เครื่องมือตรวจสอบข้อปฏิบัติการตั้งชื่อในโปรแกรมอ็อบเจกทีฟซี

นางสาวฤชุตา นันตะพานา

โครงงานมหาบัณฑิตนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2558

Objective C Naming Convention Checker

Miss Ruchuta Nundhapana

A Master Project Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science in Software Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2015

หัวข้อโครงงานมหาบัณฑิต	เครื่องมือตรวจสอบข้อปฏิบัติการตั้งชื่อในโปรแกรมอ็อบเจกทีฟซี
โดย	นางสาวฤชุตา นันตะพานา
ภาควิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับโครงงานมหาบัณฑิตฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... หัวหน้าภาควิชาวิศวกรรมคอมพิวเตอร์

(ผศ. ดร.ณัฐวุฒิ หนูไพโรจน์)

..... อาจารย์ที่ปรึกษา

(รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

ฤชุตา นันตะพานา : เครื่องมือตรวจสอบข้อปฏิบัติการตั้งชื่อในโปรแกรมอ็อบเจกทีฟซี (Objective C Naming Convention Checker) อาจารย์ที่ปรึกษา : รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา, จำนวน 75 หน้า

ความสามารถในการอ่านได้ง่ายของโปรแกรมเป็นสิ่งสำคัญต่อนักพัฒนาในการทำความเข้าใจกระบวนการทำงานและข้อมูลที่ปรากฏในโปรแกรม เพื่อให้สามารถปรับปรุงหรือแก้ไขให้เป็นไปตามความต้องการ ทีมพัฒนาโปรแกรมในองค์กรส่วนใหญ่ ไม่มีการกำหนดมาตรฐานข้อปฏิบัติการเขียนโปรแกรม นักพัฒนาจึงมีรูปแบบการเขียนที่แตกต่างกันตามทักษะและประสบการณ์ของตน เมื่อมีการพัฒนาโปรแกรมร่วมกัน นักพัฒนาจะต้องเข้าใจโปรแกรมทั้งหมดก่อนที่จะทำการแก้ไขเพิ่มเติม ความหลากหลายของรูปแบบการเขียนโปรแกรม จึงส่งผลให้ใช้ระยะเวลานานในการอ่านทำความเข้าใจโปรแกรม หรืออาจแก้ไขโปรแกรมผิดตำแหน่ง งานวิจัยนี้เล็งเห็นถึงปัญหาที่เกิดขึ้นในสภาพแวดล้อมการทำงานจริงของทีมพัฒนาแอปพลิเคชันบนระบบปฏิบัติการ iOS ในองค์กรขนาดใหญ่แห่งหนึ่ง จึงรวบรวมมาตรฐานข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซี บนระบบปฏิบัติการ Mac OS โดยเน้นที่การตั้งชื่อ การละเว้นการใช้ Magic Number และการละเว้นการใช้ Literal String และพัฒนาเครื่องมือตรวจสอบ โดยแจ้งเตือนจุดที่โปรแกรมละเมิดข้อปฏิบัติ เพื่อให้นักพัฒนาทำการแก้ไข ในการประเมินผลพบว่านักพัฒนาสามารถอ่านและทำความเข้าใจโปรแกรมที่ได้รับการปรับปรุงตามคำเตือนของเครื่องมือ ได้ง่ายขึ้นที่ระดับนัยสำคัญ 0.05

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่ออาจารย์ที่ปรึกษา.....

ปีการศึกษา 2558

5770957121 : MAJOR SOFTWARE ENGINEERING
KEY WORD : NAMING CONVENTION, CODING CONVENTION, OBJECTIVE C
RUCHUTA NUNDHAPANA: OBJECTIVE C CONVENTION CHECKER.
MASTER PROJECT ADVISOR : ASSOC.PROF. DR.TWITTIE SENIVONGSE, 75 pp.

Readability is important for developers to understand how a program works in order to improve or fix bugs in the program so that it follows software requirements. Most organizations do not leverage coding standards or conventions, so developers code programs in their own styles. When other developers join the team and need to work on certain programs, they need to understand the code before any addition or modification can be made to the code. Many coding styles make it difficult and require more time to read and understand the programs clearly. Misunderstanding may also lead to incorrect modification of the programs. We realize this problem that an iOS development team of a large organization is facing. So we survey coding conventions for Objective C on Mac OS, focusing on Naming Convention and avoidance of Magic Number and Literal String, and present a coding convention checker tool. Violations will be displayed as warnings to inform developers to revise the programs. In an evaluation, it is found that the programs are more readable and developers can better understand the programs that are revised following the convention violation warnings of the tool, at the significance level of 0.05.

Department of Computer Engineering Student's signature.....
Field of study Software Engineering Advisor's signature.....
Academic year 2015

กิตติกรรมประกาศ

ขอกราบขอบพระคุณ รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา อาจารย์ที่ปรึกษาโครงงาน เป็นอย่างยิ่งที่ได้สละเวลาให้คำปรึกษา คำแนะนำ และแนวทางสำหรับการทำโครงงานมหาบัณฑิต รวมทั้งเป็นผู้ประสานงานให้ความช่วยเหลือแก่นิสิตที่ทำโครงงานทุกคน

ขอกราบขอบพระคุณ รศ. ดร. สมชาย ประสิทธิ์จูตระกูล รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา และรศ.ดร.พรศิริ หมื่นไชยศรี คณะกรรมการคุมสอบโครงงานมหาบัณฑิตเป็นอย่างยิ่ง ที่ได้กรุณาแนะนำแนวทาง รวมถึงการตรวจสอบและแก้ไขโครงงานมหาบัณฑิตนี้

ขอขอบคุณ คณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ให้คำแนะนำ ความรู้และแนวทางการทำโครงงาน

ขอขอบคุณ เพื่อนๆ หลักสูตรวิศวกรรมซอฟต์แวร์ สำหรับกำลังใจและคำแนะนำในการจัดทำโครงงานมหาบัณฑิต

สุดท้ายนี้ ขอกราบขอบพระคุณ บิดา มารดา รวมถึงสมาชิกในครอบครัวที่ให้การสนับสนุนและให้กำลังใจที่ดีเสมอมา

ฤชุตา นันตะพานา

สารบัญ

หน้า

สารบัญ (ต่อ)

บทที่ **หน้า**

สารบัญ (ต่อ)

บทที่ **หน้า**

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

สารบัญภาพ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

องค์กรต่างๆในปัจจุบันต่างเล็งเห็นถึงความสำคัญของการพัฒนาเทคโนโลยีสารสนเทศและการนำเทคโนโลยีสารสนเทศเข้ามาใช้ เพื่อให้เกิดประโยชน์กับองค์กรของตน เช่น การพัฒนาเว็บไซต์เพื่อประชาสัมพันธ์การให้บริการหรือสินค้า หรือการพัฒนาแอปพลิเคชันเพื่อตอบสนองความต้องการของลูกค้า จึงเป็นที่มาของการเกิดฝ่ายงานสนับสนุน ค้นคว้าและพัฒนาเครื่องมือทางด้านเทคโนโลยีสารสนเทศขึ้นในองค์กร เมื่อมีผลิตภัณฑ์ซอฟต์แวร์เกิดขึ้น สิ่งสำคัญที่ตามมาคือการควบคุมคุณภาพซอฟต์แวร์ รวมทั้งองค์กรยังมุ่งเน้นที่จะใช้ทรัพยากรที่มีอยู่ให้เกิดประสิทธิภาพสูงสุด คุณภาพในด้านการดูแลบำรุงรักษา (Maintainability) จึงเป็นสิ่งที่องค์กรต้องให้ความสำคัญ มาตรฐาน ISO/IEC 9126 [1] ได้ให้คำนิยามของความสามารถในการบำรุงรักษาว่า เป็นความสามารถในการปรับเปลี่ยนผลิตภัณฑ์ซอฟต์แวร์ โดยการปรับเปลี่ยนอาจรวมถึงการปรับแก้ ปรับปรุงหรือดัดแปลงซอฟต์แวร์ตามสภาพแวดล้อมและตามความต้องการ โดยจะมีปัจจัยที่ส่งผลกระทบต่อความสามารถในการบำรุงรักษา ได้แก่ ความสามารถในการอ่าน ทำความเข้าใจและความซับซ้อนของโปรแกรม

จากการสำรวจฝ่ายงานพัฒนาซอฟต์แวร์ภายในองค์กร ในหนึ่งโครงการพัฒนาซอฟต์แวร์ จะมีนักพัฒนาโปรแกรมมากกว่าหนึ่งคน หรือโครงการอาจมีการเปลี่ยนแปลงผู้รับผิดชอบในการพัฒนาซอฟต์แวร์จึงถูกส่งมอบให้นักพัฒนาโปรแกรมคนอื่นพัฒนาต่อ นักพัฒนาโปรแกรมจึงมักประสบปัญหาในการอ่านซึ่งพัฒนามาก่อนหน้าโดยนักเขียนโปรแกรมคนอื่น ก่อนที่จะพัฒนาในส่วนของงานที่ต้องแก้ไขหรือเพิ่มเติม สาเหตุของปัญหานี้ มาจากการที่ฝ่ายงานไม่ได้มีการกำหนดมาตรฐานข้อปฏิบัติในการเขียนโปรแกรม ทำให้นักพัฒนาโปรแกรม ใช้คำและรูปแบบในการเขียนโปรแกรมที่แตกต่างกันออกไป

จากกรณีศึกษาของฝ่ายงานพัฒนาโมบายล์แอปพลิเคชันบนระบบปฏิบัติการ iOS ขององค์กรแห่งหนึ่ง ฝ่ายงานกำลังประสบกับปัญหาดังกล่าวโดยที่การพัฒนาแอปพลิเคชันร่วมกันและการรับช่วงพัฒนาต่อจากนักเขียนโปรแกรมคนก่อนทำได้ยาก ใช้เวลานานในการศึกษาโปรแกรม หัวหน้าฝ่ายงานจึงมีความต้องการที่จะสร้างมาตรฐานการเขียนโปรแกรมในภาษาอ็อบเจ็กทีฟซีสำหรับระบบปฏิบัติการ iOS ให้เกิดขึ้นในองค์กร จากการค้นคว้าพบว่า บริษัทแอปเปิลได้มีการเผยแพร่เอกสารข้อแนะนำในการเขียนโปรแกรมภาษาอ็อบเจ็กทีฟซีไว้ [2] โครงงานนี้จึงมีแนวคิดในการพัฒนาเครื่องมือสำหรับตรวจสอบข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซีบนระบบปฏิบัติการ OSX เพื่อสนับสนุนการใช้มาตรฐานในการเขียนโปรแกรมอ็อบเจกทีฟซีสำหรับฝ่ายงานดังกล่าว โดยจะยึดหลักข้อปฏิบัติจากเอกสารข้อแนะนำในการเขียนโปรแกรมของบริษัทแอปเปิล และเพิ่มเติมข้อปฏิบัติจากเอกสารคำแนะนำอื่นๆ รวมถึงพิจารณาข้อปฏิบัติที่ได้รับเสนอจากฝ่ายงานพัฒนา โดยเน้นเฉพาะ

การตรวจสอบการตั้งชื่อ การใช้ Magic Number และการใช้ Literal String ในโปรแกรม หลังจากมีการประกาศใช้ข้อปฏิบัตินี้แล้ว ฝ่ายงานจะต้องมีการตรวจสอบว่านักพัฒนาโปรแกรมในฝ่ายงานได้ปฏิบัติตามข้อกำหนดหรือไม่ ฝ่ายงานสามารถใช้เครื่องมือนี้เพื่อระบุจุดต่างๆในโปรแกรมที่ละเมิดข้อปฏิบัติและควรได้รับการปรับปรุง รวมทั้งสามารถใช้เครื่องมือนี้ในการตรวจสอบซอฟต์แวร์ที่ได้พัฒนาขึ้นใหม่หลังจากมีการประกาศใช้มาตรฐานว่านักพัฒนาโปรแกรมในฝ่ายงาน ยึดถือตามข้อปฏิบัติการเขียนโปรแกรมที่ได้กำหนดไว้หรือไม่

วัตถุประสงค์ของโครงงาน

เพื่อพัฒนาเครื่องมือในการตรวจสอบข้อปฏิบัติการตั้งชื่อในโปรแกรมอ็อบเจกทีฟซี

ขอบเขตของโครงงาน

1) เครื่องมือสามารถรองรับชุดคำสั่งโปรแกรมในภาษาอ็อบเจกทีฟซีเท่านั้น
2) ข้อปฏิบัติที่ใช้ตรวจสอบจะครอบคลุมเรื่องการตั้งชื่อ, Magic Number และ Literal String เท่านั้น
3) ผลลัพธ์ที่ได้หลังจากนำเข้าโปรแกรม เครื่องมือจะแสดงจุดที่โปรแกรมละเมิดข้อปฏิบัติและคำอธิบายเท่านั้น

ขั้นตอนและวิธีการดำเนินโครงงาน

1) ศึกษางานวิจัยที่เกี่ยวข้องกับมาตรฐานการเขียนโปรแกรม อ็อบเจกทีฟซี และเครื่องมือตรวจสอบการเขียนโปรแกรมที่มีอยู่ในปัจจุบัน
2) รวบรวมข้อมูลมาตรฐานและข้อปฏิบัติในการเขียนโปรแกรม
3) นำเสนอรายการข้อปฏิบัติการเขียนโปรแกรมให้กับทีมพัฒนาโมบายล์แอปพลิเคชันขององค์กรกรณีศึกษา
4) กำหนดรายการข้อปฏิบัติในการเขียนโปรแกรมที่จะตรวจสอบ
5) กำหนดวัตถุประสงค์และขอบเขตของโครงงาน
6) ศึกษาข้อมูลเชิงเทคนิคและทฤษฎีที่ต้องใช้ในการตรวจสอบโปรแกรมตามรายการข้อปฏิบัติ
7) ออกแบบส่วนต่อประสานของเครื่องมือในการตรวจสอบการเขียนโปรแกรม
8) พัฒนาเครื่องมือตามที่ได้ออกแบบไว้
9) ทดสอบและประเมินผลการทำงานของเครื่องมือ
10) จัดทำเอกสารโครงงานมหาบัณฑิตและบทความทางวิชาการ

ประโยชน์ที่คาดว่าจะได้รับ

1) ได้เครื่องมือตรวจสอบข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซี
2) ช่วยให้โปรแกรมสามารถอ่านและทำความเข้าใจได้ง่ายยิ่งขึ้น
3) เพิ่มความสามารถในการบำรุงรักษาโปรแกรมที่มีอยู่เดิม

โครงสร้างของเนื้อหาโครงงาน

โครงสร้างของเนื้อหารายงานโครงงานมหาบัณฑิตประกอบด้วยรายละเอียด 8 บท และภาคผนวก 8 ภาคผนวก ดังต่อไปนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน ขั้นตอนและวิธีการดำเนินโครงงาน ประโยชน์ที่คาดว่าจะได้รับจากโครงงานมหาบัณฑิต

บทที่ 2 กล่าวถึงทฤษฎี งานวิจัย และเครื่องมือที่เกี่ยวข้อง

บทที่ 3 กล่าวถึงข้อปฏิบัติการเขียนโปรแกรมและวิธีการตรวจสอบ

บทที่ 4 กล่าวถึงการวิเคราะห์ระบบ

บทที่ 5 กล่าวถึงการออกแบบระบบ

บทที่ 6 กล่าวถึงการพัฒนาระบบ

บทที่ 7 กล่าวถึงการทดสอบระบบ

บทที่ 8 กล่าวถึงบทสรุปของโครงงาน ปัญหาและข้อจำกัดในการทำโครงงาน รวมถึงข้อเสนอแนะ

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 เอกสารข้อแนะนำการเขียนโปรแกรมสำหรับภาษาอ็อบเจกทีฟซี

การกำหนดมาตรฐานข้อปฏิบัติการเขียนโปรแกรม เริ่มจากการรวบรวมข้อปฏิบัติในการเขียนโปรแกรมอ็อบเจกทีฟซี โดยยึดถือเอกสารข้อแนะนำในการเขียนโปรแกรมสำหรับโคโค่ของแอปเปิล (Coding Guidelines for Cocoa) เป็นหลัก เอกสารนี้จะกล่าวถึงข้อปฏิบัติการตั้งชื่อในโปรแกรมที่แอปเปิลใช้สำหรับ Cocoa Framework ซึ่งเป็น Framework ที่เกี่ยวข้องกับคลาสและวัตถุต่างๆในโปรแกรมอ็อบเจกทีฟซี โดยเอกสารจะกล่าวถึงข้อปฏิบัติที่นักพัฒนาควรยึดถือตาม เพื่อให้รูปแบบการตั้งชื่อในโปรแกรม มีความสอดคล้องและเป็นไปในทิศทางเดียวกัน ข้อปฏิบัตินี้ประกอบไปด้วยหัวข้อการตั้งชื่อพื้นฐานและกลุ่มขององค์ประกอบต่าง ๆ ได้แก่ ชื่อเมท็อด ชื่อฟังก์ชัน ชื่อตัวแปรและคำย่อที่อนุญาตให้ใช้ในการตั้งชื่อ โดยข้อแนะนำจะอธิบายลักษณะการตั้งชื่อและยกตัวอย่างชื่อที่ควรใช้และไม่ควรใช้ นอกจากนี้ ผู้วิจัยได้รวบรวมเอกสารข้อแนะนำการเขียนโปรแกรมอ็อบเจกทีฟซีจากแหล่งอ้างอิงอื่นๆจากเวบไซต์ของนักพัฒนาโปรแกรมอ็อบเจกทีฟซีที่ได้รับความนิยมในโลกออนไลน์ ได้แก่ ข้อแนะนำในการเขียนโปรแกรมอ็อบเจกทีฟซีของทีมพัฒนา iOS บริษัท New York Times [3], ข้อแนะนำในการเขียนโปรแกรมอ็อบเจกทีฟซีของเว็บไซต์ Ray Wenderlich [4] และรูปแบบ Cocoa สำหรับโปรแกรมอ็อบเจกทีฟซีของ Scott Stevenson [5] โดยจะรวบรวมเฉพาะข้อแนะนำที่เกี่ยวข้องกับการตั้งชื่อในโปรแกรมเท่านั้น จากแหล่งอ้างอิงทั้ง 4 แหล่ง โครงงานได้สรุปข้อแนะนำของเอกสารได้ ดังนี้

1) เอกสารข้อแนะนำในการเขียนโปรแกรมสำหรับโคโค่ของแอปเปิล

เอกสารข้อแนะนำในการเขียนโปรแกรมสำหรับโคโค่ของแอปเปิล จะแบ่งหัวข้อการตั้งชื่อออกเป็น 4 หัวข้อใหญ่ ได้แก่ พื้นฐานการตั้งชื่อในโปรแกรม, การตั้งชื่อเมท็อด, การตั้งชื่อฟังก์ชันและการตั้งชื่อตัวแปร รวมถึงหัวข้อรายการคำย่อและอักษรย่อที่อนุญาตให้ใช้ได้ รายละเอียดในแต่ละหัวข้อ ดังนี

(1) พื้นฐานการตั้งชื่อในโปรแกรมประกอบไปด้วยหัวข้อย่อยที่น่าสนใจ ได้แก่ ข้อปฏิบัติพื้นฐาน, การใช้คำเติมหน้า, ข้อปฏิบัติที่เกี่ยวกับรูปแบบการใช้ตัวอักขระและการตั้งชื่อคลาส ดังนี้

- ควรสื่อความหมายชัดเจนและสั้นกระชับ ละเว้นการใช้คำย่อหรืออักษรย่อยกเว้นคำที่ระบุไว้ในหัวข้อรายการคำย่อและอักษรย่อที่อนุญาตให้ใช้ได้ ตัวอย่าง ดังรูปภาพที่ 2.1

Code	Commentary
`destinationSelection`	Good.
`destSel`	Not clear.
`setBackgroundColor:`	Good.
`setBkgdColor:`	Not clear.

รูปที่ 2.1 ตัวอย่างการตั้งชื่อพื้นฐาน

- ควรตั้งชื่อให้สอดคล้องกับส่วนต่อประสานของการโปรแกรมโคโค่

เพื่อเป็นประโยชน์ต่อคลาสที่มีการสืบทอดเมท็อดมาจากคลาสหลักและชื่อเมท็อดที่มีการทำงานเหมือนกันในต่างคลาส ควรตั้งชื่อให้เหมือนกัน ตัวอย่างดังรูปภาพที่ 2.2

Code	Commentary
`- (NSInteger)tag`	Defined in `NSView`, `NSCell`, `NSControl`.
`- (void)setStringValue:(NSString *)`	Defined in a number of Cocoa classes.

รูปที่ 2.2 ตัวอย่างการตั้งชื่อที่มีความสอดคล้อง

- ไม่ควรตั้งชื่อที่ใช้คำอ้างอิงถึงตนเองหรือคลาสที่ตนสืบทอดมา

ตัวอย่าง ดังรูปภาพที่ 2.3 ชื่อตัวแปร NSString สืบทอดคุณสมบัติรมาจากคลาส NSObject

Code	Commentary
`NSString`	Okay.
`NSStringObject`	Self-referential.

รูปที่ 2.3 ตัวอย่างการตั้งชื่อที่ไม่อ้างอิงคลาสแม่

- ควรใช้คำเติมหน้า เพื่อแยกส่วนการทำงานในโปรแกรม โดยเฉพาะเมื่อ

โปรแกรมมีการนำเข้า Thrid Party มาใช้ จะช่วยป้องกันกรณีที่มีชื่อซ้ำกัน การใช้คำเติมหน้าจะมีอักขระ 2-3 ตัวอักขระ อยู่ในรูปของตัวพิมพ์ใหญ่ทั้งหมดและไม่ใช้เครื่องหมาย Under score โดยชื่อที่อนุญาตให้ใช้คำเติมหน้า ได้แก่ ชื่อคลาส, ชื่อโปรโตคอล, ชื่อฟังก์ชัน, ชื่อค่าคงที่และโครงสร้าง ดังรูปที่ 2.4

Prefix	Cocoa Framework
NS	Foundation
NS	Application Kit
AB	Address Book
IB	Interface Builder

รูปที่ 2.4 ตัวอย่างคำเติมหน้าใน Cocoa Framework

- ชื่อควรประกอบไปด้วยคำหลายคำและไม่ใช้เครื่องหมายวรรคตอนหรือสัญลักษณ์ในการแบ่งคำเช่น Underscore รวมถึงชื่อจะต้องตั้งตามรูปแบบ Camel-case ได้แก่ อักขระแรกของคำจะต้องเป็นตัวพิมพ์ใหญ่ นอกนั้นเป็นตัวพิมพ์เล็ก กรณีเป็นชื่อเมท็อดอักขระแรกของชื่อจะเป็นตัวพิมพ์เล็ก เช่น runTheWordsTogether
- หลีกเลี่ยงการใช้ Underscore เป็นคำเติมหน้าในเมท็อดชนิด Private (Underscore อนุญาตให้ใช้เป็นคำเติมหน้าของตัวแปรชนิด instance ได้)
- ควรตั้งชื่อคลาสที่ประกอบไปด้วยคำนามที่ระบุชัดเจนว่าคลาสนี้ทำอะไรและชื่อควรประกอบไปด้วยคำเติมหน้าที่เหมาะสม

(2) การตั้งชื่อเมท็อดประกอบไปด้วยหัวข้อย่อยที่น่าสนใจ ได้แก่ ข้อปฏิบัติพื้นฐาน, การตั้งชื่อเมท็อดชนิด Delegate และการตั้งชื่อเมท็อดชนิด Private ดังนี้

- ควรเริ่มต้นชื่อด้วยอักขระตัวพิมพ์เล็กและขึ้นต้นคำถัดไปด้วยอักขระตัวพิมพ์ใหญ่โดยละเว้นคำเติมหน้า ยกเว้นคำเติมหน้าที่ระบุไว้ในหัวข้อรายการคำย่อและอักษรย่อที่อนุญาตให้ใช้ได้
- ควรขึ้นต้นชื่อเมท็อดที่แสดงการกระทำใดกับวัตถุโดยคำกริยาและไม่ใช้คำว่า ‘do’ หรือ ‘does’ เนื่องจากคำกริยาแสดงความหมายแล้ว รวมถึงไม่ใช้คำกริยาวิเศษณ์และคำคุณศัพท์ก่อนหน้าคำกริยา ดังตัวอย่าง ในรูปที่ 2.5

```
- (void)invokeWithTarget:(id)target;
- (void)selectTabViewItem:(NSTabViewItem *)tabViewItem
```

รูปที่ 2.5 ตัวอย่างการตั้งชื่อเมท็อด

- ไม่ควรใช้ ‘get’ ขึ้นต้นชื่อเมท็อดที่คืนค่าคุณลักษณะของผู้รับ ดังตัวอย่างในรูปที่ 2.6

`- (NSSize)cellSize;`	Right.
`- (NSSize)calcCellSize;`	Wrong.
`- (NSSize)getCellSize;`	Wrong.

รูปที่ 2.6 ตัวอย่างการตั้งชื่อเมท็อด (2)

- ควรมีคีย์เวิร์ดหรือคำอธิบายก่อนหน้าอาร์กิวเมนต์เพื่ออธิบายอาร์กิวเมนต์ ดังตัวอย่าง ในรูปที่ 2.7

`- (void)sendAction:(SEL)aSelector toObject:(id)anObject forAllCells:(BOOL)flag;`	Right.

รูปที่ 2.7 ตัวอย่างการตั้งชื่อเมท็อด (3)

- ไม่ควรใช้ ‘and’ เชื่อมคีย์เวิร์ดของอาร์กิวเมนต์ของเมท็อด ดังตัวอย่างรูปที่ 2.8

`- (int)runModalForDirectory:(NSString *)path file:(NSString *)name types:(NSArray *)fileTypes;`	Right.
`- (int)runModalForDirectory:(NSString *)path andFile:(NSString *)name andTypes:(NSArray *)fileTypes;`	Wrong.

รูปที่ 2.8 ตัวอย่างการตั้งชื่อเมท็อด (4)

- ควรใช้ ‘and’ เชื่อมคีย์เวิร์ดการกระทำ (Action) ของเมท็อด ดังตัวอย่างในรูปที่ 2.9

`- (BOOL)openFile:(NSString *)fullPath withApplication:(NSString *)appName andDeactivate:(BOOL)flag;`	NSWorkspace.

รูปที่ 2.9 ตัวอย่างการตั้งชื่อเมท็อด (5)

- ควรเริ่มต้นชื่อเมท็อดชนิด Delegate ด้วยชื่อคลาสที่ละเว้นคำเติมหน้าและอักขระตัวแรกเป็นตัวพิมพ์เล็ก ดังตัวอย่าง ในรูปที่ 2.10

```
- (void)browserDidScroll:(NSBrowser *)sender;
- (NSUndoManager *)windowWillReturnUndoManager:(NSWindow *)window;
```

รูปที่ 2.10 ตัวอย่างการตั้งชื่อเมท็อดชนิด Delegate (1)

- ควรใช้ ‘did’, ‘will’ หรือ ‘should’ สำหรับเมท็อดที่รอการกระทำของกิจกรรมบางอย่าง ดังตัวอย่าง ในรูปที่ 2.11

```
- (BOOL)tableView:(NSTableView *)tableView shouldSelectRow:(int)row;
- (BOOL)application:(NSApplication *)sender openFile:(NSString *)filename;
```

รูปที่ 2.11 ตัวอย่างการตั้งชื่อเมท็อดชนิด Delegate (2)

- ไม่ควรใช้ Under Score เป็นคำเติมหน้าเนื่องจากรูปแบบนี้ถูกระบุเป็นข้อปฏิบัติสำหรับรูปแบบเมท็อดชนิด Private ของแอปเปิลแล้ว แต่ให้ใช้คำเติมหน้าแล้วตามด้วย Under Score แทน เช่น BF_addobject:

(3) การตั้งชื่อฟังก์ชัน ข้อแนะนำในการตั้งชื่อคล้ายกับชื่อเมท็อด โดยจะมีข้อแนะนำที่เพิ่มเติมมา ดังนี้

- ควรใช้คำเติมหน้าเดียวกับที่ใช้สำหรับชื่อคลาสและชื่อค่าคงที่ โดยอักขระตัวแรกหลังคำเติมหน้าจะเป็นตัวอักษรพิมพ์ใหญ่ ตัวอย่างเช่น float NSHeight
- ถ้าฟังก์ชันคืนค่าคุณสมบัติของตัวมันเองที่อาร์กิวเมนต์ตำแหน่งแรก ให้ละเว้นคำกริยา แต่ถ้าคืนอื่นให้ใช้ ‘Get’ ตัวอย่างเช่น const char *NSGetSizeAndAlignment(const char *typePtr, unsigned int *sizep,unsigned int *alignp)
- ถ้าฟังก์ชันคืนค่า Boolean ให้ขึ้นต้นชื่อด้วย Inflected Verb ตัวอย่าง เช่น BOOL NSDecimalIsNotANumber(const NSDecimal *decimal)

(4) การตั้งชื่อตัวแปร

- ถ้าตัวแปรเป็นคำนามหรือคำกริยาจะอยู่ในรูปแบบ nounOrVerb หากเป็นคำคุณศัพท์จะต้องไม่ขึ้นต้นด้วย ‘is’
- ควรใช้คำเติมหน้าเป็น Underscore หากตัวแปรเป็นชนิด instance เช่น _showTitle
- ควรใช้คำเติมหน้าหากตัวแปรเป็นค่าคงที่

2) ข้อแนะนำในการเขียนโปรแกรมอ็อบเจกทีฟซีของ New York Times มีส่วนที่

เกี่ยวกับการตั้งชื่อในโปรแกรม ดังนี้

- การตั้งชื่อตัวแปร ควรตั้งชื่อให้สามารถอธิบายและสื่อความได้ชัดเจนว่าเป็นตัวแปรอะไร เก็บข้อมูลชนิดใดอย่างเหมาะสม ตัวอย่างเช่น
- NSString *title : สามารถคาดเดาจากคำว่า “title” ได้ว่าตัวแปรนี้เก็บข้อมูลชนิดสตริง
- NSString *titleHTML : สามารถระบุถึงข้อมูลของตัวแปรนี้ว่าเป็นตัวแปรที่จัดเก็บข้อมูล HTML
- NSDate *now : สามารถสื่อความได้ชัดเจน ไม่ต้องใช้ข้อมูลเพิ่มเติม
- NSDate *lastModifiedDate : หากมีเพียง lastModified อาจสื่อความกำกวม ไม่ชัดเจน
- NSString *releaseDateString : ค่าของตัวแปร (Date) ที่ถูกแสดงอยู่ในรูปของคลาสอื่น การต่อท้ายด้วยชนิดของคลาสจะช่วยลดความกำกวม นอกจากนี้ ควรหลีกเลี่ยงการตั้งชื่อด้วยอักขระเดียวยกเว้นตัวแปรที่เป็นตัวนับในการทำงานของลูป
- การตั้งชื่อควรตั้งให้ยาวเพื่ออธิบายความหมาย เช่น
    - รูปแบบที่เหมาะสม : UIButton *settingButton;
    - รูปแบบที่ไม่เหมาะสม : UIButton *setBut;
- ใช้คำเติมหน้าสำหรับชื่อคลาสและชื่อค่าคงที่ โดยค่าคงที่จะใช้รูปแบบ Camel-case ขึ้นต้นคำด้วยอักษรพิมพ์ใหญ่และคำเติมหน้าที่สัมพันธ์กับชื่อคลาส เช่น
    - รูปแบบที่เหมาะสม : static const NSTimeInterval NYTArticleViewControllerNavigationFadeAnimationDuration
    - รูปแบบที่ไม่เหมาะสม : static const NSTimeInterval fadetime

3) ข้อแนะนำในการเขียนโปรแกรมอ็อบเจกทีฟซีของเวบไซต์ Ray Wenderlic มีส่วนที่เกี่ยวกับการตั้งชื่อในโปรแกรม ดังนี้

- การตั้งชื่อควรตั้งให้ยาวเพื่ออธิบายความหมาย เช่น
    - รูปแบบที่เหมาะสม : UIButton *settingButton;
    - รูปแบบที่ไม่เหมาะสม : UIButton *setBut;
- ใช้คำเติมหน้าสำหรับชื่อคลาสและชื่อค่าคงที่ โดยค่าคงที่จะใช้รูปแบบ Camel-case โดยขึ้นต้นคำด้วยอักษรพิมพ์ใหญ่และคำเติมหน้าที่สัมพันธ์กับชื่อคลาส เช่น

- รูปแบบที่เหมาะสม : static const NSTimeInterval RWTTutorialViewControllerNavigationFadeAnimationDuration
- รูปแบบที่ไม่เหมาะสม : static const NSTimeInterval fadetime

- ชื่อตัวแปรควรใช้รูปแบบ Camel-case โดยขึ้นต้นด้วยตัวอักษรพิมพ์เล็ก เช่น
  - รูปแบบที่เหมาะสม : NSString *descriptiveVariableName
  - รูปแบบที่ไม่เหมาะสม : id varnm;
- ตัวแปรท้องถิ่นไม่ควรใช้ Underscore เป็นส่วนประกอบของชื่อ เช่น
  - รูปแบบที่เหมาะสม : variableName
  - รูปแบบที่ไม่เหมาะสม : _variableName
- ชื่อเมท็อดควรประกอบด้วย Keyword และคำก่อนหน้าอาร์กิวเมนต์เพื่ออธิบายอาร์กิวเมนต์และไม่ควรใช้ “and” เป็นคำเชื่อมระหว่างอาร์กิวเมนต์ เช่น
  - รูปแบบที่เหมาะสม :
    - (void)sendAction:(SEL)aSelector to:(id)anObject forAllCells:(BOOL)flag
    - (instancetype)initWithWidth :(CGFloat)width height:(CGFloat)height
  - รูปแบบที่ไม่เหมาะสม :
    - (void)sendAction:(SEL)aSelector :(id)anObject:(BOOL)flag
    - (instancetype)initWith:(int)width and:(int)height
- ค่าคงที่ไม่ใช้รูปแบบการตั้งชื่อแบบ Older k-style เช่น
  - รูปแบบที่เหมาะสม : RWTPinSizeMin
  - รูปแบบที่ไม่เหมาะสม : kMaxPinSize

4) รูปแบบ Cocoa สำหรับโปรแกรมอ็อบเจกทีฟซีของ Scott Stevenson มีส่วนที่เกี่ยวกับการตั้งชื่อในโปรแกรม ดังนี้

- ชื่อคลาสต้องสื่อความหมายชัดเจน, ไม่กำกวม และมีคำเติมหน้า เช่น CDCRecipient, CDCEmail
- ชื่อตัวแปรจะต้องขึ้นต้นด้วยอักษรตัวพิมพ์เล็กและคำถัดไปขึ้นต้นด้วยอักษรตัวพิมพ์ใหญ่ สื่อความหมายชัดเจน เช่น
  - รูปแบบที่เหมาะสม : NSString *hostname

: NSNumber *ipAddress

: NSArray *accounts

- รูปแบบที่ไม่เหมาะสม : NSString *HST_NM

: NSNumber *theip

: NSArray *nsma

- ชื่อตัวแปรชนิด Private ไม่ควรใช้ Underscore เป็นคำเติมหน้า เช่น
  - รูปแบบที่เหมาะสม : NSString *name
  - รูปแบบที่ไม่เหมาะสม : NSString *_name
- ชื่อตัวแปรควรระบุชนิดของตัวแปรภายในชื่อ ยกเว้นตัวแปรชนิด NSString, NSArray, NSNumber หรือ Bool เช่น
  - รูปแบบที่เหมาะสม : NSString *accountName

: NSString *mailboxes

: NSString *defaultHeaders

: NSString *userInputWasUpdated

: NSImage *previewPaneImage

  - รูปแบบที่ไม่เหมาะสม : NSString *accountNameString:

: NSString *mailboxeArray

: NSString *defaultHeadersArray

: NSString *userInputWasUpdatedBOOL

หลังจากรวบรวมข้อแนะนำจากเอกสารข้างต้น ผู้วิจัยได้นำรายการข้อแนะนำนี้ มาประชุมกับทีมพัฒนาโมบายล์แอปพลิเคชันบนระบบปฏิบัติการ iOS เพื่อคัดเลือกและปรับปรุงข้อแนะนำให้เหมาะสมกับการพัฒนาโปรแกรมอ็อบเจกทีฟซี เพื่อนำไปสู่การสร้างมาตรฐานข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซีขององค์กร

2.1.2 ข้อปฏิบัติการเขียนโปรแกรม

ข้อปฏิบัติการเขียนโปรแกรม (Coding Convention) คือกฎบางอย่างสำหรับการเขียนชุดคำสั่งในโปรแกรม โดยจะใช้ในการเขียนโปรแกรมในขั้นตอนของการพัฒนาซอฟต์แวร์ ข้อปฏิบัติจะเกี่ยวข้องกับรูปแบบของการเขียนโปรแกรม เช่น การตั้งชื่อ, การย่อหน้า โดยมีเป้าหมายในการทำให้ชุดคำสั่งซอฟต์แวร์ง่ายต่อการอ่านและทำความเข้าใจ จากการศึกษาข้อปฏิบัติการเขียนโปรแกรมและ

งานวิจัยที่เกี่ยวข้อง โครงงานนี้ได้คัดเลือกข้อปฏิบัติที่สนใจ 3 หัวข้อ คือข้อปฏิบัติการตั้งชื่อ, การใช้ Magic Number และการใช้ Literal String โดย โดยมีรายละเอียด ดังนี้

- **ข้อปฏิบัติการตั้งชื่อ (Naming Convention)**

ข้อปฏิบัติการตั้งชื่อ คือ การทำให้โปรแกรมมีความสามารถในการทำความเข้าใจมากขึ้น [6] โดยการทำให้โปรแกรมอ่านง่ายขึ้น โดยชื่อจะต้องสื่อความหมายตามการใช้งาน เช่น getTotalOfStudent ()

- **Magic Number**

Magic Number คือ ค่าที่แท้จริงของตัวอักษรที่ปรากฏในโปรแกรม [7] ตัวอย่าง เช่น total = 1.08 * price จากคำสั่ง โปรแกรมนี้ หมายเลข 1.08 คือ Magic Number ที่ไม่บ่งบอกว่าเป็นค่าของอะไร ดังนั้นจึงควรนำ Magic Number ไปประกาศเป็นค่าคงทีเพื่อให้สื่อความหมาย เช่น const double TAX_RATE_IN_TEXAS = 1.08 เป็นต้น

- **Multiple Literal String**

Multiple Literal String คือชุดของข้อมูลสตริงที่อยู่ใน " " (Double quote) และ ' ' (Single quote) [8] โดยจะมีค่าซ้ำๆกันในหลายจุดของโปรแกรม แทนที่จะประกาศค่าไว้เป็นค่าคงที่ เช่น “Hello” เป็น Literal String

2.1.3 เครื่องมือตรวจสอบข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซีในปัจจุบัน

ปัจจุบันมีเครื่องมือตรวจสอบข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซีหลากหลายเครื่องมือ ไม่ว่าจะเป็นการตรวจสอบโปรแกรมเพื่อค้นหาข้อผิดพลาด การตรวจสอบและกำหนดรูปแบบการจัดวางตำแหน่งของชุดคำสั่ง เช่น เช่นการขึ้นบรรทัดใหม่, ตำแหน่งปีกกาเปิด-ปิดของเมท็อด รวมไปถึงการตรวจสอบการตั้งชื่อในโปรแกรม จากการค้นคว้าพบเครื่องมือตรวจสอบที่น่าสนใจ ได้แก่ เครื่องมือ Objective-Clean ซึ่งมีขั้นตอนการทำงานลักษณะคล้ายคลึงกับเครื่องมือตรวจสอบในโครงงาน ผู้พัฒนาจึงให้ Objective-Clean [9] เป็นตัวต้นแบบ มีรายละเอียดการทำงาน ดังนี้

1) โปรแกรม Objective Clean

โปรแกรม Objective Clean เป็นเครื่องมือบนระบบปฏิบัติการ OSX ที่ช่วยในการกำหนดมาตรฐานและตรวจสอบการเขียนโปรแกรมภาษาอ็อกเจกทีฟซี ให้เป็นไปตามที่ได้กำหนดไว้ โดยจะตรวจสอบเรื่องของตำแหน่งเครื่องหมายปีกกา ตำแหน่งเว้นวรรค การขึ้นต้นบรรทัดใหม่เท่านั้น การกำหนดมาตรฐานโดยผู้ใช้จะอยู่ในรูปแบบของการทำแบบสอบถาม 40 ข้อ

ตัวอย่างแบบสอบถาม

```
-(void)applicationWillResignActive
```

1. Should there be a space after the (+/-) method type declaration?

Yes

No

Doesn't matter

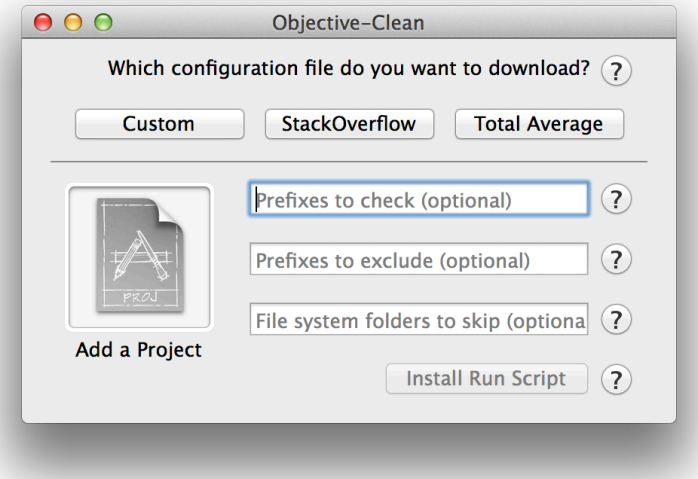
```
NSString* string; //A
NSString * string1; //B
NSString *string2; //C
NSString*string3; //D
```

2. Which image best portrays the correct syntax for declaring an object variable?

- A
- B
- C
- D
- Doesn't matter

หลังจากทำแบบสอบถาม จะได้ไฟล์โครงแบบ (Configuration) ที่จะต้องอัปโหลดเข้าเครื่องมือก่อนเริ่มใช้งาน โดยขั้นตอนการทำงานของเครื่องมืออ็อบเจกทีฟคลีน มีดังนี้

ขั้นตอนที่ 1 : เปิดใช้งานเครื่องมืออ็อบเจกทีฟคลีน จะพบส่วนต่อประสาน ดังรูปที่ 2.12



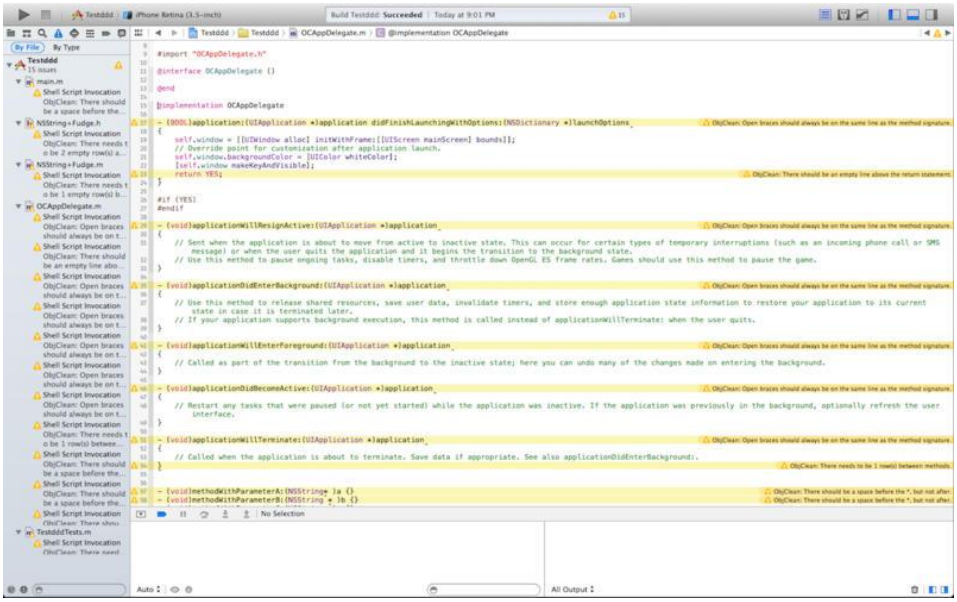
รูปที่ 2.12 หน้าจอหลักของโปรแกรมอ็อบเจกทีฟคลีน

ขั้นตอนที่ 2: เลือกไฟล์โปรเจคที่ต้องการตรวจสอบ แล้วกดติดตั้งเครื่องมือ

ขั้นตอนที่ 3: เปิดไฟล์โปรเจคด้วยโปรแกรม Xcode จะพบว่ามีการฝังการทำงานของเครื่องมืออ็อบเจกทีฟคลีนไว้ในโปรเจคแล้ว

ขั้นตอนที่ 4 : กดรันโปรแกรมเพื่อประมวลผลโปรแกรมตามการทำงานปกติ

ขั้นตอนที่ 5 : เครื่องมือแสดงผลแจ้งเตือนตำแหน่งคำสั่งในโปรแกรมที่ละเมิดกฎ พร้อมคำอธิบาย ดังรูปที่ 2.13



รูปที่ 2.13 การแสดงผลการแจ้งเตือนของเครื่องมืออ็อบเจกทีฟคลีน

นอกจากนี้ยังพบเครื่องมือตรวจสอบที่น่าสนใจและมีความเกี่ยวข้องกับการตรวจสอบการตั้งชื่อในโปรแกรม ดังนี้

2) Clang

Clang [10] เครื่องมือในการวิเคราะห์ชุดคำสั่งในโปรแกรมเพื่อค้นหาข้อผิดพลาดในโปรแกรม ภาษาซี ซีพลัสพลัส และอ็อบเจกทีฟซี โดยจะครอบคลุมการตรวจสอบที่หลากหลาย มีเป้าหมายในการค้นหาข้อบกพร่อง ด้านความปลอดภัยและการใช้งาน API การค้นหา dead code และข้อผิดพลาดของตรรกะอื่นๆ ตัวอย่างเช่น

1. core.CallAndMessage (C, C++, ObjC) : ตรวจสอบข้อผิดพลาดสำหรับการเรียกใช้ฟังก์ชันและตัวแปรที่ส่งผ่านเข้ามายังฟังก์ชัน เช่น ฟังก์ชันดึงค่าวัตถุในอาร์เรย์ โดยการส่งตำแหน่งของ Index ที่ต้องการเข้าไป โดยที่ยังไม่มีการให้ค่าเริ่มต้นกับอาร์เรย์

```
// Objective-C
@interface Subscriptable : NSObject
- (id)objectAtIndexedSubscript:(unsigned int)index;
@end

@interface MyClass : Subscriptable
@property (readwrite,assign) id x;
- (long double)longDoubleM;
@end

void test() {
  MyClass *obj1;
  id i = obj1[0]; // warn: uninitialized object pointer
}
```

2. osx.cocoa.IncompatibleMethodTypes (ObjC) : ตรวจสอบความถูกต้องของชนิดตัวแปรที่ถูกคืนค่าเมื่อเมท็อดถูกนำไป Override

```
@interface MyClass1 : NSObject
- (int)foo;
@end

@implementation MyClass1
- (int)foo { return 1; }
@end

@interface MyClass2 : MyClass1
- (float)foo;
@end

@implementation MyClass2
- (float)foo { return 1.0; } // warn
@end
```

3) OCLint

OCLing [11] เครื่องมือในการวิเคราะห์ชุดคำสั่งคอมพิวเตอร์เพื่อปรับปรุงคุณภาพและลดจำนวนข้อผิดพลาดในโปรแกรมภาษาซี ซีพลัสพลัสและอ็อบเจกทีฟซี รวมถึงค้นหาปัญหาที่อาจเกิดขึ้นใน 6 เรื่อง ดังนี้

- Possible bugs - empty if/else/try/catch/finally statements
- Unused code - unused local variables and parameters

- Complicated code - high cyclomatic complexity, NPath complexity and high NCSS
- Redundant code - redundant if statement and useless parentheses
- Code smells - long method and long parameter list
- Bad practices - inverted logic and parameter reassignment

การตรวจสอบของเครื่องมือ OCLint จะมีข้อกำหนดที่เกี่ยวข้องกับเรื่องการตั้งชื่อ ดังนี้

1. LongVariableName ตัวแปรจะต้องมีจำนวนอักขระ ไม่เกิน 20 อักขระ
2. ShortVariableName ตัวแปรจะต้องมีจำนวนอักขระมากกว่า 3 อักขระขึ้นไป

4) Uncrustify

Uncrustify [12] เครื่องมือในการเพิ่มความมีระเบียบ สวยงามให้กับชุดคำสั่งคอมพิวเตอร์ในโปรแกรมภาษาซี ซีพลัสพลัส อ็อบเจกทีฟซี จาวา Pawn และ VALA โดยมีเป้าหมายในการเพิ่มความสมบูรณ์ให้โปรแกรม สามารทำการแก้ไขได้ง่าย และชุดคำสั่งมีระเบียบ สวยงาม โดยการตรวจสอบจะเป็นไปตามหัวข้อปฏิบัติ ดังนี้

Ident code, aligning on parens, assignments, etc
Align on '=' and variable definitions
Align structure initializers
Align #define stuff
Align backslash-newline stuff
Reformat comments (a little bit)
Fix inter-character spacing
Add or remove parens on return statements
Add or remove braces on single-statement if/do/while/for statements
Supports embedded SQL 'EXEC SQL' stuff
Highly configurable

5) fauxpasapp

fauxpasapp [13] เครื่องมือในการค้นหาข้อผิดพลาดในโปรแกรมของระบบปฏิบัติการ โดยทำการแจ้งเตือนจุดบกพร่องที่อาจเกิดขึ้น รวมไปถึงการบำรุงรักษาโปรแกรม เครื่องมือ fauxpasapp จะมีข้อกำหนดที่เกี่ยวข้องกับเรื่องของการตั้งชื่อ ได้แก่

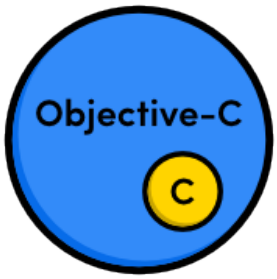
1. Unidiomatic accessor naming : แจ้งเตือนเมื่อเมท็อดในการดึงค่า เริ่มต้นด้วยคำว่า get

2. Identifier naming : อนุญาตให้ผู้ใช้สามารถปรับเปลี่ยนรูปแบบการตั้งชื่อต่างๆ ได้ตามชนิดของตัวแปรโดยใช้ความสามารถของ Regular Expression

2.1.4 ภาษาอ็อบเจกทีฟซีและโปรแกรม Xcode

1) ภาษาอ็อบเจกทีฟซี

ภาษาอ็อบเจกทีฟซี เป็นภาษาเริ่มต้นที่ใช้ในการเขียนโปรแกรมบนระบบปฏิบัติการ OSX และ iOS ซึ่งมีรากฐานมาจากภาษาซีดังรูปที่ 2.14 โดยมีการเพิ่มความสามารถของ Object-Oriented หรือการมองส่วนประกอบของโปรแกรมให้เป็นวัตถุใดๆ รวมถึงความสามารถในการจัดการวัตถุในขณะที่โปรแกรมกำลังทำงาน อ็อบเจกทีฟซีสืบทอดโครงสร้างภาษา ชนิดข้อมูลพื้นฐาน และคำสั่งควบคุมการทำงานของโปรแกรมมาจากภาษาซี โดยมีการเพิ่มเติมในส่วนของโครงสร้างสำหรับการสร้างคลาสและเมท็อด ปัจจุบันภาษาอ็อบเจกทีฟซีมีการพัฒนาต่อจากโครงสร้างดั้งเดิม โดยเวอร์ชันปัจจุบัน คืออ็อบเจกทีฟซี 2.0 [14]



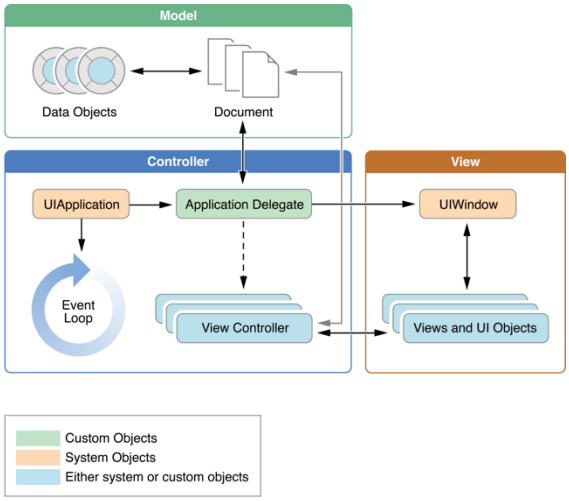
รูปที่ 2.14 ภาษาอ็อบเจกทีฟซีสืบทอดความสามารถมาจากภาษาซี

2) โปรแกรม Xcode

เครื่องมือในการพัฒนาโปรแกรมของบริษัทแอปเปิล โดยจะใช้ในการพัฒนาแอปพลิเคชันที่ใช้งานบนผลิตภัณฑ์ของแอปเปิล ได้แก่ iPad, iPhone, Apple Watch และ Mac โปรแกรม Xcode จะมีกระบวนการพัฒนาเริ่มต้นตั้งแต่สร้างแอปพลิเคชัน ทดสอบ เพิ่มประสิทธิภาพในการทำงาน จนถึงขั้นตอนของการส่งมอบแอปพลิเคชันไปยังคลังแอปพลิเคชัน (Apple Store) ภาษาโปรแกรมหลักที่ใช้ในการพัฒนาแอปพลิเคชันคือภาษาอ็อบเจกทีฟซี (Objective C) และ Swift โดยภาษา Swift เป็นภาษาใหม่ที่ทางแอปเปิลพัฒนาขึ้น เริ่มประกาศใช้อย่างเป็นทางการเมื่อปี ค.ศ 2014 โดยในช่วงเริ่มต้นโครงงานนี้ ภาษา Swift ยังอยู่ในช่วงของการพัฒนาและยังไม่เป็นที่แพร่หลายในการนำมาพัฒนาแอปพลิเคชันในอุตสาหกรรมซอฟต์แวร์

โครงสร้างการทำงานของโปรแกรม Xcode จะใช้สถาปัตยกรรมแบบ MVC

(Model-View-Controller) โดยรูปแบบนี้จะแบ่งส่วนของข้อมูล (Model) และส่วนของคำสั่งในการทำงานของโปรแกรม (Controller) ออกจากส่วนของการแสดงผล (View) ดังรูปที่ 2.15



รูปที่ 2.15 โครงสร้างการทำงานของโปรแกรม Xcode

2.1.5 Cocoa (Touch)

Cocoa และ Cocoa Touch เป็นสภาพแวดล้อมในการพัฒนาแอปพลิเคชันสำหรับระบบปฏิบัติการ OSX และ iOS เพื่ออ้างอิงถึงคลาสหรือวัตถุใดๆที่ใช้ในการสร้างส่วนต่อประสานของโปรแกรม โดยมีส่วนประกอบหลัก 2 ส่วน ได้แก่

1. Objective C Runtime รายละเอียดกล่าวในหัวข้อถัดไป
2. Two Core framework

- Appkit/UIKit Framework คลังจัดเก็บคลาสและวัตถุพื้นฐานที่เป็นส่วนประกอบในการสร้างส่วนต่อประสานของแอปพลิเคชัน ตัวอย่างคลาส เช่น UITableView, UIButton, UIImage และ UIAlertView เป็นต้น

- Foundation Framework คลังจัดเก็บคลาสและวัตถุพื้นฐานที่ใช้ในการเขียนโปรแกรม โดยสืบทอดมาจากคลาสราก คือ คลาส NSObject เพื่อระบุพฤติกรรมพื้นฐาน ชนิดของข้อมูล และการจัดเก็บข้อมูลของวัตถุใดๆ ตัวอย่างคลาส เช่น NSString, NSArray, NSDictionary และ NSCalendar เป็นต้น

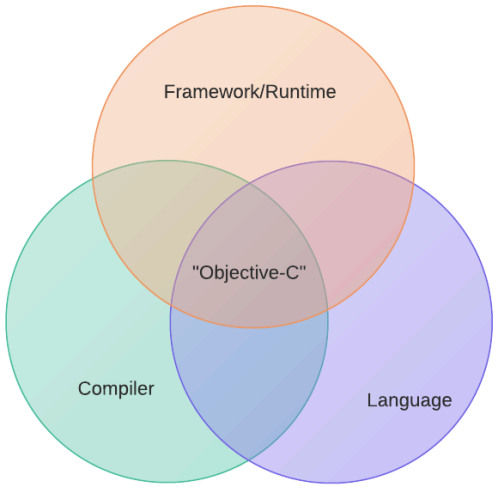
สำหรับโครงงานนี้ มีการใช้ Foundation Framework ในการพัฒนาเครื่องมือตรวจสอบข้อปฏิบัติการเขียนโปรแกรมที่ สำคัญ ได้แก่ NSRegular Expression คลาสหนึ่งในภาษาอ็อบเจกทีฟ ซี ที่มีความสามารถใน การใช้ Regular Expression กับข้อมูลสตริงในโปรแกรม โดยรูปแบบไวยากรณ์ที่ใช้ในปัจจุบันสนับสนุนการใช้ Regular Expression แบบ ICU พื้นฐานเมท็อดในการเปรียบเทียบข้อมูล จะเป็นการทำงานแบบ Block

2.1.6 คลังอ็อบเจกทีฟซีรันไทม์ (Objective C Runtime library)

เครื่องมือพัฒนา Xcode สำหรับภาษาอ็อบเจกทีฟซี มีส่วนการทำงานที่เรียกว่ารันไทม์ (Runtime) [15] โดยสามารถจัดการกับวัตถุในขณะที่โปรแกรมทำงานอยู่ ดังนั้นอ็อบเจกทีฟซีจึงไม่ได้ต้องการเพียงเฉพาะการทำงานของคอมไพเลอร์ แต่ยังต้องอาศัยระบบรันไทม์เพื่อประมวลผลโปรแกรมที่ถูกคอมไพล์แล้วด้วยดังรูปที่ 2.16 การสื่อสารระหว่างภาษาอ็อบเจกทีฟซีกับตัวรันไทม์ จะแบ่งเป็น 3 ระดับ ดังนี้

1. สื่อสารผ่านชุดคำสั่งของอ็อบเจกทีฟซี
2. สื่อสารผ่านเมท็อดในคลาส NSObject
3. สื่อสารผ่านการเรียกใช้งานรันไทม์โดยตรง

การสื่อสารระหว่างอ็อบเจกทีฟซีกับตัวรันไทม์ ผ่านการเรียกใช้งานรันไทม์ จะต้องทำการนำเข้าคลังโปรแกรม (Library) ที่ชื่อว่า "OS X Objective-C 2.0 runtime library" โดยจะรวบรวมฟังก์ชันงานที่สนับสนุนการทำงานที่เกี่ยวข้องกับโครงสร้างของข้อมูลในโปรแกรม เช่น ตัวแปรและฟังก์ชัน



รูปที่ 2.16 อ็อบเจกทีฟซีมีการทำงานร่วมกันระหว่างภาษา รันไทม์และคอมไพเลอร์

2.1.7 รันสคริปต์และภาษาเชลล์

นอกเหนือจากการประมวลผลโปรแกรมผ่านคอมไพเลอร์และรันไทม์ โปรแกรม Xcode ยังมีความสามารถในการประมวลผลผ่านคำสั่งเชลล์สคริป (Shell Script) ที่เขียนโดยภาษาเชลล์ใด ๆ โดยเชลล์เป็นโปรแกรมบนระบบยูนิกซ์ (UNIX) ทำหน้าที่ในการติดต่อระหว่างผู้ใช้งานและยูนิกซ์ ทำให้

ผู้ใช้สามารถจะป้อนคำสั่งให้ยูนิกซ์ทำงานตามที่ต้องการ คำสั่งจะเปรียบได้กับโปรแกรม หากนำคำสั่งต่างๆมาเรียงต่อกัน จะได้กลุ่มของชุดคำสั่งที่เรียกว่าคำสั่งเชลล์สคริป (Shell Script)

ส่วนการทำงานของคำสั่งเชลล์ในโปรแกรม Xcode จะอยู่ภายในส่วนของการตั้งค่าโปรแกรม โดยนักเขียนโปรแกรมจะต้องทำการเพิ่มเมนูรันสคริปต์ (Run Script) และทำการเขียนโปรแกรมคำสั่งตามที่ต้องการ ดังรูปที่ 2.17

รูปที่ 2.17 ส่วนการทำงานของเชลล์สคริปในโปรแกรม Xcode

2.1.8 Regular Expression

Regular Expression คือรูปแบบที่ถูกใช้ในการนิยามและระบุข้อมูลชนิดสตริง [16] รูปแบบเกิดจากการรวมกันของอักขระและสัญลักษณ์พิเศษ ดังตัวอย่างในตารางที่ 2.1 Regular Expression สามารถอธิบายถึงความซับซ้อนของรูปแบบสตริงได้โดยการกำหนดกฎเกณฑ์ด้วยข้อความสั้นๆ นอกจากนี้ยังสามารถใช้ในการปรับเปลี่ยนแก้ไขและเปรียบเทียบกับข้อมูลสตริงได้

ตารางที่ 2.1 ตัวอย่างอักขระที่ใช้สร้าง Regular Expression และความหมาย

Symbol	Description
character-match	Match any digit and character except \ ^ $. ? * + ( )
. (dot)	Match any single character
? (question mark)	Makes the preceding item optional
* (star)	Repeat the previous item zero or more times
+ (plus)	Repeat the previous item once or more times
| (pipe)	Cause the regex engine to match either the part on the left side or the right side
^ (caret)	Match at the start of the string the regex pattern is applied to
$	Match at the end of the string the regex pattern is applied to
{n}	Repeat the previous item exactly *n* times
{m, n}	Repeat the previous item between *n* and *m* times
/s	Shorthand character classes matching whitespace (spaces, tabs, and line breaks)
/n	Match an LF character
/d	Shorthand character classes matching word characters (letters, digits, and underscores)
/r	Match an CR character

ยกตัวอย่าง Regular expression เช่น /t$/ หมายถึง การค้นหาข้อความที่ลงท้ายด้วยอักษร t หากเปรียบเทียบกับ คำว่า match จะถือว่าไม่ตรงตามรูปแบบ แต่หากเปรียบเทียบคำว่า eat จะถือว่าตรงตามรูปแบบ

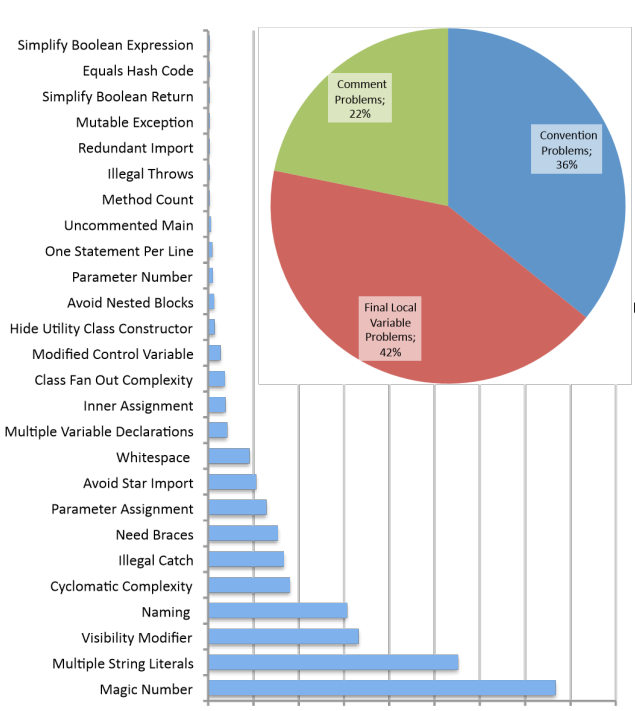
2.2 งานวิจัยที่เกี่ยวข้อง

ผู้วิจัยได้ทำการค้นคว้าแอพพลิเคชัน และงานวิจัยที่เกี่ยวข้องกับงานวิจัยที่จะพัฒนาขึ้นนี้ โดยสามารถอธิบายตามหัวข้อย่อยดังต่อไปนี้

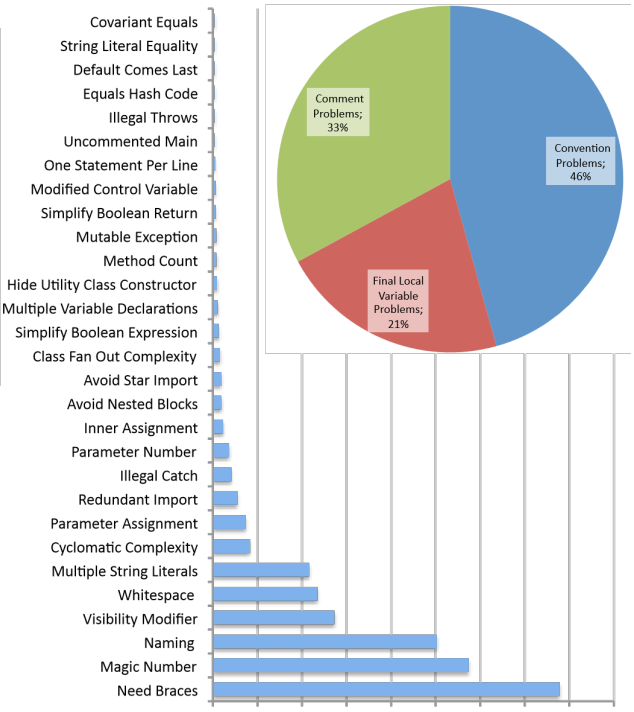
2.2.1 ศึกษาการละเมิดข้อปฏิบัติการเขียนโปรแกรม

งานวิจัย Maintainability and Source Code Conventions: An Analysis of Open Source Projects [17] โดย Michael Smit และคณะได้กล่าวถึงปัจจัยที่มีความสำคัญต่อการบำรุงรักษาซอฟต์แวร์ ไม่ได้มีเพียงความซับซ้อนเท่านั้น แต่มองว่าข้อปฏิบัติการเขียนโปรแกรมก็เป็นปัจจัยอีกอย่างหนึ่งที่สำคัญ ในการเขียนโปรแกรม นักพัฒนาจะเป็นคนตัดสินใจว่าจะเขียนโปรแกรมอย่างไร เช่น การใช้ Magic Number หรือการ Hard Code String อาจส่งผลลบต่อความสามารถในการอ่านและทำความเข้าใจ รวมไปถึงการบำรุงรักษาซอฟต์แวร์ โดยจะทำให้การเปลี่ยนแปลง

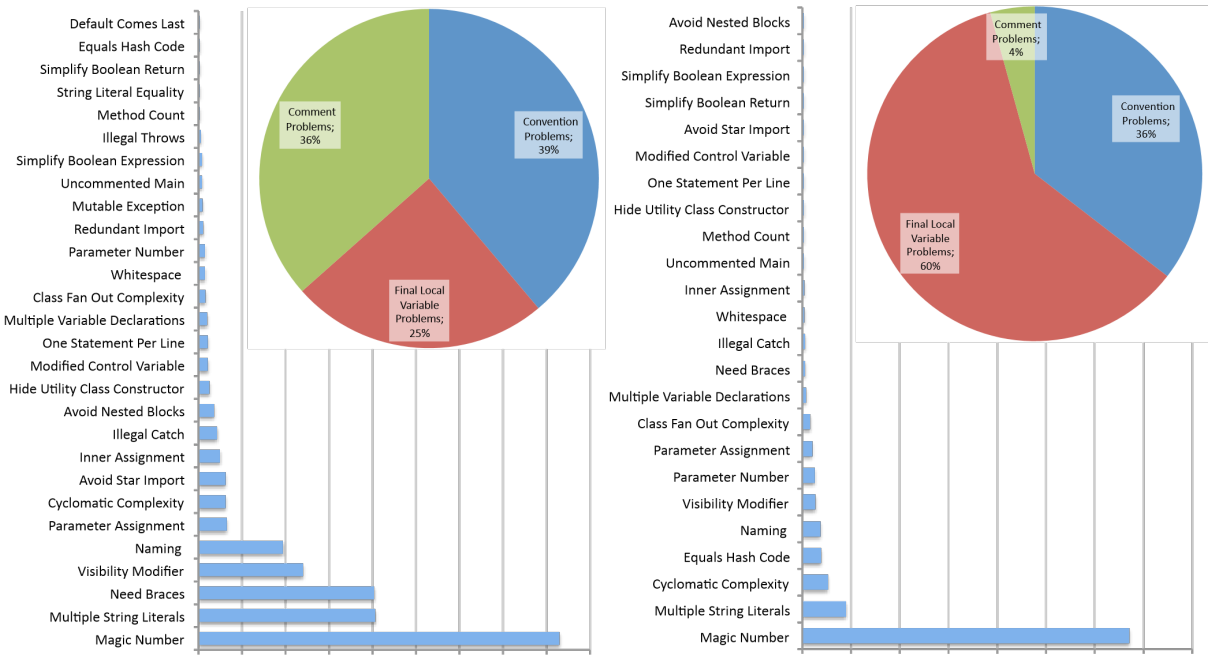
โปรแกรมเป็นไปได้ยาก งานวิจัยนี้ได้ทำการรวบรวมข้อปฏิบัติการเขียนโปรแกรมที่สัมพันธ์กับการบำรุงรักษาซอฟต์แวร์มาทั้งหมด 71 ข้อ จากการเก็บรวบรวมข้อเสนอแนะจากนักวิศวกรรมซอฟต์แวร์และนำเสนอเมทริกซ์ แสดงมุมมองต่างๆของการบำรุงรักษาซอฟต์แวร์ ชื่อ “Convention adherence” โดยใช้ข้อมูลของจำนวนและความรุนแรงของการละเมิดข้อปฏิบัติในการเขียนโปรแกรม งานวิจัยนี้คัดเลือกโปรแกรมที่จะนำมาใช้พิจารณา คือโปรแกรม Open Source ภาษาจาวา (JAVA) จำนวน 4 โปรแกรม โดยส่วนหนึ่งของการวิจัยนี้ ได้แก่ การวิเคราะห์การละเมิดข้อปฏิบัติที่พบในโปรแกรมภาษา Java พบว่าการละเมิดข้อปฏิบัติที่พบมากที่สุด 3 อันดับได้แก่ การใช้ Magic Number, การใช้ Multiple Literal String และ Naming โดยดูกราฟแสดงผลได้จาก รูปที่ 2.18



(a) Apache Ant



(b) Apache Derby



(c) Hadoop Common (d) JFreeChart

รูปที่ 2.18 กราฟสรุปผลจำนวนการละเมิดข้อปฏิบัติ

จากการศึกษางานวิจัยนี้ โครงงานพิจารณาเลือกข้อปฏิบัติ 3 ข้อดังที่กล่าวมา ในการพัฒนาเครื่องมือตรวจสอบข้อปฏิบัติการเขียนโปรแกรมภาษาอ็อบเจกทีฟซี

2.2.2 รูปแบบการตั้งชื่อที่พบในโปรแกรม

งานวิจัย Identifier Naming Conventions and Software Coding Standards: A Case Study in One School of Software [18] โดย Yanqing Wang และคณะ มองเรื่องความสำคัญของการกำหนดมาตรฐานในการเขียนโปรแกรมว่า มาตรฐานนี้จะช่วยให้นักพัฒนาโปรแกรมสามารถสื่อสารและทำงานร่วมกันได้อย่างมีประสิทธิภาพ ส่งผลต่อคุณภาพของผลิตภัณฑ์ โดยในปัจจุบันมีบริษัทที่เป็นที่ยอมรับในระดับสากล ประกาศใช้เกณฑ์ข้อปฏิบัติในการเขียนโปรแกรมอย่างเคร่งครัด การค้นคว้าและจัดอบรมที่เกี่ยวข้องกับเรื่องดังกล่าวมีมากขึ้น มาตรฐานในการเขียนโปรแกรมจึงกลายเป็นอีกปัจจัยที่ส่งผลต่อการแข่งขันกันในอุตสาหกรรมการผลิตซอฟต์แวร์

งานวิจัยนี้กล่าวถึงมาตรฐานการเขียนโปรแกรมว่าถูกแบ่งเป็น 4 หมวดหมู่ได้แก่ การจัดวาง (Layout) การตั้งชื่อ (Naming) คำอธิบาย (Comment) และการเขียนโปรแกรม (Coding) นักวิจัยได้เน้นถึงปัญหาและความสำคัญของการตั้งชื่อ เพราะเป็นปัญหาที่พบมากในปัจจุบัน ตัวอย่างปัญหาที่

พบคือการตั้งชื่อตัวแปรที่ไม่สื่อความหมาย เช่น I, jj, kkk เป็นต้น นักพัฒนาโปรแกรมส่วนมากไม่ให้ความสำคัญ ส่งผลให้โปรแกรมยากต่อการทำความเข้าใจ การบำรุงรักษาและการนำกลับมาใช้ใหม่

งานวิจัยนี้ได้ทำการเลือกรูปแบบของการตั้งชื่อที่ได้รับความนิยมมาทั้งหมด 4 รูปแบบ ได้แก่ Hungarian, Camel, Pascal และ Underscore โดยลักษณะการตั้งชื่อแต่ละรูปแบบเป็น ดังนี้

1) Hungarian ชื่อจะขึ้นต้นด้วยตัวอักษรตัวพิมพ์เล็กที่แสดงถึงขอบเขตหรือชนิดตามที่ได้มีระบุไว้ หลังจากนั้น ทุกคำจะขึ้นต้นด้วยอักษรตัวพิมพ์ใหญ่ เช่น iStudentNumber เป็นข้อมูลประเภท int

2) Camel หรือ camel case จะใช้ตัวอักษรตัวพิมพ์ใหญ่ในการแบ่งแยกคำ ส่วนที่เหลือจะเป็นตัวอักษรตัวพิมพ์เล็ก การตั้งชื่อจะเริ่มต้นด้วยตัวอักษรตัวพิมพ์เล็ก และตัวอักษรแรกของคำที่ตามมาจะเป็นตัวอักษรตัวพิมพ์ใหญ่ เช่น printEmployeePaychecks();

3) Pascal ข้อปฏิบัติการตั้งชื่อจะมีรูปแบบคล้ายกับ Camel ยกเว้นอักษรตัวแรกของชื่อจะต้องเป็นตัวพิมพ์ใหญ่ เช่น PrintEmployeePaychecks();

4) Underscore ข้อปฏิบัติการตั้งชื่อจะมีรูปแบบคล้ายกับ Camel แต่เปลี่ยนการใช้ตัวแยกคำเป็นเครื่องหมาย underscore และใช้ตัวอักษรพิมพ์เล็กทั้งหมด เช่น print_employee_pay_checks()

งานวิจัยได้ทำการนำเข้ากลุ่มไฟล์โปรแกรมตัวอย่างในภาษาจาวา จากโครงการของนักเรียนในสถาบันแห่งหนึ่ง ที่ได้ทำการรวบรวมมาเป็นระยะเวลา 3 ปี เพื่อนำมาตรวจสอบข้อปฏิบัติในการตั้งชื่อตามรูปแบบต่างๆ โดยใช้เทคนิคของคอมไพเลอร์ (Compiler) และ Regular Expression เป็นตัวสกัดชุดคำสั่งในโปรแกรม Regular Expression จะทำการหาคำที่สั้นกระชับ ตัดได้และมีความหมาย เพื่อระบุสตริงที่อยู่ในข้อความ เช่น แยกตามตัวอักษร คำ หรือรูปแบบของตัวอักษร และทำการสกัดคำออกจากชุดคำสั่งในโปรแกรม เพื่อทำการเปรียบเทียบกับข้อปฏิบัติในการตั้งชื่อทั้ง 4 แบบ โดย Regular Expression ของแต่ละรูปแบบ เป็นดังตารางที่ 2.2

ตารางที่ 2.2 Regular Expression ของการตั้งชื่อในรูปแบบต่างๆ

รูปแบบในการตั้งชื่อ	**Regular expression**
Hungary	[ab(by)(cb)(cr)(cx)(cy)(dw)(fn)hil(lp)(m_) n(np)ps(sz)w]+([A-Z][a-z]+)+
Camel	[a-z]+([A-Z][a-z]+)+
Pascal	([A-Z][a-z]+)+
Underscore	([a-z]+_)+[a-z]+

หลังจากนั้น นำคำที่สกัดมาได้มาจัดหมวดหมู่ตามรูปแบบที่ได้กล่าวข้างต้น เพื่อนับจำนวนการตั้งชื่อในรูปแบบต่างๆและวิเคราะห์หาค่าความสอดคล้องกันของการตั้งชื่อในไฟล์โปรแกรมที่รวบรวมมาได้ในแต่ละปี โดยได้ผลว่าการตั้งชื่อโดยใช้รูปแบบ Camel Case ได้รับความนิยมสูงสุด ผู้วิจัยจึงแนะนำว่า Camel Case เป็นรูปแบบที่ควรจะให้ความสำคัญและสนับสนุนให้ใช้ในวงกว้าง

จากการศึกษางานวิจัยดังกล่าว แสดงให้เห็นถึงขั้นตอนวิธีการเบื้องต้นในการตรวจสอบรูปแบบของชุดคำสั่งในโปรแกรม โดยจะต้องทำการสกัดชื่อในโปรแกรมออกมาและนำมาตรวจสอบรูปแบบของชื่อโดยใช้วิธีการ Regular Expression และจากผลของงานวิจัย ทำให้ทราบว่าการเขียนโปรแกรมส่วนมากจะนิยมใช้ในรูปแบบของ Camel Case โดยโครงงานนี้จะนำรูปแบบดังกล่าว ไปใช้เป็นข้อปฏิบัติหนึ่งในการตรวจสอบ

2.2.3 ศึกษาโครงสร้างของคำที่ใช้ในการตั้งชื่อในโปรแกรม

งานวิจัย Investigating naming convention adherence in Java references โดย Butler และคณะ [19] ศึกษารูปแบบและองค์ประกอบที่นักพัฒนาใช้ในการตั้งชื่อตัวแปรในภาษาจาวา โดยวิเคราะห์จากตัวแปรในคลังข้อมูล 60 well known java projects โดยนำชื่อมาแยกและจัดกลุ่มโทเค็นออกเป็น 6 กลุ่ม ได้แก่ Cipher, ตัวย่อชนิดของชื่อ, คำที่ปรากฏในพจนานุกรม, คำเติมหน้า (Prefix), อักษรย่อที่เป็นที่รู้จัก และโทเค็นที่ไม่รู้จัก นอกจากนี้ยังศึกษาเกี่ยวกับโครงสร้างของวลีและวลีที่ใช้ในการตั้งชื่อ พบว่ามี 5 แบบ ได้แก่ วลีนาม, วลีกริยา, วลีคุณศัพท์, วลีกริยาวิเศษณ์และวลีบุพบท

จากงานวิจัยที่เกี่ยวข้องรวมถึงเครื่องมือตรวจสอบการเขียนโปรแกรมอ็อบเจกทีฟซีที่กล่าวถึงแล้วข้างต้น พบว่ายังไม่มีงานใดที่ทำการตรวจสอบข้อปฏิบัติที่ครอบคลุมเรื่องการตั้งชื่อทั้งหมด รวมทั้งการใช้ Magic Number และ Literal String และสามารถระบุจุดที่ละเมิดในโปรแกรมได้เพื่อให้เกิดการปรับปรุงโปรแกรมต่อไป

2.3 เครื่องมือที่เกี่ยวข้อง

2.3.1 Word Segment

Word Segment [20] เป็นมอดูลหนึ่งที่ถูกพัฒนาขึ้นในภาษา Python สำหรับใช้ในการตัดคำภาษาอังกฤษ โดยอ้างอิงข้อมูลคำศัพท์จาก Google Web Trillion Word Corpus ผู้ใช้จะต้องทำการเรียกใช้ฟังก์ชัน segment โดยส่งข้อความภาษาอังกฤษที่มีจำนวนอักขระไม่เกิน 24 ตัวอักษร จากนั้น segment จะทำการกำจัดอักขระพิเศษและเว้นวรรคออก แล้วคืนค่าผลลัพธ์เป็นกลุ่มคำในรูปแบบข้อมูลอาร์เรย์ ตัวอย่าง ดังรูปที่ 2.19

```
>>> from wordsegment import segment
>>> segment('thisisatest')
['this', 'is', 'a', 'test']
```

รูปที่ 2.19 ตัวอย่างการใช้ Word Segment

เครื่องมือในการตรวจสอบของโครงงานนี้ สามารถเรียกใช้งาน Word Segment ได้โดยตรง เนื่องจากโปรแกรม Xcode และระบบปฏิบัติการ OS X มี Framework Python อยู่แล้ว โปรแกรมจึงสามารถเขียนภาษา Python ใน Xcode เพื่อเรียกใช้มอดูล Word Segment

2.3.2 WordsAPI

WordsAPI [21] เป็น API ที่ให้ผู้ใช้สามารถดึงข้อมูลคำศัพท์ภาษาอังกฤษ เช่น คำนิยาม คำพ้องและคำตรงกันข้าม ผ่านการเรียกใช้งาน API แบบ RESTful โดยอ้างอิงข้อมูลและรายละเอียดจาก Princeton WordNet และจำนวนพยางค์และการออกเสียงจาก Moby Project การทำงานของ WordsAPI ผู้ใช้จะต้องทำการร้องขอข้อมูลไปที่ URL https://wordsapiv1.p.mashape.com/words/ โดยส่งคำที่ต้องการข้อมูลต่อท้าย URL ดังกล่าว จากนั้น API จะคืนค่าผลลัพธ์ในรูปของข้อมูล JSON คืนมา ตัวอย่าง ดังในรูปที่ 2.20


```
{
  "word": "open",
  "results": [
    {
      "definition": "(of textures) full of small openings or gaps",
      "partOfSpeech": "adjective",
      "synonyms": [
        "loose"
      ],
      "similarTo": [
        "harsh",
        "coarse"
      ],
      "examples": [
        "an open texture"
      ]
    },
```


รูปที่ 2.20 ตัวอย่างผลลัพธ์การใช้งาน WordsAPI

2.3.3 WebKnox Word API

WebKnox Word API [22] เป็น API ที่ช่วยในการค้นหาคำในรูปทั่วไปในกรณีที่คำถูกเปลี่ยนรูปไป ได้แก่ คำพหูพจน์, คำกริยาช่อง 2 หรือคำกริยาช่อง 3 การทำงานของ WordsAPI ผู้ใช้จะต้องทำการร้องขอข้อมูลไปที่ URL : https://webknox-

words.p.mashape.com/words/{word}/simplePast โดยส่งคำที่ต้องการค้นหาไปใน URL จากนั้น API จะคืนค่าผลลัพธ์ในรูปของข้อมูล JSON คืนมา ดังตัวอย่าง ในรูปที่ 2.21 ส่งคำว่า gone จะได้คำในรูปทั่วไปคืนมาในคีย์ simplePresent

รูปที่ 2.21 ตัวอย่างผลลัพธ์การใช้งาน WebKnox Word API

บทที่ 3
ข้อปฏิบัติการเขียนโปรแกรมและวิธีการตรวจสอบ

3.1 ข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซี

หลังจากรวบรวมข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซีจากเอกสารข้อแนะนำที่ได้กล่าว ผู้วิจัยได้นำข้อแนะนำมาประชุมกับทีมพัฒนา เพื่อสร้างมาตรฐานข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซีขององค์กร สรุปผลได้เป็นรายการข้อปฏิบัติทั้งหมด 36 ข้อ แสดงดังรูปที่ 3.1

Group	Convention	Correct	Wrong
General	Names should consist of meaningful words [2]	`index,string,channel`	`x,str,ch`
	Names should not use abbreviation [2],[3]	`setBackgroundColor:`	`setBkgdColor:`
	Names should consist of multiple words [2]	`checkUserPermission`	`permission`
	Names should not contain name of parent class [2]	`string`	`stringObject`
Class name	Names should contain a noun [2]	`CategoryView`	`CategorizeView`
	Names should have prefix [2],[3]	`TMVMainView`	`MainView`
	Names should be camel-case with all words capitalized [2], [5]	`TMVLoginView`	`TMVloginview`
	Names should have a suffix that indicates type [iOS Development Team]	`TMVRegisterViewController`	`TMVRegister`
Method/ Function	Names should be camel-case [2]	`indexOfObject`	`index_of_object`
	Names should start with a verb followed by a noun [iOS Development Team]	`selectTabViewItem`	`tableViewSelected`
	Names should not contain “do” or “does” [2]	`loginToSystem`	`doLoginToSystem`
	Names of getter method should not start with “get” [2]	`cellSize`	`getCellSize`
	Names should have keywords before all arguments [2]	`sendAction:(SEL)aSelector toObject:(id)anObject`	`sendAction:(SEL)aSelector : (id)anObject`
	Names should have a word before the argument that describes the argument [2]	`viewWithTag:(NSInteger)aTag`	`taggedView:(int)aTag`
	Names should not have “and” to link arguments [2]	`runFilePath:(NSString *)path file:(NSString *)name types :(NSArray *)fileTypes`	`runFilePath:(NSString *)path andFile:(NSString*)name andTypes :(NSArray *)fileTypes`
	Names should have “and” to link actions [2]	`openFile:(NSString *)fullPath andDeactivate:(BOOL)flag`	`openFile:(NSString *)fullPath Deactivate:(BOOL)flag`
Method	Names should not have prefix [2]	`shareLocation`	`TMVShareLocation`
- Delegate Method	Names should start with Class name and omit prefix [2]	`alertViewReceiveButtonAtIndex`	`receiveButtonAtIndex`
	Names should contain “did”,”will”,”should” for notifying when something has happened [2]	`alertViewWillDismiss`	`alertViewDismiss`
- Private Method	Names should not use underscore as a prefix [2]	`updateContent`	`_updateContent`
	Names should have prefix followed by underscore [2]	`TMV_updateContent`	`_updateContent`
Function	Names should have prefix [2]	`NSHighlightRect`	`HighlightRect`
	Names should be camel-case with all words capitalized [2]	`NSDeallocateObject`	`NSdeallocateObject`
	Names should omit verb if function returns object directly [2]	`height`	`getHeight`
	Names should omit “is” if function returns Boolean [2]	`(Bool)editable`	`(Bool)isEditable`
Property/ Variable	Names should have a suffix that indicates type [5]	`userNameString`	`userName`
	Names should be camel-case with the first word starting with lowercase letter [3]	`localizedUppercaseString`	`localizeduppercasestring`
	Names of Boolean type should start with is,has,show [2]	`showAdvertise`	`advertise`
	Names should omit “is” if it is expressed as an adjective [2]	`editable`	`isEditable`
	Names should be expressed as a Noun, Verb or Adjective [2]	`title,userName,logo`	`from,with,after`
Instance variable	Names should have underscore as a prefix [2]	`_contentName`	`contentName`
Constant	Names should be camel-case with all words capitalized [2]	`TMVEncodingDetectionSuggestedKey`	`TMVencodingdetectionsuggestedkey`
	Names should not start with ‘k' (Older k-style) [4]	`TMVTelephoneNumber,TMVEmail`	`kTel,kEmail`
	Names should have prefix [2],[3],[4]	`TMVLoginViewHeight`	`LoginViewHeight`
Magic Number	Program should not use Magic Number [iOS Development Team]	`const float TMVHeightOfMenuView = 56.0;`	`area = width x 56.0;`
Literal String	Program should not use Literal String [iOS Development Team]	`static NSString *const TMVCompanyEmail = “abc@gmail.com”`	`mail.sender = “abc@gmail.com”`

รูปที่ 3.1 รายการข้อปฏิบัติทั้งหมด

ข้อปฏิบัตินี้จะจำแนกเป็นหมวดหมู่ ได้แก่ข้อปฏิบัติพื้นฐาน 4 ข้อ การตั้งชื่อคลาส 4 ข้อ การตั้งชื่อเมท็อดและฟังก์ชัน 17 ข้อ การตั้งชื่อตัวแปร 6 ข้อ และการตั้งชื่อค่าคงที่ 3 ข้อ นอกจากนี้ยังเพิ่มเติมข้อปฏิบัติในการละเว้นการใช้ Magic Number และ Literal String (ในจำนวนนี้มี 4 ข้อที่เป็นข้อปฏิบัติที่ฝ่ายงานพัฒนาขององค์กรกรณีศึกษา [iOS Development Team] เสนอเพิ่มเติมจากข้อปฏิบัติโดยทั่วไปของอ็อบเจกทีฟซี) ทั้งนี้ Magic Number หมายถึงค่าตัวเลขที่ปรากฏในโปรแกรมและไม่ได้ถูกนำไปประกาศเป็นค่าคงที่ โดยจะยกเว้นค่าตัวเลข -1, 0, 1 และ 2 [7], [13] ตัวอย่างเช่น view.frame = CGRectMake(20, 30, 320, 568) ใช้ Magic Number ในการกำหนดตำแหน่งความกว้างและความสูงของหน้าแสดงผล view ทำให้ตัวเลขที่ระบุไม่สื่อความหมายว่าเป็นค่าของอะไร ส่วน Literal String หมายถึงค่าสตริงที่อยู่ภายในอัญประกาศซึ่งไม่ได้ถูกนำไปประกาศเป็นค่าคงที่ [7], [8] และอาจมีค่าสตริงเดียวกันในหลายจุดของโปรแกรม ตัวอย่างเช่น self.titleLabel.font = [UIFont fontWithName:@"True Bold" size:28.0] หากโปรแกรมมีการเปลี่ยนแปลงรูปแบบอักษร จะต้องแก้ไขชื่อรูปแบบอักษรในทุกจุดของโปรแกรม

3.2 ตรวจสอบชื่อตามข้อปฏิบัติการตั้งชื่อ

การตรวจสอบข้อปฏิบัติการตั้งชื่อ จะเริ่มต้นจากการตรวจสอบการตั้งชื่อเบื้องต้น หลังจากนั้นจึงแยกตรวจสอบตามประเภท

3.2.1 การตรวจสอบการตั้งชื่อเบื้องต้น

รายการข้อปฏิบัติการตั้งชื่อเบื้องต้น

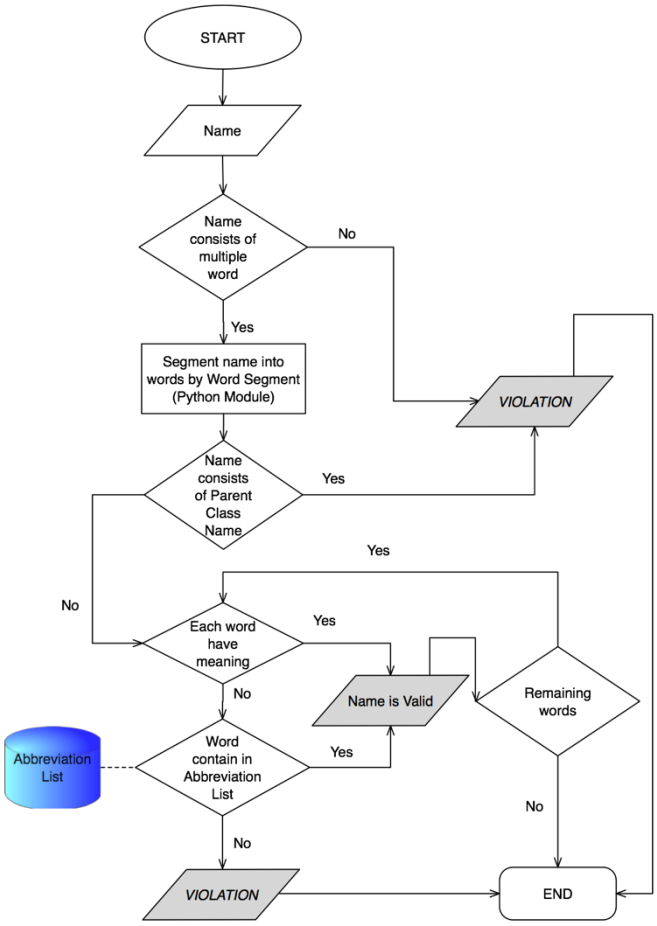
ข้อปฏิบัติ	ชื่อควรประกอบด้วยคำที่มีความหมาย	**รหัส : CG01**
วิธีการตรวจสอบ	ตรวจสอบความหมายของคำด้วย WordsAPI	
ผ่าน	API แสดงผลลัพธ์รายละเอียดข้อมูลของคำ	
ไม่ผ่าน	API แสดงผลลัพธ์ว่าไม่พบคำดังกล่าว	

ข้อปฏิบัติ	ชื่อไม่ควรใช้คำย่อหรืออักษรย่อ	**รหัส : CG02**
วิธีการตรวจสอบ	ตรวจสอบความหมายของคำด้วย WordsAPI	
ผ่าน	API แสดงผลลัพธ์รายละเอียดข้อมูลของคำ	
ไม่ผ่าน	API แสดงผลลัพธ์ว่าไม่พบคำดังกล่าว	

ข้อปฏิบัติ	ชื่อควรประกอบไปด้วยคำมากกว่าหนึ่งคำ	**รหัส : CG03**
วิธีการตรวจสอบ	ตรวจสอบจำนวนคำของชื่อ	
ผ่าน	จำนวนคำมากกว่าหนึ่ง	
ไม่ผ่าน	จำนวนคำเท่ากับหนึ่ง	

ข้อปฏิบัติ	ชื่อไม่ควรมีชื่อของคลาสแม่เป็นส่วนประกอบ	**รหัส : CG04**
วิธีการตรวจสอบ	ตรวจสอบคำที่เป็นส่วนประกอบของชื่อ	
ผ่าน	ไม่มีชื่อคลาสแม่เป็นส่วนประกอบของชื่อ	
ไม่ผ่าน	มีชื่อคลาสแม่เป็นส่วนประกอบของชื่อ	

ขั้นตอนการตรวจสอบเบื้องต้น ดังรูปที่ 3.2



รูปที่ 3.2 ขั้นตอนการตรวจสอบการตั้งชื่อเบื้องต้น

3.2.2 การตรวจสอบการตั้งชื่อคลาส

รายการข้อปฏิบัติการตั้งชื่อคลาส

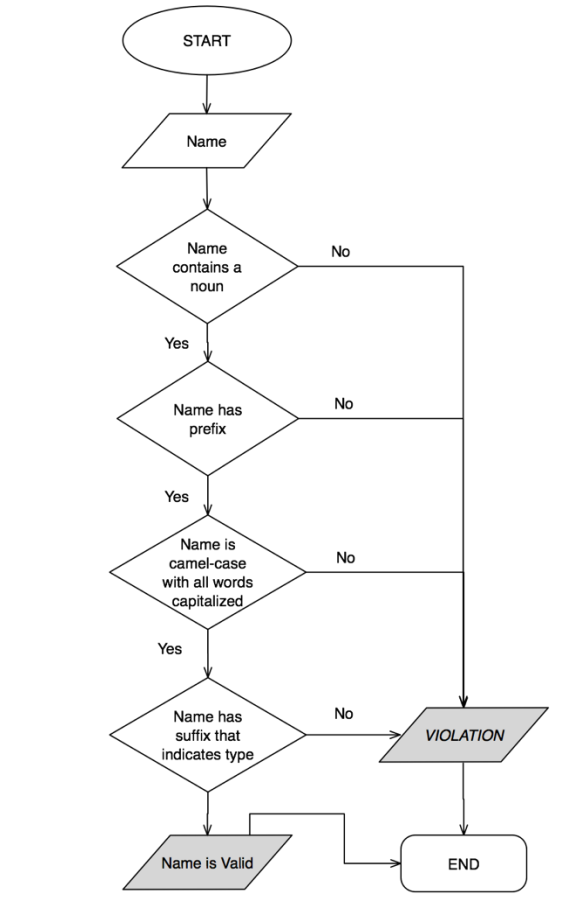
ข้อปฏิบัติ	ชื่อควรประกอบไปด้วยคำนาม	**รหัส : CC01**
วิธีการตรวจสอบ	ตรวจสอบชนิดของคำโดยใช้ WordsAPI	
ผ่าน	ชื่อมีคำนามเป็นส่วนประกอบ	
ไม่ผ่าน	ชื่อไม่มีคำนามเป็นส่วนประกอบ	

ข้อปฏิบัติ	ชื่อควรมีคำเติมหน้า	**รหัส : CC02**
วิธีการตรวจสอบ	ตรวจสอบคำแรกของชื่อและคำเติมหน้า	
ผ่าน	คำแรกของชื่อเป็นคำเติมหน้าที่ผู้ใช้กำหนด	
ไม่ผ่าน	คำแรกของชื่อไม่ใช่คำเติมหน้าที่ผู้ใช้กำหนด	

ข้อปฏิบัติ	ชื่อควรอยู่ในรูปแบบ Camel-Case โดยอักษรแรกของชื่อจะต้องเป็นอักษรตัวพิมพ์ใหญ่	**รหัส : CC03**
วิธีการตรวจสอบ	ตรวจสอบอักขระตัวพิมพ์เล็ก - ตัวพิมพ์ใหญ่	
ผ่าน	ตัวอักษรแรกของทุกคำเป็นตัวพิมพ์ใหญ่และตัวอักษรที่เหลือเป็นตัวพิมพ์เล็ก	
ไม่ผ่าน	ตัวอักษรแรกของทุกคำไม่เป็นตัวพิมพ์ใหญ่หรือตัวอักษรที่เหลือไม่เป็นตัวพิมพ์เล็ก	

ข้อปฏิบัติ	ชื่อควรต่อท้ายด้วยการระบุชนิดของคลาส	**รหัส : CC04**
วิธีการตรวจสอบ	ตรวจสอบคำสุดท้ายของชื่อ	
ผ่าน	คำสุดท้ายของชื่อเป็นชนิดของคลาส	
ไม่ผ่าน	คำสุดท้ายของชื่อไม่เป็นชนิดของคลาส	

ขั้นตอนการตรวจสอบชื่อคลาส ดังรูปที่ 3.3



รูปที่ 3.3 ขั้นตอนการตรวจสอบการตั้งชื่อคลาส

3.2.3 การตรวจสอบการตั้งชื่อเมท็อดและฟังก์ชัน

รายการข้อปฏิบัติการตั้งชื่อเมท็อดและฟังก์ชัน

ข้อปฏิบัติ	ชื่อควรอยู่ในรูปแบบ Camel-Case โดยอักษรแรกของชื่อจะต้องเป็นอักษรตัวพิมพ์เล็ก	**รหัส : CMF01**
วิธีการตรวจสอบ	ตรวจสอบอักขระตัวพิมพ์เล็ก - ตัวพิมพ์ใหญ่	
ผ่าน	ตัวอักษรแรกของคำที่สองเป็นต้นไปเป็นตัวพิมพ์ใหญ่และตัวอักษรที่เหลือเป็นตัวพิมพ์เล็ก	
ไม่ผ่าน	ตัวอักษรแรกของคำที่สองเป็นต้นไปไม่เป็นตัวพิมพ์ใหญ่หรือตัวอักษรที่เหลือไม่เป็นตัวพิมพ์เล็ก	

ข้อปฏิบัติ	ชื่อควรเริ่มต้นด้วยคำกริยาตามด้วยคำนาม	**รหัส : CMF02**
วิธีการตรวจสอบ	ตรวจสอบชนิดของคำโดยใช้ WordsAPI	
ผ่าน	คำแรกเป็นคำกริยาและคำที่สองเป็นคำนาม	
ไม่ผ่าน	คำแรกเป็นไม่คำกริยาหรือคำที่สองไม่เป็นคำนาม	

ข้อปฏิบัติ	ชื่อไม่ควรใช้ ‘do’ หรือ ‘does’	**รหัส : CMF03**
วิธีการตรวจสอบ	ตรวจสอบคำที่เป็นส่วนประกอบของชื่อ	
ผ่าน	ไม่มีคำว่า ‘do’ หรือ ‘does’ เป็นส่วนประกอบของชื่อ	
ไม่ผ่าน	มีคำว่า ‘do’ หรือ ‘does’ เป็นส่วนประกอบของชื่อ	

ข้อปฏิบัติ	ชื่อ Getter เมท็อด ไม่ควรเริ่มต้นด้วย ‘get’	**รหัส : CMF04**
วิธีการตรวจสอบ	ตรวจสอบคำที่เป็นส่วนประกอบของชื่อ	
ผ่าน	คำแรกของชื่อไม่ใช่ ‘get’	
ไม่ผ่าน	คำแรกของชื่อเป็นคำว่า ‘get’	

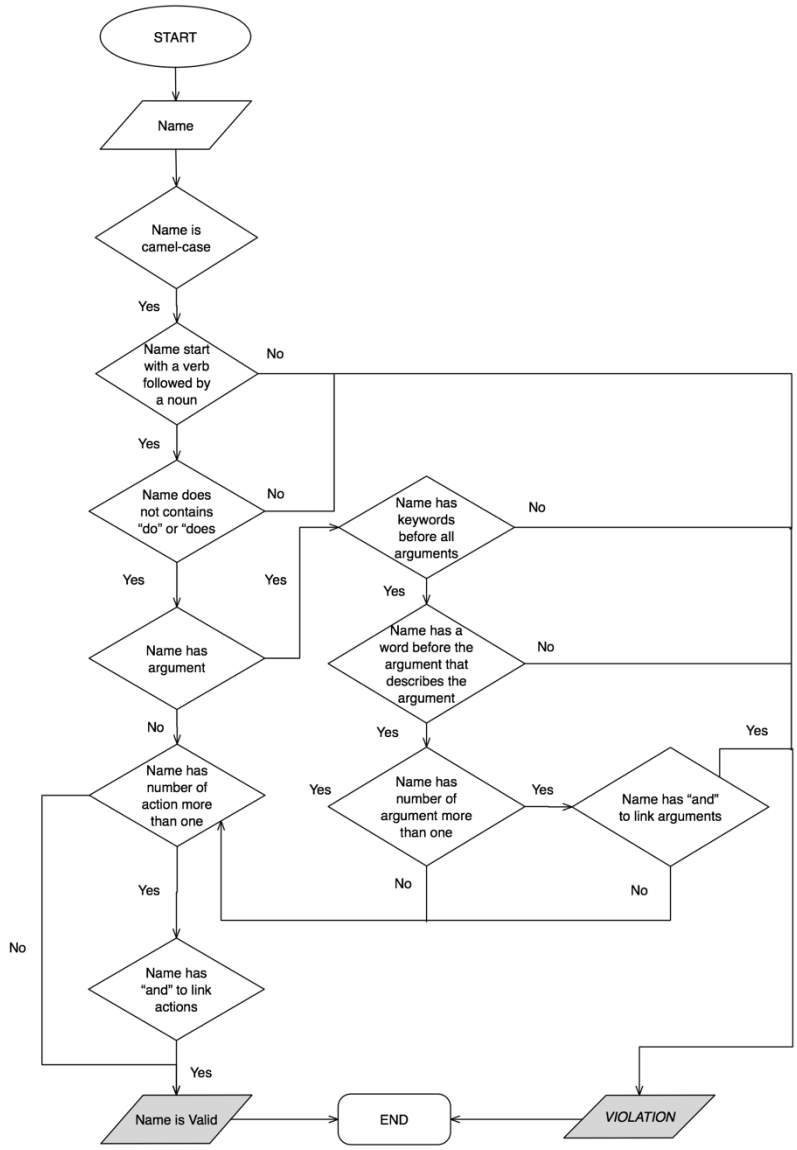
ข้อปฏิบัติ	ชื่อควรมีคีย์เวิร์ดก่อนอาร์กิวเมนต์	**รหัส : CMF05**
วิธีการตรวจสอบ	ตรวจสอบคำก่อนอาร์กิวเมนต์	
ผ่าน	มีคำก่อนหน้าอาร์กิวเมนต์ทุกตัว (ก่อนสัญลักษณ์ :)	
ไม่ผ่าน	ไม่มีคำก่อนหน้าอาร์กิวเมนต์ทุกตัว (ก่อนสัญลักษณ์ :)	

ข้อปฏิบัติ	ชื่อควรมีคำก่อนอาร์กิวเมนต์เพื่ออธิบายอาร์กิวเมนต์	**รหัส : CMF6**
วิธีการตรวจสอบ	ตรวจสอบคำก่อนอาร์กิวเมนต์	
ผ่าน	คำก่อนอาร์กิวเมนต์เป็นชื่อเดียวกันกับอาร์กิวเมนต์	
ไม่ผ่าน	คำก่อนอาร์กิวเมนต์ไม่เป็นชื่อเดียวกันกับอาร์กิวเมนต์	

ข้อปฏิบัติ	หากมีอาร์กิวเมนต์มากกว่าหนึ่ง ชื่อควรมีคำว่า and เชื่อมระหว่างอาร์กิวเมนต์	**รหัส : CMF07**
วิธีการตรวจสอบ	ตรวจสอบคำระหว่างอาร์กิวเมนต์	
ผ่าน	มีคำว่า ‘and’ ระหว่างอาร์กิวเมนต์	
ไม่ผ่าน	ไม่มีคำว่า ‘and’ ระหว่างอาร์กิวเมนต์	

ข้อปฏิบัติ	หากมี action มากกว่าหนึ่ง ไม่ชื่อควรใช้คำว่า and เชื่อมระหว่าง action	**รหัส : CMF08**
วิธีการตรวจสอบ	ตรวจสอบคำระหว่าง action	
ผ่าน	ไม่มีคำว่า ‘and’ ระหว่าง action	
ไม่ผ่าน	มีคำว่า ‘and’ ระหว่าง action	

ขั้นตอนการตรวจสอบเมท็อดและฟังก์ชัน ดังรูปที่ 3.4



รูปที่ 3.4 ขั้นตอนการตรวจสอบการตั้งชื่อเมท็อดและฟังก์ชัน

3.2.4 การตรวจสอบการตั้งชื่อเมท็อด

รายการข้อปฏิบัติการตั้งชื่อเมท็อด

ข้อปฏิบัติ	ชื่อไม่ควรมีคำเติมหน้า	**รหัส : CM01**
วิธีการตรวจสอบ	ตรวจสอบคำเติมหน้า	
ผ่าน	ชื่อไม่มีคำเติมหน้า	
ไม่ผ่าน	ชื่อมีคำเติมหน้า	

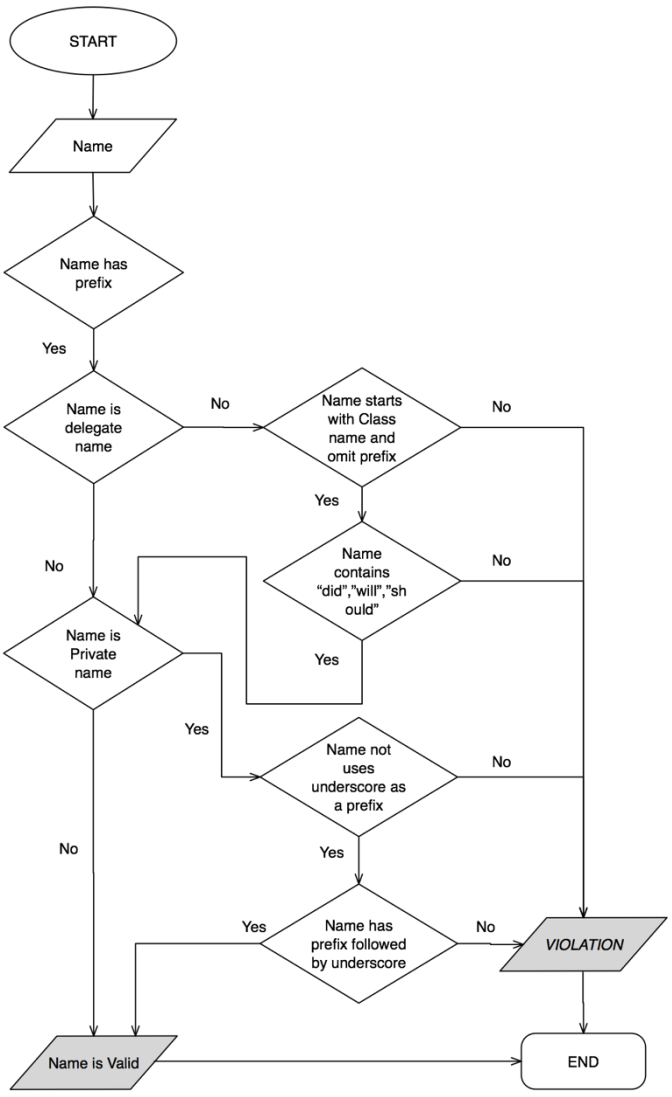
ข้อปฏิบัติ	ชื่อของเมท็อด Delegate ควรขึ้นต้นด้วยชื่อคลาส	**รหัส : CM02**
วิธีการตรวจสอบ	ตรวจสอบคำแรกของชื่อ	
ผ่าน	คำแรกของชื่อเป็นชื่อคลาส	
ไม่ผ่าน	คำแรกของชื่อไม่ใช่ชื่อคลาส	

ข้อปฏิบัติ	ชื่อเมท็อด Delegate ควรประกอบไปด้วยคำว่า ‘did’, ‘will’ `หรือ ‘should’	**รหัส : CM03**
วิธีการตรวจสอบ	ตรวจสอบคำที่เป็นส่วนประกอบของชื่อ	
ผ่าน	มีคำว่า ‘did’ , ‘will’ หรือ ‘should’ เป็นส่วนประกอบของชื่อ	
ไม่ผ่าน	ไม่มีคำว่า ‘did’ , ‘will’ หรือ ‘should’ เป็นส่วนประกอบของชื่อ	

ข้อปฏิบัติ	ชื่อเมท็อด Private ไม่ควรใช้คำเติมหน้าเป็น underscore	**รหัส : CM04**
วิธีการตรวจสอบ	ตรวจสอบคำเติมหน้า	
ผ่าน	ไม่มีคำเติมหน้าเป็น underscore	
ไม่ผ่าน	มีคำเติมหน้าเป็น underscore	

ข้อปฏิบัติ	ชื่อเมท็อด Private ควรมีคำเติมหน้า	**รหัส : CM05**
วิธีการตรวจสอบ	ตรวจสอบคำเติมหน้า	
ผ่าน	ชื่อมีคำเติมหน้า	
ไม่ผ่าน	ชื่อไม่มีคำเติมหน้า	

ขั้นตอนการตรวจสอบเมท็อด ดังรูปที่ 3.5



รูปที่ 3.5 ขั้นตอนการตรวจสอบการตั้งชื่อเมท็อด

3.2.5 การตรวจสอบการตั้งชื่อฟังก์ชัน

รายการข้อปฏิบัติการตั้งชื่อฟังก์ชัน

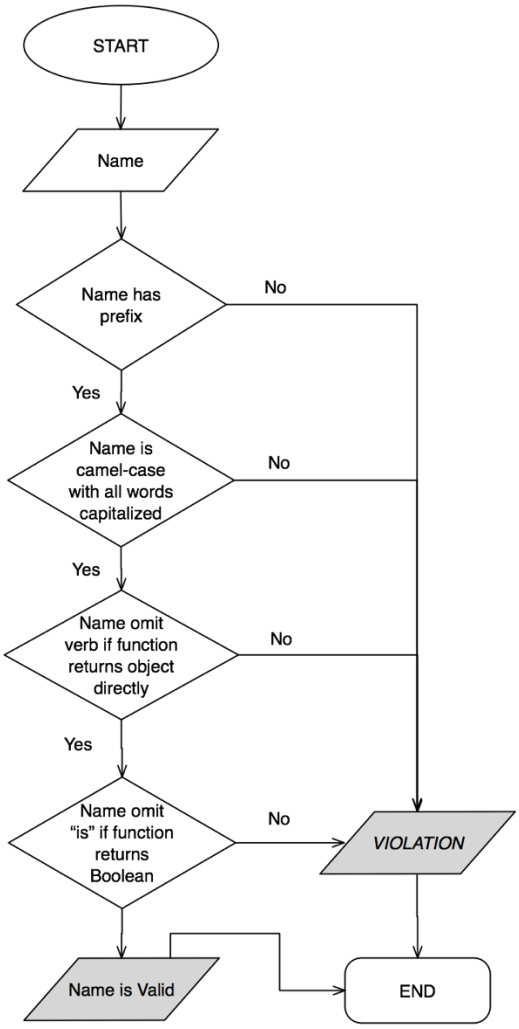
ข้อปฏิบัติ	ชื่อควรมีคำเติมหน้า	**รหัส : CF01**
วิธีการตรวจสอบ	ตรวจสอบคำเติมหน้า	
ผ่าน	ชื่อมีคำเติมหน้า	
ไม่ผ่าน	ชื่อไม่มีคำเติมหน้า	

ข้อปฏิบัติ	ชื่อควรอยู่ในรูปแบบ Camel-Case โดยตัวอักษรแรกเป็นตัวพิมพ์ใหญ่	**รหัส : CF02**
วิธีการตรวจสอบ	ตรวจสอบอักขระตัวพิมพ์เล็ก - ตัวพิมพ์ใหญ่	
ผ่าน	ตัวอักษรแรกของทุกคำเป็นตัวพิมพ์ใหญ่และตัวอักษรที่เหลือเป็นตัวพิมพ์เล็ก	
ไม่ผ่าน	ตัวอักษรแรกของทุกคำไม่เป็นตัวพิมพ์ใหญ่หรือตัวอักษรที่เหลือไม่เป็นตัวพิมพ์เล็ก	

ข้อปฏิบัติ	ชื่อไม่ควรขึ้นต้นด้วยคำกริยาถ้าฟังก์ชัน return คุณลักษณะของตัวมันเอง	**รหัส : CF03**
วิธีการตรวจสอบ	ตรวจสอบชนิดของคำโดยใช้ WordsAPI	
ผ่าน	คำแรกของชื่อเป็นคำกริยา	
ไม่ผ่าน	คำแรกของชื่อไม่ใช่คำกริยา	

ข้อปฏิบัติ	ชื่อไม่ควรขึ้นต้นด้วย is ถ้าฟังก์ชันคืนค่า Boolean	**รหัส : CF04**
วิธีการตรวจสอบ	ตรวจสอบคำแรกของชื่อ	
ผ่าน	คำแรกของชื่อไม่ใช่คำว่า ‘is’	
ไม่ผ่าน	คำแรกของชื่อเป็นคำว่า ‘is’	

ขั้นตอนการตรวจสอบฟังก์ชัน ดังรูปที่ 3.6



รูปที่ 3.6 ขั้นตอนการตรวจสอบการตั้งชื่อฟังก์ชัน

3.2.6 การตรวจสอบการตั้งชื่อตัวแปร

รายการข้อปฏิบัติการตั้งชื่อตัวแปร

ข้อปฏิบัติ	ชื่อควรต่อท้ายด้วยการระบุชนิด	**รหัส : CV01**
วิธีการตรวจสอบ	ตรวจสอบคำสุดท้ายของชื่อ	
ผ่าน	คำสุดท้ายของชื่อเป็นชนิดของตัวแปร	
ไม่ผ่าน	คำสุดท้ายของชื่อไม่ใช่ชนิดของตัวแปร	

ข้อปฏิบัติ	ชื่อควรอยู่ในรูปแบบ Camel-Case โดยอักษรแรกของชื่อจะต้องเป็นอักษรตัวพิมพ์เล็ก	**รหัส : CV02**
วิธีการตรวจสอบ	ตรวจสอบอักขระตัวพิมพ์เล็ก - ตัวพิมพ์ใหญ่	
ผ่าน	ตัวอักษรแรกของคำที่สองเป็นต้นไปเป็นตัวพิมพ์ใหญ่และตัวอักษรที่เหลือเป็นตัวพิมพ์เล็ก	
ไม่ผ่าน	ตัวอักษรแรกของคำที่สองเป็นต้นไปไม่เป็นตัวพิมพ์ใหญ่หรือตัวอักษรที่เหลือไม่เป็นตัวพิมพ์เล็ก	

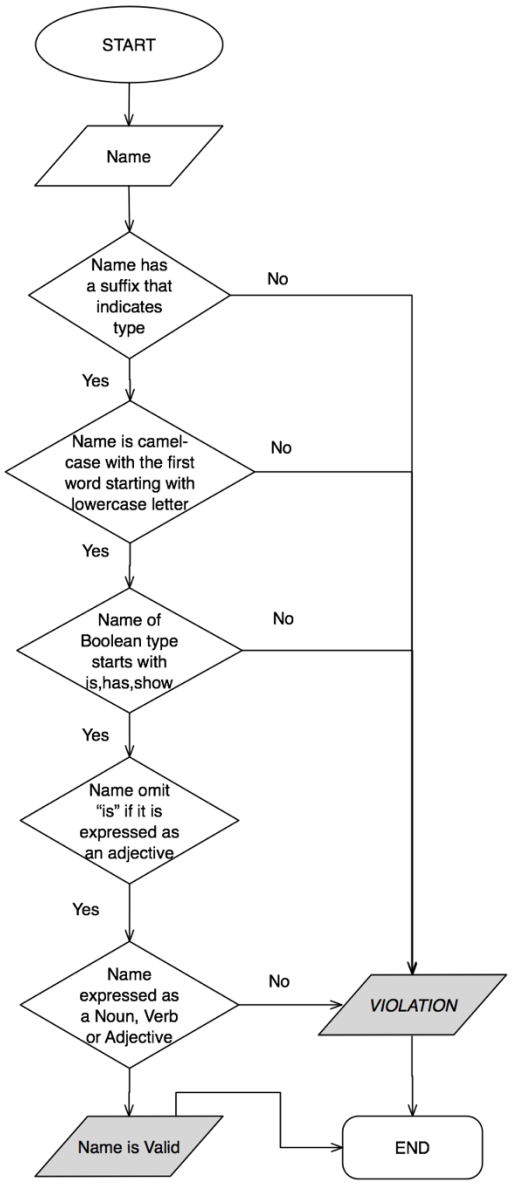
ข้อปฏิบัติ	ชื่อที่เป็นชนิด Boolean ควรขึ้นต้นด้วย ‘is’ , ‘has’ หรือ ‘show’	**รหัส : CV03**
วิธีการตรวจสอบ	ตรวจสอบคำแรกของชื่อ	
ผ่าน	มีคำว่า ‘is’ , ‘has’ หรือ ‘show’ เป็นคำแรกของชื่อ	
ไม่ผ่าน	ไม่มีคำว่า ‘is’ , ‘has’ หรือ ‘show’ เป็นคำแรกของชื่อ	

ข้อปฏิบัติ	ชื่อไม่ควรเริ่มต้นด้วย is หากชื่อเป็นคำคุณศัพท์	**รหัส : CV04**
วิธีการตรวจสอบ	ตรวจสอบคำแรกของชื่อ	
ผ่าน	คำแรกของชื่อไม่ใช่คำว่า ‘is’	
ไม่ผ่าน	คำแรกของชื่อเป็นคำว่า ‘is’	

ข้อปฏิบัติ	ชื่อควรเป็นคำนาม คำกริยาและคำคุณศัพท์	**รหัส : CV05**
วิธีการตรวจสอบ	ตรวจสอบชนิดของชื่อ	
ผ่าน	ชื่อเป็นคำนาม คำกริยาและคำคุณศัพท์	
ไม่ผ่าน	ชื่อไม่ใช่คำนาม คำกริยาและคำคุณศัพท์	

ข้อปฏิบัติ	ชื่อตัวแปร instance ควรขึ้นต้นด้วย underscore	**รหัส : CV06**
วิธีการตรวจสอบ	ตรวจสอบอักขระแรกของชื่อ	
ผ่าน	อักขระแรกของชื่อเป็น underscore	
ไม่ผ่าน	อักขระแรกของชื่อไม่ใช่ underscore	

ขั้นตอนการตรวจสอบชื่อตัวแปร ดังรูปที่ 3.7



รูปที่ 3.7 ขั้นตอนการตรวจสอบชื่อตัวแปร

3.2.7 การตรวจสอบการตั้งชื่อค่าคงที่

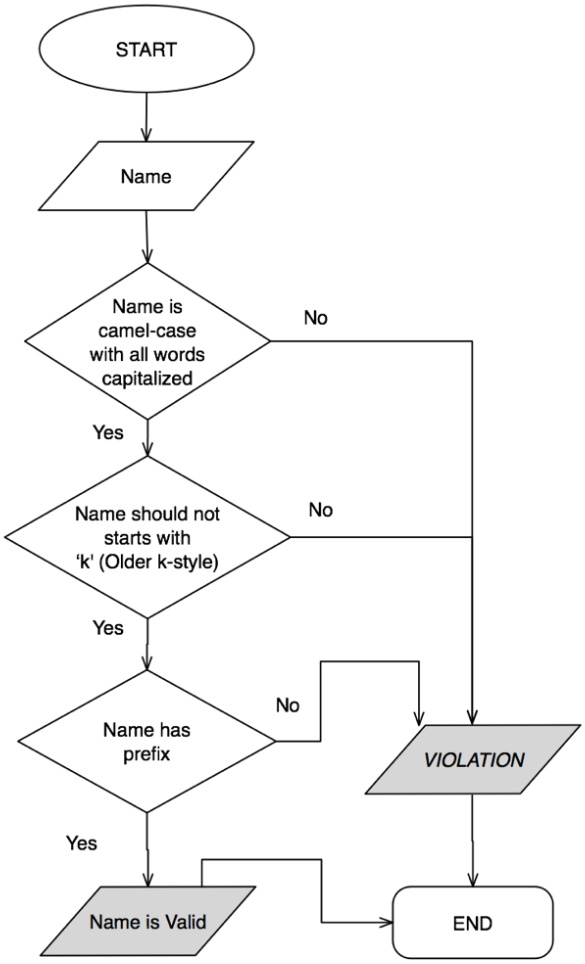
รายการข้อปฏิบัติการตั้งชื่อค่าคงที่

ข้อปฏิบัติ	ชื่อควรอยู่ในรูปแบบ Camel-Case โดยตัวอักษรแรกเป็นตัวพิมพ์ใหญ่	**รหัส : CCO01**
วิธีการตรวจสอบ	ตรวจสอบอักขระตัวพิมพ์เล็ก - ตัวพิมพ์ใหญ่	
ผ่าน	ตัวอักษรแรกของทุกคำเป็นตัวพิมพ์ใหญ่และตัวอักษรที่เหลือเป็นตัวพิมพ์เล็ก	
ไม่ผ่าน	ตัวอักษรแรกของทุกคำไม่เป็นตัวพิมพ์ใหญ่หรือตัวอักษรที่เหลือไม่เป็นตัวพิมพ์เล็ก	

ข้อปฏิบัติ	ชื่อไม่ควรขึ้นต้นด้วย ‘k’ (รูปแบบ Older k-style)	**รหัส : CCO02**
วิธีการตรวจสอบ	ตรวจสอบอักขระตัวแรก	
ผ่าน	อักขระตัวแรกของชื่อไม่ใช่ตัว k	
ไม่ผ่าน	อักขระตัวแรกของชื่อเป็นตัว k	

ข้อปฏิบัติ	ชื่อควรมีคำเติมหน้า	**รหัส : CCO03**
วิธีการตรวจสอบ	ตรวจสอบคำเติมหน้า	
ผ่าน	ชื่อมีคำเติมหน้า	
ไม่ผ่าน	ชื่อไม่มีคำเติมหน้า	

ขั้นตอนการตรวจสอบชื่อค่าคงที่ ดังรูปที่ 3.8



รูปที่ 3.8 แสดงขั้นตอนการตรวจสอบการตั้งชื่อค่าคงที่

บทที่ 4
การวิเคราะห์ระบบ

การพัฒนาเครื่องมือในการตรวจสอบโปรแกรมอ็อบเจกทีฟซี ผู้วิจัยได้วิเคราะห์ความสามารถของเครื่องมือเพื่อให้เป็นไปตามวัตถุประสงค์ที่ได้กำหนดไว้ โดยมีการศึกษาเพิ่มเติมเกี่ยวกับการตัดคำ การตรวจสอบความหมายและชนิดของคำ หลังจากนั้นจึงกำหนดความต้องการของระบบ

4.1 การศึกษาการตัดคำ

การตรวจสอบการตั้งชื่อตามข้อปฏิบัติที่โครงงานได้รวบรวมและกำหนดขึ้น ข้อมูลก่อนการตรวจสอบจะต้องอยู่ในรูปแบบของกลุ่มคำ เนื่องจากการข้อปฏิบัติส่วนใหญ่จะตรวจสอบแยกคำ ผู้วิจัยจึงศึกษาค้นคว้าซอฟต์แวร์ที่ช่วยในการตัดคำโดยจะต้องสามารถนำเข้าข้อมูลข้อความที่เกิดจากคำหลายคำเชื่อมต่อกันและจะต้องสามารถนำมาพัฒนาเครื่องมือร่วมกับโปรแกรม Xcode ได้ด้วย

4.2 การตรวจสอบความหมายและชนิดของคำ

รายการข้อปฏิบัติที่โครงงานได้รวบรวมและกำหนดขึ้น มีข้อปฏิบัติที่เกี่ยวข้องกับการตรวจสอบความหมายและชนิดของคำ ผู้วิจัยจึงต้องค้นหาวิธีในการตรวจสอบคำ โดยคัดเลือกการให้บริการในรูปแบบของ API เนื่องจากนำมาพัฒนาเครื่องมือร่วมกับโปรแกรม Xcode ได้ง่ายและได้ข้อมูลที่ทันสมัย

4.3 การวิเคราะห์ความต้องการของระบบ

จากการวิเคราะห์ความต้องการของเครื่องมือ ได้รายการความต้องการเชิงหน้าที่ ตามระบบการทำงานหลัก 3 ส่วนของโปรแกรม ดังนี้

4.3.1 จัดเก็บข้อมูล

เครื่องมือจะต้องจัดเก็บข้อมูลชื่อคลาส, ชื่อเมท็อด, ชื่อฟังก์ชัน, ชื่อตัวแปร, ชื่อค่าคงที่, Magic Number และ Literal String ในโปรแกรมที่ผู้ใช้งานต้องการตรวจสอบได้ทุกต้องครบถ้วน

4.3.2 ตรวจสอบชื่อ

เครื่องมือจะต้องตรวจสอบรายการชื่อตามข้อปฏิบัติที่กำหนดไว้ โดยเริ่มจากการตรวจสอบเบื้องต้น จากนั้นจะต้องแยกตรวจสอบตามประเภท

4.3.3 แสดงผลการแจ้งเตือนจุดที่โปรแกรมละเมิดข้อปฏิบัติได้

เครื่องมือจะต้องแจ้งเตือนผู้ใช้งานว่าโปรแกรมมีการละเมิดข้อปฏิบัติใดบ้างในรูปแบบของการแจ้งเตือน ระบุคำหรือตัวเลขที่ละเมิดข้อปฏิบัติพร้อมคำอธิบาย

จากการวิเคราะห์ความต้องการของเครื่องมือ สามารถแบ่งความต้องการเชิงหน้าที่ (Functional Requirement: FR) ตามส่วนการทำงานหลักของเครื่องมือและการตรวจสอบข้อปฏิบัติ ดังตารางที่ 4.1

ตารางที่ 4.1 ความต้องการเชิงหน้าที่

รหัสความต้องการ	**คำอธิบาย**
FR01	เครื่องมือสามารถเก็บรวบรวมตัวเลข Magic Number ได้
FR02	เครื่องมือสามารถเก็บรวบรวมข้อมูล Literal String ได้
FR03	เครื่องมือสามารถเก็บรวบรวมข้อมูลชื่อได้
FR04	เครื่องมือสามารถจำแนกชื่อออกเป็นกลุ่มคำได้
FR05	เครื่องมือสามารถตรวจสอบคำที่เป็นส่วนประกอบของชื่อว่ามีความหมายหรือไม่ได้
FR06	เครื่องมือสามารถตรวจสอบจำนวนคำที่เป็นส่วนประกอบของชื่อได้
FR07	เครื่องมือสามารถตรวจสอบว่าชื่อประกอบด้วยชื่อของคลาสแม่หรือไม่ได้
FR08	เครื่องมือสามารถตรวจสอบชนิดของคำที่เป็นส่วนประกอบของชื่อได้
FR09	เครื่องมือสามารถตรวจสอบว่าชื่อประกอบด้วยคำเติมหน้าหรือไม่ได้
FR10	เครื่องมือสามารถตรวจสอบว่าชื่ออยู่ในรูปแบบ Camel-Case หรือไม่ได้
FR11	เครื่องมือสามารถตรวจสอบว่าชื่อต่อท้ายด้วยชนิดหรือไม่ได้
FR12	เครื่องมือสามารถตรวจสอบคำที่เป็นส่วนประกอบของชื่อเป็นคำชนิดใดได้
FR13	เครื่องมือสามารถตรวจสอบอักขระที่ควรมีหรือไม่ควรมีในชื่อได้
FR14	เครื่องมือสามารถตรวจสอบคำที่ควรมีหรือไม่ควรมีในชื่อได้
FR15	เครื่องมือสามารถตรวจสอบคำตามตำแหน่งที่ถูกต้องได้

ตารางที่ 4.1 ความต้องการเชิงหน้าที่ (ต่อ)

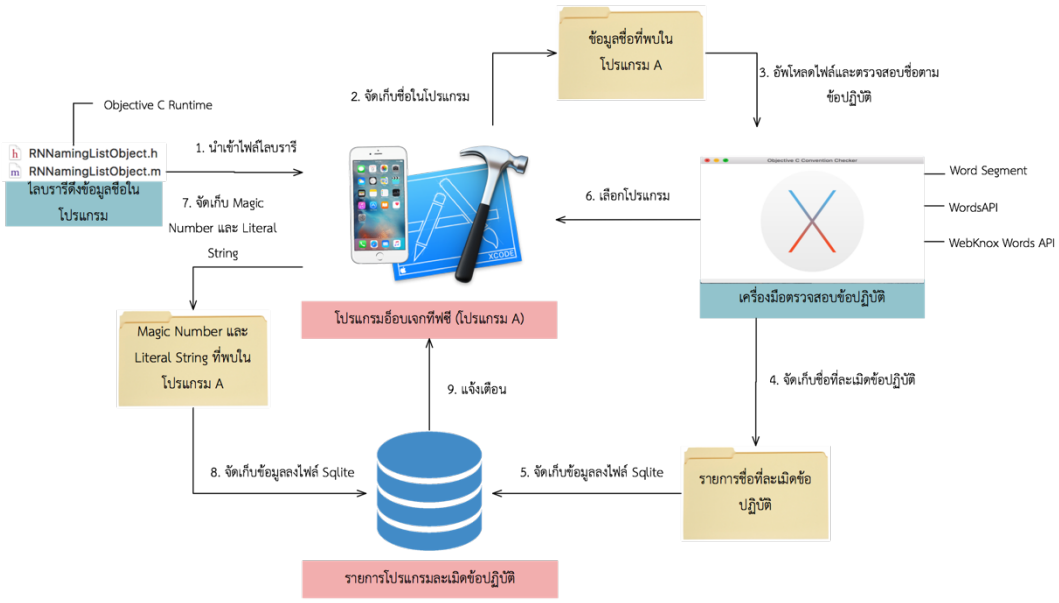
รหัสความต้องการ	คำอธิบาย
FR16	เครื่องมือสามารถตรวจสอบอักขระว่าเป็นตัวอักษรพิมพ์เล็กหรือพิมพ์ใหญ่ได้
FR17	เครื่องมือสามารถตรวจสอบว่าชื่อเมท็อดและฟังก์ชันประกอบด้วยคำอธิบายอาร์กิวเมนต์หรือไม่ได้
FR18	เครื่องมือสามารถตรวจสอบว่าชื่อเมท็อดและฟังก์ชันมีคำว่า 'and' ระหว่างอาร์กิวเมนต์หรือไม่ได้
FR19	เครื่องมือสามารถตรวจสอบว่าชื่อเมท็อดและฟังก์ชันมีคำว่า 'and' ระหว่าง action หรือไม่ได้
FR20	เครื่องมือสามารถแสดงผลรายการที่ละเมิดข้อปฏิบัติในโปรแกรมที่ตรวจสอบได้ครบถ้วน

บทที่ 5
การออกแบบระบบ

ในบทนี้จะกล่าวถึงการออกแบบระบบซึ่งประกอบด้วย ภาพรวมของเครื่องมือ การออกแบบหน้าที่การทำงานของเครื่องมือและการออกแบบส่วนต่อประสานของผู้ใช้งานเครื่องมือ

5.1 ภาพรวมของเครื่องมือ

เครื่องมือในการตรวจสอบข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซี พัฒนาขึ้นเพื่อตรวจสอบโปรแกรมว่าเป็นไปตามข้อปฏิบัติที่กำหนดหรือไม่ การทำงานของเครื่องมือ เริ่มต้นจากการจัดเก็บข้อมูลในโปรแกรมที่ต้องการตรวจสอบ ได้แก่ ข้อมูลชื่อ (ชื่อคลาส ชื่อเมท็อด ชื่อฟังก์ชัน ชื่อตัวแปรและชื่อค่าคงที่), ค่าตัวเลข Magic Number และ Literal String โดยการจัดเก็บข้อมูลชื่อจะทำโดยการเรียกใช้ฟังก์ชันรันไทม์ เนื่องจากว่าฟังก์ชันรันไทม์สามารถเรียกใช้งานได้ภายในโปรแกรมของตนเองเท่านั้น ผู้วิจัยจึงต้องพัฒนาไลบรารีเพื่อให้ผู้ใช้งานนำเข้าสู่โปรแกรมที่ต้องการตรวจสอบก่อนเพื่อเก็บรวบรวมข้อมูลชื่อ เมื่อทำการจัดเก็บข้อมูลแล้วจะเข้าสู่ขั้นตอนการตรวจสอบด้วยเครื่องมือตรวจสอบ ซึ่งพัฒนาขึ้นในรูปแบบแอปพลิเคชันบนระบบปฏิบัติการ OSX หลังการตรวจสอบจะได้ผลลัพธ์เป็นรายการโปรแกรมที่ละเมิดข้อปฏิบัติ การแสดงผลการละเมิดข้อปฏิบัติ ณ จุดที่พบในโปรแกรม สามารถทำได้โดยการใช้คำสั่ง Shell Script ดังนั้นเครื่องมือจึงทำการสร้างไฟล์ Shell Script จากฐานข้อมูลการละเมิดข้อปฏิบัติขึ้น เพื่อให้ผู้ใช้นำไฟล์ดังกล่าวเข้าสู่โปรแกรมที่ตรวจสอบ เมื่อรันโปรแกรม ไฟล์ Shell Script จะสั่งให้โปรแกรมแสดงผลการละเมิด ณ จุดที่พบในรูปแบบของการแจ้งเตือน (Warning) โดยภาพรวมของระบบเป็นดังรูปที่ 5.1



รูปที่ 5.1 ภาพรวมของระบบ

ขั้นตอนการทำงานของเครื่องมือจะประกอบไปด้วย 9 ขั้นตอน รายละเอียด ดังนี้

1. การนำเข้าไฟล์ไลบรารี ผู้ใช้งานจะต้องนำเข้าไฟล์ไลบรารี (RNNamingListObject) เข้าสู่โปรแกรมที่ต้องการตรวจสอบ จากนั้นเรียกใช้งานไลบรารีโดยใช้คำสั่ง

[RNNamingListObject startGetListName]

การเรียกใช้งานจะต้องเรียกคำสั่งข้างต้นในไฟล์ Appdelegate.m ซึ่งเป็นไฟล์ควบคุมการทำงานหลักของโปรแกรมและสั่งประมวลผล โปรแกรมไลบรารีจะทำดึงข้อมูลเพื่อจัดเก็บในขั้นตอนถัดไป

2. จัดเก็บชื่อในโปรแกรม โดยจะแบ่งการจัดเก็บตามประเภทของชื่อได้แก่ ชื่อคลาส, ชื่อฟังก์ชัน, ชื่อเมท็อด, ชื่อตัวแปรและชื่อค่าคงที่ ข้อมูลรายการชื่อจะถูกจัดเก็บในรูปของไฟล์ข้อความ โดยข้อมูลแต่ละรายการจะอยู่ในรูปแบบของ {ชื่อ-ชนิด} โดยผู้ใช้จะต้องทำการนำไฟล์ดังกล่าวไปใช้ในขั้นตอนถัดไป

3. อัปโหลดไฟล์และตรวจสอบชื่อตามข้อปฏิบัติ เครื่องมือจะมีส่วนต่อประสานให้ผู้ใช้งานเลือกไฟล์รายการชื่อจากขั้นตอนที่ 2 และอัปโหลดเข้าสู่เครื่องมือเพื่อทำการตรวจสอบหาชื่อที่ละเมิดข้อปฏิบัติ

4. จัดเก็บชื่อที่ละเมิดข้อปฏิบัติ โดยแต่ละรายการจะประกอบด้วยข้อมูลชื่อ, ชื่อคลาสที่พบ, ตำแหน่งบรรทัดที่พบ, รหัสข้อปฏิบัติที่ละเมิด

5. จัดเก็บข้อมูลลงไฟล์ Sqlite ซึ่งเป็นฐานข้อมูลของเครื่องมือ

6. เลือกโปรแกรมที่ตรวจสอบ จากเมนูในส่วนต่อประสานของเครื่องมือ และเลือกไปที่โฟลเดอร์ของโปรแกรมที่ต้องการตรวจสอบ

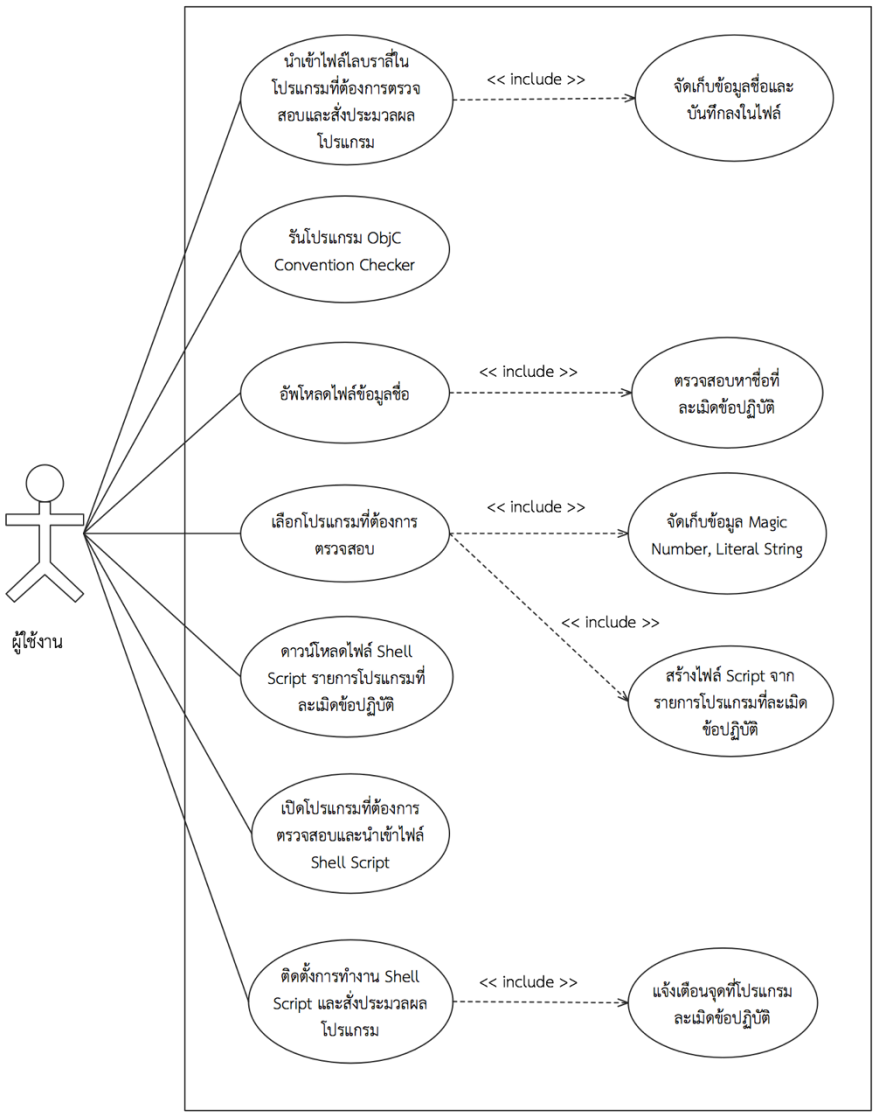
7. จัดเก็บ Magic Number และ Literal String โดยเครื่องมือจะทำการอ่านไฟล์ในโฟลเดอร์และทำการตรวจสอบหา Magic Number และ Literal String ในไฟล์คลาส (ไฟล์ Implement สกุล .m)

8. จัดเก็บข้อมูลลงไฟล์ Sqlite ซึ่งเป็นฐานข้อมูลของเครื่องมือ

9. แจ้งเตือนโปรแกรมที่ตรวจสอบ โดยให้ผู้ใช้งานดาวน์โหลดไฟล์ Shell Script ที่ถูกสร้างขึ้นจากรายการละเมิดข้อปฏิบัติในฐานข้อมูล Sqlite นำไฟล์ Shell Script เข้าสู่โปรแกรมแล้วสั่งประมวลผล Shell Script จะแสดงตำแหน่งที่ละเมิดข้อปฏิบัติ พร้อมคำอธิบาย

5.2 การออกแบบหน้าที่การทำงานของระบบ

จากภาพรวมและความต้องการของเครื่องมือตรวจสอบข้อปฏิบัติ สามารถอธิบายภาพรวมการทำงานของเครื่องมือด้วยแผนภาพยูสเคส ดังรูปที่ 5.2

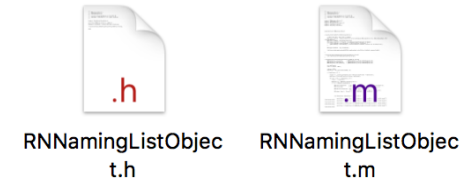


รูปที่ 5.2 แผนภาพยูสเคสการทำงานของเครื่องมือ

5.3 การออกแบบส่วนต่อประสานของผู้ใช้งานระบบ

การออกแบบส่วนต่อประสานของเครื่องมือจะแบ่งเป็น 2 ส่วน ได้แก่ ไลบรารีจัดเก็บข้อมูลชื่อในโปรแกรมและเครื่องมือตรวจสอบข้อปฏิบัติ

1) ไลบรารีจัดเก็บข้อมูลชื่อในโปรแกรม ดังรูปที่ 5.3



รูปที่ 5.3 ไฟล์ไลบรารีจัดเก็บข้อมูลชื่อในโปรแกรม

- ภายในไลบรารีจัดเก็บข้อมูล จะประกอบไปด้วยฟังก์ชันรันไทม์ในการดึงข้อมูลรายการชื่อคลาส, ชื่อเมท็อด, ชื่อฟังก์ชัน, ชื่อตัวแปรและชื่อค่าคงที่ ที่พบในโปรแกรม โดยผู้ใช้จะต้องนำเข้าไลบรารีและเขียนโปรแกรมคำสั่งในการเรียกใช้งานไลบรารี จากนั้นจึงทำการรันโปรแกรม ไลบรารีจะแสดงข้อความแจ้งเตือนผลลัพธ์การเก็บข้อมูลและสร้างไฟล์รายการชื่อในรูปแบบไฟล์ข้อความเพื่อให้ผู้ใช้ดาวน์โหลดและนำไฟล์ไปตรวจสอบในเครื่องมือตรวจสอบ ตัวอย่างไฟล์ข้อความรายการชื่อที่จัดเก็บ เป็นดังรูปที่ 5.4

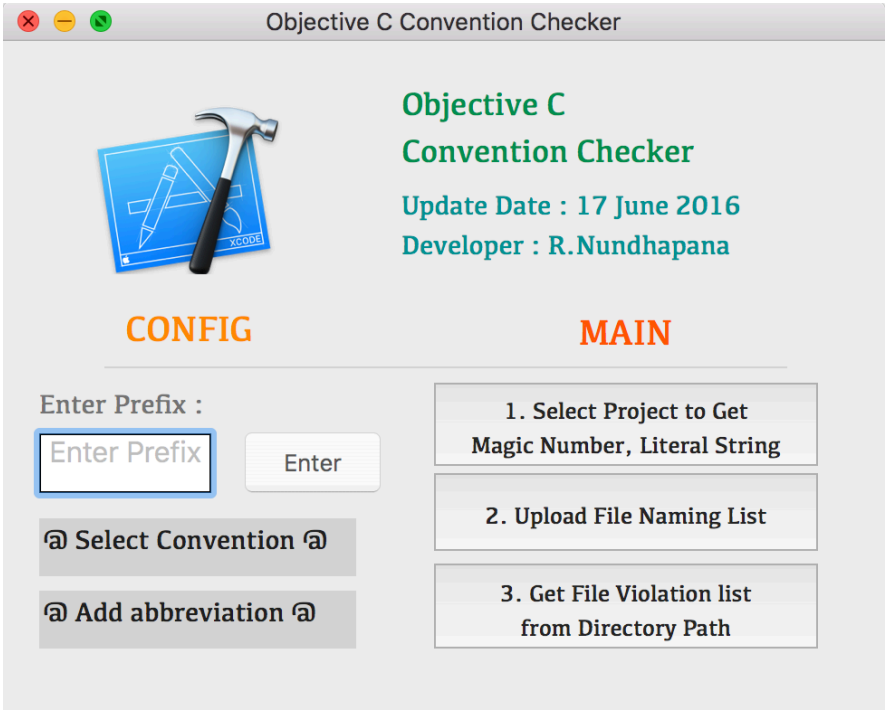
NameListFile.txt

```
ProjectCell-tableviewcell,EachRoomVC-viewcontroller,APIService-object,UserInfoVC-
viewcontroller,EachProjectVC-viewcontroller,AppDelegate-responder,BillVC-
viewcontroller,MainViewVC-viewcontroller,ForgotPwVC-viewcontroller,LoginVC-
viewcontroller,AutoPurgeCache-cache,RoomCVCell-collectionviewcell,ProjectInfoFooter-
collectionreusableview,_AFURLSessionTaskSwizzling-object,ViewController-
viewcontroller,ProjectInfoHeader-collectionreusableview###setTitleLB:-v,setDescLB:-
v,setThumbImgV:-v,.cxx_destruct-v,setSelected:animated:-v,awakeFromNib-
v,gotoMWImageBrowser-v,setRoomDict:-v,numberOfPhotosInPhotoBrowser:-Q,setContent-
v,seeRoomAlbum:-v,seeRoomPlan:-v,seeDemo:-v,setMainView:-v,setHeaderImg:-v,setTitleLB:-
v,setThumbImg:-v,setDescLB:-v,setRoomPlanImg:-v,setDemoImg:-v,.cxx_destruct-
v,supportedInterfaceOrientations-Q,didReceiveMemoryWarning-v,setScrollView:-
v,viewDidLoad-v,getBarCodeByCSRunno:completeBlock:-
v,loginByUsername:Password:completeBlock:-v,forgotPasswordByEmail:completeBlock:-
v,getCustomerInfo:completeBlock:-v,setDelegate:-v,.cxx_destruct-v,getUserInfo-
v,setNameLB:-v,setIdCardLB:-v,setEmailLB:-v,.cxx_destruct-
v,supportedInterfaceOrientations-Q,didReceiveMemoryWarning-v,viewDidLoad-v,close:-
v,gotoMWImageBrowser-v,touchPlanImgV-v,touchMapImgV-v,numberOfPhotosInPhotoBrowser:-
Q,setImageV:-v,setRoomCV:-v,setProjectDict:-v,.cxx_destruct-
v,collectionView:numberOfItemsInSection:-q,collectionView:didSelectItemAtIndexPath:-
v,supportedInterfaceOrientations-Q,didReceiveMemoryWarning-
v,collectionView:layout:sizeForItemAtIndexPath:-{CGSize=dd},viewDidLoad-
```

รูปที่ 5.4 ตัวอย่างไฟล์ข้อความรายการชื่อที่จัดเก็บ

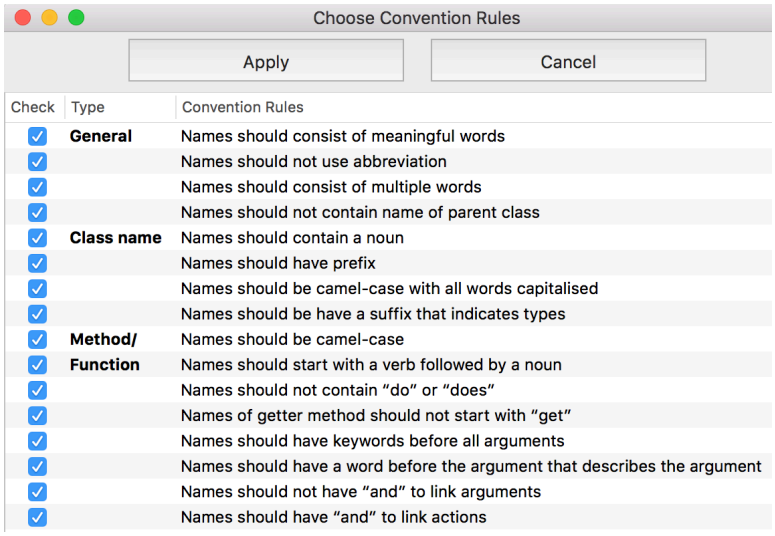
2) เครื่องมือตรวจสอบข้อปฏิบัติ

เครื่องมือตรวจสอบข้อปฏิบัติจะมีส่วนต่อประสานหน้าหลักประกอบไปด้วยการทำงาน 2 ส่วนได้แก่ การตั้งค่าเครื่องมือ (Config) โดยผู้ใช้จะต้องกำหนดคำเติมหน้า (Prefix) ที่ใช้สำหรับโปรแกรมที่ตรวจสอบ และส่วนการทำงานหลัก (Main) 3 เมนู ส่วนต่อประสานของเครื่องมือเป็นดังรูปที่ 5.5

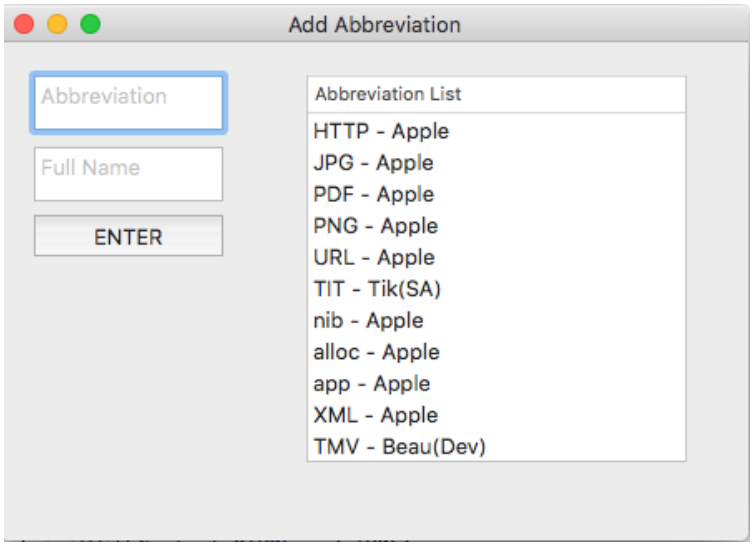


รูปที่ 5.5 หน้าจอหลักของเครื่องมือ

การตั้งค่าเครื่องมือผู้ใช้สามารถกำหนดและปรับแต่งการตรวจสอบได้โดยเลือกข้อปฏิบัติที่ไม่ต้องการให้ตรวจสอบดังรูปที่ 5.6 รวมถึงสามารถกำหนดคำย่อหรืออักษรย่อที่อนุญาตให้ใช้ได้ดังรูปที่ 5.7 หากโปรแกรมตรวจสอบเจอคำในรายการดังกล่าว จะถูกละเว้นการตรวจสอบ

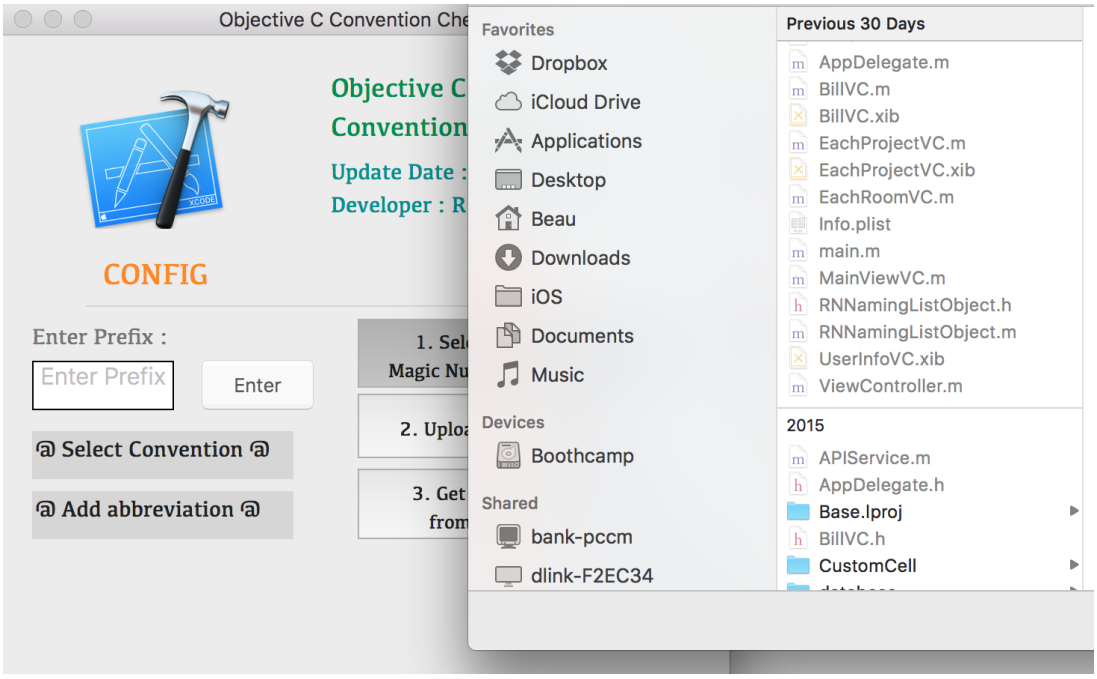


ภาพที่ 5.6 หน้าจอการเลือกข้อปฏิบัติที่จะตรวจสอบ



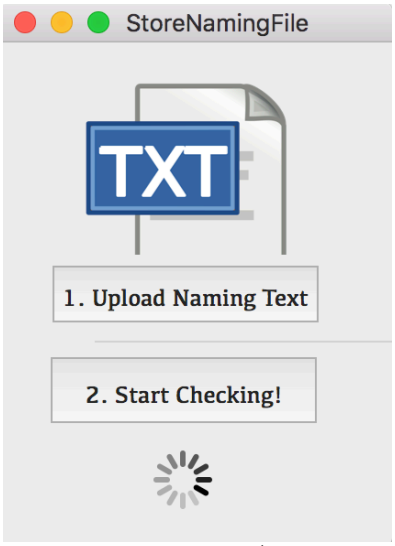
รูปที่ 5.7 หน้าจอการป้อนข้อมูลคำย่อ

ส่วนการทำงานหลักของโปรแกรม เริ่มต้นโดยผู้ใช้เลือกเมนูแรกเพื่อทำการจัดเก็บ Magic Number และ Literal String โดยเลือกโฟลเดอร์ของโปรแกรมที่ต้องการตรวจสอบ ดังตัวอย่างรูปที่ 5.8



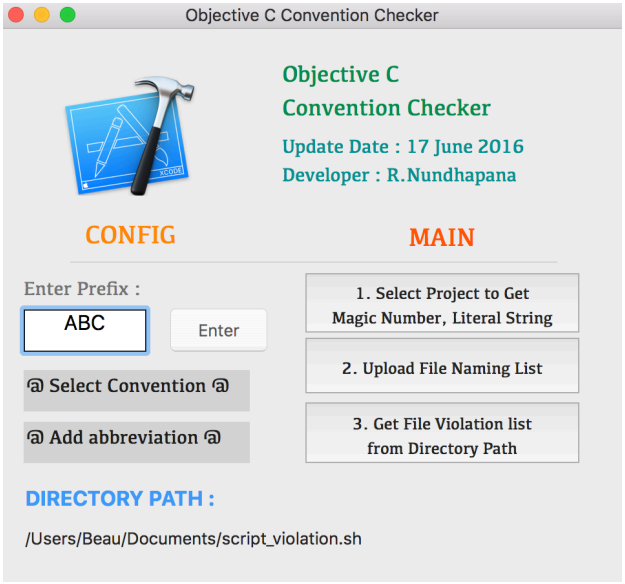
รูปที่ 5.8 หน้าจอตัวอย่างการเลือกโฟลเดอร์โปรแกรมที่ตรวจสอบ

ผู้ใช้เลือกเมนูที่สองเพื่อนำเข้าไฟล์ข้อความรายการชื่อในโปรแกรมที่จัดเก็บได้จากไลบรารี ดังรูปที่ 5.9 จากนั้นกดปุ่มเริ่มต้นการตรวจสอบ เครื่องมือจะทำการตรวจสอบชื่อตามข้อปฏิบัติเพื่อหาส่วนของโปรแกรมที่ละเมิดข้อปฏิบัติและจัดเก็บลงฐานข้อมูล



รูปที่ 5.9 หน้าจอการนำเข้าไฟล์รายการชื่อและการตรวจสอบ

หลังจากจัดเก็บ Magic Number, Literal String และรายชื่อที่ละเมิดข้อปฏิบัติแล้ว เครื่องมือจะสร้างไฟล์ Shell Script เพื่อใช้แจ้งเตือนจุดที่โปรแกรมละเมิดข้อปฏิบัติ จากนั้นกดเมนูที่สาม เครื่องมือจะแสดงตำแหน่งของไฟล์ Shell Script ดังรูปที่ 5.10

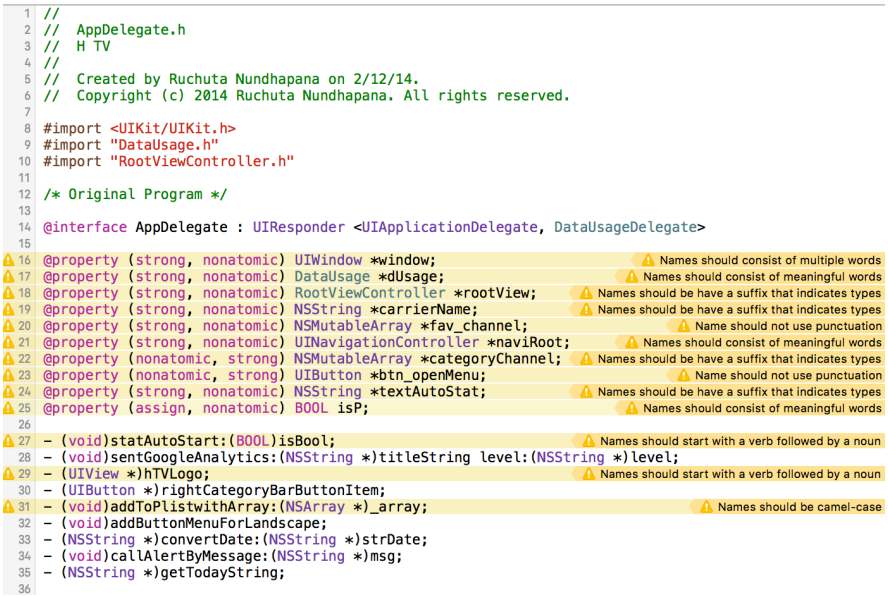


รูปที่ 5.10 หน้าจอแสดงตำแหน่งไฟล์ Shell Script

ผู้ใช้เปิดโปรแกรมที่ต้องการตรวจสอบ และนำไฟล์ Shell Script ที่ได้จากขั้นตอนก่อนหน้าเข้าสู่โปรแกรม จากนั้นทำการติดตั้งการเรียกใช้ไฟล์ Shell Script ในส่วนการทำงานของ Run Script เพื่อรอการเรียกใช้งานเมื่อโปรแกรมประมวลผล ดังรูปที่ 5.11

รูปที่ 5.11 การติดตั้งไฟล์ Shell Script ส่วนการทำงาน Run script

ผู้ใช้ทำการประมวลผลโปรแกรม ไฟล์ Shell Script จะทำการแสดงผลจุดที่โปรแกรมละเมิดข้อปฏิบัติเพื่อให้นักพัฒนาทำการแก้ไข ดังรูปที่ 5.12 และตัวอย่างหลังโปรแกรมถูกปรับแก้ตามคำอธิบายแล้ว รูปที่ 5.13



รูปที่ 5.12 การแจ้งเตือนโปรแกรมที่ละเมิดข้อปฏิบัติ

```
//
//  AppDelegate.h
//  H TV
//
//  Created by Ruchuta Nundhapana on 2/12/14.
//  Copyright (c) 2014 Ruchuta Nundhapana. All rights reserved.

#import <UIKit/UIKit.h>
#import "DataUsage.h"
#import "RootViewController.h"

@interface AppDelegate : UIResponder <UIApplicationDelegate, DataUsageDelegate>

@property (strong, nonatomic) UIWindow *mainWindow;
@property (strong, nonatomic) DataUsage *dataUsage;
@property (strong, nonatomic) RootViewController *rootViewController;
@property (strong, nonatomic) NSString *carrierNameString;
@property (strong, nonatomic) NSMutableArray *favoriteChannelArray;
@property (strong, nonatomic) UINavigationController *rootNvaigationController;
@property (nonatomic, strong) NSMutableArray *categoryChannelArray;
@property (nonatomic, strong) UIButton *openMenuButton;
@property (strong, nonatomic) NSString *textAutoStatString;
@property (assign, nonatomic) BOOL isPortrait;

- (void)startAutoStat:(BOOL)isBool;
- (void)sentGoogleAnalytics:(NSString *)titleString level:(NSString *)level;
- (UIView *)createHtvLogoView;
- (UIButton *)rightCategoryBarButtonItem;
- (void)addToPlistWithArray:(NSArray *)_array;
- (void)addButtonMenuForLandscape;
- (NSString *)convertDate:(NSString *)strDate;
- (void)callAlertByMessage:(NSString *)msg;
- (NSString *)getTodayString;

@end

```

รูปที่ 5.13 โปรแกรมหลังการแก้ไข

บทที่ 6
การพัฒนาระบบ

การพัฒนาระบบตามเอกสารการวิเคราะห์และออกแบบระบบนั้นอาศัยเครื่องมือทั้งฮาร์ดแวร์และซอฟต์แวร์ที่เหมาะสม รวมถึงเทคนิคและการพัฒนาชุดคำสั่ง เพื่อให้เกิดประโยชน์สูงสุดในการพัฒนาระบบที่สมบูรณ์และมีประสิทธิภาพในการใช้งาน

6.1 เครื่องมือที่ใช้ในการพัฒนาโปรแกรม

6.1.1 ฮาร์ดแวร์ที่ใช้ในการพัฒนาโปรแกรม

เครื่องคอมพิวเตอร์ที่ใช้พัฒนาระบบ

1) Macbook Pro (ปลายปี 2013)

2) หน่วยประมวลผล อินเทล คอร์ไอ 7 ความเร็ว 2.3 กิกะเฮิร์ท

3) หน่วยความจำ ดีดีอาร์ 3 ที่ 1,600 ขนาด 16 กิกะไบต์

4) หน้าจอเรตินา ขนาด 15 นิ้ว

6.1.2 ซอฟต์แวร์ที่ใช้ในการพัฒนาโปรแกรม

1) ระบบปฏิบัติการ OS X EL Capitan

2) เครื่องมือที่ใช้ในการออกแบบและจัดทำเอกสารของกระบวนการ

- ไมโครซอฟท์ออฟฟิซ รุ่น 2013
- เอ็กซ์ไดอะแกรม เวอชัน 3.0

3) เครื่องมือที่ใช้ในการพัฒนาเครื่องมือ

- Xcode เวอชัน 7.3.1

6.2 ขั้นตอนการพัฒนาระบบ

การพัฒนาเครื่องมือตรวจสอบข้อปฏิบัติการเขียนโปรแกรม ประกอบไปด้วย 4 ขั้นตอนหลัก ได้แก่ การจัดเก็บข้อมูลชื่อ, การจัดเก็บข้อมูล Magic Number และ Literal String, การตรวจสอบชื่อที่ละเมิดข้อปฏิบัติ และการสร้างไฟล์ Shell Script จากรายการโปรแกรมที่ละเมิดข้อปฏิบัติ โดยรายละเอียดในแต่ละขั้นตอนเป็นดังนี้

6.2.1 การจัดเก็บข้อมูลชื่อ

การจัดเก็บข้อมูลชื่อ ใช้ความสามารถของฟังก์ชันในอ็อบเจกทีฟซีรันไทม์ ในการดึงข้อมูลชื่อคลาส, ชื่อเมท็อด, ชื่อฟังก์ชัน, ชื่อตัวแปรและชื่อค่าคงที่

6.2.2 การจัดเก็บข้อมูล Magic Number และ Literal String

การจัดเก็บข้อมูล Magic Number และ Literal String เครื่องมือจะทำการตรวจสอบไฟล์ของคลาสในโปรแกรม และทำการประมวลผลชุดคำสั่งในโปรแกรมทีละบรรทัด และค้นหาข้อมูลด้วย Regular Expression ขั้นตอนการค้นหา ดังนี้

- ขั้นตอนการค้นหา Magic Number

1 จัดเก็บคำสั่งในโปรแกรมทั้งไฟล์ในรูปของข้อมูลสตริง

2 แปลงข้อมูลสตริงเป็นอาร์เรย์ของชุดคำสั่งแต่ละบรรทัด

3 ขจัดบรรทัดที่เป็นคอมเมนต์

4 โปรแกรมทำการเริ่มอ่านชุดคำสั่งทีละบรรทัด

4.1 ขจัดอักขระช่องว่าง, อักขระพิเศษ และ Literal String

4.2 ตรวจสอบว่าชุดคำสั่งประกอบด้วยอักขระที่เป็นตัวเลขหรือไม่

4.3 ตรวจสอบว่าพบ Magic Number ตาม Regular Expression ดัง ตารางที่ 6.1

4.4 ตรวจสอบ Magic Number ที่พบ หากเป็นตัวเลข -1,0,1,2 ให้ละเว้น

4.5 ทำการจัดเก็บข้อมูล Magic Number ลงในฐานข้อมูล

ตารางที่ 6.1 รูปแบบ Regular Expression ของ Magic Number

ตำแหน่งที่พบ	**รูปแบบนิพจน์ปกติ**	**ตัวอย่าง**
ตัวเลขอยู่หลังเครื่องหมาย = หรือ :	[\:\=]([\d\.]+)	originX = 15.0; setWidth:30.0
ตัวเลขอยู่ก่อนหน้า ;	([\d\.]+)[;]	menuWidth =viewWidth/4.0;
ตัวเลขอยู่หลังเครื่องหมาย +, -, x, /, <, >	[\>\<\+\-*\/] ([\d\.]+)	salary = month*25*100;
ตัวเลขอยู่ก่อนเครื่องหมาย +, -, x, /, <, >	([\d\.]+)[\+\- *\/\>\<]	days = 7*week;
ตัวเลขอยู่หลัง (และอยู่ก่อน ,	[\(\[]([\d|\.]+)[,]	setFrame(20, originY, width, height)
ตัวเลขอยู่หลัง , และอยู่ก่อน)	[,]([\d|\.]+)[\)\]]	setFrame(originX, originX, width,568)
ตัวเลขอยู่ระหว่าง ,	[,]([\d|\.]+)[,]	setFrame(originX, origin, 185, height)

- ขั้นตอนการค้นหา Literal String

1 จัดเก็บคำสั่งในโปรแกรมทั้งไฟล์ในรูปของข้อมูลสตริง

2 แปลงข้อมูลสตริงเป็นอาร์เรย์ของชุดคำสั่งแต่ละบรรทัด

3 ขจัดบรรทัดที่เป็นคอมเมนท์

4 โปรแกรมทำการเริ่มอ่านชุดคำสั่งทีละบรรทัด

4.1 ขจัดอักขระช่องว่าง, อักขระพิเศษ และสตริงของค่าคงที่

4.2 ตรวจสอบว่าพบ Literal String ตาม Regular Expression ดังตารางที่ 6.2

4.3 ทำการจัดเก็บข้อมูล Literal String ลงในไฟล์ sqlite

รายการของ Magic Number และ Literal String พบตามขั้นตอนข้างต้น จะถูกจัดเก็บลงไฟล์รายการละเมิดข้อปฏิบัติ เพื่อนำไปสู่ขั้นตอนถัดไป

ตารางที่ 6.2 รูปแบบ Regular Expression ของ Literal String

ตำแหน่งที่พบ	**รูปแบบนิพจน์ปกติ**	**ตัวอย่าง**
ข้อมูลระหว่าง "	\"([^\\\"]|\\.)*\"	"Hello"

6.2.3 การตรวจสอบชื่อที่ละเมิดข้อปฏิบัติ

การตรวจสอบข้อปฏิบัติการตั้งชื่อเริ่มต้นจากการตรวจสอบพื้นฐาน จากนั้นจึงตรวจสอบตามประเภทของชื่อ วิธีที่ใช้ในการตรวจสอบจะแบ่งออกเป็น 5 กลุ่มตามลักษณะของข้อปฏิบัติ ดังนี้

1) การตรวจสอบจำนวนคำของชื่อ เพื่อตรวจสอบว่าชื่อมีจำนวนคำที่เป็นไปตามข้อปฏิบัติหรือไม่ ตัวอย่างเช่น การตรวจสอบว่าชื่อประกอบด้วยจำนวนคำที่มากกว่าหนึ่งคำหรือไม่ โค้ดเทียมของตัวอย่างการตรวจสอบนี้เป็นดังนี้

Convention : Name should consist of multiple words

Algorithm : Check number of word more than one

Input: Name (array of words)

Output: Result (yes or no)

1: Declare integer variable count

2: Read name

3: Set count to number of words

4: **If** count > 1

5: Print “name is valid” and return yes

6: **Else**

7: Print “name is violating” and return no

---

2) การตรวจสอบส่วนประกอบชื่อด้วยข้อมูลสตริงและอักขระ เพื่อดูว่าชื่อมีส่วนประกอบหรือละเว้นค่าข้อมูลที่ได้กำหนดไว้ในข้อปฏิบัติหรือไม่ ตัวอย่างเช่น การตรวจสอบการใช้เส้นใต้อักขระ (Underscore) เป็นคำเติมหน้าในชื่อตัวแปรอินสแตนซ์ (Instance Variable) จะนำอักขระแรกของชื่อมาเปรียบเทียบกับเส้นใต้อักขระ หากเงื่อนไขถูกต้องจะถือว่าชื่อผ่านข้อปฏิบัติ โค้ดเทียมของตัวอย่างการตรวจสอบนี้เป็นดังนี้

Convention : Instance variable should have underscore as a prefix

Algorithm : Check instance variable name underscore prefix

Input: Instance Variable name (array of words)

Output: Result (yes or no)

1: Declare a string variable char

2: Read instance variable name

3: Set char to first character of name

4: **If** char is **equal to** underscore

5: Print “name is valid” and return yes

6: **Else**

7: Print “name is violating” and return no

---

3) การตรวจสอบรูปแบบของชื่อด้วย Regular Expression ตัวอย่างเช่น การตรวจสอบชื่อคลาสว่ามีคำเติมหน้าหรือไม่ เริ่มจากสร้างรูปแบบสตริง Regular Expression จากคำเติมหน้าที่ได้กำหนดไว้ของโปรแกรม (เช่น คำเติมหน้าที่ระบุโดยองค์กร) จากนั้นนำชื่อคลาสมา

เปรียบเทียบกับรูปแบบนี้ หากเงื่อนไขถูกต้องจะถือว่าชื่อผ่านข้อปฏิบัติ โค้ดเทียมของตัวอย่างการตรวจสอบนี้เป็นดังนี้

Convention: Class name should have prefix

Algorithm: Check class name prefix

Input: Class name (array of words) and Prefix

Output: Result (yes or no)

1: Declare a string variable name

2: Declare a string variable regex pattern

3: Read class name

4: Read prefix

5: Set name to the class name

6: Set regex pattern includes prefix to

"(?<={prefix})[A-Z][a-z].*"

7: **If** name matches regex pattern

8: Print “name is valid” and return yes

9: **Else**

10: Print “name is violating” and return no

---

4) การตรวจสอบความหมายและชนิดของคำ ตัวอย่างเช่น การตรวจสอบชื่อเมท็อดว่าขึ้นต้นด้วยคำกริยาและตามด้วยคำนามหรือไม่ โดยนำคำแรกและคำที่สองมาตรวจสอบชนิดของคำผ่าน WordsAPI แล้วนำผลลัพธ์จาก API มาตรวจสอบว่าคำแรกเป็นคำกริยาและคำที่สองเป็นคำนามหรือไม่ หากเงื่อนไขถูกต้องจะถือว่าชื่อผ่านข้อปฏิบัติ โค้ดเทียมของตัวอย่างการตรวจสอบนี้เป็นดังนี้

Convention: Method name should start with a verb followed by a noun

Algorithm: Check method name verb and noun

Input: Method name (array of words)

Output: Result (yes or no)

```
1:      Declare a string variable word
2:      Read method name
3:      Set firstWord to the first word of the name
4:      Set secondWord to the second word of the name
5:      Add firstWord and secondWord to wordArray
6:      Set authentication to access WordsAPI by Key
7:      For each word in wordArray
8:          Invoke WordsAPI Url passing word as
            argument
9:          Get part of speech of word from response
            from WordsAPI
10:         If word == firstWord
11:             If part of speech of word is not equal
                    to verb
12:                 print “name is violating” and
                        return no
13:         Else
14:             If part of speech of word is equal
                    to noun
15:                 if word is the last word
```

16: print “name is valid” and

return yes

17: **Else**

18: print “name is violating” and return no

6.2.4 การสร้างไฟล์ Shell Script

หลังจากการรวบรวมรายการละเมิดข้อปฏิบัติและบันทึกลงในไฟล์ฐานข้อมูล Sqlite เครื่องมือจะทำการสร้างไฟล์ Shell Script จาก Sqlite เพื่อใช้ในการแจ้งเตือนโปรแกรมในจุดที่พบการละเมิดข้อปฏิบัติ โดยแต่ละรายการที่ละเมิดข้อปฏิบัติ จะถูกแปลงเป็นคำสั่ง Shell โดยกำหนดคีย์เวิร์ดหรือค่าข้อมูลที่ตรวจสอบ และคำอธิบายข้อปฏิบัติที่ค่าข้อมูลละเมิด รูปแบบเป็นดังนี้

```
KEYWORDS="@\"{ค่าข้อมูล}"

find "${SRCROOT}" \( -name "*.h" -or -name "*.m" \) -print0 | xargs -0 egrep --with-filename --line-number --only-matching "($KEYWORDS).*\$" | perl -p -e "s/($KEYWORDS3)/ warning: {คำอธิบายของข้อปฏิบัติที่ถูกละเมิด}/"
```

โปรแกรม Xcode จะทำการค้นหา Keywords ที่ระบุและแจ้งเตือน ณ จุดที่พบคำดังกล่าว พร้อมแสดงคำอธิบาย

บทที่ 7

การทดสอบระบบ

การทดสอบเครื่องมือตรวจข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซี แบ่งการทดสอบออกเป็น การทดสอบเครื่องมือในด้านความถูกต้องกับความสามารถในการอ่านโปรแกรม หลังจากที่โปรแกรมได้รับการปรับปรุงแก้ไขตามคำแนะนำของเครื่องมือ

7.1 การประเมินความถูกต้องของเครื่องมือ

การทดสอบระบบมีวัตถุประสงค์เพื่อตรวจสอบและค้นหาข้อผิดพลาดในการทำงานของระบบที่พัฒนา ซึ่งมีรายละเอียดดังต่อไปนี้

7.1.1 ประเภทของการทดสอบระบบ

ในการทดสอบเครื่องมือตรวจสอบข้อปฏิบัติการเขียนโปรแกรม แบ่งประเภทของการทดสอบเป็น 4 ประเภท ดังนี้

- **การทดสอบหน่วยย่อย (Unit Testing)**

การทดสอบหน่วยย่อย คือการทดสอบขณะที่ทำการพัฒนาระบบ ซึ่งผู้วิจัยเป็นผู้ทำการทดสอบด้วยตนเอง โดยจะพิจารณาจากส่วนการทำงานหลัก 3 ส่วน ได้แก่

1) การเก็บข้อมูลรายการที่จะตรวจสอบ โดยการทดสอบจะแบ่งตามประเภทการจัดเก็บ ดังนี้

- ทดสอบจัดเก็บข้อมูลชื่อด้วยฟังก์ชันของรันไทม์ โดยการเรียกใช้ฟังก์ชันรันไทม์กับโปรแกรมทดสอบ สำหรับข้อมูลชื่อทุกประเภท
- ทดสอบจัดเก็บข้อมูล Magic Number และ Literal String ด้วย Regular Expression โดยทำการทดสอบกับรูปแบบ Regular Expression ทีละแบบเพื่อตรวจสอบว่า Regular Expression ที่กำหนด สามารถทำงานได้ถูกต้องหรือไม่

2) การตรวจสอบรายการชื่อตามข้อปฏิบัติ โดยการทดสอบฟังก์ชันสำหรับตรวจสอบโปรแกรมตามข้อปฏิบัติแต่ละข้อ เพื่อดูว่าการตรวจสอบทุกข้อปฏิบัติ สามารถตรวจสอบและทำงานได้ถูกต้อง

3) การแจ้งเตือนโปรแกรม ณ จุดที่โปรแกรมละเมิดข้อปฏิบัติ โดยการทดสอบโปรแกรม Shell Script ที่สร้างขึ้นว่าสามารถแสดงตำแหน่งที่ต้องการในโปรแกรมได้ พร้อมแสดงรายละเอียด

- **การทดสอบแบบบูรณาการ (Integration Testing)**

การทดสอบแบบบูรณาการ คือ การทดสอบการทำงานส่วนการทำงานหลักในระบบทั้งสามส่วน เพื่อดูว่าเมื่อการทำงานหลักทั้งสามส่วนทำงานร่วมกัน เครื่องมือตรวจสอบสามารถทำงานได้ถูกต้องหรือไม่ โดยผู้วิจัยได้สร้างโปรแกรมขึ้นมาเพื่อตรวจสอบการทำงานของเครื่องมือ

โปรแกรมที่สร้างขึ้นจะมีการใส่ข้อมูลชื่อที่ไม่เป็นไปตามข้อปฏิบัติ, ข้อมูล Magic Number และ Literal String รวมทั้งหมด 36 ข้อปฏิบัติที่ได้กำหนดไว้ และทดสอบเครื่องมือกับโปรแกรมดังกล่าว เพื่อตรวจหาข้อผิดพลาดในการทำงาน

- **การทดสอบระบบ (System Testing)**

การทดสอบระบบ คือ การทดสอบที่ทำหลังจากที่ได้ทำการพัฒนาเครื่องมือเสร็จสิ้นครบทุกหน้าที่ของเครื่องมือแล้ว ซึ่งผู้วิจัยได้ทำการทดสอบเครื่องมือโดยใช้สภาพแวดล้อมของการทำงานจริง โดยทำการตรวจสอบข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซีกับโปรแกรมจริงในองค์กร ทั้งหมด 3 โปรแกรม โดยโปรแกรมที่ 1 (ขนาดเล็ก) มีจำนวนคลาส 5 คลาส โปรแกรมที่ 2 (ขนาดกลาง) มีจำนวนคลาส 12 คลาส และโปรแกรมที่ 3 (ขนาดใหญ่) มีจำนวนคลาส 19 คลาส โดยผู้วิจัยจะเตรียมผลลัพธ์ของโปรแกรมไว้ เพื่อตรวจสอบกับผลลัพธ์จากเครื่องมือ ดังนี้

1) รายการข้อมูลชื่อ, รายการ Magic Number, รายการ Literal String
2) รายการโปรแกรมที่ละเมิดข้อปฏิบัติ

จากนั้นตรวจสอบโปรแกรมทั้ง 3 โปรแกรมด้วยเครื่องมือตรวจสอบข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซีที่พัฒนาขึ้น เปรียบเทียบผลลัพธ์ที่ได้จากเครื่องมือและผลลัพธ์ที่ถูกต้อง

- **การทดสอบเพื่อยอมรับ (Acceptance Testing)**

การทดสอบเพื่อการยอมรับ คือการทดสอบที่กระทำโดยกลุ่มผู้ใช้งานระบบจริง ได้แก่ ทีมนักพัฒนาโปรแกรมอ็อบเจกทีฟซีขององค์กร โดยทำการทดสอบการใช้งานเครื่องมือกับโปรแกรมที่พัฒนาอยู่จริง รวมถึงฮาร์ดแวร์และซอฟต์แวร์ที่ใช้พัฒนาจริงด้วย เพื่อทำการทดสอบความถูกต้องในการทำงาน และ ประเมินการใช้งานของเครื่องมือ รวมทั้งเสนอข้อคิดเห็นเกี่ยวกับการทำงานของเครื่องมือเพื่อนำมาใช้ในการปรับปรุงเครื่องมือต่อไป

7.1.2 กรณีทดสอบ

ผู้วิจัยได้ทำการทดสอบเครื่องมือที่พัฒนาขึ้น โดยผู้วิจัยจะยกตัวอย่างเฉพาะกรณีทดสอบสำหรับการตรวจสอบโปรแกรมตามข้อปฏิบัติเท่านั้น โดยการสร้างกรณีทดสอบมีขั้นตอนดังต่อไปนี้

- การเตรียมข้อมูลสำหรับกรณีทดสอบ

ในการทดสอบการตรวจสอบโปรแกรมตามข้อปฏิบัตินั้น จำเป็นจะต้องมีการเตรียมข้อมูลที่ละเมิดข้อปฏิบัติสำหรับข้อปฏิบัติทั้งหมด 36 ข้อ เพื่อใช้ในการตรวจสอบว่าเครื่องมือสามารถตรวจจับโปรแกรมที่ละเมิดได้หรือไม่ โดยข้อมูลที่นำมาทำการทดสอบระบบ ดังนี้

1) ข้อมูล Magic Number ตามรูปแบบ Regular Expression ที่เป็นไปได้
2) ข้อมูล Literal String ตามรูปแบบ Regular Expression ที่เป็นไปได้
3) รายการข้อมูลชื่อที่ละเมิดข้อปฏิบัติ โดยจะต้องครอบคลุมข้อปฏิบัติการตั้งชื่อ

ทั้งหมด 34 ข้อ

ส่วนการทดสอบออกเป็น 2 ส่วนใหญ่ๆ ดังนี้

1) การทดสอบค้นหา Magic Number และ Literal String
2) การทดสอบการตรวจสอบรายการข้อมูลชื่อ
   - การทดสอบข้อปฏิบัติเบื้องต้น
   - การทดสอบข้อปฏิบัติสำหรับชื่อประเภทคลาส
   - การทดสอบข้อปฏิบัติสำหรับชื่อประเภทฟังก์ชัน
   - การทดสอบข้อปฏิบัติสำหรับชื่อประเภทเมท็อด
   - การทดสอบข้อปฏิบัติสำหรับชื่อประเภทตัวแปร
   - การทดสอบข้อปฏิบัติสำหรับชื่อประเภทค่าคงที่

- ตัวอย่างกรณีทดสอบ

ตัวอย่างกรณีทดสอบนี้ เป็นเพียงส่วนหนึ่งของกรณีทดสอบจริง ซึ่งผู้วิจัยสามารถสรุปผลการทดสอบและบันทึกผลการทดสอบตามกรณีทดสอบ ดังตารางที่ 7.1 และตัวอย่างข้อมูลแสดงผลการทดสอบ เป็นดังตารางที่ 7.2

ตารางที่ 7.1 ตารางกรณีทดสอบ

ชื่อกรณีทดสอบ	
ข้อมูลที่ทดสอบ	
รหัสข้อปฏิบัติ	
ชื่อคลาสที่พบ	
หมายเลขบรรทัดที่พบ	
สรุปผลการทดสอบ :	☐ ผ่าน ☐ ไม่ผ่าน หมายเหตุ_____

ตารางที่ 7.2 ตัวอย่างข้อมูลแสดงผลการทดสอบ

ชื่อกรณีทดสอบ	ตรวจสอบว่าชื่ออยู่ในรูปแบบ Camel-Case หรือไม่
ข้อมูลที่ทดสอบ	Loginbyusername
รหัสข้อปฏิบัติ	CC03
ชื่อคลาสที่พบ	LoginViewController
หมายเลขบรรทัดที่พบ	58
ผลการทดสอบ	ละเมิดข้อปฏิบัติ
สรุปผลการทดสอบ :	✓ ผ่าน ☐ ไม่ผ่าน หมายเหตุ_____

7.1.3 สรุปผลการทดสอบระบบ

การทดสอบโปรแกรมที่แสดงในโครงงานมหาบัณฑิตนี้ เป็นกรณีที่ใช้ทดสอบหน้าที่ของเครื่องมือตรวจสอบข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซี เพื่อตรวจสอบการทำงานและค้นหาข้อผิดพลาดของระบบโดยการทดสอบข้อปฏิบัติและผลการทดสอบจะแสดงในตารางที่ 7.3

ตารางที่ 7.3 การประเมินความถูกต้องของเครื่องมือ

รหัสกรณีทดสอบ	**ชื่อกรณีทดสอบ**	**ฟังก์ชันที่ทดสอบ**	**ผลการทดสอบ**		**หมายเหตุ**
			ผ่าน	**ไม่ผ่าน**	
การทดสอบการจัดการข้อมูล					
T0101	เก็บรวบรวมตัวเลข Magic Number	FR01	✓		
T0102	เก็บรวบรวมข้อมูล Literal String	FR02	✓		
T0103	เก็บรวบรวมข้อมูลชื่อคลาส, เมท็อด,ฟังก์ชัน,ตัวแปร,ค่าคงที่	FR03	✓		
T0104	จำแนกชื่อออกเป็นกลุ่มคำ	FR04	✓		
การทดสอบการตรวจสอบข้อปฏิบัติการเขียนโปรแกรม					
T0201	ตรวจสอบคำที่เป็นส่วนประกอบของชื่อว่ามีความหมาย	FR05	✓		
T0202	ตรวจสอบจำนวนคำที่เป็นส่วนประกอบของชื่อ	FR06	✓		
T0203	ตรวจสอบว่าชื่อประกอบด้วยชื่อของคลาสแม่หรือไม่	FR07	✓		

ตารางที่ 7.3 การประเมินความถูกต้องของเครื่องมือ (ต่อ)

รหัสกรณีทดสอบ	ชื่อกรณีทดสอบ	ฟังก์ชันที่ทดสอบ	ผลการทดสอบ		หมายเหตุ
			ผ่าน	ไม่ผ่าน	
T0204	ตรวจสอบชนิดของคำที่เป็นส่วนประกอบของชื่อ	FR08	✓		
T0205	ตรวจสอบว่าชื่อประกอบด้วยคำเติมหน้าหรือไม่	FR09	✓		
T0206	ตรวจสอบว่าชื่ออยู่ในรูปแบบ Camel-Case หรือไม่	FR10	✓		
T0207	ตรวจสอบว่าชื่อต่อท้ายด้วยชนิดหรือไม่	FR11	✓		
T0208	ตรวจสอบคำที่เป็นส่วนประกอบของชื่อเป็นคำชนิดใด	FR12	✓		
T0209	ตรวจสอบอักขระที่ควรมีหรือไม่ควรมีในชื่อ	FR13	✓		
T0210	ตรวจสอบคำที่ควรมีหรือไม่ควรมีในชื่อ	FR14	✓		
T0211	ตรวจสอบคำตามตำแหน่งที่ถูกต้อง	FR15	✓		
T0212	ตรวจสอบอักขระว่าเป็นตัวอักษรพิมพ์เล็กหรือพิมพ์ใหญ่	FR16	✓		
T0213	ตรวจสอบว่าชื่อเมท็อดและฟังก์ชันประกอบด้วยคำอธิบายอาร์กิวเมนต์หรือไม่	FR17	✓		
T0214	ตรวจสอบว่าชื่อเมท็อดและฟังก์ชันมีคำว่า 'and' ระหว่างอาร์กิวเมนต์หรือไม่	FR18	✓		
T0215	ตรวจสอบว่าชื่อเมท็อดและฟังก์ชันมีคำว่า 'and' ระหว่าง action หรือไม่	FR19	✓		

ตารางที่ 7.3 การประเมินความถูกต้องของเครื่องมือ (ต่อ)

รหัสกรณีทดสอบ	ชื่อกรณีทดสอบ	ฟังก์ชันที่ทดสอบ	ผลการทดสอบ		หมายเหตุ
			ผ่าน	ไม่ผ่าน	
การทดสอบการแจ้งเตือนโปรแกรมที่ละเมิดข้อปฏิบัติ					
T0301	เครื่องมือสามารถแสดงผลรายการที่ละเมิดข้อปฏิบัติในโปรแกรมที่ตรวจสอบได้ครบถ้วน	FR20	✓		

7.2 การประเมินผลการใช้งานเครื่องมือ

เพื่อแสดงให้เห็นว่าโปรแกรมที่เป็นไปตามข้อปฏิบัติการเขียนโปรแกรมนั้นสามารถช่วยเพิ่มประสิทธิภาพด้านการอ่านทำความเข้าใจได้ง่ายจริง ผู้วิจัยจึงทำการทดสอบสัมฤทธิผลจากการใช้เครื่องมือ โดยให้นักพัฒนาอ่านโปรแกรมดั้งเดิมและโปรแกรมที่ได้รับการแก้ไขตามข้อปฏิบัติหลังผ่านการแจ้งเตือนโดยเครื่องมือที่พัฒนาขึ้นหลังจากนั้นให้นักพัฒนาตอบคำถามที่เกี่ยวข้องกับการตั้งชื่อ การใช้ Magic Number และ Literal String ซึ่งต้องอาศัยการอ่านทำความเข้าใจโปรแกรม ผู้วิจัยจะนำเวลาที่นักพัฒนาใช้ตอบคำถามจนถูกต้องทุกข้อระหว่างโปรแกรมทั้งสองเวอร์ชันมาเปรียบเทียบกัน เพื่อทดสอบว่าระยะเวลาที่ใช้ในการทำแบบทดสอบโปรแกรมดั้งเดิมมากกว่าระยะเวลาที่ใช้ในการทำแบบทดสอบโปรแกรมที่เป็นไปตามข้อปฏิบัติที่ระดับนัยสำคัญ 0.05 โดยผู้วิจัยได้กำหนดตัวแปรและตั้งสมมติฐานสำหรับการทดสอบไว้ดังนี้

$$H0: \mu1 - \mu2 = 0$$

$$H1: \mu1 - \mu2 > 0$$

โดยที่ $\mu1$ คือ ระยะเวลาที่ใช้ในการทำแบบทดสอบถูกต้องทุกข้อของโปรแกรมดั้งเดิม

และ $\mu2$ คือ ระยะเวลาที่ใช้ในการทำแบบทดสอบถูกต้องทุกข้อของโปรแกรมที่เป็นไปตามข้อปฏิบัติ

สถิติที่ผู้วิจัยนำมาใช้ในการทดสอบสมมติฐานนั้นคือ Paired T-Test ซึ่งสามารถคำนวณได้จากสมการที่ (1)

$$t = \frac{\overline{d} - d_0}{s_d/\sqrt{n}} \quad (1)$$

ผู้วิจัยคัดเลือกนักพัฒนาโปรแกรมอ็อบเจกทีฟซีจำนวน 4 คนที่มีประสบการณ์ทำงานเฉลี่ย 3-4 ปี และคัดเลือกโปรแกรมที่ใช้ในการทดสอบทั้งหมด 4 โปรแกรม ดังการทดสอบรูปที่ 7.1

การทำแบบทดสอบโดยนักพัฒนาคู่ที่ 1	โปรแกรมชุดที่ 1		โปรแกรมชุดที่ 2	
	โปรแกรม P1 (9 Class, 3565 LOC)		โปรแกรม P2 (14 Class, 4684 LOC)	
	เวอร์ชันดั้งเดิม	เวอร์ชันตามข้อปฏิบัติ	เวอร์ชันดั้งเดิม	เวอร์ชันตามข้อปฏิบัติ
นักพัฒนา A ประสบการณ์ 4 ปี	ทำก่อน	ทำหลัง	ทำหลัง	ทำก่อน
นักพัฒนา B ประสบการณ์ 4 ปี	ทำหลัง	ทำก่อน	ทำก่อน	ทำหลัง
การทำแบบทดสอบโดยนักพัฒนาคู่ที่ 2	**โปรแกรม P3 (6 Class, 1060 LOC)**		**โปรแกรม P4 (9 Class, 2038 LOC)**	
	เวอร์ชันดั้งเดิม	เวอร์ชันตามข้อปฏิบัติ	เวอร์ชันดั้งเดิม	เวอร์ชันตามข้อปฏิบัติ
นักพัฒนา C ประสบการณ์ 3 ปี	ทำก่อน	ทำหลัง	ทำหลัง	ทำก่อน
นักพัฒนา D ประสบการณ์ 3 ปี	ทำหลัง	ทำก่อน	ทำก่อน	ทำหลัง

รูปที่ 7.1 ข้อมูลการทดสอบโปรแกรม

ในการทดสอบแต่ละครั้งจะทำการจับคู่นักพัฒนา เพื่อทดสอบโปรแกรม 2 ชุด ซึ่งนักพัฒนาไม่คุ้นเคยมาก่อน โปรแกรมแต่ละชุดจะประกอบไปด้วย 2 เวอร์ชันคือโปรแกรมดั้งเดิมและโปรแกรมที่ได้รับการแก้ไขให้เป็นไปตามข้อปฏิบัติหลังผ่านการแจ้งเตือนโดยเครื่องมือแล้ว ในแต่ละคู่ของนักพัฒนานั้น สำหรับโปรแกรมชุดที่ 1 นักพัฒนาคนแรกจะเริ่มทำแบบทดสอบกับโปรแกรมเวอร์ชันดั้งเดิมก่อน ส่วนนักพัฒนาอีกคนจะเริ่มทำแบบทดสอบกับโปรแกรมเวอร์ชันที่เป็นไปตามข้อปฏิบัติก่อน แต่เพื่อลดความได้เปรียบของโปรแกรมเวอร์ชันที่ทำแบบทดสอบทีหลัง จากการที่นักพัฒนาอาจมีความเข้าใจในโปรแกรมมาจากโปรแกรมเวอร์ชันที่ทำแบบทดสอบไปก่อนแล้ว ดังนั้นสำหรับโปรแกรมชุดที่ 2 จึงกำหนดให้นักพัฒนาทำแบบทดสอบในลำดับกลับกันคือนักพัฒนาคนแรกจะเริ่มทำแบบทดสอบกับโปรแกรมเวอร์ชันที่เป็นไปตามข้อปฏิบัติก่อน ส่วนนักพัฒนาอีกคนจะเริ่มทำแบบทดสอบโปรแกรมเวอร์ชันดั้งเดิมก่อน ตัวอย่างคำถามและตัวอย่างโปรแกรม P1 สองเวอร์ชันเป็นดังรูปที่ 7.2

1. จงบอกชื่อ Notification สำหรับแจ้งเตือนเมื่อผู้ใช้กดเลือกสติกเกอร์บนรูปภาพ (Literal string)	
เวอร์ชันดั้งเดิม	`[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(getSticker:) name: @"GET_STICKER_KEY" object: nil];`
เวอร์ชันตามข้อปฏิบัติ	`static NSString *const FZGetStickerNotification = @"GET_STICKER_KEY";` `[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(getSticker:) name:FZGetStickerNotification object:nil];`
2. จงบอกระยะเวลาที่ Loading Indicator แสดงผลเมื่อผู้ใช้กดปุ่มบันทึก (Magic number)	
เวอร์ชันดั้งเดิม	`UIView animateWithDuration:0.3 delay:0.0 options: UIViewAnimationOptionCurveEaseInOut`
เวอร์ชันตามข้อปฏิบัติ	`const float FZAnimationTimeDelay = 0.3` `UIView animateWithDuration: FZAnimationTimeDelay delay:0.0 options:UIViewAnimationOptionCurveEaseInOut`
3. จงบอกชื่อตัวแปรสำหรับจัดเก็บรายการของอัลบั้มภาพจาก Camera Roll (Variable name)	
เวอร์ชันดั้งเดิม	`NSMutableArray *groups;`
เวอร์ชันตามข้อปฏิบัติ	`NSMutableArray *albumArray;`
4. จงบอกชื่อเมท็อดที่ถูกเรียกใช้งานเมื่อผู้ใช้กดดูเมนูรายการชุดสติกเกอร์ทั้งหมด (Method name)	
เวอร์ชันดั้งเดิม	`-(IBAction)MySticker:(id)sender;`
เวอร์ชันตามข้อปฏิบัติ	`-(IBAction)selectMenuMySticker:(id)sender;`
5. จงบอกชื่อคลาสหน้าแสดงผลข้อมูลเกี่ยวกับแอพพลิเคชัน (Class name)	
เวอร์ชันดั้งเดิม	`AbAppView`
เวอร์ชันตามข้อปฏิบัติ	`AboutAppViewController`
6. จงบอกชื่อเมท็อดที่ถูกเรียกใช้งานเมื่อผู้ใช้กดไอคอน Twitter (Method name)	
เวอร์ชันดั้งเดิม	`-(IBAction)TweetBtn:(id)sender`
เวอร์ชันตามข้อปฏิบัติ	`-(IBAction)selectMenuTwitter:(id)sender`

รูปที่ 7.2 ตัวอย่างคำถามและโปรแกรมที่ใช้ในการประเมินผล

การทดสอบในแต่ละโปรแกรมจะเริ่มต้นโดยให้นักพัฒนาอ่านโปรแกรมจากเครื่องมือ Xcode ซึ่งเปรียบเสมือนสภาพแวดล้อมจริงในการพัฒนาโปรแกรม และทำการตอบคำถามบนแบบฟอร์มออนไลน์ ก่อนการเริ่มต้นทดสอบผู้วิจัยจะให้นักพัฒนารันโปรแกรมหนึ่งครั้งเพื่อดูการทำงานของแอปพลิเคชัน จากนั้นสั่งหยุดโปรแกรมเพื่อให้นักพัฒนาเริ่มทำการทดสอบโดยการตอบคำถามทั้งหมด 30 ข้อและจับเวลา

ตารางที่ 7.4 เวลาเฉลี่ยที่นักพัฒนาใช้ในการตอบคำถาม

นักพัฒนา	**เวลาที่ใช้ตอบคำถามโปรแกรมดั้งเดิม (วินาที s [นาที m])**	**เวลาที่ใช้ตอบคำถามโปรแกรมตามข้อปฏิบัติ (วินาที s [นาที m])**	**ผลต่างเวลา (วินาที s [นาที m])**
A	3,632s [60m 32s]	2,363s [39m 23s]	1,269s [21m 9s]
B	3,767s [62m 47s]	2,420.5s [40m 20.5s]	1,346.5s [22m 26.5s]
C	2,564.5s [42m 44.5s]	1,686s [28m 6s]	878.5s [14m 38.5s]
D	2,912s [48m 32s]	2,133s [35m 33s]	779s [12m 59s]

จากการทดสอบเปรียบเทียบเวลาที่นักพัฒนาใช้ในการตอบคำถามถูกทุกข้อระหว่างโปรแกรมดั้งเดิมและโปรแกรมที่เป็นไปตามข้อปฏิบัติ ได้ผลดังตารางที่ 7.4 จากตารางผลการทดสอบ ผู้วิจัยได้ค่าประมาณของผลต่างระหว่างค่าเฉลี่ยของสองประชากรแบบจับคู่ (t) ดังการคำนวณต่อไปนี้

$$t = \frac{\overline{d} - d_0}{s_d/\sqrt{n}} = \frac{1068.25 - 0}{281.303/\sqrt{4}}$$

$$= \frac{1068.25}{140.6515}$$

$$= 7.595$$

ผู้วิจัยจะสรุปผลโดยการสร้างเขตปฏิเสธ ซึ่งจะปฏิเสธสมมติฐานถ้า $t > t_{.95;3} = 2.353$ สรุปผลการทดสอบได้ว่า ปฏิเสธ H_0 เนื่องจาก $t > 2.3534$ คือระยะเวลาที่ใช้ในการทำแบบทดสอบถูกต้องทุกข้อของโปรแกรมดั้งเดิมมากกว่าโปรแกรมที่เป็นไปตามข้อปฏิบัติ ที่ระดับนัยสำคัญ 0.05

บทที่ 8
บทสรุปโครงงานและข้อเสนอแนะ

8.1 สรุปผลโครงงานมหาบัณฑิต

การพัฒนาโปรแกรมร่วมกันของคนในทีมพัฒนา มีรูปแบบที่แตกต่างกันเนื่องจากไม่มีมาตรฐานในการเขียนโปรแกรมร่วมกัน ก่อให้เกิดปัญหาในการอ่านและทำความเข้าใจโปรแกรม จากกรณีศึกษาทีมพัฒนาขององค์กรแห่งหนึ่งที่ประสบกับปัญหาดังกล่าว ผู้วิจัยจึงรวบรวมข้อแนะนำในการเขียนโปรแกรมอ็อบเจกทีฟซีจากแหล่งข้อมูลต่าง ๆ เพื่อสร้างเป็นมาตรฐานข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซีสำหรับองค์กร รวมถึงพัฒนาเครื่องมือตรวจสอบว่าโปรแกรมเป็นไปตามข้อปฏิบัตินี้หรือไม่ เครื่องมือผ่านการทดสอบและสามารถตรวจสอบการละเมิดข้อปฏิบัติได้ถูกต้อง ส่วนผลจากการปรับปรุงโปรแกรมให้เป็นมาตรฐานตามข้อปฏิบัติตามคำแนะนำของเครื่องมือพบว่าโปรแกรมสามารถอ่านและเข้าใจได้ง่ายขึ้น นักพัฒนาใช้เวลาในการทำความเข้าใจโปรแกรมน้อยลง

8.2 ปัญหาและข้อจำกัดในการทำโครงงาน

การจัดทำโครงงานมหาบัณฑิตนี้ พบประเด็นปัญหาและข้อจำกัดในการทำโครงงานดังต่อไปนี้

1 ในการทดสอบการทำงานของเครื่องมือ ผู้วิจัยพบปัญหาในการตรวจสอบชนิดของคำด้วย WordsAPI โดยหากคำที่ตรวจสอบไม่ใช่คำนามเอกพจน์ เช่น users และไม่ใช่คำกริยาช่องที่หนึ่ง เช่น went, gone, requested เมื่อตรวจสอบคำเหล่านี้ WordsAPI จะคืนค่าผลลัพธ์ว่าไม่พบคำดังกล่าว หรือคืนค่าชนิดของคำเป็นค่า Null ทำให้เครื่องมือไม่สามารถระบุชนิดของคำเพื่อนำไปตรวจสอบต่อได้ เมื่อพบกรณีนี้ ผู้วิจัยจะนำคำมาตรวจสอบเพิ่มเติมโดยใช้ WebKnox Words APIในการหาคำนามเอกพจน์หรือคำกริยาช่องที่หนึ่งของคำ จากนั้นจึงนำมาตรวจสอบชนิดของคำด้วย WordsAPI อีกครั้ง จึงทำให้เครื่องมือสามารถตรวจสอบข้อปฏิบัติได้อย่างถูกต้อง

2 การตรวจสอบข้อปฏิบัติบางกรณี เครื่องมือจะต้องทำการตรวจสอบข้อมูลจาก API ดังนั้นเครื่องมือจะสามารถใช้งานได้ในขณะที่มีการเชื่อมต่อกับอินเทอร์เน็ตเท่านั้น

3 โปรแกรมที่นำมาตรวจสอบจะต้องสามารถรันได้

8.3 ข้อเสนอแนะ

สำหรับแนวทางในการดำเนินงานต่อ ผู้วิจัยจะปรับเครื่องมือตรวจสอบให้ผู้ใช้งานสามารถใช้งานได้ง่ายขึ้น โดยจะพัฒนาในรูปแบบของปลั๊กอินสำหรับ Xcode เพื่อลดขั้นตอนในการทำงาน รวมถึงปรับปรุงโปรแกรมให้สามารถตั้งค่าการตรวจสอบได้มากขึ้น นอกจากนี้ผู้วิจัยจะศึกษาข้อแนะนำในการเขียนโปรแกรมใน ภาษา Swift ซึ่งใช้สภาพแวดล้อมการทำงานเดียวกันกับโปรแกรมอ็อบเจกทีฟซี เพื่อกำหนดมาตรฐานข้อปฏิบัติการเขียนโปรแกรม Swift และพัฒนาเครื่องมือในการตรวจสอบ โดยยึดถือโครงแบบในการทำงานเดียวกับเครื่องมือในงานวิจัย

รายการอ้างอิง

[1] ISO/IEC, "ISO/IEC 9126-1 Software engineering- Product quality- Part 1: Quality model," 2001

[2] "Coding Guidelines for Cocoa" [Online]. Available: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CodingGuidelines/CodingGuidelines.pdf [Accessed: 2-11-2015]

[3] "NYTimes Objective-C Style Guide" [Online]. Available: https://github.com/NYTimes/objective-c-style-guide [Accessed: 2-11-2015]

[4] "The official raywenderlich.com Objective-C style guide" [Online]. Available: https://github.com/raywenderlich/objective-c-style-guide [Accessed: 2-11-2015]

[5] "Google Objective-C Style Guide" [Online]. Available: https://google.github.io/styleguide/objcguide.xml [Accessed: 2-11-2015]

[6] "Naming Convention" [Online]. Available: http://www.oracle.com/technetwork/java/codeconventions-135099.html [Accessed: 1-11-2015].

[7] "Magic Number" [Online]. Available: http://c2.com/cgi/wiki?MagicNumber [Accessed: 1-11-2015].

[8] "Literal String" [Online]. Available: http://www.computerhope.com/jargon/l/literal.htm [Accessed: 15-11-2015].

[9] "OBJECTIVE-CLEAN" [Online]. Available: http://objclean.com/index.php [Accessed: 5-11-2015]

[10] "Clang 3.9 documentation" [Online]. Available: http://clang.llvm.org/docs/ObjectiveCLiterals.html [Accessed: 3-4-2016]

[11] "OCLint" [Online]. Available: http://oclint.org/ [Accessed: 3-4-2016]

[12] "Uncrustify" [Online]. Available: http://uncrustify.sourceforge.net/ [Accessed: 3-4-2016]

[13] "Faux Pas" [Online]. Available: http://fauxpasapp.com/ [Accessed: 3-4-2016]

[14] "Programming with Objective C" [Online]. Available: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/ProgrammingWithObjectiveC.pdf. [Accessed: 2-11-2015].

[15] "Objective-C Runtime Programming Guide" [Online]. Available:

[16] [1] Y. Wen, X. Tang, L. Ju, and T. Chen, “PeRex : A power efficient FPGA-based architecture for regular expression matching,” 2011.

[17] [1] M. Smit, B. Gergel, H. J. Hoover, and E. Stroulia, “Maintainability and Source Code Conventions : An Analysis of Open Source Projects,” pp. 1–10.

[18] Y. Wang, S. Wang, X. Li, H. Li, and J. Du, “Identifier Naming Conventions and Software Coding Standards: A Case Study in One School of Software,” Comput. Intell. Softw. Eng. (CiSE), 2010 Int. Conf., pp. 1–4, 2010.

[19] S. Butler, M. Wermelinger, and Y. Yu, “Investigating naming convention adherence in Java references,” IEEE Int. Conf. Software Maintenance and Evolution (ICSME), 2015

[20] "PYTHON WORDSEGMENT MODULE" [Online]. Available: http://www.grantjenks.com/blog/portfolio-post/english-word-segmentation-python/ [Accessed: 3-4-2016]

[21] "WordsAPI" [Online]. Available: https://www.wordsapi.com [Accessed: 3-4-2016]

[22] "WebKnox REST API" [Online]. Available: https://webknox.com/api [Accessed: 3-4-2016]