน(user) ไม่ใช่การทดสอบจากผู้พัฒนา(developer)

2.1.2 Unit test

ภาพที่ 2.2 Unit Testing

Unit test คือการทดสอบ Code Unit หรือโดยทั่วไปเรียกว่า Function/Method โดยที่การทดสอบนั้นมีเพื่อเอาไว้ยืนยันว่าแต่ละ Unit นั้นทำงานได้ถูกต้อง โดย Unit Test ถือว่าเป็นการทดสอบที่เล็กที่สุดของการทำ Software Testing

การทำ Unit Test โดยทั่วไปจะทำแบบ Manual ก็ได้ เช่น การเรียกฟังก์ชันด้วย Input ที่แตกต่างกัน แสดงผลลัพธ์ออกมาทาง console แล้วตรวจสอบว่าผลลัพท์ที่ได้นั้นตรงกับผลลัพธ์ที่คาดหวังไว้หรือไม่ แต่ข้อเสียของวิธีนี้ก็คือเสียเวลา และอาจจะมีข้อผิดพลาดได้

ตัวอย่างการทดสอบ Unit test

สมมติมีฟังก์ชันการทำงานดังต่อไปนี้

```
int absolute(const int& a);
```

ภาพที่ 2.3 source code ตัวอย่างการทดสอบ Unit test ภาพที่ 1

เราสามารถทดสอบด้วยวิธี Manual โดยการป้อน Input เข้าไปแล้วดูผลลัพธ์ที่ console ว่าออกมาตรงกับผลลัพธ์ที่คาดหวังไว้หรือไม่

หรือเราอาจจะทดสอบด้วยวิธี Automate test โดยการเขียนโปรแกรมขึ้นมาเพื่อทดสอบฟังก์ชัน โดยอาศัย Unit testing framework ในการสร้างโปรแกรมขึ้นมาเป็นชุดการทดสอบ (Unit test suite) โดยตัวที่ได้รับความนิยมก็คือ JUnit ของภาษา Java ดังนี้

```
void test_absolute() {
  assert(absolute(10) == 10);
  assert(absolute(-5) == 5);
  assert(absolute(0) == 0);
}
```

ภาพที่ 2.4 source code ตัวอย่างการทดสอบ Unit test ภาพที่ 2

โค้ดด้านบนอธิบายได้ว่า เมื่อเรียกใช้ฟังก์ชัน absolute(int) ด้วย Input ที่ต่างกัน โดยคาดหวังว่าเมื่อใส่ Input ต่างๆ จะได้ Output ออกมาตามที่เราคาดหวังไว้ โดยใช้คำสั่ง assert(expect results, real result) เพื่อเปรียบเทียบผลลัพธ์จริงที่ออกมาจากการเรียกใช้ฟังก์ชันการทำงาน (real result) กับผลลัพธ์ที่เราคาดหวังไว้ (Expect result)

เมื่อตัว unit test ถูกเรียกใช้งาน ถ้าหากผลลัพธ์ที่เกิดขึ้นจริงตรงกับผลลัพธ์ที่คาดหวังไว้แสดงว่าฟังก์ชันทำงานถูกต้อง ในทางกลับกันถ้าหากว่ามี test case ไหนที่ผลลัพธ์ที่เกิดขึ้นจริงไม่ตรงกับผลลัพธ์ที่คาดหวังไว้ ฟังก์ชัน assert จะแสดง exception ออกมา เพื่อให้รู้ว่าฟังก์ชันทำงานไม่ถูกต้อง

2.1.3 วิธีการทดสอบโปรแกรม

วิธีการทดสอบโปรแกรมนั้นแบ่งได้เป็น 2 วิธี ได้แก่

1) **Manual Test** เป็นการทดสอบการทำงานของโปรแกรม โดยให้ tester เข้ามาทดสอบด้วยตัวเองเลย โดยที่ไม่ต้องเสียเวลาในการวางแผนการทดสอบมากนัก

ตารางที่ 2.1 ข้อดี-ข้อเสีย Manual test

Manual test	
ข้อดี	**ข้อเสีย**
เหมาะสำหรับการทดสอบ project ที่ไม่ต้อง run test บ่อยๆ	ลงทุนน้อยกว่า Automate test
เหมาะสำหรับการทำ ac-hoc test (การสุ่มทดสอบ โดยที่ไม่ได้ต้องมี test case ไว้ก่อน)	เมื่อ tester ต้องมาทดสอบโปรแกรมเดิมซ้ำๆ จะเกิดความเบื่อหน่าย
คุ้มค่าสำหรับ project ในระยะสั้น	

2) **Automate Test** เป็นการทดสอบโปรแกรมโดยการพัฒนาชุดทดสอบโปรแกรมขึ้นมาเพื่อทดสอบโปรแกรมอัตโนมัติตาม test case ที่เรากำหนดไว้ ว่าจะให้ทดสอบการทำงานส่วนไหน โดยใส่ข้อมูลอะไรลงไป เพื่อให้เกิดผลลัพธ์อย่างไร เพื่อแก้ปัญหาบางอย่างที่เกิดจากการทำ Manual test

ตารางที่ 2.2 ข้อดี-ข้อเสีย Automate test

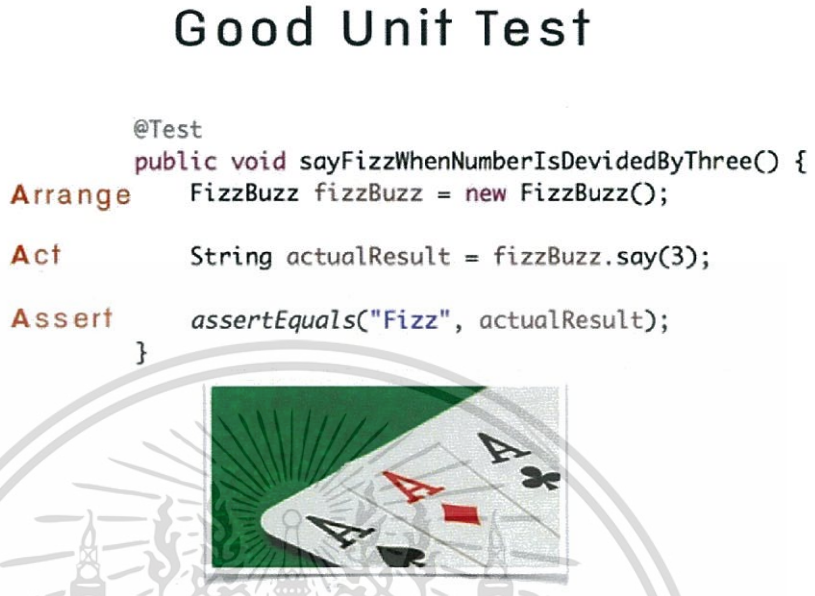
Automate test	
ข้อดี	**ข้อเสีย**
ช่วยในการ run test แบบเดิมซ้ำๆกันหลายๆครั้ง	ลงทุนสูงกว่า Manual test
ช่วยในการ run test โปรแกรมเดิมซ้ำๆโดยมีชุด configuration ที่ต่างกัน	ในบางอย่างเราไม่สามารถใช้ Automate test ได้ ก็ต้องใช้ Manual test ช่วย
ช่วยให้สามารถทำ regression test ได้อย่างรวดเร็ว	ต้องใช้คนที่มีทักษะการเขียนโปรแกรมในการสร้าง Automate test
ใช้เวลาในการ test น้อยกว่า Manual test	
คุ้มค่าสำหรับ project ระยะยาว	

แนวทางสำหรับการเขียน Unit test

แนวทางต่างๆที่ควรปฏิบัติสำหรับการเขียน Unit test มีดังนี้

- **Know what you're testing** ผู้เขียน Unit test จะต้องรู้ว่า Unit test นี้ต้องการจะทดสอบอะไร ไม่ควรทดสอบหลายๆกรณีใน Unit test เดียวกัน
- **Should be isolate** การทดสอบแต่ละ Unit test ควรจะทดสอบอย่างอิสระ ไม่ขึ้นกับการทำงานของส่วนอื่น อาทิเช่น Database, File system ถ้าจำเป็นให้ใช้การ Mock ข้อมูลขึ้นมาแทน
- **Should be deterministic** ถ้าทดสอบ Unit test ไหนผ่านแล้ว ควรจะทดสอบ Unit test นั้นผ่านทุกครั้ง
- **Use Isolation Frameworks** ถ้าการเตรียมข้อมูลที่จะนำมาทดสอบทำได้ยาก (มี dependency จำนวนมาก) isolate framework ช่วยในการเตรียมข้อมูล
- **Naming Conventions** ควรตั้งชื่อ method ของแต่ละ Unit test ให้สื่อความหมายว่าต้องการจะ test เรื่องใด

โครงสร้างของ Unit test ที่ดี



ภาพที่ 2.5 โครงสร้างของ Unit test ที่ดี

โครงสร้างของ Unit test ที่ดี มีหลายแบบ แต่แบบที่ต้องการนำเสนอคือ โครงสร้างแบบ 3A ซึ่งมีลักษณะ คือ

Arrange เป็นขั้นตอนของการเตรียมข้อมูลที่จะนำมาทดสอบการทำงานของโปรแกรม

Act เป็นขั้นตอนการเรียกใช้ฟังก์ชันที่ต้องการทดสอบ โดยเรียกใช้และป้อนข้อมูล Input ที่เตรียมไว้เข้าไป

Assert เป็นขั้นตอนการนำผลที่ได้รับจากการเรียกใช้ฟังก์ชันมาเปรียบเทียบกับผลที่เราคาดหวังไว้ก่อนการเรียกใช้โปรแกรม เพื่อตรวจสอบว่าฟังก์ชันที่เราเรียกใช้นั้นสามารถทำงานได้ตามวัตถุประสงค์ที่เรากำหนดไว้หรือไม่

2.1.4 JUnit framework

ภาพที่ 2.6 สัญลักษณ์ของ JUnit version 5

JUnit framework เป็น unit testing framework ที่พัฒนาโดย Kent Beck and Erich Gamma นิยมใช้กันในภาษา Java มี library อยู่ใน IDE ต่างๆไม่ว่าจะเป็น Eclipse, NetBeans, IntelliJ ปัจจุบันพัฒนามาถึงเวอร์ชัน 5

รูปแบบการเขียนด้วย JUnit framework

รูปแบบการเขียนมีหลายแบบ แต่จะยกตัวอย่างแนวทางการพัฒนา unit test ด้วย JUnit ตามแนวคิด Data Driven มาอธิบาย 3 แบบ ดังต่อไปนี้

1) การใช้ Parameterized ของ JUnit

เป็นความสามารถพื้นฐานที่มาพร้อมกับ JUnit มีการใช้งานดังนี้

- ในการทดสอบจะต้อง run ผ่าน Parameterized โดยสามารถประกาศผ่าน @RunWith(Parameterized.class)
- method สำหรับสร้างข้อมูลทดสอบ จะต้องใส่ @Parameters
- class ทดสอบจะต้องสร้าง constructor ที่มีจำนวน arguments เท่ากับข้อมูลที่ต้องการทดสอบ

ตัวอย่าง source code แสดงการใช้ Parameterized ของ JUnit

```
@RunWith(Parameterized.class)
public class MemberStatusTest {

   @Parameters(name = "{index}: {0} initially had {1} points, earns {2} points, should become {3} ")
   public static Iterable<Object[]> data() {
      return Arrays.asList(new Object[][]{
            {Regular,   0,   100,  Regular},
            {Regular,   0,    50,  Bronze},
            {Regular,   0,   200,  Silver},
            {Regular, 100,   100,  Silver},
            {Regular,   0,   500,  Gold},
            {Regular,   0,  1000,  Platinum},
      });
   }


      private String currentStatus;
      private int currentPoint;
      private int earnPoint;
      private String expectedStatus;

      public MemberStatusTest(String currentStatus, int currentPoint, int earnPoint, String expectedStatus) {
            this.currentStatus = currentStatus;
            this.currentPoint = currentPoint;
            this.earnPoint = earnPoint;
            this.expectedStatus = expectedStatus;
      }

      @Test
      public void memberShouldUpdateStatusBasedOnPoint() {
            LoyaltyMember loyaltyMember = LoyaltyMember.withMemberNumber("1234")
                        .named("Somkiat", "Puisungnoen")
                        .withCurrentPoint(this.currentPoint)
                        .withCurrentStatus(this.currentStatus);

            loyaltyMember.earnPoint(this.earnPoint);
            assertEquals(this.expectedStatus, loyaltyMember.getStatus());
      }
}
```

ภาพที่ 2.7 source code แสดงการใช้ Parameterized ของ JUnit

จากการใช้งานพบว่า ต้องเขียน code ขึ้นมามากพอสมควร รวมทั้งมีข้อจำกัดในการใช้งาน คือ จะทำงานเป็น class ไม่ใช่ test case ซึ่งเป็นข้อจำกัดที่ส่งผลให้ในแต่ละ class มีข้อมูลเพียงชุดเดียวเท่านั้น ซึ่งอาจจะไม่สะดวกในการใช้งานมากนัก ดังนั้นจึงมีคนสร้าง library มาใช้กัน หนึ่งในนั้น คือ JUnitParams

2) การใช้ JUnitParams

เป็น library สร้างขึ้นมา เพื่อให้เขียน unit test ได้สะดวกมากยิ่งขึ้น โดย library อยู่ที่ https://code.google.com/archive/p/junitparams/

การใช้งาน จะเปลี่ยนไปจากวิธีการใช้ Parameterized ของ JUnit ไม่มากนัก แต่สามารถกำหนดข้อมูลทดสอบให้แต่ละ test case ได้เลย โดยมีการใช้งานดังนี้

- เปลี่ยนตัว Runner เป็นตัวใหม่ก็คือ JUnitParamsRunner
- ในแต่ละ test case สามารถระบุ Parameters ซึ่งต่างจากวิธีการใช้ Parameterized ของ JUnit

ตัวอย่าง source code แสดงการใช้ JUnitParams

```
@RunWith(JUnitParamsRunner.class)
public class MemberStatusWithJUnitParamsTest {

    @Test
    @Parameters({
      "Regular,  0,  100,  Regular",
      "Regular,  0,   50,  Bronze",
      "Regular,  0,  200,  Silver",
      "Regular, 100,  100,  Silver",
      "Regular,  0,  500,  Gold",
      "Regular,  0, 1000,  Platinum",
    })
    public void memberShouldUpdateStatusBasedOnPoint(String currentStatus, int currentPoint, int earnPoint, String expectedStatus) {
        LoyaltyMember loyaltyMember = LoyaltyMember.withMemberNumber("1234")
                    .named("Somkiat", "Puisungnoen")
                    .withCurrentPoint(currentPoint)
                    .withCurrentStatus(currentStatus);

        loyaltyMember.earnPoint(earnPoint);
        assertEquals(expectedStatus, loyaltyMember.getStatus());
    }
}
```

ภาพที่ 2.8 source code แสดงการใช้ JUnitParams

จากข้างต้นจะเห็นว่าสามารถกำหนดข้อมูลสำหรับการทดสอบแต่ละ test case ได้เลย

3) การใช้ JUnitParams โดยดึงข้อมูลจาก class/method อื่นๆ

จากวิธีการใช้ JUnitParams จะเกิดปัญหาเมื่อข้อมูลที่นำมาทดสอบอยู่ในไฟล์ เช่น CSV, Excel เป็นต้น จึงทำให้เกิดการเตรียมข้อมูลไว้อีก 2 วิธี ได้แก่

3.1) ระบุข้อมูลที่ต้องการทดสอบ ไปยัง method

```
@RunWith(JUnitParamsRunner.class)
public class MemberStatusWithJUnitParamsExternalTest {

	@Test
	@Parameters( method="sampleData")
	public void memberShouldUpdateStatusBasedOnPoint(String currentStatus, int currentPoint, int earnPoint, String expectedStatus) {
		LoyaltyMember loyaltyMember = LoyaltyMember.withMemberNumber("1234")
				.named("Somkiat", "Puisungnoen")
				.withCurrentPoint(currentPoint)
				.withCurrentStatus(currentStatus);

		loyaltyMember.earnPoint(earnPoint);
		assertEquals(expectedStatus, loyaltyMember.getStatus());
	}

	private Object[] sampleData() {
		return $(
			$(Regular,   0,   100,  Regular),
			$(Regular,   0,    50,  Bronze),
			$(Regular,   0,   200,  Silver),
			$(Regular, 100,   100,  Silver),
			$(Regular,   0,   500,  Gold),
			$(Regular,   0,  1000,  Platinum)
		);
	}
}
```

ภาพที่ 2.9 source code แสดงการระบุข้อมูลที่ต้องการทดสอบ ไปยัง method

จะสังเกตเห็นว่า JUnitParams ได้เตรียม method $ มาให้ สำหรับการแปลงข้อมูลที่ต้องการทดสอบ ไปยัง array ของ object

3.2) ระบุข้อมูลที่ต้องการทดสอบ ไปยัง class

มีข้อกำหนดว่า method ใน class จะต้องมีรูปแบบ ดังนี้ public static Object[] provideXXX(){ } และชื่อ method ต้องขึ้นต้นด้วยคำว่า provide และ return เป็น array ของ Object

ตัวอย่าง source code แสดงการระบุข้อมูลที่ต้องการทดสอบ ไปยัง class

```
@RunWith(JUnitParamsRunner.class)
public class MemberStatusWithJUnitParamsExternalClassTest {

	@Test
	@Parameters( source=MemberDataTest.class)
	public void memberShouldUpdateStatusBasedOnPoint(String currentStatus, int currentPoint, int earnPoint, String expectedStatus) {
		LoyaltyMember loyaltyMember = LoyaltyMember.withMemberNumber("1234")
				.named("Somkiat", "Puisungnoen")
				.withCurrentPoint(currentPoint)
				.withCurrentStatus(currentStatus);

		loyaltyMember.earnPoint(earnPoint);
		assertEquals(expectedStatus, loyaltyMember.getStatus());
	}
}
```

ภาพที่ 2.10 source code แสดงการระบุข้อมูลที่ต้องการทดสอบ ไปยัง class

ตัวอย่างการเขียน JUnit บน Eclipse

ด้านล่างจะแสดงตัวอย่างการเขียน JUnit บน Eclipse โดยจะใช้โครงสร้างนี้สำหรับการพัฒนาชุดสอบโปรแกรม ที่จะนำมาใช้ทดสอบโปรแกรม Web GUI version 2 สำหรับโครงงานชิ้นนี้

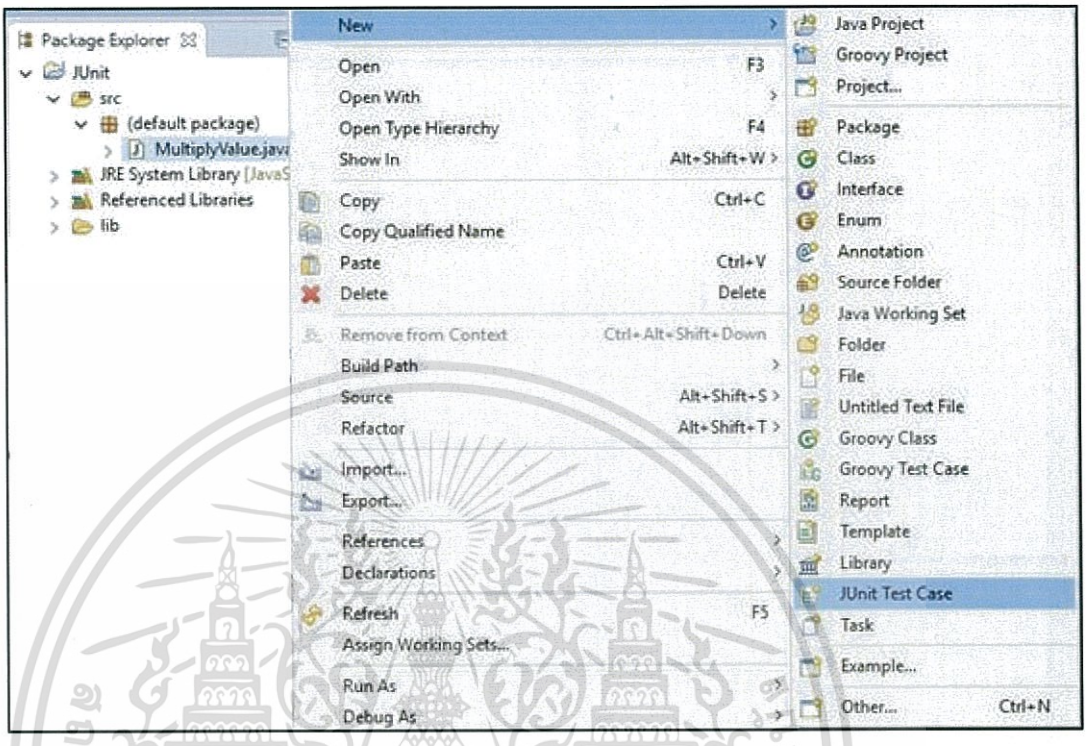
โดยจะทำการทดลองสร้างคลาส MultiplyValue.class ซึ่งเป็นคลาสที่ใช้ในการคำนวณผลของการคูณเลข 2 จำนวน และสร้างคลาส Test โดยใช้ JUnit เพื่อตรวจสอบผลการทำงานของคลาส MultiplyValue.class มีวิธีการดังต่อไปนี้

1) สร้าง Java Project ขึ้นมาแล้วสร้างคลาส MultiplyValue ดังนี้

```
2  public class MultiplyValue {
3      public int multiplyValaue(int value1, int value2){
4          int result;
5
6          result = value1 + value2;
7
8          return result;
9      }
10 }
```

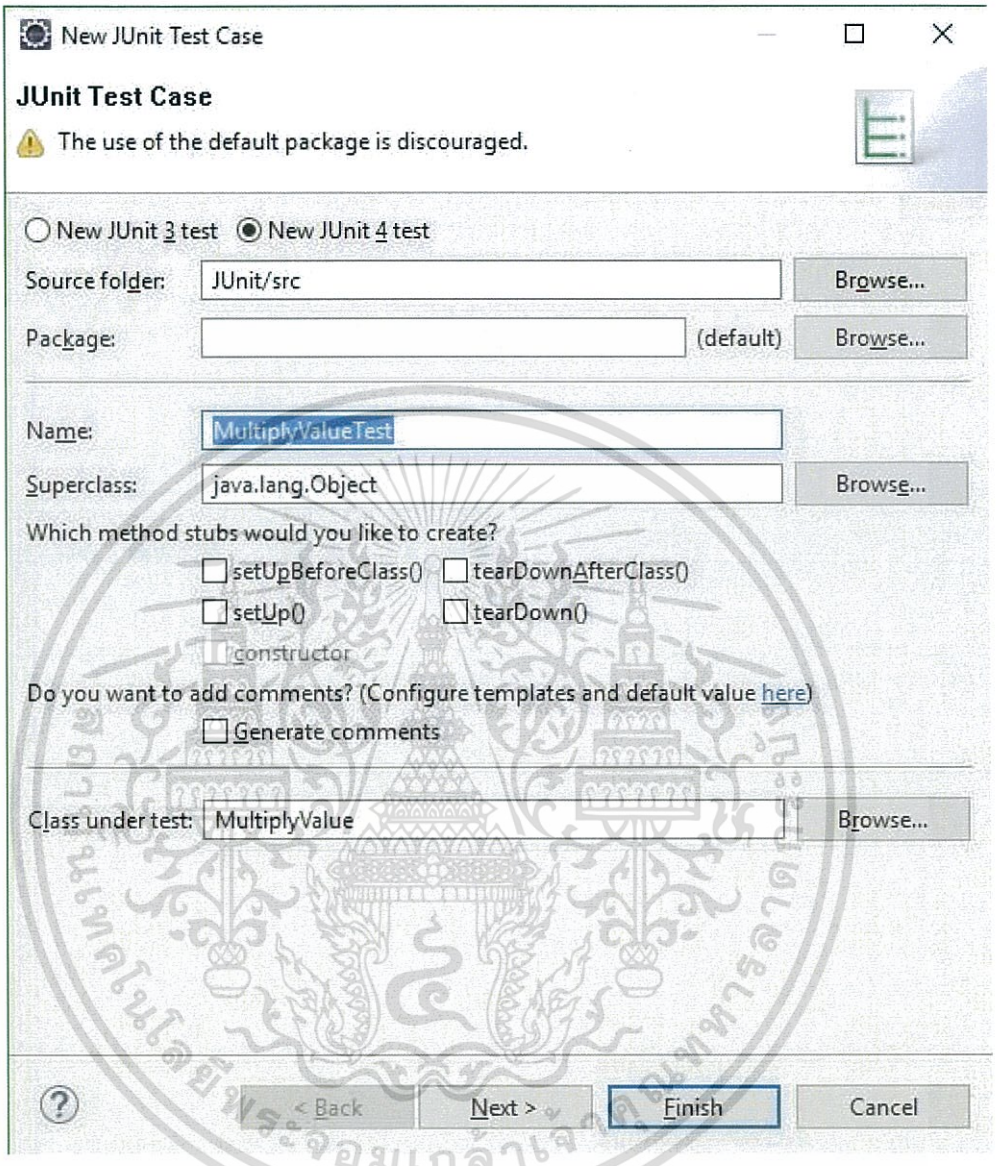
ภาพที่ 2.11 source code แสดงการเขียน JUnit บน Eclipse

2) ที่หน้าต่าง Package Explorer คลิกขวาที่ชื่อ MultiplyValue->New->JUnit Test Case



ภาพที่ 2.12 แสดงการสร้าง JUnit Test Case

จะมี dialog แสดงขึ้นดังนี้



ภาพที่ 2.13 แสดง dialog ในการสร้าง JUnit Test Case

จะได้คลาส ดังนี้

```
1⊕ import static org.junit.Assert.*;
4
5  public class MultiplyValueTest {
6
7⊖      @Test
8       public void test() {
9           fail("Not yet implemented");
10      }
11
12  }
```

ภาพที่ 2.14 source code เริ่มต้นหลังจากสร้าง JUnit Test Case

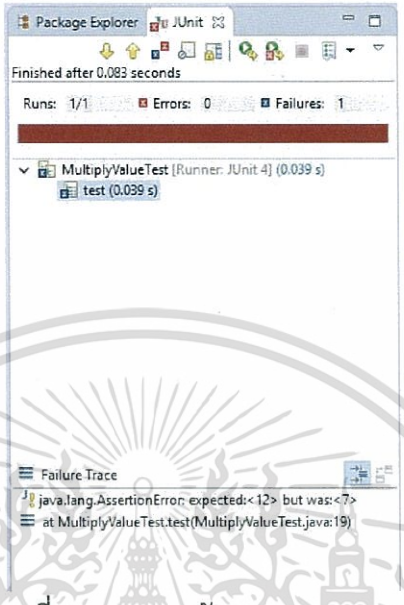
3) แก้ไข Test Case ใหม่ ดังนี้

```
1⊕ import static org.junit.Assert.*;
4
5  public class MultiplyValueTest {
6
7⊖      @Test
8       public void test() {
9           /** Prepare Data*/
10          int value1 = 3;
11          int value2 = 4;
12          int result;
13
14          /** Execute Function*/
15          MultiplyValue multiplyValue = new MultiplyValue();
16          result = multiplyValue.multiplyValaue(value1, value2);
17
18          /** Check Result*/
19          assertEquals(12,result);
20      }
21 }
```

ภาพที่ 2.15 source code JUnit Test Case ที่แก้ไขแล้ว

อธิบาย code คือเรียกใช้ฟังก์ชัน multiplyValue(int value1, int value2) ด้วยการส่ง Input เป็น 3 และ 4 และใช้คำสั่ง assertEquals(long expected, long actual) เพื่อเปรียบเทียบผลลัพธ์ที่คาดหวังเอาไว้ ในที่นี้คือ 12 กับผลลัพธ์ที่ออกมาจากการเรียกใช้ฟังก์ชันว่ามีค่าเท่ากันหรือไม่

4) เมื่อรันคลาส Test ด้วยการคลิกขวาที่คลาสแล้วเลือก Run As -> JUnit Test จะปรากฏหน้าจอ ดังนี้



ภาพที่ 2.16 ผลการรัน JUnit Test Case

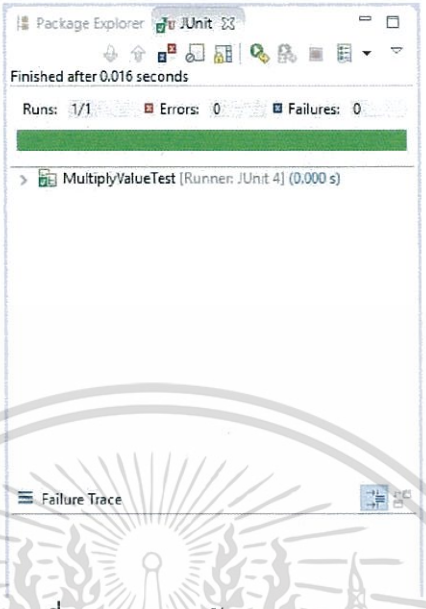
จากผลการทดสอบแสดงว่าคลาส MultiplyValue ทำงานผิดพลาด

5) เมื่อมาตรวจสอบดู code ของคลาส MultiplyValue พบว่ามี code การทำงานที่ผิด เนื่องจากเป็นฟังก์ชันการคูณตัวเลข ในส่วนของการคำนวณจึงควรจะเป็น value1 * value2 ไม่ใช่ value1 + value2 จึงแก้ไขให้ถูกต้อง เป็นดังนี้

```
1
2  public class MultiplyValue {
3      public int multiplyValaue(int value1, int value2){
4          int result;
5
6          result = value1 * value2;
7
8          return result;
9      }
10 }
```

ภาพที่ 2.17 source code ของคลาส MultiplyValue ที่แก้ไขแล้ว

6) จากนั้นกลับไปรัน Test Case ใหม่ได้ผลลัพธ์เป็นดังนี้



ภาพที่ 2.18 ผลการรัน JUnit Test Case

7) จากผลการทดสอบแสดงว่าคลาส MultiplyValue ทำงานได้ถูกต้องตามที่เราคาดหวังไว้แล้ว

จะเห็นว่าการใช้ JUnit มาช่วยในการทดสอบจะสามารถทำให้ได้ผลการทดสอบที่ถูกต้องและเหมือนกันในทุกๆครั้ง จึงได้นำแนวทางและโครงสร้างนี้มาใช้ในการพัฒนาชุดทดสอบที่จะนำไปทดสอบตัวโปรแกรม Web GUI version2 ต่อไป

2.1.5 Test Isolation

Test Isolation คือ อิสรภาพทางการทดสอบ การเขียน Unit test ที่ดีควรจะต้องเขียน code test ที่เป็นอิสระจาก code ของโปรแกรมที่พัฒนาขึ้นมาจริงๆ เช่น Database, Network เพื่อให้การทดสอบมีความคล่องตัว และเพื่อไม่ให้การทำงานของส่วนที่ไม่ได้ทดสอบส่งผลต่อส่วนที่ต้องการทดสอบ

ปัญหาที่พบจากการเขียน code test ที่มี Dependency สูงๆมีมากมาย ได้แก่

- ในส่วนของ Database ระบบไฟล์ และ API นั้น Tester ไม่สามารถควบคุมและตรวจสอบได้เลยสำหรับปัญหาที่เกิดขึ้น อาทิเช่น API Server ล่ม การทดสอบก็จะผิดพลาดไปด้วย
- ในการทำงานที่เกี่ยวกับ Database ระบบไฟล์ และ API จะทำให้การทำการทดสอบช้าลงอย่างมาก แต่เป้าหมายของการทำ Unit test คือการนำมาทดสอบอยู่บ่อยๆ จึงควรทดสอบได้อย่างรวดเร็ว เพื่อจะได้นำผลจากการทดสอบไปปรับปรุงแก้ไขโปรแกรมให้ทำงานได้อย่างถูกต้อง

- การทำงานที่ต้องรับข้อมูลจาก Document เราไม่สามารถควบคุมให้ Document นั้นเหมือนเดิมในทุกๆครั้งที่ทดสอบ ไม่ถูกแก้ไข ทำให้ผลการทดสอบออกมาคลาดเคลื่อนได้

วิธีแก้ไขปัญหาดังกล่าว ทำได้โดยการจำลองข้อมูลที่จะใช้ทดสอบขึ้นมาเอง (Mock Object) ให้เป็นอิสระจากฟังก์ชันการทำงานอื่นๆ โดยอาจจะใช้ Library เข้ามาช่วย อาทิเช่น

- Mockito
- EasyMock
- jMock

ซึ่งเป็น Library ที่ถูกพัฒนาขึ้นมาช่วยในการจำลองข้อมูลสำหรับทำ Unit test โดยเฉพาะ เพื่อทำให้สามารถแยกส่วนต่างๆ ของระบบออกมาทดสอบได้ (isolation)

2.1.6 Mockito framework

ภาพที่ 2.19 สัญลักษณ์ของ Mockito

Mockito เป็น library mocking framework ที่พัฒนามาในภาษา Java ใช้ในการสร้าง unit test ช่วยจำลองข้อมูลเพื่อให้เป็นอิสระจากการทำงานของคลาสที่ไม่ได้ทดสอบ

ประโยชน์จากการ Mock Object

เมื่อใช้การ Mock Object มาแทนการเรียกใช้ Database, Web server และระบบอื่นๆที่เกี่ยวข้องกับการทำงาน แต่ไม่ใช่ส่วนที่ต้องการทดสอบ จะได้ประโยชน์ดังนี้

- ทำการทดสอบเร็วขึ้น
- ทำให้การทดสอบไม่อ่อนไหวต่อความผิดพลาด ที่อาจเกิดขึ้นจากการ configuration ของ object ที่ถูก Mock ขึ้นมา
- ทำให้ง่ายต่อการทดสอบในทุกๆกรณี
- ในการสร้าง Mock object นั้น จะไม่ส่งค่าของ Mock object อื่นๆ กลับมาโดยเด็ดขาด

และประโยชน์ที่สำคัญที่สุดของการ Mock Object คือ จะทำให้ทราบขอบเขตของสถาปัตยกรรมของระบบ เมื่อมองเห็นขอบเขตการทำงานที่ชัดเจนแล้ว จะทำให้สามารถสร้าง interface ของส่วนการทำงานนั้นๆขึ้นมาเพื่อแบ่งงานกันทำแล้วนำแต่ละส่วนที่แบ่งกันทำมาประกอบกันได้โดยไม่เกิดปัญหา ส่งผลให้การจัดการระบบทำได้ง่าย คือ ง่ายต่อการพัฒนา และ ติดตั้งระบบงานเนื่องจากแต่ละส่วนเป็นอิสระต่อกัน

เมื่อใดที่ควรใช้การ Mock Object

เหตุผลต่างๆที่ควรใช้การ Mock Object ได้แก่

เมื่อ real object มีพฤติกรรมที่ยากต่อการคาดเดา
เมื่อ real object สร้างหรือกำหนดค่าขึ้นมาได้ยาก
เมื่อ real object ทำงานช้า
เมื่อ real object คือ User interface ของระบบ

เครื่องมือต่างๆของ Mockito

เครื่องมือที่สามารถใช้ได้ใน library ของ Mockito มีหลายเครื่องมือโดยเครื่องมือพื้นฐาน ได้แก่

Mock เพื่อจำลอง Object ขึ้นมาทั้ง Object โดยไม่ได้สร้าง Object ขึ้นมาจริงๆ สามารถควบคุมการทำงานของ Object ที่ Mock ขึ้นมาด้วยคำสั่ง when()
Spy เป็นการสร้าง Object นั้นขึ้นมาจริงๆ แต่สามารถควบคุมการทำงานของ Object ที่ Mock ขึ้นมาด้วยคำสั่ง when() เช่นเดียวกับ Mock
Verify ใช้เพื่อตรวจสอบว่าระหว่างการทำงานของโปรแกรมมีการเรียกใช้ Method นั้นๆหรือไม่

ตัวอย่างการใช้ Mockito ช่วยในการจำลอง Object สำหรับการเขียน Unit test

ด้านล่างจะแสดงตัวอย่างการใช้ Mockito เพื่อจำลองข้อมูล มีวิธีการดังนี้

1) สร้างคลาส MyProfile ซึ่งเป็นคลาสที่มี field name และ field phone ชนิด String และมี Method Set/Get สำหรับทั้ง 2 field ดังภาพ

```
3  public class MyProfile {
4    private String name = null;
5    private String phone = null;
6
7    public String getName() {
8      return name;
9    }
10   public void setName(String name) {
11     this.name = name;
12   }
13   public String getPhone() {
14     return phone;
15   }
16   public void setPhone(String phone) {
17     this.phone = phone;
18   }
19 }
```

ภาพที่ 2.20 source code ของคลาส MyProfile

2) ในฟังก์ชัน main เขียนโค้ดเพื่อเรียกใช้ฟังก์ชัน profile() และฟังก์ชัน profileMockito() ซึ่งทั้ง 2 ฟังกัชันเป็นฟังก์ชันสำหรับแสดงผลหมายเลขโทรศัพท์ลงบน console ทั้งคู่แต่การทำงานต่างกัน คือ

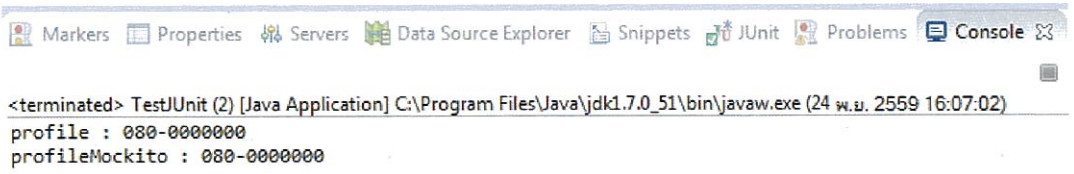
profile() เป็นฟังก์ชันที่เรียกใช้คำสั่ง setPhone() เพื่อกำหนดค่า field phone และใช้คำสั่ง getPhone() ในการดึงค่าของ field phone ออกมาแสดงผล

profileMockito() ใช้การ mock object จำลอง object profile ขึ้นมา และใช้คำสั่ง when ไปควบคุมการทำงานว่าเมื่อใดก็ตามที่ object profile เรียกใช้ฟังก์ชัน getPhone() ให้ return ข้อมูล “080-0000000” ออกมา ซึ่งได้ source ดังภาพ

```
10   public static void main(String[] args) {
11     profile();
12     profileMockito();
13   }
14
15   public static void profile(){
16     MyProfile profile = new MyProfile();
17     profile.setPhone("080-0000000");
18
19     System.out.println(profile.getPhone());
20   }
21
22   public static void profileMockito(){
23     MyProfile profile = mock(MyProfile.class);
24     when(profile.getPhone()).thenReturn("080-0000000");
25
26     System.out.println(profile.getPhone());
27   }
```

ภาพที่ 2.21 source code ของฟังก์ชัน main

3) เมื่อรันฟังก์ชัน main จะได้ผลลัพธ์ที่ console ดังภาพ



ภาพที่ 2.22 console แสดงผลของการรันฟังก์ชัน main

จะเห็นว่าการใช้ Mockito มาจำลอง object จะให้ผลการทดสอบคลาสที่ต้องการเหมือนกับการสร้าง object จริง แต่การให้ Mockito ทำให้การทดสอบที่แน่นอนกว่าการสร้าง object จริงก็ต่อเมื่อการทำงานของส่วนที่ไม่ได้ทดสอบมีความซับซ้อนมาก ซึ่งระหว่างการสร้าง object ขึ้นมาอาจเกิดความผิดพลาดทำให้ส่งผลต่อผลการทดสอบของคลาสที่นำมาทดสอบ

2.2 เครื่องมือที่ใช้ในการพัฒนา

2.2.1 Eclipse

ภาพที่ 2.23 สัญลักษณ์ของ Eclipse

Eclipse เป็นเครื่องมือที่เรียกว่า integrated development environment (IDE) สำหรับพัฒนา applications โดยใช้ Java หรือภาษาอื่นๆเช่น C/C++, Python, PERL, Ruby ฯลฯ

Eclipse มีการรองรับปลั๊กอินที่หลากหลาย ผู้พัฒนาที่ใช้ภาษา Java ในการพัฒนา application ต่างๆ สามารถใช้ Eclipse ในการพัฒนาได้ โดยตัว Eclipse มีสภาวะแวดล้อมที่สมบูรณ์ คือมีเครื่องมือต่างๆให้ใช้มากมาย นอกจากนี้ Eclipse ยังสามารถใช้พัฒนาโปรแกรมภาษาอื่นๆได้ถ้ามีตัวปลั๊กอินนั้นอยู่ เช่น ถ้าเราต้องการพัฒนา application โดยใช้ภาษา PHP ถ้า Eclipse มีปลั๊กอินสำหรับภาษานี้ เราสามารถใช้ Eclipse ในการพัฒนาได้

Eclipse และ ปลั๊กอินต่างๆ ของ Eclipse พัฒนาอยู่ภายใต้ Eclipse Public License (EPL) เพื่อให้ Eclipse สามารถดาวน์โหลด และติดตั้งได้ฟรี นอกจากนี้ยังสามารถปรับปรุงแก้ไขและนำไป

จัดจำหน่ายได้ MVC ย่อมาจาก Model View Controller เป็นสถาปัตยกรรมซอฟแวร์ซึ่งในปัจจุบันเว็บแอพพลิเคชันเกือบทั้งหมดมีโครงสร้างแบบ MVC โดยแต่ละเว็บแอพพลิเคชันจะถูกแบ่งออกเป็นสามส่วน ได้แก่ Model เป็นส่วนที่ใช้ติดต่อฐานข้อมูลทำหน้าที่ดึงข้อมูลจากฐานข้อมูลมาจัดการให้อยู่ในรูปแบบที่เหมาะสม, View เป็นส่วนที่จะนำข้อมูลจากโมเดลไปแสดงผลให้ผู้ใช้เห็นผลลัพธ์ออกมาใน User Interface, Controller เป็นส่วนที่คอยรับ Input จาก Client เข้ามาแล้วนำคำสั่งไปประมวลผลเพื่อสั่งงาน View และ Model ให้ประมวลผลออกมา

2.2.2 SVN (Subversion)

ภาพที่ 2.24 สัญลักษณ์ของ Subversion

SVN คือ โปรแกรมที่ทำหน้าที่จัดการกับ Version Control ของไฟล์ต่างๆไม่ว่าจะเป็นการจัดการกับเอกสาร Document , ไฟล์รูปภาพ และที่สำคัญที่สุดก็คือ จัดการเกี่ยวกับ Source Code ของโปรแกรม โดยหน้าที่ของ SVN คือจัดเก็บไฟล์ต่างๆเหล่านั้นไว้ในคลัง แล้วแยกไฟล์นั้นเป็นเวอร์ชั่นต่างๆ โดยเราเรียกมันว่า Revision ซึ่งเวอร์ชั่นของไฟล์ที่จัดเก็บนั้นจะเกี่ยวข้องกับ User หลายๆผู้ใช้งาน ซึ่งประโยชน์ของ SVN ก็คือโดยปกติแล้ว SVN จะมี Server ทำหน้าที่จัดเก็บไฟล์ ฉะนั้นไฟล์ต่างๆจะถูกจัดเก็บไว้บน Server และเรียกใช้งานผ่าน Protocol : TCP/HTTP ฉะนั้น SVN Server ที่ทำหน้าที่จัดเก็บไฟล์ จะเป็นตัวกลางในการแลกเปลี่ยน Version ของไฟล์ ซึ่งวิธีนี้จะเป็นประโยชน์มากในการป้องกันไฟล์หาย และจะได้ไฟล์ล่าสุดเสมอ เมื่อทำการ Checkout หรืออัพเดตจาก SVN Server และประโยชน์อื่นๆของ SVN เช่น ดู Log หรือ History ของการแก้ไขไฟล์ และ ยังสามารถนำ Revision ของไฟล์กลับมาใช้งานได้ ในกรณีที่ต้องการกลับไปใช้ Version เก่า

ในปัจจุบัน SVN ได้รับความนิยมอย่างมากมาย โดยเฉพาะโปรเจค Open Source ในต่างประเทศก็ใช้ SVN เป็น Source Version Control เข้ามาจัดการกับการทำงานเป็นทีม และในเมืองไทยในหลายๆบริษัทก็ใช้กันเกือบทุกๆที่ที่มีการพัฒนาโปรแกรมที่มีหลายๆคนเข้ามาเกี่ยวข้องส่วนหนึ่งเพราะ SVN เป็น Open Source ที่สามารถใช้งานได้ฟรี โปรแกรมมีขนาดเล็ก อีกทั้งยังมี Plugin อีกมากมายที่รองรับบน IDE ร่วมกับการพัฒนาโปรแกรมหลายๆประเภท เช่น PHP , Java หรือจะเป็นโปรแกรมบน .Net Application ที่ได้รับความนิยมมากในปัจจุบัน

บทที่ 3
ขั้นตอนการทำโครงงาน

บริษัท เอ็นทีที เดต้า (ประเทศไทย) จำกัด มีความต้องการพัฒนาโปรแกรม Web GUI ขึ้นมาใหม่ ผู้จัดทำจึงได้รับมอบหมายให้รับผิดชอบในส่วนของการทำ Unit test suite เพื่อทดสอบการทำงานของโปรแกรม Web GUI version 2 ที่พัฒนาขึ้น โดยพัฒนาชุดทดสอบโปรแกรมขึ้นมาโดยใช้ JUnit framework ควบคู่กับ Mockito framework โดยมีขั้นตอนการทำงาน ดังต่อไปนี้

3.1 ทำความเข้าใจการทำงานโปรแกรม Web GUI version 2

3.2 กำหนด test case ที่ต้องการทดสอบ

3.3 จัดทำเอกสารประกอบการทดสอบ

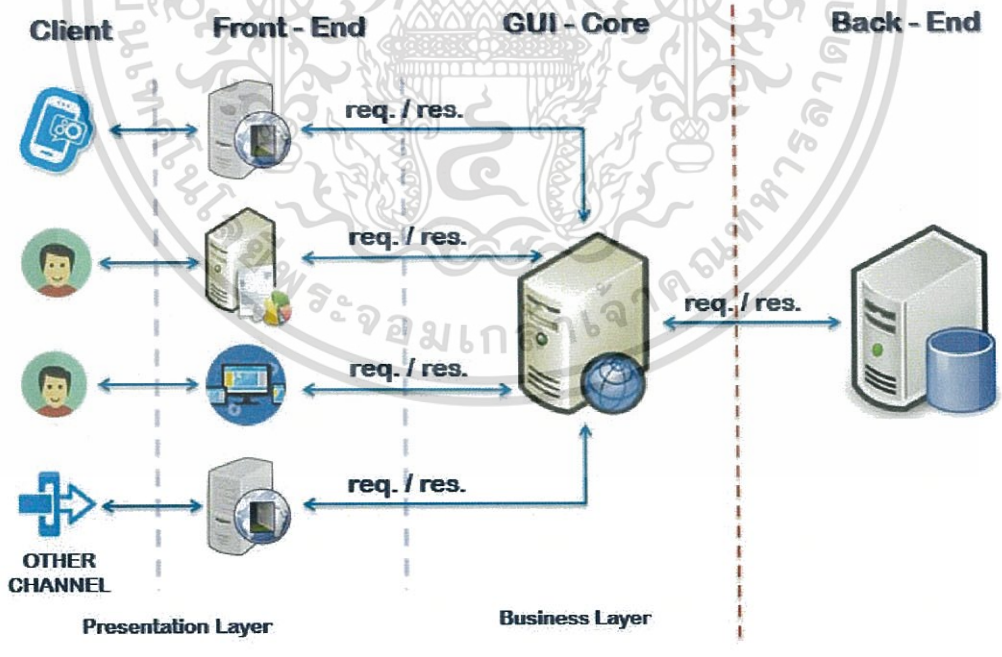
3.4 พัฒนาชุดทดสอบโปรแกรมตามที่ได้ออกแบบไว้

3.5 นำชุดทดสอบโปรแกรมที่พัฒนามาทดสอบการทำงานของโปรแกรมและบันทึกผลการทดสอบโปรแกรม

3.6 นำเสนอผลการพัฒนาชุดทดสอบโปรแกรม

3.1 ทำความเข้าใจการทำงานโปรแกรม Web GUI version 2

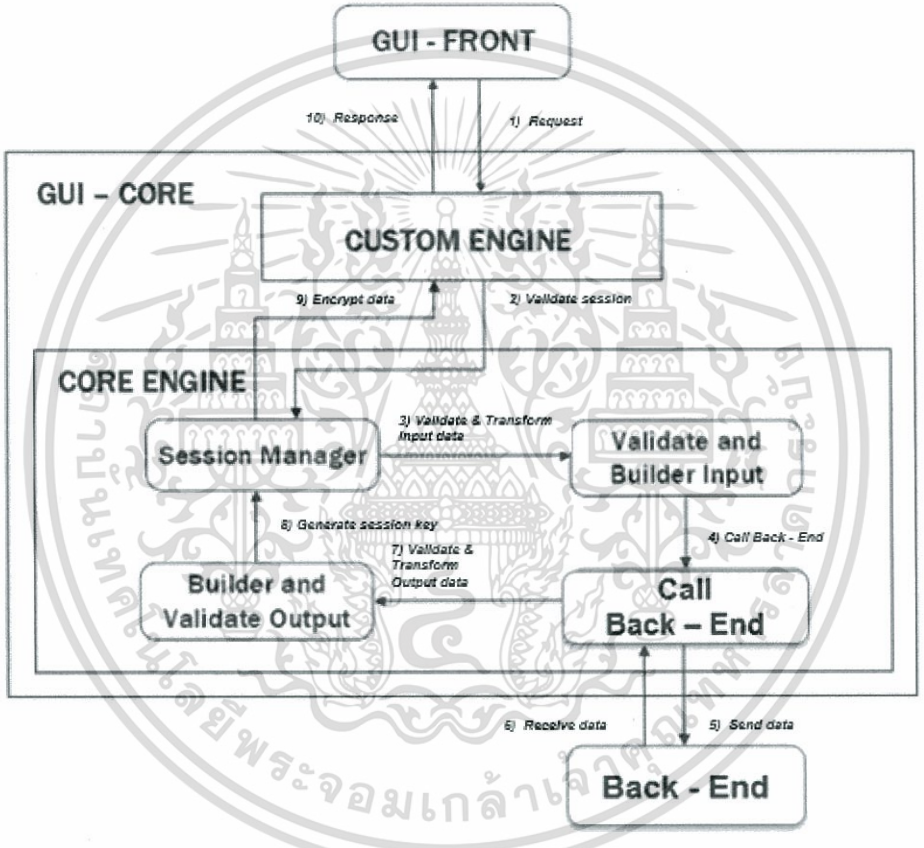
3.1.1 ภาพรวมระบบ



ภาพที่ 3.1 ภาพรวมของระบบ

ส่วนที่ผู้จัดทำได้รับมอบหมายให้พัฒนาชุดทดสอบโปรแกรมเป็นส่วนการทำงานในชั้น Business Layer มีหน้าที่เป็นตัวกลางรับส่งข้อมูลระหว่างชั้น Presentation Layer กับส่วนของ Back-End ซึ่งเป็นโปรแกรมการทำงานเกี่ยวกับระบบบัตรเครดิตที่ใช้ภายในบริษัท โดยเหตุผลที่มีชั้น Business Layer ขึ้นมาเพื่อทำให้โครงสร้างของระบบเป็นแบบ MVC และรองรับการใช้งานของผู้ใช้งานในหลายๆ Platform เพื่อจัดเตรียมข้อมูลให้พร้อมสำหรับการเรียกใช้ฟังก์ชันการทำงานที่ส่วนของ Back-End

3.1.2 ผังการทำงานของระบบ



ภาพที่ 3.2 ผังการทำงานของระบบ

ภาพด้านบนเป็นภาพแสดงผังการทำงานของระบบโดยเป็นผังการทำงานในระดับ High - level ไม่ได้แสดงถึงระดับที่เล็กที่สุดของระบบเนื่องจากโปรแกรมดังกล่าวเป็นผลิตภัณฑ์ของบริษัท ทำให้ไม่สามารถเปิดเผยรายละเอียดได้

3.2 กำหนด test case ที่ต้องการทดสอบ

หลังจากทำความเข้าใจระบบที่จะทดสอบแล้ว จึงได้ออกแบบ case ที่จะทดสอบในแต่ละฟังก์ชันการทำงานของโปรแกรม ว่าในแต่ละฟังก์ชันการทำงานนั้นจะป้อน Input อย่างไรได้บ้าง ผลที่ได้จากการป้อน Input แต่ละแบบควรจะได้ผลลัพธ์ออกมาเป็นอย่างไร (โปรแกรมทำงานสำเร็จ หรือโปรแกรมทำงานผิดพลาดแล้วเกิด Error อย่างไร) จัดทำเป็นเอกสารเพื่อเป็นแนวทางในการพัฒนาชุดทดสอบโปรแกรม เพื่อไม่ให้ผู้พัฒนาชุดทดสอบโปรแกรมพัฒนาขึ้นมาไม่ตรงตามที่ได้ออกแบบไว้

3.3 จัดทำเอกสารประกอบการทดสอบ

จัดทำเอกสารประกอบการทดสอบ โดยแต่ละเอกสารจะประกอบด้วยรายละเอียดต่างๆ ที่มีความสำคัญต่อการทดสอบระบบ

3.4 พัฒนาชุดทดสอบโปรแกรมตามที่ได้ออกแบบไว้

หลังจากกำหนด test case ที่จะนำไปใช้ในการทดสอบการทำงานของโปรแกรม Web GUI version 2 แล้วจึงลงมือพัฒนาชุดทดสอบโปรแกรม โดยพัฒนาด้วยภาษา Java ลงบน Eclipse และใช้ SVN ช่วยในการจัดการ version ของ source code โดยแนวทางของการพัฒนาคือทุกๆ Unit test จะมีการแบ่ง code เป็น 3 ส่วน ได้แก่

Prepare data เป็นขั้นตอนของการเตรียมข้อมูลที่จะนำมาทดสอบการทำงานของโปรแกรม

Execute function เป็นขั้นตอนการเรียกใช้ฟังก์ชันที่ต้องการทดสอบ โดยเรียกใช้และป้อนข้อมูล Input ที่เตรียมไว้เข้าไป

Check result เป็นขั้นตอนการนำผลที่ได้รับจากการเรียกใช้ฟังก์ชันมาเปรียบเทียบกับผลที่เราคาดหวังไว้ก่อนการเรียกใช้โปรแกรม เพื่อตรวจสอบว่าฟังก์ชันที่เรียกใช้นั้นสามารถทำงานได้ตามวัตถุประสงค์ที่เรากำหนดไว้หรือไม่

และได้นำ library ที่มีอยู่แล้วในภาษา Java ได้แก่ JUnit framework และ Mockito framework เข้ามาช่วยในการพัฒนาชุดทดสอบโปรแกรม

ตัวอย่างการใช้ Mockito ช่วยในการจำลอง Object คู่กับการใช้ JUnit สร้าง Unit test suite

ด้านล่างจะแสดงตัวอย่างการใช้ Mockito คู่กับ JUnit จะทดสอบการทำงานของฟังก์ชัน volumeSquareBasedPyramid() โดยที่จะต้องเรียกใช้ฟังก์ชัน areaSquare() ขึ้นมาก่อน ดังนี้

1) สร้างคลาสขึ้นมา โดยภายในคลาสประกอบไปด้วย method ดังต่อไปนี้

testFunctionVolumeSquareBasedPyramid() เป็นฟังก์ชันที่สร้างขึ้นมาเพื่อทดสอบการทำงานของฟังก์ชัน volumeSquareBasedPyramid() โดยแบ่ง code เป็น 3 ส่วนคือ ส่วนที่ 1 เป็นส่วนเตรียมข้อมูล มีการประกาศตัวแปรต่างๆและกำหนดค่าไว้ มีการจำลอง object Mockito ขึ้นมา และควบคุมการทำงานด้วยการใช้คำสั่ง when ควบคุมว่าเมื่อใดก็ตามที่เรียกใช้ฟังก์ชัน areaSquare() โดยป้อน input ตามที่กำหนด จะให้ return ค่าออกมาเท่ากับ 12 ส่วนที่ 2 เป็นส่วนการเรียกใช้ฟังก์ชันการทำงาน ส่วนที่ 3 เป็นส่วนตรวจสอบผลการทำงานว่าได้ผลลัพธ์เท่ากับที่เราคาดหวังไว้หรือไม่ ในที่นี่คาดหวังว่าผลลัพธ์จะเท่ากับ 20

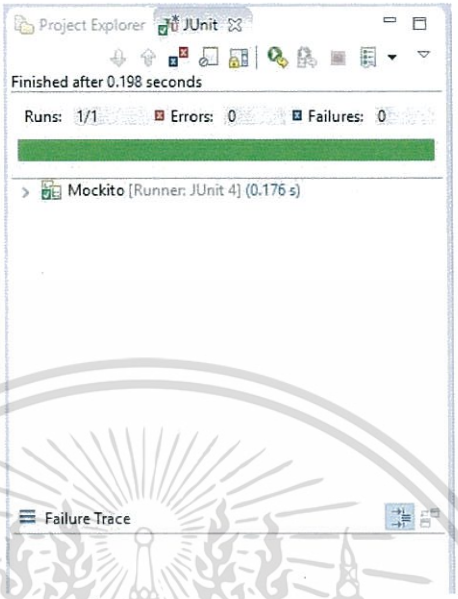
areaSquare() เป็นฟังก์ชันการคำนวณพื้นที่ฐานของพีระมิดฐานสี่เหลี่ยม โดยรับ input เป็นความยาวของด้านกว้างและยาวฐานสี่เหลี่ยมของพีระมิด และส่ง output ออกมาเป็นพื้นที่ฐานของพีระมิดฐานสี่เหลี่ยม

volumeSquareBasedPyramid() เป็นฟังก์ชันการคำนวณปริมาตรของพีระมิดฐานสี่เหลี่ยม โดยรับ input เป็นพื้นที่ฐานของพีระมิดฐานสี่เหลี่ยมและความสูงของพีระมิดฐานสี่เหลี่ยม และส่ง output ออกมาเป็นปริมาตรของพีระมิดฐานสี่เหลี่ยม แสดง source code ดังภาพ

```
6  public class Mockito {
7      @Test
8      public void testFunctionVolumeSquareBasedPyramid() {
9          /** Prepare Data*/
10         int value1 = 3, value2 = 4, high = 5;
11         int area, volume;
12         Mockito mockito = mock(Mockito.class);
13         when(mockito.areaSquare(value1, value2)).thenReturn(12);
14
15         /** Execute Function*/
16         area = mockito.areaSquare(value1, value2);
17         volume = volumeSquareBasedPyramid(area, high);
18
19         /** Check Result*/
20         assertEquals(20,volume);
21     }
22
23     public int areaSquare(int value1, int value2){
24         int area;
25         area = value1 * value2;
26         return area;
27     }
28
29     public int volumeSquareBasedPyramid(int area, int high){
30         long volume;
31         volume = (long) (1.0/3 * area * high);
32         return (int)volume;
33     }
34 }
```

ภาพที่ 3.3 source code แสดงตัวอย่างการใช้ Mockito

2) เมื่อรัน test จะได้ผลดังภาพ



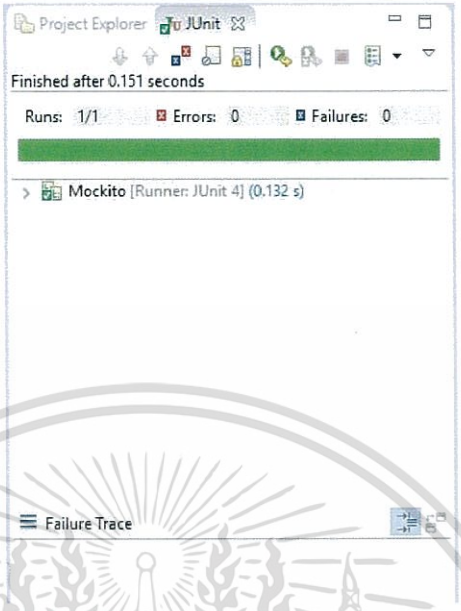
ภาพที่ 3.4 ผลการรันฟังก์ชัน testFunctionVolumeSquareBasedPyramid

3) เมื่อทดลองลองแก้ไข code ในส่วนของฟังก์ชัน areaSquare() (แก้ไข code จาก area = value1 * value2 เป็น area = value1 + value2) ดังภาพ

```
6  public class Mockito {
7      @Test
8      public void testFunctionVolumeSquareBasedPyramid() {
9          /** Prepare Data*/
10         int value1 = 3, value2 = 4, high = 5;
11         int area, volume;
12         Mockito mockito = mock(Mockito.class);
13         when(mockito.areaSquare(value1, value2)).thenReturn(12);
14
15         /** Execute Function*/
16         area = mockito.areaSquare(value1, value2);
17         volume = volumeSquareBasedPyramid(area, high);
18
19         /** Check Result*/
20         assertEquals(20,volume);
21     }
22
23     public int areaSquare(int value1, int value2){
24         int area;
25         area = value1 + value2;
26         return area;
27     }
28
29     public int volumeSquareBasedPyramid(int area, int high){
30         long volume;
31         volume = (long) (1.0/3 * area * high);
32         return (int)volume;
33     }
34 }
```

ภาพที่ 3.5 source code แสดงการแก้ไขฟังก์ชัน areaSquare

4) เมื่อรัน test จะได้ผลดังภาพ



ภาพที่ 3.6 ผลการรันฟังก์ชัน testFunctionVolumeSquareBasedPyramid

จะเห็นว่าทั้ง 2 กรณีให้ผลลัพธ์ที่เหมือนกันแม้ว่าจะแก้ไขฟังก์ชันที่ไม่ได้ทดสอบให้ทำงานผิดพลาด สรุปได้ว่าเมื่อใช้ Mockito มาจำลองและควบคุม object ที่ไม่ได้ทดสอบจะทำให้ผลของการทดสอบเป็นอิสระจากฟังก์ชันที่ไม่ได้ทดสอบ (ผลของฟังก์ชันที่ไม่ได้ทดสอบไม่ส่งผลต่อผลของการทดสอบ) ตรงตามแนวทางของการเขียน Unit test จึงได้นำโครงสร้างนี้ไปใช้ในการพัฒนาชุดทดสอบโปรแกรมทั้งหมด

3.5 นำชุดทดสอบโปรแกรมที่พัฒนามาทดสอบการทำงานของโปรแกรมและทำเอกสารบันทึกผลการทดสอบโปรแกรม

เมื่อพัฒนาชุดทดสอบโปรแกรมตาม test case ที่กำหนดไว้ครบทุกฟังก์ชันแล้วจึงนำชุดทดสอบโปรแกรมที่พัฒนาขึ้นมา ไปทดสอบการทำงานของโปรแกรม Web GUI version 2 และบันทึกผลการทดสอบโปรแกรม เพื่อส่งผลการทดสอบไปให้ทีมพัฒนาโปรแกรม Web GUI version 2 ปรับปรุงแก้ไขให้สามารถทำงานได้อย่างถูกต้องต่อไป

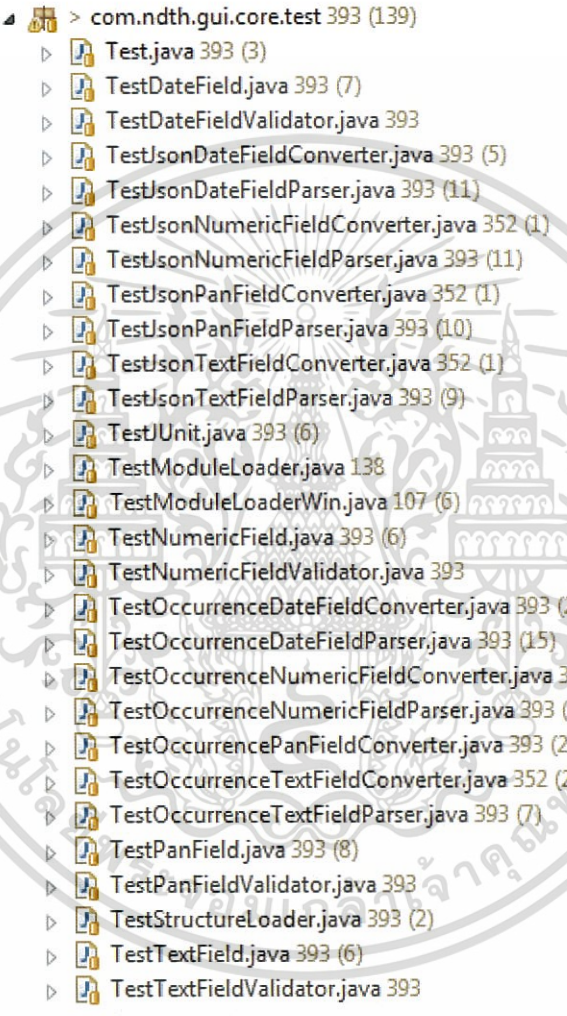
3.6 นำเสนอผลการพัฒนาชุดทดสอบโปรแกรม

เมื่อพัฒนาชุดทดสอบโปรแกรมสำเร็จแล้ว จึงได้มีการเตรียมเอกสารนำเสนอผลการพัฒนาชุดทดสอบโปรแกรม และส่งมอบตัวโปรแกรมให้กับทางบริษัท

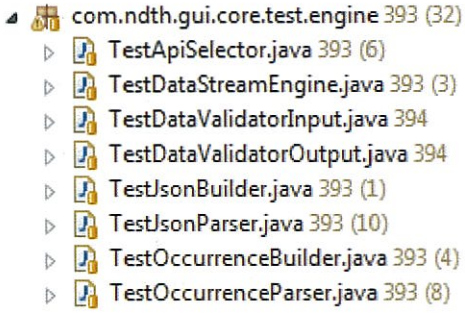
บทที่ 4
ผลการดำเนินงาน

4.1 ผลของการพัฒนาชุดทดสอบโปรแกรม

จากการพัฒนาชุดทดสอบโปรแกรมทำให้ได้ชุดทดสอบโปรแกรมที่สามารถนำไปทดสอบฟังก์ชันการทำงานของโปรแกรม Web GUI version 2 ดังต่อไปนี้



ภาพที่ 4.1 คลาสต่างๆที่ใช้ในการทดสอบฟังก์ชันการทำงานของโปรแกรม

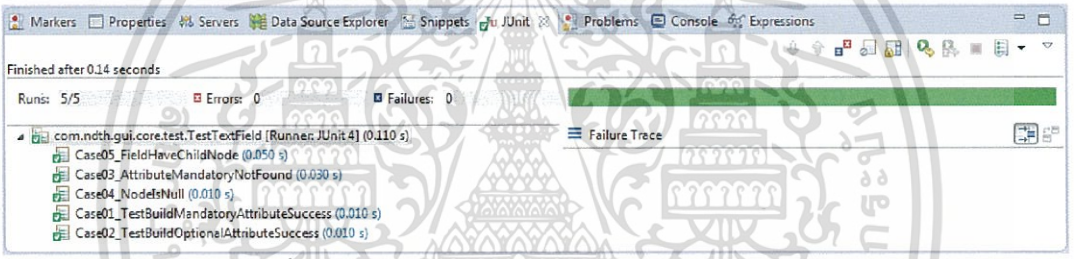


ภาพที่ 4.2 คลาสต่างๆที่ใช้ในการทดสอบ Engine การทำงานของโปรแกรม

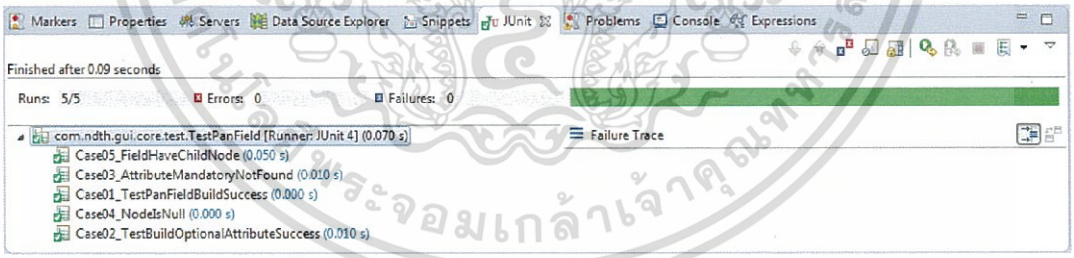
4.2 ผลของการนำชุดทดสอบโปรแกรมไปใช้ทดสอบโปรแกรม Web GUI version 2

จากการชุดทดสอบโปรแกรมที่พัฒนาขึ้นมา ทดสอบฟังก์ชันการทำงานของโปรแกรม Web GUI version 2 ได้ผลการทดสอบเป็นดังนี้

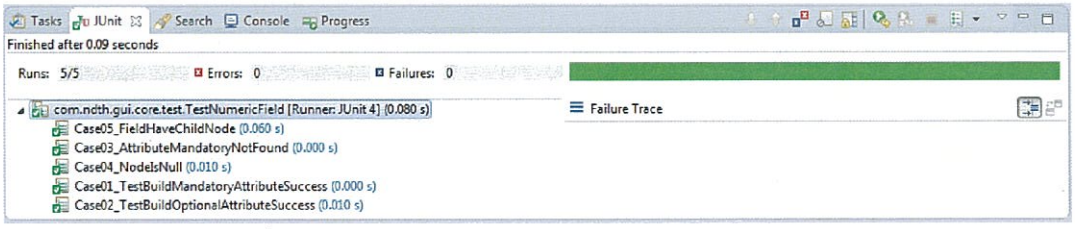
4.2.1 ผลการทดสอบส่วนการทำงานสร้าง BeanElement



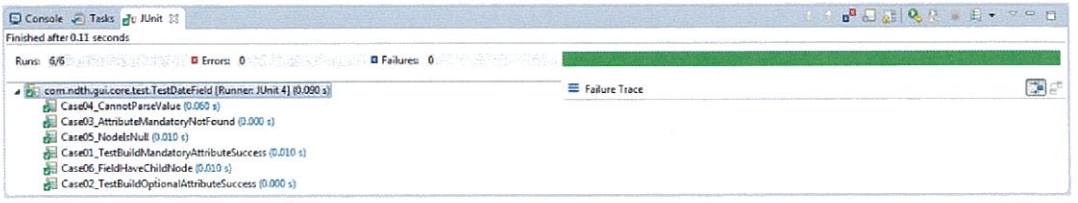
ภาพที่ 4.3 ผลการทดสอบการทำงานของคลาส TextField



ภาพที่ 4.4 ผลการทดสอบการทำงานของคลาส PanField

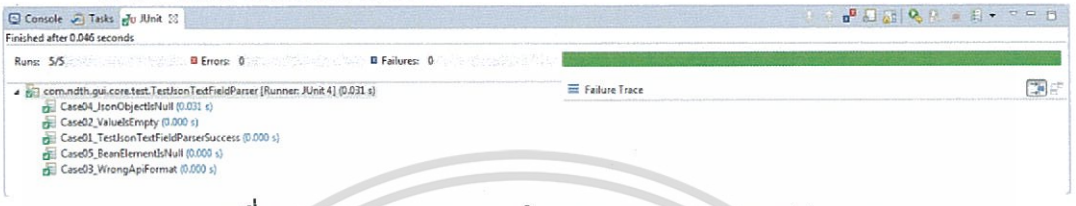


ภาพที่ 4.5 ผลการทดสอบการทำงานของคลาส NumericField



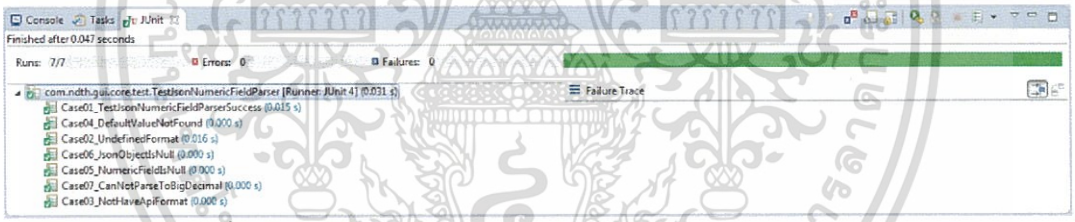
ภาพที่ 4.6 ผลการทดสอบการทำงานของคลาส DateField

4.2.2 ผลการทดสอบส่วนการทำงาน Json Parser

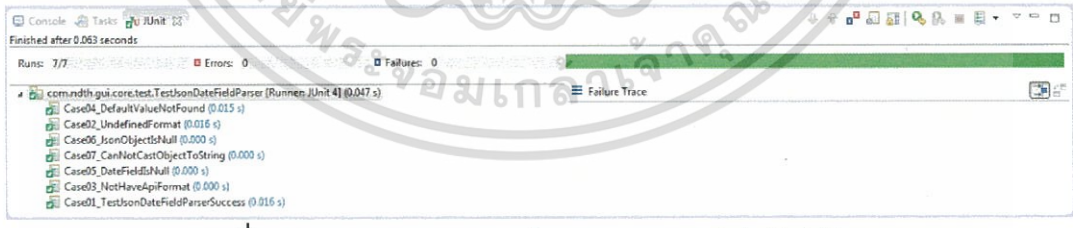


ภาพที่ 4.7 ผลการทดสอบการทำงานของคลาส TextFieldParser

ภาพที่ 4.8 ผลการทดสอบการทำงานของคลาส PanFieldParser

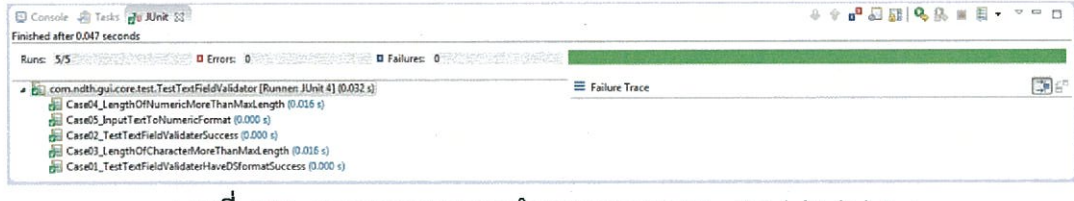


ภาพที่ 4.9 ผลการทดสอบการทำงานของคลาส NumericFieldParser

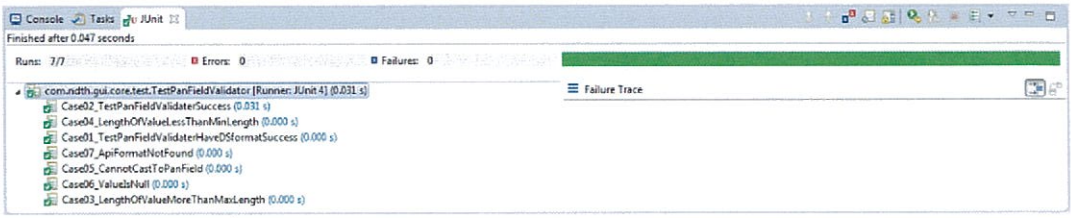


ภาพที่ 4.10 ผลการทดสอบการทำงานของคลาส DateFieldParser

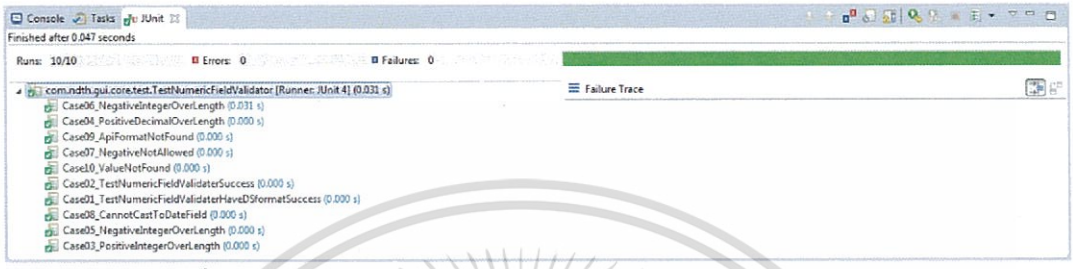
4.2.3 ผลการทดสอบส่วนการทำงาน DataValidator



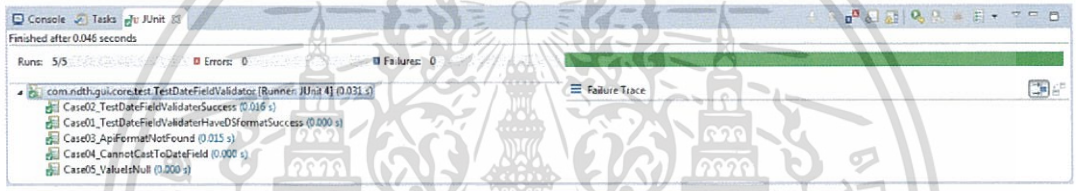
ภาพที่ 4.11 ผลการทดสอบการทำงานของคลาส TextFieldValidator



ภาพที่ 4.12 ผลการทดสอบการทำงานของคลาส PanFieldValidator



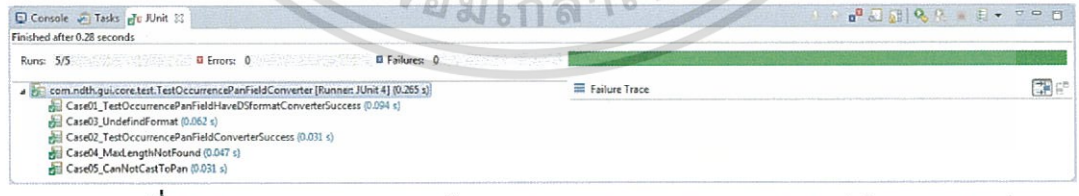
ภาพที่ 4.13 ผลการทดสอบการทำงานของคลาส NumericFieldValidator



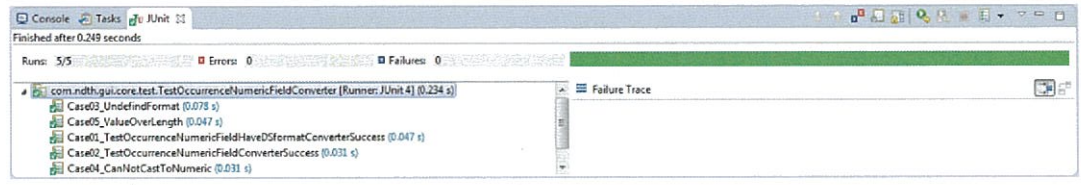
ภาพที่ 4.14 ผลการทดสอบการทำงานของคลาส DateFieldValidator

4.2.4 ผลการทดสอบส่วนการทำงาน OccurrenceBuilder

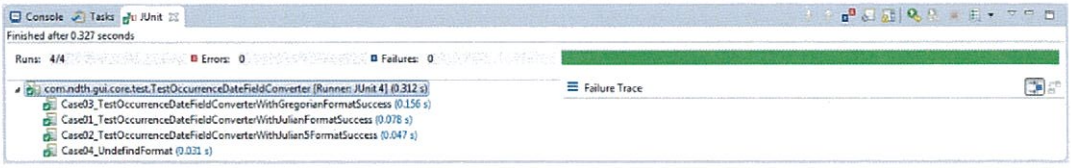
ภาพที่ 4.15 ผลการทดสอบการทำงานของคลาส OccurrenceTextFieldConverter



ภาพที่ 4.16 ผลการทดสอบการทำงานของคลาส OccurrencePanFieldConverter

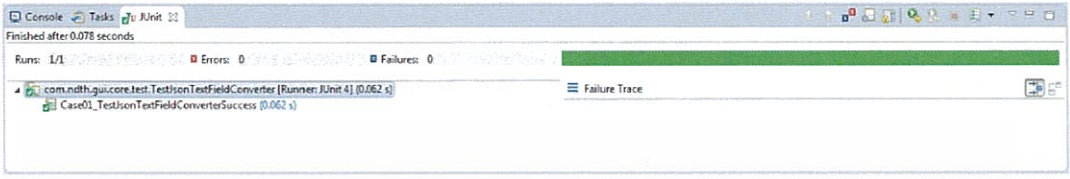


ภาพที่ 4.17 ผลการทดสอบการทำงานของคลาส OccurrenceNumericFieldConverter

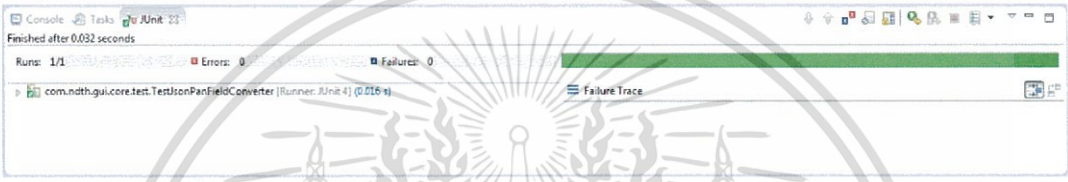


ภาพที่ 4.18 ผลการทดสอบการทำงานของคลาส OccurrenceDateFieldConverter

4.2.5 ผลการทดสอบส่วนการทำงาน Json Builder

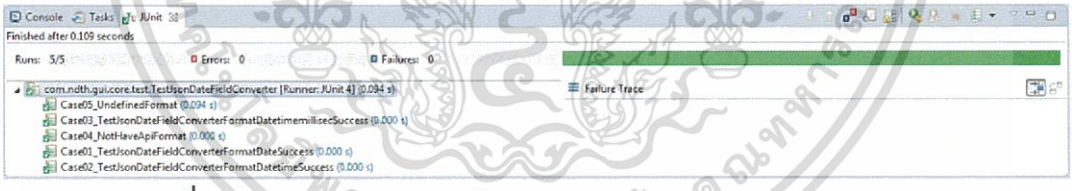


ภาพที่ 4.19 ผลการทดสอบการทำงานของคลาส JsonTextFieldConverter



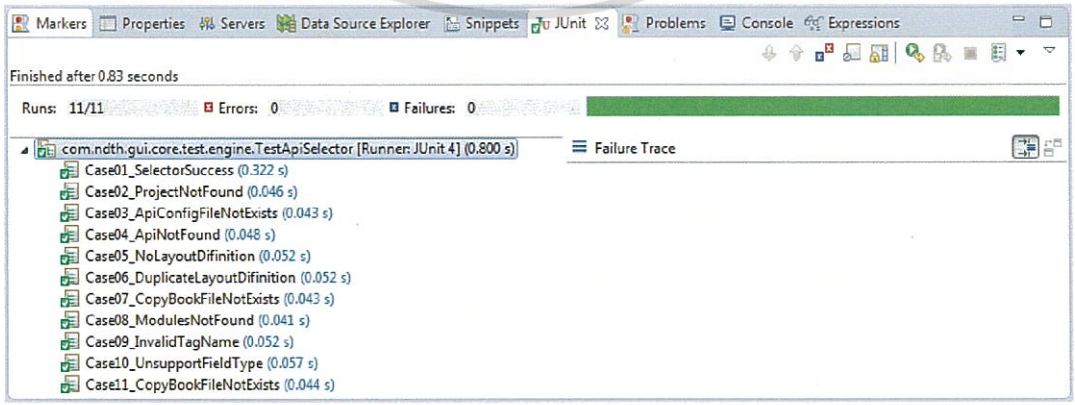
ภาพที่ 4.20 ผลการทดสอบการทำงานของคลาส JsonPanFieldConverter

ภาพที่ 4.21 ผลการทดสอบการทำงานของคลาส JsonNumericFieldConverter

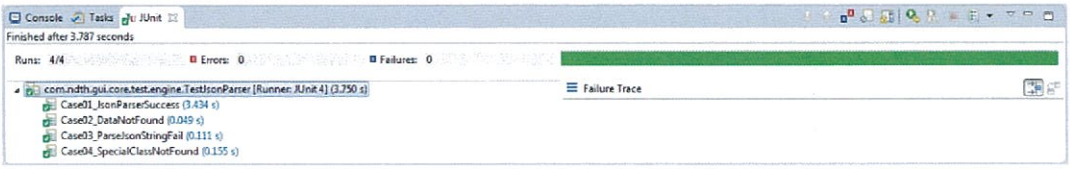


ภาพที่ 4.22 ผลการทดสอบการทำงานของคลาส JsonDateFieldConverter

4.2.6 ผลการทดสอบส่วนการทำงานของ Engine



ภาพที่ 4.23 ผลการทดสอบการทำงานของคลาส ApiSelector

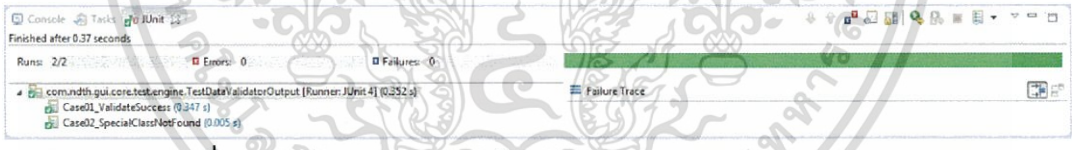


ภาพที่ 4.24 ผลการทดสอบการทำงานของคลาส JsonParser

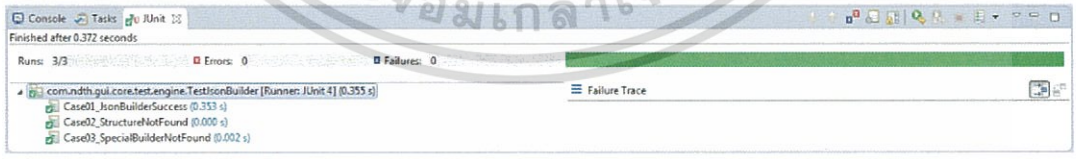
ภาพที่ 4.25 ผลการทดสอบการทำงานของคลาส DataValidatorInput

ภาพที่ 4.26 ผลการทดสอบการทำงานของคลาส OccurrenceBuilder

ภาพที่ 4.27 ผลการทดสอบการทำงานของคลาส OccurrenceParser



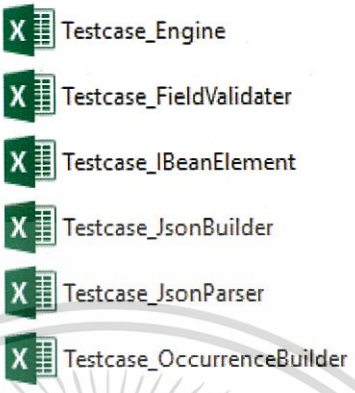
ภาพที่ 4.28 ผลการทดสอบการทำงานของคลาส DataValidatorOutput



ภาพที่ 4.29 ผลการทดสอบการทำงานของคลาส JsonBuilder

4.3 เอกสารประกอบการพัฒนาชุดทดสอบโปรแกรม

ผู้ทำโครงงานได้ทำเอกสารประกอบการพัฒนาชุดทดสอบโปรแกรม โดยมีรายการเอกสารดังต่อไปนี้



ภาพที่ 4.30 รายการเอกสารประกอบการพัฒนาชุดทดสอบโปรแกรม

โดยภายในแต่ละเอกสารจะประกอบไปด้วยข้อมูลต่างๆ ดังนี้

- ชื่อของคลาสที่ถูกทดสอบและชื่อคลาสชุดทดสอบ
- ชื่อของ case ที่ทดสอบ
- รายละเอียดของ case ที่ต้องการให้เกิดขึ้น
- Method ที่ถูกใช้งานระหว่างการทดสอบ
- ผลลัพธ์ที่คาดหวังว่าจะเกิดขึ้น (Expected Result)
- ผลลัพธ์ที่เกิดขึ้นจริง (Actual Result)
- ชื่อผู้ทดสอบ / วันที่ทดสอบ
- ผลการทดสอบ
- ชื่อผู้ตรวจผลการทดสอบ / วันที่ตรวจผลการทดสอบ

บทที่ 5
สรุปผลการดำเนินงาน

5.1 สรุปผลการดำเนินการ

จากการพัฒนาชุดทดสอบโปรแกรมโดยใช้ JUnit framework คู่กับการใช้ Mockito framework ช่วยจำลอง object ในการทดสอบ โดยพัฒนาตามโครงสร้างของการเขียน Unit test ที่ดี ทำให้ได้ชุดทดสอบโปรแกรมที่สามารถนำมาทดสอบการทำงานส่วนต่างๆของโปรแกรม Web GUI version 2 ช่วยให้พบข้อผิดพลาดของโปรแกรมในหลายๆจุด ทำให้ทีมพัฒนาโปรแกรม Web GUI version 2 สามารถแก้ไขปัญหาข้อผิดพลาดที่เกิดขึ้นเพื่อให้โปรแกรมสามารถทำงานได้ตามวัตถุประสงค์ที่ออกแบบไว้ ถือว่าประสบความสำเร็จในการพัฒนาชุดทดสอบโปรแกรมที่มีวัตถุประสงค์เพื่อตรวจสอบการทำงานของโปรแกรม Web GUI version 2 อย่างมาก

จากการที่ได้นำเสนอผลการพัฒนาชุดทดสอบโปรแกรมที่พัฒนาขึ้นมากับทางบริษัทที่มอบหมายโครงงานชิ้นนี้ให้ ทางบริษัทได้ตอบรับว่าได้ผลออกมาบรรลุวัตถุประสงค์อย่างมาก ชุดทดสอบที่พัฒนามาสามารถนำไปทดสอบได้จริงตามที่ต้องการ

5.2 ปัญหาที่พบและแนวทางการแก้ไข

5.2.1 ปัญหาที่พบ

1) จากการวางแผนการพัฒนาชุดทดสอบโปรแกรม ผู้จัดทำไม่สามารถรู้ว่าการออกแบบ case ของการทดสอบนั้น จะสามารถครอบคลุมทุกกรณีที่อาจจะเกิดขึ้นได้หรือไม่ เนื่องจากระยะเวลาในการพัฒนามีน้อย ประกอบกับโปรแกรม Web GUI version 2 เป็นโปรแกรมที่มีระบบการทำงานใหญ่และซับซ้อนอย่างมาก

2) จากที่ชุดทดสอบที่พัฒนาขึ้น สามารถใช้ทดสอบได้เฉพาะโปรแกรม Web GUI version 2 เท่านั้น แต่ตัวโปรแกรม Web GUI version 2 เป็นผลิตภัณฑ์ของ บริษัท เอ็นทีที เดต้า (ประเทศไทย) จำกัด ทำให้ไม่สามารถแสดงรายละเอียดการทำงานภายในโปรแกรมแบบละเอียดได้ สามารถแสดงได้เพียงผังการทำงานแบบ High level ทำให้ผู้ที่ศึกษาโครงงานชิ้นนี้อาจจะไม่เข้าใจการทำงานในบางส่วน

5.2.2 แนวทางการแก้ไข

1) ถ้ามีโอกาสพัฒนาชุดทดสอบโปรแกรมต่อ ควรจะนำ tool อาทิเช่น EclEmma ซึ่งเป็น Java Code Coverage for Eclipse มาช่วยในการพัฒนาชุดทดสอบโปรแกรม

2) ผู้จัดทำพยายามอธิบายการทำงานของโปรแกรม Web GUI version 2 ให้ละเอียดที่สุดเท่านี้สามารถนำมาแสดงได้ ภายใต้ขอบเขตการเผยแพร่ข้อมูลของทางบริษัท

บรรณานุกรม

KANNIQUE. **Automated Testing vs. Manual Testing เลือกอะไรดี.**[Online]. Avaliable : http://www.chapterpiece.com/software-development-process/2010/02/06/choosing-automated-testing-or-manual-testing/

Kasidech Tapang. **มาทำความรู้จักกับ Unit Test กันเถอะ.**[Online]. Avaliable : http://ascended.in.th/unit-testing-in-action/

MagicKiat. **ตัวอย่างการทำ Unit Test แบบง่ายๆ.**[Online]. Avaliable : https://magickiat.wordpress.com/2012/01/25/simple-java-unit-test-with-junit/

nttdata. **NTT data Global IT Invovator.**[Online].Avaliable : http://th.nttdata.com/

Somkiat. **junit Archive.**[Online].Avaliable : http://www.somkiat.cc/tag/junit/

Szczepan Faber and friends. **mockito.**[Online].Avaliable : http://site.mockito.org/

wikipedia. **Unit testing.**[Online].Avaliable : https://en.wikipedia.org/wiki/Unit_testing

Wutipong Wongsakuldej. **Unit Test คืออะไร ?.**[Online]. Avaliable : http://blog.playground-soft.com/2014/10/unit-test-คืออะไร/