ข้อเสนอโครงงานมหาบัณฑิต
(MASTER PROJECT PROPOSAL)

ชื่อเรื่อง (ภาษาไทย)	เครื่องมือตรวจสอบข้อปฏิบัติการตั้งชื่อในโปรแกรมอ็อบเจกทีฟซี
ชื่อเรื่อง (ภาษาอังกฤษ)	Objective C Naming Convention Checker
เสนอโดย	นางสาวฤชุตา นันตะพานา
รหัสนิสิต	5770957121
หลักสูตร	วิศวกรรมซอฟต์แวร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
คณะ	วิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
สถานที่ติดต่อ	876/10 คอนโดบ้านสวนลาซาล ถนนสุขุมวิท 105 แขวงบางนา เขตบางนา กรุงเทพฯ 10260
โทรศัพท์	083-032-5625
อีเมล	Ruchuta.N@student.chula.ac.th
อาจารย์ที่ปรึกษา	รศ.ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา
คำสำคัญ (ภาษาไทย)	ข้อปฏิบัติการเขียนโปรแกรม, ข้อปฏิบัติการตั้งชื่อ, อ็อบเจกทีฟซี
คำสำคัญ (ภาษาอังกฤษ)	Code Convention, Naming Convention, Objective C

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 จะมีนักเขียนโปรแกรมมากกว่าหนึ่งคน หรือโครงการอาจมีการเปลี่ยนแปลงผู้รับผิดชอบในการพัฒนา ซอฟต์แวร์จึงถูกส่งมอบให้นักเขียนโปรแกรมคนอื่นพัฒนาต่อ นักเขียนโปรแกรมจึงมักประสบปัญหาในการอ่านซึ่งพัฒนามาก่อนหน้าโดยนักเขียนโปรแกรมคนอื่น ก่อนที่จะพัฒนาในส่วนของงานที่ต้องแก้ไขหรือเพิ่มเติม สาเหตุของปัญหานี้ มาจากการที่ฝ่ายงานไม่ได้มีการกำหนดมาตรฐานข้อปฏิบัติในการเขียนโปรแกรม ทำให้นักเขียนโปรแกรม ใช้คำและรูปแบบในการเขียนโปรแกรมที่แตกต่างกันออกไป

จากกรณีศึกษาของฝ่ายงานพัฒนาโมบายล์แอปพลิเคชันบนระบบปฏิบัติการ iOS ขององค์กรแห่งหนึ่ง ฝ่ายงานที่กำลังประสบกับปัญหาดังกล่าวโดยที่การพัฒนาแอปพลิเคชันร่วมกันและการรับช่วงพัฒนาต่อจากนักเขียนโปรแกรมคนก่อนทำได้ยาก ใช้เวลานานในการศึกษาโปรแกรม หัวหน้าฝ่ายงานจึงมีความต้องการที่จะสร้างมาตรฐานการเขียนโปรแกรมในภาษาอ็อบเจ็กทีฟซีสำหรับระบบปฏิบัติการ iOS ให้เกิดขึ้นในองค์กร จากการค้นคว้าพบว่า บริษัทแอปเปิลได้มีการเผยแพร่เอกสารข้อแนะนำในการเขียนโปรแกรมภาษาอ็อบเจ็กทีฟซีไว้ โครงงานนี้จึงมีแนวคิดในการพัฒนาเครื่องมือสำหรับตรวจสอบข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซี เพื่อสนับสนุนการใช้มาตรฐานในการเขียนโปรแกรมอ็อบเจกทีฟซีสำหรับฝ่ายงานดังกล่าว โดยจะยึดหลักข้อปฏิบัติจากเอกสารข้อแนะนำในการเขียนโปรแกรมของบริษัทแอปเปิล[12] และเพิ่มเติมข้อปฏิบัติจากเอกสารคำแนะนำอื่นๆ รวมถึงพิจารณาข้อปฏิบัติที่ได้รับเสนอจากฝ่ายงานพัฒนา โดยเน้นเฉพาะการตรวจสอบการตั้งชื่อและการกำหนดค่าคงที่ในโปรแกรม หลังจากมีการประกาศใช้ข้อปฏิบัตินี้แล้ว ฝ่ายงานจะต้องมีการตรวจสอบว่านักเขียนโปรแกรมในฝ่ายงานได้ปฏิบัติตามข้อกำหนดหรือไม่ ฝ่ายงานสามารถใช้เครื่องมือนี้เพื่อระบุจุดต่างๆในโปรแกรมที่ละเมิดข้อปฏิบัติและควรได้รับการปรับปรุง รวมทั้งสามารถใช้เครื่องมือนี้ในการตรวจสอบซอฟต์แวร์ที่ได้พัฒนาขึ้นใหม่

หลังจากมีการประกาศใช้มาตรฐานว่านักเขียนโปรแกรมในฝ่ายงาน ยึดถือตามข้อปฏิบัติการเขียนโปรแกรมที่ได้กำหนดไว้หรือไม่

2. ทฤษฎีที่เกี่ยวข้อง

2.1 ข้อปฏิบัติการเขียนโปรแกรม

ข้อปฏิบัติการเขียนโปรแกรม (Coding Convention) คือกฎบางอย่างสำหรับการเขียนชุดคำสั่งในโปรแกรม [2] โดยจะใช้ในการเขียนโปรแกรมในขั้นตอนของการพัฒนาซอฟต์แวร์ ข้อปฏิบัติส่วนมากในปัจจุบันจะเกี่ยวข้องกับรูปแบบของการเขียนโปรแกรม เช่น การตั้งชื่อ, การย่อหน้า โดยมีเป้าหมายหลักคือการทำให้ชุดคำสั่งซอฟต์แวร์ง่ายต่อการอ่านและการบำรุงรักษา

โครงงานนี้สนใจข้อปฏิบัติการเขียนโปรแกรมใน 3 หัวข้อ คือการใช้ Magic Number, การใช้ Multiple Literal String และข้อปฏิบัติในการตั้งชื่อ

- Magic Number

  Magic Number คือ ค่าที่แท้จริงของตัวอักษรที่ปรากฏในโปรแกรม [3] ยกตัวอย่าง 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 qoute) และ ' ' (Single qoute) [4] โดยจะมีค่าซ้ำๆกันในหลายจุดของโปรแกรม [5] แทนที่จะประกาศค่าไว้เป็นค่าคงที่ เช่น “Hello” เป็น Literal String

- Naming Convention (ข้อปฏิบัติการตั้งชื่อ)

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

2.2 เอกสารข้อแนะนำการเขียนโปรแกรมสำหรับภาษาอ็อบเจกทีฟซี

<u>พื้นฐานการตั้งชื่อในโปรแกรม</u>

1. หลักการตั้งชื่อพื้นฐาน

   1.1 ความชัดเจน (Clarity)

- ชื่อควรสื่อความหมายชัดเจนและสั้นกระชับที่สุดเท่าที่จะทำได้ แต่ไม่สั้นเกินไป
- ไม่ควรใช้อักษรย่อชื่อ
- กลุ่มคำย่อที่สามารถใช้ได้ ดังนี้

Abbreviation	Meaning and comments
alloc	Allocate.
alt	Alternate.
app	Application. For example, NSApp the global application object. However, "application" is spelled out in delegate methods, notifications, and so on.
calc	Calculate.
dealloc	Deallocate.
func	Function.
horiz	Horizontal.
info	Information.
init	Initialize (for methods that initialize new objects).
int	Integer (in the context of a C `int`—for an `NSInteger` value, use `integer`).
max	Maximum.
min	Minimum.
msg	Message.
nib	Interface Builder archive.
pboard	Pasteboard (but only in constants).
rect	Rectangle.
Rep	Representation (used in class name such as `NSBitmapImageRep`).
temp	Temporary.
vert	Vertical.

1.2 ความสอดคล้อง

- เป็นสิ่งสำคัญโดยเฉพาะอย่างยิ่งเมื่อมีคลาสที่เมท็อด มีความสามารถในการสืบทอด

(Polymorphism)

- เมท็อดที่ทำสิ่งเดียวกน แต่อยู่ต่างคลาสกัน ควรใช้ชื่อเดียวกัน

1.3 ไม่อ้างอิงตัวเอง

- ชื่อไม่ควรอ้างอิงตัวเอง เช่น NSStringObject แค่ NSString ก็พอ
- ยกเว้นชื่อของค่าคงที่หรือชื่อของตัวแปรชนิด Notification

2. คำเติมหน้า (Prefix)

2.1 คำเติมหน้าเป็นวิ่งสำคัญมากในการตั้งชื่อ เนื่องจากจะทำให้สามารถแยกพื้นที่การทำงานของฟังก์ชันในโปรแกรมได้ เช่นการเรียกใช้ Framework อาจมีการตั้งชื่อ Package ภายในเหมือนกัน โดยฟังก์ชันในแต่ละ Framework จะถูกแยกด้วยคำเติมหน้า เพื่อป้องกันการขัดแย้งกันของชื่อ กรณีที่ชื่อเหมือนกัน

2.2 ใช้คำเติมหน้ากับกลุ่มชื่อบางกลุ่ม ได้แก่

- ชื่อ Class
- ชื่อ Protocol
- ชื่อ Function
- ชื่อ Constant
- ชื่อ Typedef Structure

2.3 ไม่ใช้คำเติมหน้ากับชื่อเมท็อด

3. ข้อกำหนดของรูปแบบการใช้คำ

3.1 ชื่อประกอบไปด้วยคำหลายคำ ไม่ใช้เครื่องหมายเว้นวรรค หรือสัญลักษณ์ในการแบ่งคำ เช่น underscore(_) หรือ dashes (-)

3.2 ใช้ camel-casing : อักษรตัวแรกของคำเป็นอักษรพิมพ์ใหญ่

- สำหรับ เมท็อด : เริ่มต้นด้วยตัวพิมพ์เล็ก คำต่อไปตักแรกเป็นตัวพิมพ์ใหญ่ ยกเว้นกรณีเริ่มต้นด้วยคำย่อที่เป็นที่รู้จัก ก็จะขึ้นต้นด้วยตัวใหญ่
- สำหรับ Function / Constant ใช้ Prefixเดียวกันสำหรับคลาสที่สัมพันธ์กัน และอักษรตัวแรกเป็นตัวพิมพ์ใหญ่
- หลีกเลี่ยงการใช้ underscore ในชื่อเมท็อด

4. หลักการตั้งชื่อ Class และ Protocol

4.1 ชื่อคลาสควรประกอบไปด้วยคำนามที่ชัดเจนเพื่อที่จะระบุว่าคลาสต้องการที่จะทำอะไร

4.2 มีคำเติมหน้าที่เหมาะสม ดูตัวอย่างใน Foundation กับ Framework เช่น ‘NS’ จะนำหน้าตัวแปรที่เป็นอ็อบเจกต์ เช่น NSString และ NSDate

4.3 ชื่อ Protocol ควรตั้งชื่อตามกลุ่มลักษณะการทำงานกลุ่ม Protocol ส่วนใหญ่มีความสัมพันธ์กับเมท็อดที่ไม่ได้เกี่ยวข้องกับคลาส ชนิดของ Protocol ควรที่จะตั้งชื่อ ไม่ให้สับสนกับชื่อคลาส

<u>หลักการตั้งชื่อเมท็อด</u>

1. หลักการตั้งชื่อเมท็อดพื้นฐาน

1.1 เริ่มต้นด้วยอักษรตัวพิมพ์เล็ก คำต่อไปเริ่มต้นด้วยตัวพิมพ์ใหญ่ และไม่ใช้ Prefix (มีข้อยกเว้นสำหรับตัวย่อพื้นฐานที่ทั่วไปใช้ เช่น TIFF หรือ PDF สามารถขึ้นต้นด้วยอักษรพิมพ์ใหญ่ได้และ ใช้ Prefix ได้)

1.2 เมท็อดที่แสดงการกระทำกับอ็อบเจกต์ใดๆ ให้ขึ้นต้นด้วยคำกริยา ยกเว้นคำกริยา do และ dose และไม่ใช้ adverb หรือ adjective ก่อน verb

1.3 เมท็อดที่คืนค่าของตัวแปรโดยตรง ไม่จำเป็นต้องใช้ Get ยกเว้น มีค่าที่ไม่ได้ Return ออกมาตรงๆ เช่น ต้องผ่านการประมวลผลก่อน

1.4 ใช้ Keyword ก่อนตัวแปรที่รับเข้ามาในเมท็อด **(**argument**)** เพื่ออธิบายตัวแปรนั้นๆ

1.5 เพิ่ม Keyword ไว้ด้านท้ายของชื่อเมท็อดเดิม เมื่อสร้าง เมท็อดที่เฉพาะเจาะจงกว่าเมท็อดเดิมที่สืบทอดมา

1.6 ไม่ใช้คำว่า and เป็น Keyword ในการเชื่อมตัวแปรที่จะรับเข้ามาในฟังก์ชัน

2. Accessor เมท็อด

2.1 ถ้าคุณลักษณะของอ็อบเจกต์เป็นคำนาม เมท็อดจะอยู่ในรูปแบบ ดังนี้

Getter: - (type)noun;

Setter: - (void)setNoun:(type)aNoun;

2.2 ถ้าคุณลักษณะของอ็อบเจกต์เป็นคำคุณศัพท์ เมท็อดจะอยู่ในรูปแบบ ดังนี้

Getter: - (BOOL)isAdjective;

Setter: - (void)setAdjective:(BOOL)flag;

2.3 ถ้าคุณลักษณะของอ็อบเจกต์เป็นคำกริยาในรูปของ Present Simple Tense โดย เมท็อดจะอยู่ในรูปแบบ ดังนี้

Getter: - (BOOL)verbObject;

Setter: - (void)setVerbObject:(BOOL)flag;

2.4 ไม่แปลงคำกริยาให้กลายเป็นคำคุณศัพท์ โดยทำให้เป็นกริยาช่องที่สาม เช่น

ชื่อเมท็อดที่ควร

Getter: - (BOOL)acceptsGlyphInfo;

Setter: - (void)setAcceptsGlyphInfo:(BOOL)flag;

ชื่อเมท็อดที่ไม่ควรใช้

Getter: - (BOOL)glyphInfoAccepted;

Setter: - (void)setGlyphInfoAccepted:(BOOL)flag;

2.5 ควรใช้กริยาช่วย เช่น can, should หรือ will เพื่อช่วยให้ชื่อเมท็อดสื่อความหมายชัดเจนยิ่งขึ้น แต่ห้ามใช้ do หรือ does เช่น

ชื่อเมท็อดที่ควร

Getter: - (void)setCanHide:(BOOL)flag;

Setter: - (BOOL)canHide;

ชื่อเมท็อดที่ไม่ควรใช้

Getter: - (BOOL)doesAcceptGlyphInfo;

Setter: - (void)setDoesAcceptGlyphInfo:(BOOL)flag;

2.6 ขึ้นต้นชื่อเมท็อดด้วย Get ในกรณีที่เมท็อดไม่ได้คืนค่าอ็อบเจ็กโดยตรง หรือในกรณีที่เมท็อดมีการคืนค่ามากกว่าหนึ่งค่า เช่น

```
-- (void)getLineDash:(float *)pattern count:(int *)count

phase:(float *)phase;
```

3. หลักการตั้งชื่อฟังก์ชัน

3.1 การตั้งชื่อฟังก์ชัน มีพื้นฐานข้อแนะนำเช่นเดียวกับการตั้งชื่อเมท็อด แต่จะมีข้อยกเว้นอยู่ 2 อย่าง ได้แก่

- ขึ้นต้นชื่อด้วยคำเติมหน้าเดียวกันกับที่ใช้ในชื่อคลาสกับค่าคงที่
- อักษรตัวแรกหลังคำเติมหน้าจะเป็นตัวพิมพ์ใหญ่

3.2 ชื่อฟังก์ชันส่วนใหญ่จะขึ้นต้นด้วยคำกริยาเพื่ออธิบายถึงการทำงานของฟังก์ชัน

3.3 ฟังก์ชันที่มีการคืนค่าคุณลักษณะ จะมีกฎของการตั้งชื่อที่เพิ่มเข้ามา

- ถ้าฟังก์ชันคืนค่าคุณลักษณะของตัวมันเอง ให้ละเว้นการเริ่มต้นด้วยคำกริยา
- ถ้าฟังก์ชันมีการคืนค่าโดยการอ้างอิงค่า ไม่ได้คืนค่าของคุณลักษณะโดยตรง ให้ใช้ Get
- ถ้าฟังก์ชันมีการคืนค่าเป็นชนิด Boolean ควรเริ่มต้นชื่อฟังก์ชันด้วย Inflected Verb

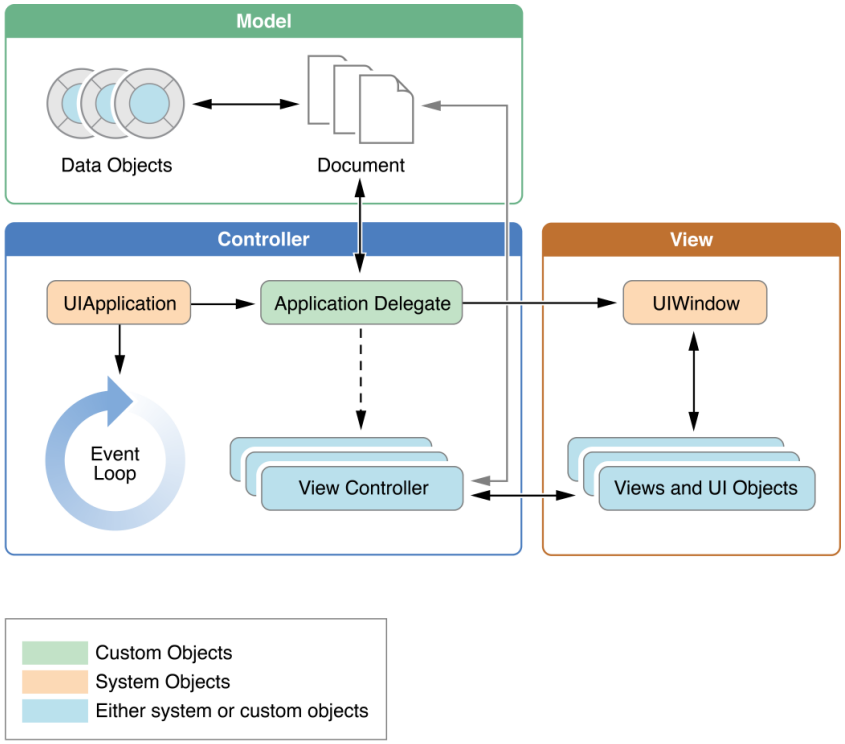
2.3 ภาษาอ็อบเจกทีฟซี

ภาษาอ็อบเจกทีฟซี เป็นภาษาเริ่มต้นที่ใช้ในการเขียนโปรแกรมบนระบบปฏิบัติการ OSX และ iOS ซึ่งมีรากฐานมาจากภาษาซีดังรูปที่ 1 โดยมีการเพิ่มความสามารถของ Object-Oriented หรือการมองส่วนประกอบของโปรแกรมให้เป็นวัตถุใดๆ รวมถึงความสามารถในการจัดการวัตถุในขณะที่โปรแกรมกำลังทำงาน อ็อบเจกทีฟซีสืบทอดโครงสร้างภาษา ชนิดข้อมูลพื้นฐาน และคำสั่งควบคุมการทำงานของโปรแกรมมาจากภาษาซี โดยมีการเพิ่มเติมในส่วนของโครงสร้างสำหรับการสร้างคลาสและเมท็อด ปัจจุบันภาษาอ็อบเจกทีฟซีมีการพัฒนาต่อจากโครงสร้างดั้งเดิม โดยเวอร์ชันปัจจุบัน คืออ็อบเจกทีฟซี 2.0 [7]

รูปที่ 1 ภาษาอ็อบเจกทีฟซีสืบทอดความสามารถมาจากภาษาซี

2.4 โปรแกรม Xcode

เครื่องมือในการพัฒนาโปรแกรมของบริษัทแอปเปิล โดยจะใช้ในการพัฒนาแอปพลิเคชันที่ใช้งานบนผลิตภัณฑ์ของแอปเปิล ได้แก่ iPad, iPhone, Apple Watch และ Mac โดย Xcode จะมีเครื่องมือในการจัดการกระบวนการพัฒนา ตั้งแต่เริ่มต้นสร้างแอปพลิเคชัน ทดสอบ เพิ่มประสิทธิภาพในการทำงาน จนถึงขั้นตอนของการส่งมอบแอปพลิเคชันไปยังคลังแอปพลิเคชัน (Apple Store) ภาษาโปรแกรมหลักที่ใช้ในการพัฒนาแอปพลิเคชันคือภาษาอ็อบเจกทีฟซี (Objective C) และ Swift โดยภาษา Swift เป็นภาษาใหม่ที่ทางแอปเปิลพัฒนาขึ้น เริ่มประกาศใช้อย่างเป็นทางการเมื่อปี ค.ศ 2014 และยังอยู่ในช่วงของการพัฒนา จึงยังไม่เป็นที่แพร่หลายในการนำมาพัฒนาแอปพลิเคชันในอุตสาหกรรมซอฟต์แวร์ โครงสร้างการทำงานของโปรแกรม จะใช้สถาปัตยกรรมแบบ MVC (Model-View-Controller) โดยรูปแบบนี้จะแบ่งส่วนของข้อมูล (Model) และส่วนของคำสั่งในการทำงานของโปรแกรม (Controller) ออกจากส่วนของการแสดงผล (View) ดังรูปที่ 2



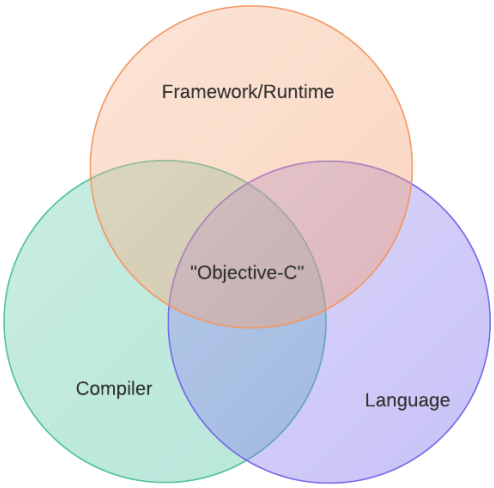
รูปที่ 2 โครงสร้างการทำงานของโปรแกรม Xcode

2.5 คลังอ็อบเจกทีฟซีรันไทม์ (Objective C Runtime library)

เครื่องมือพัฒนา Xcode สำหรับภาษาอ็อบเจกทีฟซี มีส่วนการทำงานที่เรียกว่ารันไทม์ (Runtime) [8] โดยสามารถจัดการกับวัตถุในขณะที่โปรแกรมทำงานอยู่ ดังนั้นอ็อบเจกทีฟซีจึงไม่ได้ต้องการเพียงเฉพาะการทำงานของคอมไพเลอร์ แต่ยังต้องอาศัยระบบรันไทม์เพื่อประมวลผลโปรแกรมที่ถูกคอมไพล์แล้วด้วยดังรูปที่ 3 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

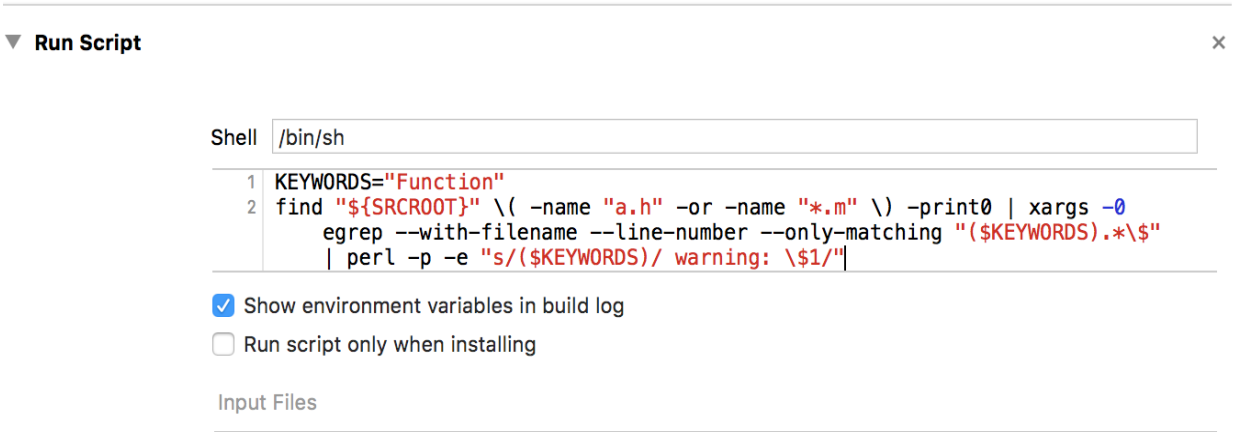


รูปที่ 3 อ็อบเจกทีฟซีมีการทำงานร่วมกันระหว่างภาษา รันไทม์และคอมไพเลอร์

2.6 ภาษาเชลล์ผ่านรันสคริปต์ (Run Script)

นอกเหนือจากการประมวลผลโปรแกรมผ่านคอมไพเลอร์และรันไทม์ โปรแกรม Xcode ยังมีความสามารถในการประมวลผลผ่านคำสั่ง (Script) ที่เขียนโดยภาษาเชลล์ใด ๆ โดยเชลล์ (Shell) เป็นโปรแกรมบนระบบยูนิกซ์ (UNIX) ทำหน้าที่ในการติดต่อระหว่างผู้ใช้งานและยูนิกซ์ ทำให้ผู้ใช้สามารถจะป้อนคำสั่งให้ยูนิกซ์ทำงานตามที่ต้องการ คำสั่งจะเปรียบได้กับโปรแกรม หากนำคำสั่งต่างๆมาเรียงต่อกัน จะได้กลุ่มของชุดคำสั่งที่เรียกว่าคำสั่งเชลล์ (Shell Script)

ส่วนการทำงานของคำสั่งเชลล์ในโปรแกรม Xcode จะอยู่ภายในส่วนของการตั้งค่าโปรแกรม โดยนักเขียนโปรแกรมจะต้องทำการเพิ่มเมนูรันสคริปต์ (Run Script) และทำการเขียนโปรแกรมคำสั่งตามที่ต้องการ ดังรูปที่ 4



รูปที่ 4 ส่วนการทำงานของเชลล์สคริปในโปรแกรม Xcode

2.7 Regular Expression

Regular Expression คือรูปแบบที่ถูกใช้ในการนิยามและระบุข้อมูลชนิดสตริง [9] รูปแบบเกิดจากการรวมกันของอักขระและสัญลักษณ์พิเศษ ดังตัวอย่างในตารางที่ 1 Regular Expression สามารถอธิบายถึงความซับซ้อนของรูปแบบสตริงได้โดยการกำหนดกฎเกณฑ์ด้วยข้อความสั้นๆ นอกจากนี้ยังสามารถใช้ในการปรับเปลี่ยนแก้ไขและเปรียบเทียบกับข้อมูลสตริงได้

ตารางที่ 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 จะถือว่าตรงตามรูป

2.8. NSRegular Expression

คลาสหนึ่งในภาษาอ็อบเจกทีฟซี ที่มีความสามารถใน การใช้ Regular Expression กับข้อมูลสตริงในโปรแกรม โดยรูปแบบไวยากรณ์ที่ใช้ในปัจจุบันสนับสนุนการใช้ Regular Expression แบบ ICU พื้นฐานเมท็อดในการเปรียบเทียบข้อมูล จะเป็นการทำงานแบบ Block

2.9 NSLinguisticTagger

คลาสหนึ่งในภาษาอ็อยเจกทีฟซีที่ถูกใช้ในการจำแนกคำที่เป็นภาษาธรรมชาติและแสดงรายละเอียดของคำ เช่น ชนิด ภาษาและรากศัพท์ของคำความสามารถในการ ตัดหรือแยก (Tokenize) ข้อมูลสตริงออกเป็นคำระบุชนิดของคำ (Part of speech) ภาษา, รากศัพท์

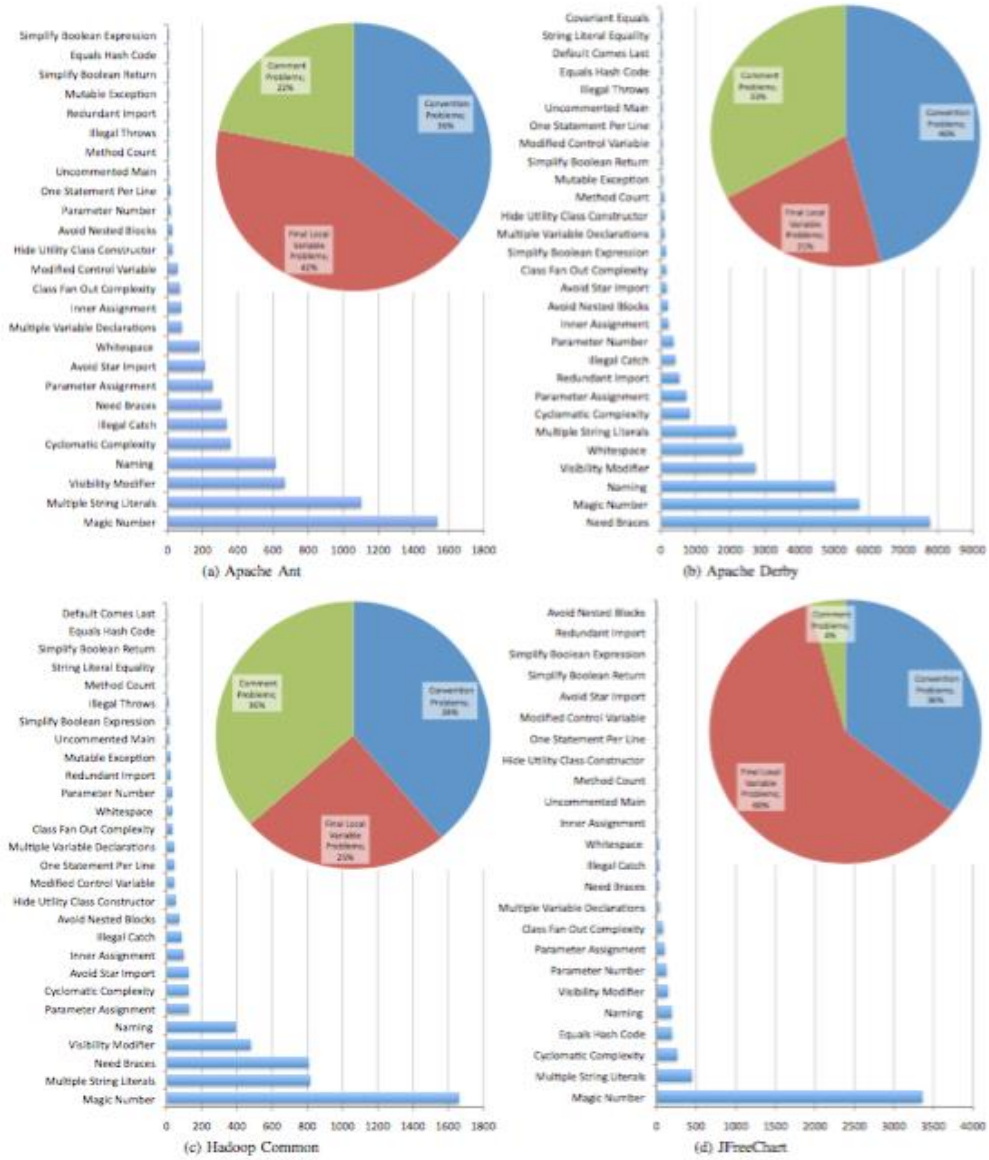
2.10 ParseKit Framework

Framework ภาษาอ็อบเจกทีฟซี ที่ถูกเขียนขึ้นโดยTodd Ditchendorf มีความสามรถในการจำแนกข้อมูลชนิดสตริง วิเคราะห์คำและโครงสร้างในประโยคของภาษาระดับสูง รวมถึงการสร้างชุดคำสั่งโปรแกรมอ็อบเจกทีฟซีจากการสร้างภาษาโดยใช้ BNF-style grammar syntax

3. งานวิจัยที่เกี่ยวข้องและเครื่องมือที่เกี่ยวข้อง

3.1 Maintainability and Source Code Conventions: An Analysis of Open Source Projects [5]

Michael Smit และผู้ร่วมงาน ได้กล่าวถึงปัจจัยที่มีความสำคัญต่อการบำรุงรักษาซอฟต์แวร์ ไม่ได้มีเพียงความซับซ้อนเท่านั้น แต่มองว่าข้อปฏิบัติการเขียนโปรแกรมก็เป็นปัจจัยอีกอย่างหนึ่งที่สำคัญ ในการเขียนโปรแกรม นักพัฒนาจะเป็นคนตัดสินใจว่าจะเขียนโปรแกรมอย่างไร เช่น การใช้ Magic Number หรือการ Hard Code String อาจส่งผลลบต่อความสามารถในการอ่านและทำความเข้าใจ รวมไปถึงการบำรุงรักษาซอฟต์แวร์ โดยจะทำให้การเปลี่ยนแปลงโปรแกรมเป็นไปได้ยาก งานวิจัยนี้ได้ทำการรวบรวมข้อปฏิบัติการเขียนโปรแกรมที่สัมพันธ์กับการบำรุงรักษาซอฟต์แวร์มาทั้งหมด 71 ข้อ จากการเก็บรวบรวมข้อเสนอแนะจากนักวิศวกรรมซอฟต์แวร์ และนำเสนอเมทริกซ์ แสดงมุมมองต่างๆของการบำรุงรักษาซอฟร์แวร์ 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 5



รูปที่ 5 กราฟสรุปผลจำนวนการละเมิดข้อปฏิบัติ

จากการศึกษางานวิจัยนี้ โครงงานพิจารณาเลือกข้อปฏิบัติ 3 ข้อดังที่กล่าวมา ในการพัฒนาเครื่องมือตรวจสอบข้อปฏิบัติการเขียนโปรแกรมภาษาอ็อบเจกทีฟซี

3.2 Identifier Naming Conventions and Software Coding Standards: A Case Study in One School of Software [10]

Yanqing Wang และผู้ร่วมงาน มองเรื่องความสำคัญของการกำหนดมาตรฐานในการเขียนโปรแกรมว่ามาตรฐานนี้จะช่วยให้นักเขียนโปรแกรมสามารถสื่อสารและทำงานร่วมกันได้อย่างมีประสิทธิภาพ ส่งผลต่อคุณภาพ

ของผลิตภัณฑ์ โดยในปัจจุบันมีบริษัทที่เป็นที่ยอมรับในระดับสากล ประกาศใช้เกณฑ์ข้อปฏิบัติในการเขียนโปรแกรมอย่างเคร่งครัด การค้นคว้าและจัดอบรมที่เกี่ยวข้องกับเรื่องดังกล่าวมีมากขึ้น มาตรฐานในการเขียนโปรแกรมจึงกลายเป็นอีกปัจจัยที่ส่งผลต่อการแข่งขันกันในอุตสาหกรรมการผลิตซอฟต์แวร์

งานวิจัยนี้กล่าวถึงมาตรฐานการเขียนโปรแกรมว่าถูกแบ่งเป็น 4 หมวดหมู่ได้แก่ การจัดวาง (Layout) การตั้งชื่อ (Naming) คำอธิบาย (Comment) และการเขียนโปรแกรม (Coding) นักวิจัยได้เน้นถึงปัญหาและความสำคัญของการตั้งชื่อ เพราะเป็นปัญหาที่พบมากในปัจจุบัน ตัวอย่างปัญหาที่พบคือการตั้งชื่อตัวแปรที่ไม่สื่อความหมาย เช่น I, jj, kkk เป็นต้น นักเขียนโปรแกรมส่วนมากไม่ให้ความสำคัญ ส่งผลให้โปรแกรมยากต่อการทำความเข้าใจ การบำรุงรักษาและการนำกลับมาใช้ใหม่

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

1. Hungarian:

ชื่อจะขึ้นต้นด้วยตัวอักษรตัวพิมพ์เล็กที่แสดงถึงขอบเขตหรือชนิดตามที่ได้มีระบุไว้ หลังจากนั้น ทุกคำจะขึ้นต้นด้วยอักษรพิมพ์ใหญ่ เช่น iStudentNumber เป็นข้อมูลประเภท int

2. Camel:

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

ตารางที่ 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 ได้ผลการตั้งชื่อโดยใช้รูปแบบ Camel Case ได้รับความนิยมสูงสุด ผู้วิจัยจึงแนะนำว่า Camel Case เป็นรูปแบบที่ควรจะให้ความสำคัญและสนับสนุนให้ใช้ในวงกว้าง

จากการศึกษางานวิจัยดังกล่าว แสดงให้เห็นถึงขั้นตอนวิธีการเบื้องต้นในการตรวจสอบรูปแบบของชุดคำสั่งในโปรแกรม โดยจะต้องทำการสกัดชื่อในโปรแกรมออกมาและนำมาตรวจสอบรูปแบบของชื่อโดยใช้วิธีการ Regular Expression และจากผลของงานวิจัย ทำให้ทราบว่าการเขียนโปรแกรมส่วนมากจะนิยมใช้ในรูปแบบของ Camel Case โดยโครงงานนี้จะนำรูปแบบดังกล่าว ไปใช้เป็นข้อปฏิบัติหนึ่งในการตรวจสอบ

3.3 โปรแกรม Objective Clean

โปรแกรม Objective Clean [11] 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

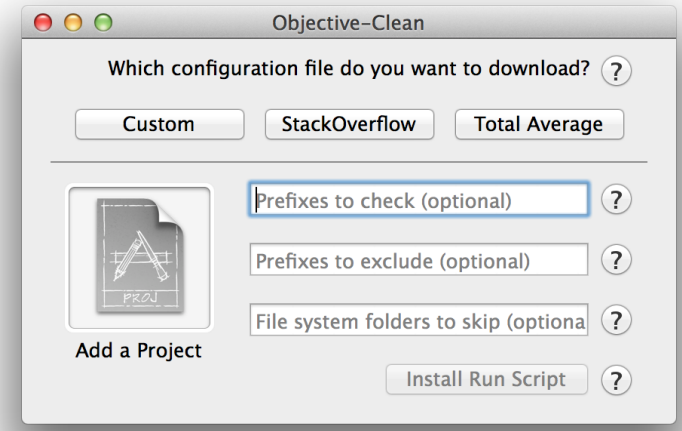
```
NSString* string; //A
NSString * string1; //B
NSString *string2; //C
NSString*string3; //D
```

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

1. เปิดใช้งานเครื่องมืออ็อบเจกทีฟคลีน จะพบส่วนต่อประสาน ดังรูปที่ 6

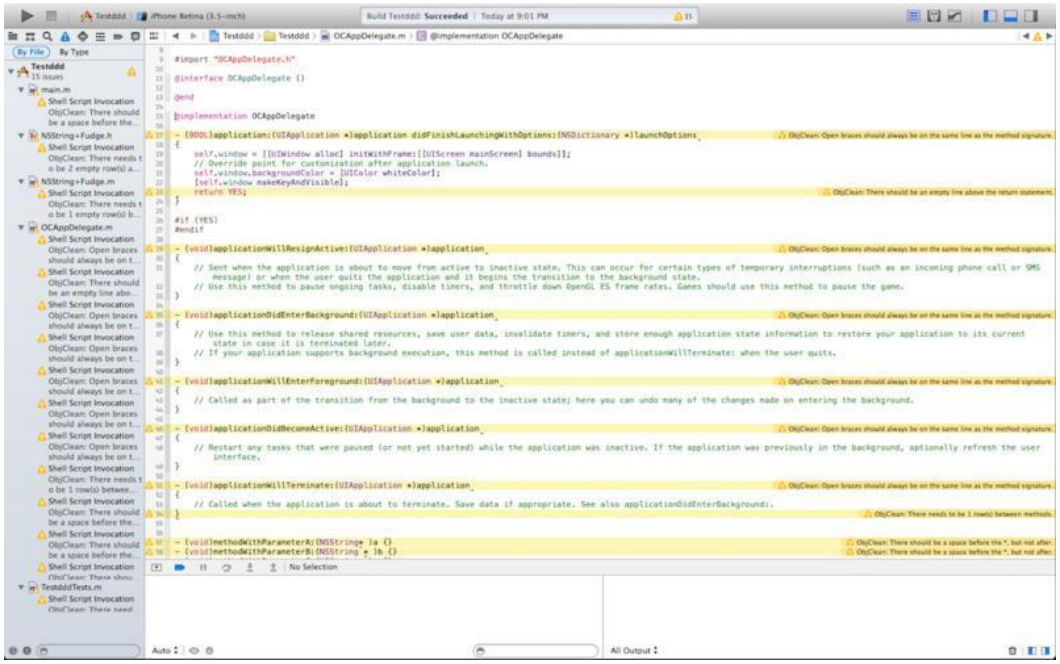


รูปที่ 6 ส่วนต่อประสานโปรแกรมอ็อบเจกทีฟคลีน

2. เลือกไฟล์โปรเจคที่ต้องการตรวจสอบ แล้วกดติดตั้งเครื่องมือ
3. เปิดไฟล์โปรเจคด้วยโปรแกรม Xcode จะพบว่ามีการฝังการทำงานของเครื่องมืออ็อบเจกทีฟคลีน ไว้ในโปรเจคแล้ว

4. กดรันโปรแกรมเพื่อประมวลผลโปรแกรมตามการทำงานปกติ

5. เครื่องมือแสดงผลแจ้งเตือนตำแหน่งคำสั่งในโปรแกรมที่ละเมิดกฎ พร้อมคำอธิบาย ดังรูปที่ 7



รูปที่ 7 การแสดงผลเครื่องมืออ็อบเจกทีฟคลีน

3.4 Clang

เครื่องมือในการวิเคราะห์ชุดคำสั่งในโปรแกรมเพื่อค้นหาข้อผิดพลาดในโปรแกรม ภาษาซี ซีพลัสพลัส และอ็อบเจกทีฟซี โดยจะครอบคลุมการตรวจสอบที่หลากหลาย มีเป้าหมายในการค้นหาข้อบกพร่อง ด้านความปลอดภัยและการใช้งาน API การรค้นหา dead code และข้อผิดพลาดของตรรกะอื่นๆ ยกตัวอย่าง เช่น

1. core.CallAndMessage (C, C++, ObjC)

ตรวจสอบข้อผิดพลาดสำหรับการเรียกใช้ฟังก์ชันและตัวแปรที่ส่งผ่านเข้ามายังฟังก์ชัน เช่น ฟังก์ชันดึงค่าวัตถุในอาร์เรย์ โดยการส่งตำแหน่งของ Index ที่ต้องการเข้าไป แต่ยังไม่มีการให้ค่าเริ่มต้นกับอาร์เรย์

```
// 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)

ตรวจสอบความถูกต้องของตัวแปรที่ที่ถูก Return เมื่อเมท็อดถูกนำไป 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
```

3.5 OCLint

เครื่องมือในการวิเคราะห์ชุดคำสั่งคอมพิวเตอร์เพื่อปรับปรุงคุณภาพและลดจำนวนข้อผิดพลาดในโปรแกรมภาษาซี ซีพลัสพลัสและอ็อบเจกทีฟซี และค้นหาปัญหาที่อาจเกิดขึ้นใน 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 จะมีข้อกำหนดที่เกี่ยวข้องกับเรื่องของการตั้งชื่อ ดังนี้

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

3.6 Uncrustify

เครื่องมือในการเพิ่มความมีระเบียบ สวยงามให้กับชุดคำสั่งคอมพิวเตอร์ในโปรแกรมภาษาซี ซีพลัสพลัส อ็อบเจกทีฟซี จาวา Pawn และ VALA โดยมีเป้าหมายคือ การเพิ่มความสมบูรณ์ให้โปรแกรม สามารทำกาแก้ไขได้ง่าย และชุดคำสั่งมีความระเบียบ 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

3.7 fauxpasapp

เครื่องมือในการค้นหาข้อผิดพลาดในโปรแกรมของระบบปฏิบัติการ โดยทำการแจ้งเตือนจุดบกพร่องที่อาจเกิดขึ้น รวมไปถึงการบำรุงรักษา และรูปแบบของการเขียนโปรแกรม โดยกฎในการตรวจสอบจะแบ่งออกเป็นกลุ่มได้ ดังนี้

- BestPractice
- Resources
- Config
- Localization
- APIUsage
- VCS
- Style
- Pedantic
- Miscellaneous

การตรวจสอบของเครื่องมือ fauxpasapp จะมีข้อกำหนดที่เกี่ยวข้องกับเรื่องของการตั้งชื่อ ดังนี

1. **Unidiomatic accessor naming**

แจ้งเตือนเมื่อเมท็อดในการดึงค่า เริ่มต้นด้วยคำว่า get

2. **Identifier naming**

อนุญาตให้ผู้ใช้สามารถปรับเปลี่ยนรูปแบบการตั้งชื่อต่างๆได้ตามชนิดของตัวแปรโดยใช้ความสามรถของ Regular Expression

4. แนวคิดและวิธีวิจัย

หัวข้อนี้จะอธิบายถึงแนวคิดในการพัฒนาเครื่องมือตรวจสอบข้อปฏิบัติในโปรแกรมอ็อบเจ็กทีฟซี โดยมีขั้นตอนทั้งหมด 4 ขั้นตอน ได้แก่ 1) กำหนดรายการข้อปฏิบัติที่จะทำการตรวจสอบ 2) กำหนดวิธีการที่ใช้ในการตรวจสอบ 3) พัฒนาเครื่องมือและจัดทำให้อยู่ในรูปแบบของคลัง (library) 4) ทดสอบการทำงานของเครื่องมือและประเมินผล 5) แนวคิดการทำงานของเครื่องมือ

4.1 กำหนดรายการข้อปฏิบัติที่จะทำการตรวจสอบ

จากที่ได้กล่าวไว้ในหัวข้อที่ 3.1 โครงงานนี้จะพัฒนาเครื่องมือเพื่อตรวจสอบข้อปฏิบัติการเขียนโปรแกรมโดยจะครอบคลุมในเรื่องของการตั้งชื่อ, การใช้ Magic Number และการใช้ Multiple Literal String เท่านั้น โดยการกำหนดรายการข้อปฏิบัติ มีขั้นตอนดังนี้

1. ทำการรวบรวมข้อปฏิบัติในการเขียนโปรแกรมในภาษาอ็อบเจกทีฟซี โดยมีแหล่งที่มา ดังนี้

- เอกสารข้อแนะนำในการตั้งชื่อของบริษัทแอปเปิล [12]

  ตัวอย่างข้อแนะนำ : รูปแบบพื้นฐานในการตั้งชื่อเป็นแบบ camel-casing เช่น openWebsite

- เอกสารคำแนะนำรูปแบบอ็อบเจกทีฟซี ของบริษัท The New York Times [13]

  ตัวอย่างข้อแนะนำ : ไม่ใช้ตัวอักษรย่อชื่อ เช่น loginBtn() ควรใช้เป็น loginButton()

- เอกสารคำแนะนำรูปแบบอ็อบเจกทีฟซี ของเว็บไซต์ Raywenderlich [14]

  ตัวอย่างข้อแนะนำ : การตั้งชื่อฟังก์ชันไม่ควรใช้ and เป็นตัวเชื่อมระหว่างพารามิเตอร์ที่รับเข้ามา เช่น calculateWithSalary(NSString *)salary andMonth(NSString *)month

- เอกสารคำแนะนำรูปแบบอ็อบเจกทีฟซี ของเว็บไซต์ Google [15]

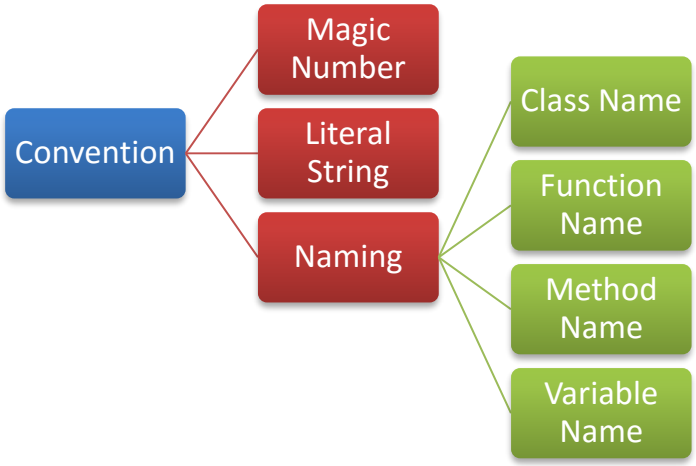
  ตัวอย่างข้อแนะนำ : หากตัวแปรเป็นค่าคงที่ ให้ขึ้นต้นชื่อตัวแปรด้วย ‘k’

- นำเสนอรายการข้อปฏิบัติที่รวบรวมมาให้กับทีมพัฒนาในองค์กรกรณีศึกษาและรวบรวมข้อเสนอเพิ่มเติม

  ตัวอย่างข้อเสนอ : การตั้งชื่อตัวแปรควรต่อท้ายด้วยชนิดของตัวแปร เช่น nameString

2. กำหนดรายการข้อปฏิบัติในการเขียนโปรแกรม ที่จะนำมาใช้ตรวจสอบในเครื่องมือ โดยรายการข้อปฏิบัติที่ใช้ในการตรวจสอบ ดูตัวอย่างข้อปฏบิติจากตารางที่ 3 จะแบ่งเป็นหมวดหมู่ ดังรูปที่ 8 ดังนี้

   1. ตรวจสอบการใช้ Magic Number
   2. ตรวจสอบการใช้ Literal String
   3. ตรวจสอบการตั้งชื่อ แบ่งเป็นหมวดหมู่ย่อย ดังนี้
      - 3.1 ตรวจสอบการตั้งชื่อคลาส
      - 3.2 ตรวจสอบการตั้งชื่อฟังก์ชัน
      - 3.3 ตรวจสอบการตั้งชื่อเมท็อด
      - 3.4 ตรวจสอบการตั้งชื่อตัวแปร



รูปที่ 8 กลุ่มรายการข้อปฏิบัติที่จะทำการตรวจสอบ

ตารางที่ 3 ตัวอย่างข้อปฏิบัติที่จะทำการตรวจสอบ

การตั้งชื่อตัวแปร
1. ชื่อตัวแปรจะต้องมีจำนวนอักขระมากกว่า 3 ตัวอักษรขึ้นไป
2. ชื่อตัวแปรจะต้องประกอบไปด้วยคำมากกว่าหนึ่งคำขึ้นไป
3. ชื่อตัวแปรจะต้องอยู่ในรูปแบบของ Camel Case
4. ต้องมีการระบุชนิดของตัวแปรไว้ด้านท้ายของชื่อตัวแปร
การตั้งชื่อฟังก์ชัน
1. ไม่ใช้เครื่องหมายเว้นวรรค Underscore หรือ Dashes
2. เริ่มต้นชื่อฟังก์ชันด้วยคำกริยา
3. ถ้าฟังก์ชันคืนค่าตัวแปรในคลาส สามารถละเว้นข้อกำหนด 2 และ 5 ได้
4. ถ้าฟังก์ชันคืนค่าที่เกี่ยวข้องในคลาส ให้ขึ้นต้นชื่อด้วย 'get'
5. ถ้าฟังก์ชันคืนค่าชนิด Bool ให้ขึ้นต้นชื่อฟังก์ชันด้วย Inflected Verb เช่น is
การตั้งชื่อคลาส
1.ชื่อคลาสต้องขึ้นต้นด้วยอักษรพิมพ์ใหญ่ ด้านหลังเป็นไปตามรูปแบบ Camel case
2. สามารถขึ้นต้นด้วยคำเติมหน้า (Prefix) เป็นอักษรพิมพ์ใหญ่ทั้งหมด ประกอบไปด้วยอักขระ 2-3 อักขระ
3. ชื่อคลาสจะต้องต่อท้ายด้วยชนิดของคลาส

4.2 กำหนดวิธีการที่ใช้ในการตรวจสอบ ดังตัวอย่างในตารางที่ 4

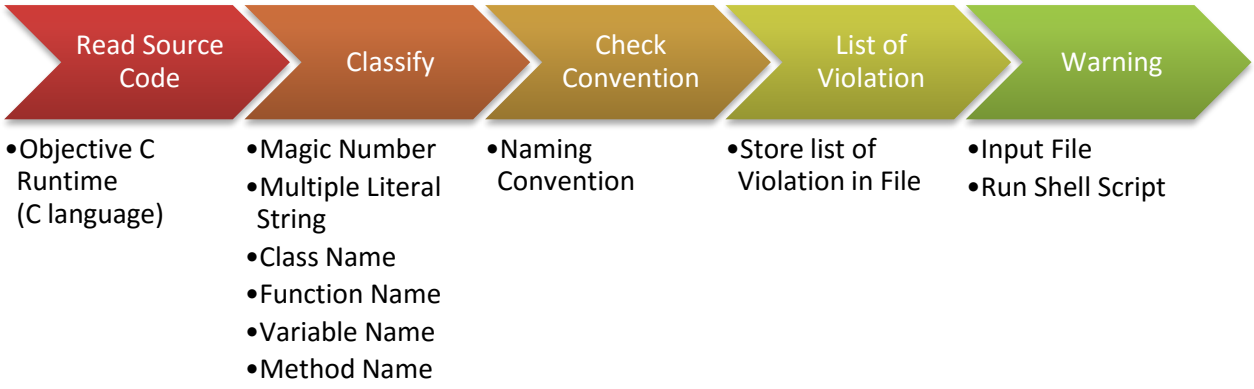
ตารางที่ 4 ตัวอย่างวิธีการตรวจสอบและคำสั่งในโปรแกรม

ข้อปฏิบัติการตั้งชื่อ	**วิธีการตรวจสอบ**	**ตัวอย่างคำสั่งในโปรแกรตม**
1. ชื่อตัวแปรจะต้องมีจำนวนอักขระมากกว่า 3 ตัวอักษรขึ้นไป	1.1 ตรวจสอบจำนวนอักขระของชื่อตัวแปร หากน้อยกว่า 3 อักขระ จะถือว่าละเมิดข้อปฏิบัติ	String *abc; If (abc.count<=3) Print (“ Identifier name must be at least three-character long”)
2. ชื่อตัวแปรจะต้องอยู่ในรูปแบบของ Camel Case	2.1 สร้าง Regular expression ตามรูปแบบ Camel case 2.2 นำ Regular expression แบบ Camel case มาสร้างเป็นตัวแปรชนิด NSRegularExpression 2.3 เรียกใช้ฟังก์ชัน matchesInString() เพื่อทำการเปรียบเทียบชื่อกับ regular expression 2.4 ตรวจสอบผลการเปรียบเทียบ หากไม่ตรงกันแสดงว่าชื่อตัวแปรไม่เป็นไปตามการตั้งชื่อในรูปแบบ Camel case	Pattern *regEx; String *loginButton; regEx = “^[a-z][a-z0-9]*([A-Z][a-z0-9]+|[A-Z]$)*$” if(!regEX matchesInString:loginButton) Print (“Identifier name must be in Camel case Pattern”)
3. ชื่อฟังก์ชันจะต้องขึ้นต้นด้วยคำกริยา	3.1 ทำการตัดคำจากชื่อโดยใช้ตัวอักษรพิมพ์ใหญ่เป็นตัวตรวจสอบ เช่น ฟังก์ชัน loginByGuest() ตัดออกมาเป็นคำได้ ดังนี้ login, By, Guest 3.2 นำคำแรกของชื่อมาตรวจสอบชนิดของคำ โดยใช้ความสามารถของคลาส NSLinguisticTagger เช่น นำเข้าคำว่า login ผลที่ได้คือ verb 3.3 ตรวจสอบผล หากคำแรกของชื่อฟังก์ชันไม่ใช่คำกริยา แสดงว่าชื่อฟังก์ชันนี้ไม่เป็นไปตามข้อปฏิบัติ	String *loginByGuest; Array wordArray = loginByGuest.separateStringbyUpperCase; String *word1 = wordArray[0]; Type *word1Type = word1.type; If (word1Type!=verb) Print (“Function name must begin with berb”)

ทั้งนี้ NSRegularExpression คือคลาสที่ใช้สำหรับอธิบายและประยุกต์ใช้ Regular Expression กับข้อมูลอักขระตามมาตรฐานยูนิโค้ดและ NSLinguisticTagger Class คือคลาสที่ใช้แยกข้อความภาษาธรรมชาติโดยอัตโนมัติ และจะระบุรายละเอียดของส่วนที่แยกได้ เช่น ชนิดของคำ นอกจากนี้คลาสยังสามารถระบุภาษาตัวเขียนและสกัดรากศัพท์ของคำได้

4.3 ขั้นตอนพัฒนาเครื่องมือและจัดทำเครื่องมือให้อยู่รูปแบบของคลัง (library)

เครื่องมือที่พัฒนาจะเริ่มทำงานเมื่อ Xcode ถูกสั่งให้ประมวลผลโปรเจคใดๆ โดยการทำงานของเครื่องมือจะทำงานอยู่เบื้องหลังการทำงานหลัก เครื่องมือจะมีขั้นตอนในการทำงานทั้งหมด 4 ขั้นตอน ได้แก่ 1) ขั้นตอนของการจำแนกกลุ่มข้อมูล 2) ขั้นตอนการตรวจสอบข้อปฏิบัติการเขียนโปรแกรม 3) ชั้นตอนการสร้างไฟล์รายการโปรแกรมที่ละเมิดข้อปฏิบัติ 4) ขั้นตอนการแสดงผลแจ้งเตือนตำแหน่งโปรแกรมที่ละเมิดข้อปฏิบัติ ดังรูปที่ 9



รูปที่ 9 ขั้นตอนการทำงานของเครื่องมือ

1) ขั้นตอนของการจำแนกกลุ่มข้อมูล

เมื่อโปรแกรมถูกประมวล เครื่องมือจะทำการเรียกฟังก์ชันของอ็อบเจกทีฟซีรันไทม์ เพื่อทำการจำแนกกลุ่มข้อมูล ดังนี้

1.1 กลุ่มของ Magic Number

1.2 กลุ่มของ Multiple Literal String

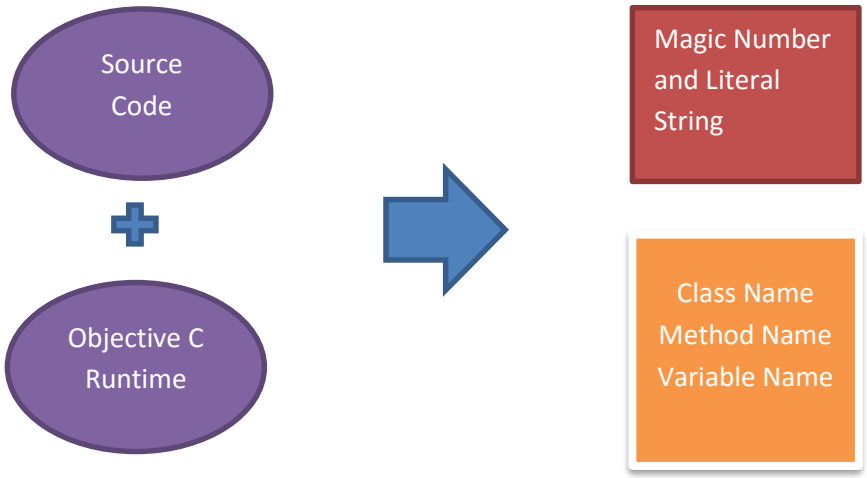
1.3 กลุ่มของชื่อคลาส

1.4 กลุ่มของชื่อฟังก์ชัน

1.5 กลุ่มของเมท็อด

1.6 กลุ่มของชื่อตัวแปร

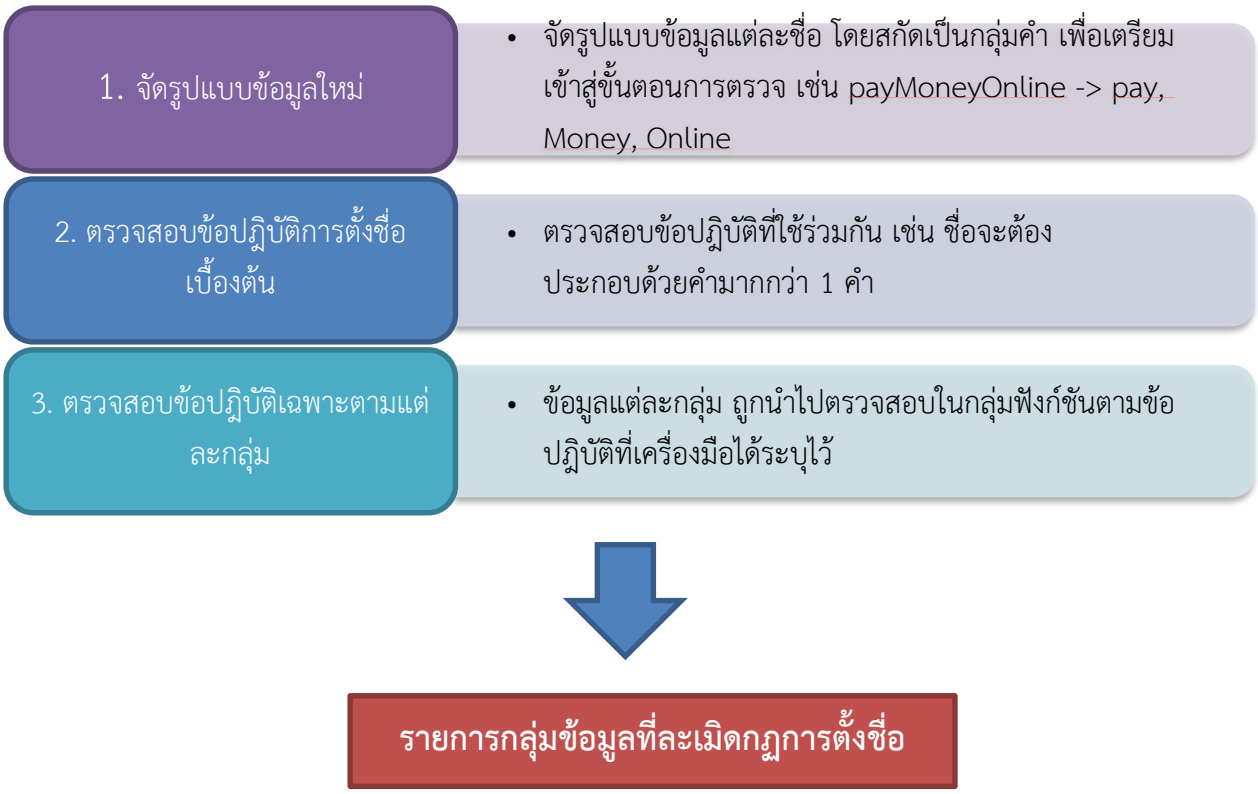
โดยข้อมูลแต่ละกลุ่ม จะถูกจัดเก็บในรูปของ Array เพื่อนำเข้าไปยังขั้นตอนในการตรวจสอบต่อไป ยกเว้นข้อมูลกลุ่ม 1.1 และ 1.2 จะถูกจัดว่าเป็นกลุ่มโปรแกรมที่ละเมิดข้อปฏิบัติแล้ว ดังรูปที่ 10



รูปที่ 10 ขั้นตอนการจำแนกข้อมูล

2) ขั้นตอนการตรวจสอบข้อปฏิบัติการเขียนโปรแกรม

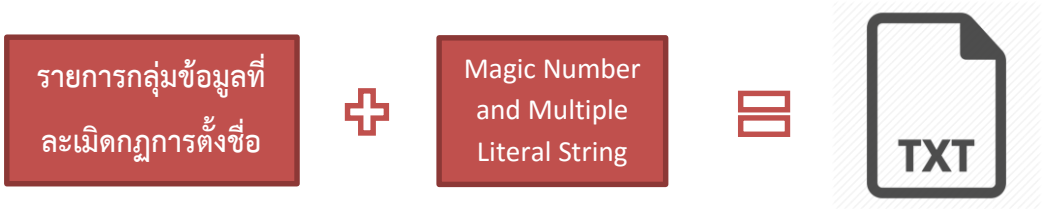
ขั้นตอนนี้จะเป็นการตรวจสอบข้อปฏิบัติที่เกี่ยวกับการตั้งชื่อ โดยข้อมูลนำเข้าได้แก่ กลุ่มของชื่อคลาส กลุ่มของชื่อฟังก์ชัน กลุ่มของชื่อเมท็อดและกลุ่มของชื่อตัวแปร โดยทั้ง 4 กลุ่มจะกระบวนการในการจัดเตรียมข้อมูลและตรวจสอบข้อปฏิบัติเบื้องต้นในการตั้งชื่อก่อน จากนั้นแต่ละกลุ่มจะแยกไปตรวจสอบตามข้อปฏิบัติเฉพาะที่ได้กำหนดไว้ ดังรูปที่ 11



รูปที่ 11 ขั้นตอนการตรวจสอบข้อปฏิบัติการเขียนโปรแกรม

3) ขั้นตอนการสร้างไฟล์รายการโปรแกรมที่ละเมิดข้อปฏิบัติ

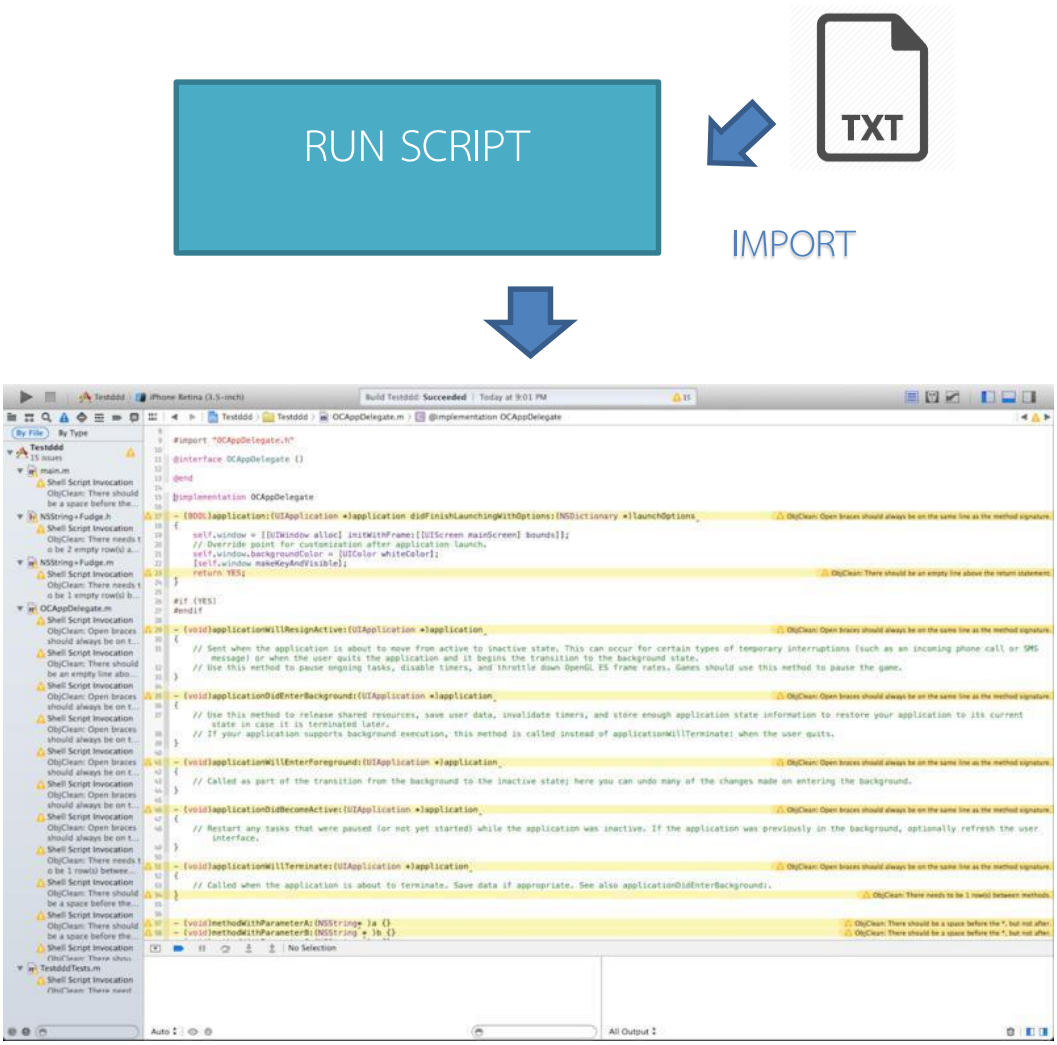
นำรายการกลุ่มข้อมูลที่ละเมิดกฎการตั้งชื่อ กลุ่มข้อมูล Magic Number และกลุ่มข้อมูล Multiple Literal String มาจัดเก็บลงไฟล์ข้อความ เพื่อนำเข้าขั้นตอนในการแสดงผล ดังรูปที่ 12



รูปที่ 12 ขั้นตอนการสร้างไฟล์ข้อความของรายการโปรแกรมที่ละเมิดข้อปฏิบัติ

4) ขั้นตอนการแสดงผลแจ้งเตือนตำแหน่งโปรแกรมที่ละเมิดข้อปฏิบัติ

นำเข้าไฟล์รายการที่ละเมิดกฎผ่านการทำงานของ Run Script และประมวลผลให้โปรแกรมแสดงการแจ้งเตือน ณ ตำแหน่งที่โปรแกรมละเมิดกฎพร้อมคำอธิบาย ดังรูปที่ 13



รูปที่ 13 ขั้นตอนการแสดงผลแจ้งเตือนตำแหน่งโปรแกรมที่ละเมิดข้อปฏิบัติ

4.4 ขั้นตอนทดสอบการทำงานของเครื่องมือและประเมินผล

ขั้นตอนการทดสอบเครื่องมือจะแบ่งการทดสอบในด้านความถูกต้องของเครื่องมือกับความสามารถในการอ่านโปรแกรม

4.4.1 การประเมินความถูกต้องของเครื่องมือ จะทำโดยการนำเข้าโปรแกรมตัวอย่างที่จะทำการตรวจสอบโดยนำเข้าข้อมูลที่ละเมิดข้อปฏิบัติเข้าไปในโปรแกรม จากนั้นใช้เครื่องมือตรวจสอบเพื่อดูว่าเครื่องมือสามารถตรวจหาจุดในโปรแกรมที่ละเมิดข้อปฏิบัติได้ครบถ้วนหรือไม่

4.4.2 การประเมินความสามารถในการอ่านโปรแกรมเวอร์ชันที่ปรับแก้จากคำแนะนำของเครื่องมือแล้วกับโปรแกรมที่ละเมิดข้อปฏิบัติ โดยวัดเวลาที่นักเขียนโปรแกรมใช้ในการหาคำตอบสำหรับข้อคำถามที่กำหนด จากการอ่านโปรแกรมทั้งสองเวอร์ชัน

4.5 แนวคิดการทำงานของเครื่องมือ

เครื่องมือในการตรวจสอบข้อปฏิบัติการเขียนโปรแกรมภาษาอ็อบเจกทีฟซี จะอยู่ในรูปแบบของไฟล์ไลบรารีที่จะต้องนำเข้าในโปรแกรม Xcode และเขียนคำสั่งในการเรียกใช้งาน

1. เปิดโปรเจคงานที่ต้องการตรวจสอบด้วยโปรแกรม Xcode
2. นำเข้าไฟล์ไลบรารีที่ใช้ในการตรวจสอบข้อปฏิบัติการเขียนโปรแกรมภาษาอ็อบเจกทีฟซี
3. เพิ่มคำสั่งในการเรียกใช้งานไลบรารี และกำหนดข้อปฏิบัติที่ต้องการยกเว้นได้
4. กด Run เพื่อสั่งให้โปรแกรมประมวลผล
5. โปรแกรม Xcode แสดงตำแหน่งบรรทัดของชุดคำสั่งที่ไม่เป็นไปตามกฎ ในรูปแบบของการแจ้งเตือนพร้อมคำอธิบาย เพื่อแจ้งให้นักเขียนโปรแกรมแก้ไข

5. วัตถุประสงค์งานวิจัย

เพื่อพัฒนาเครื่องมือในการตรวจสอบข้อปฏิบัติในโปรแกรมอ็อบเจกทีฟซี

6. ขอบเขตงานวิจัย

6.1 เครื่องมือสามารถรองรับชุดคำสั่งโปรแกรมในภาษา อ็อบเจกทีฟซี เท่านั้น

6.2 ข้อปฏิบัติที่ใช้ตรวจสอบจะครอบคลุมเรื่อง Magic Number, Multiple Literal String และการตั้งชื่อเท่านั้น

6.3 ผลลัพธ์ที่ได้หลังจากนำเข้าโปรแกรม เครื่องมือจะแสดงจุดที่โปรแกรมละเมิดกฎและคำอธิบายเท่านั้น

7. ขั้นตอนการดำเนินงาน

7.1 ศึกษางานวิจัยที่เกี่ยวข้องกับมาตรฐานการเขียนโปรแกรม อ็อบเจกทีฟซี และเครื่องมือตรวจสอบการเขียนโปรแกรม

ที่มีอยู่ในปัจจุบัน

7.2 รวบรวมข้อมูลมาตรฐานและข้อปฏิบัติในการเขียนโปรแกรม

7.3 นำเสนอรายการข้อปฏิบัติการเขียนโปรแกรมให้กับทีมพัฒนาโมบายล์แอปพลิเคชันขององค์กรกรณีศึกษา

7.4 กำหนดรายการข้อปฏิบัติในการเขียนโปรแกรมที่จะใช้ตรวจสอบ

7.5 กำหนดวัตถุประสงค์และขอบเขตของโครงงาน

7.6 ศึกษาข้อมูลเชิงเทคนิคและทฤษฎีที่ต้องใช้ในการตรวจสอบโปรแกรมตามรายการข้อปฏิบัติ

7.7 ออกแบบส่วนต่อประสานของเครื่องมือในการตรวจสอบการเขียนโปรแกรม

7.8 พัฒนาเครื่องมือตามที่ได้ออกแบบไว้

7.9 ทดสอบและประเมินผลการทำงานของเครื่องมือ

7.10 จัดทำเอกสารโครงงานมหาบัณฑิตและบทความทางวิชาการ

8. ประโยชน์ที่คาดว่าจะได้รับ

8.1 ได้เครื่องมือตรวจสอบข้อปฏิบัติการเขียนโปรแกรมอ็อบเจกทีฟซี

8.2 ช่วยให้โปรแกรมสามารถอ่านและทำความเข้าใจได้ง่ายยิ่งขึ้น

8.3 เพิ่มความสามารถในการบำรุงรักษาโปรแกรมที่มีอยู่เดิม ซึ่งจะช่วยเพิ่มความสามารถในการบำรุงรักษาโปรแกรม

9. รายการอ้างอิง

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

[2] T. Okubo and H. Tanaka, “Secure software development through coding conventions and frameworks,” Proc. - Second Int. Conf. Availability, Reliab. Secur. ARES 2007, pp. 1042–1051, 2007.

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

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

[5] [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.

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

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

[8] "Objective-C Runtime Programming Guide" [Online]. Available: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ObjCRuntimeGuide/ObjCRuntimeGuide.pdf. [Accessed: /-11-2015].

[9] [1] Y. Wen, X. Tang, L. Ju, and T. Chen, “PeRex : A power efficient FPGA-based architecture for regular expression matching,” 2011.
[10] 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.
[11] "OBJECTIVE-CLEAN" [Online]. Available: http://objclean.com/index.php [Accessed: 5-11-2015]
[12] "Coding Guidelines for Cocoa" [Online]. Available: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CodingGuidelines/CodingGuidelines.pdf [Accessed: 2-11-2015]
[13] "NYTimes Objective-C Style Guide" [Online]. Available: https://github.com/NYTimes/objective-c-style-guide [Accessed: 2-11-2015]
[14] "The official raywenderlich.com Objective-C style guide" [Online]. Available: https://github.com/raywenderlich/objective-c-style-guide [Accessed: 2-11-2015]
[15] "Google Objective-C Style Guide" [Online]. Available: https://google.github.io/styleguide/objcguide.xml [Accessed: 2-11-2015]