Lab 04 MVC, JavaFX & Maven

1. วิธีการสร้าง JavaFX Project เพื่อทำ Desktop Application

1. สร้าง Project ใหม่ ใน Intellij IDEA เลือกเป็น JavaFX > JavaFX Application
   เลือก Project SDK: เป็น 1.8 version 1.8.0_271
   กดปุ่ม Next
2. กำหนด Project location และชื่อ Project แล้วกดปุ่ม Finish
3. สังเกต Project Panel จะเห็นโครงสร้างไฟล์ดังนี้

```
/your_project_name/
|---- .idea/
|---- src/                    (Source Root สัญลักษณ์สีฟ้า)
    |---- sample/             (Package)
        |---- Controller      (Controller Class)
        |---- Main            (Main Class)
        |---- sample.fxml     (View)
```

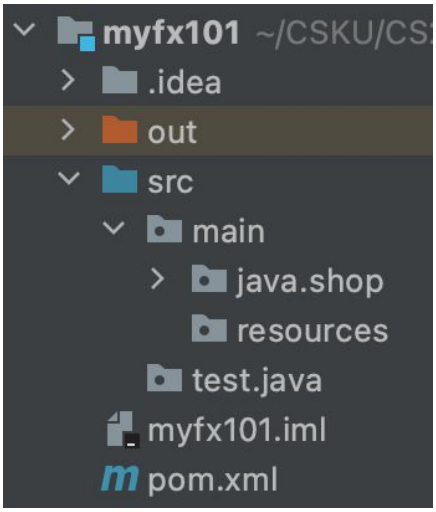
4. เปลี่ยนชื่อ package sample เป็นชื่อที่สอดคล้องกับ Project เช่น shop
   (ทุกการเปลี่ยนชื่อ ให้คลิกขวาที่ไฟล์ที่ต้องการเปลี่ยนชื่อ แล้วเลือก Refactor > Rename...
   เพื่อให้โค้ดส่วนอื่นที่อ้างอิงถึงชื่อเดิม เปลี่ยนเป็นชื่อใหม่ให้อัตโนมัติ)
5. เพิ่ม package ใน shop ชื่อว่า models และ controllers (ชื่อ package ใช้ตัวเล็กทั้งหมด)
   ตอนที่สร้าง จะใส่ข้อมูลว่า shop.models และ shop.controllers
6. สังเกตใน Main Class มีคำสั่งในการ load View sample.fxml มาใส่ primaryStage
   ลองรัน Main Class จะเห็นโปรแกรมขนาด 300 x 275 px ที่มี Title ว่า Hello World
7. เปลี่ยนชื่อ sample.fxml เป็น home.fxml
   สังเกตใน Main Class คำสั่งในการ load view เปลี่ยนชื่อจาก sample.fxml เป็น home.fxml
   ลองรันอีกครั้ง ได้ผลลัพธ์เดิม
8. กำหนด Title เป็นรหัสนิสิตของตนเอง และกำหนดขนาดหน้าจอโปรแกรมเป็น 1280 x 1024 px
   แล้วรันอีกครั้ง

2. วิธีการเพิ่ม Maven Framework ไปใน JavaFX Project

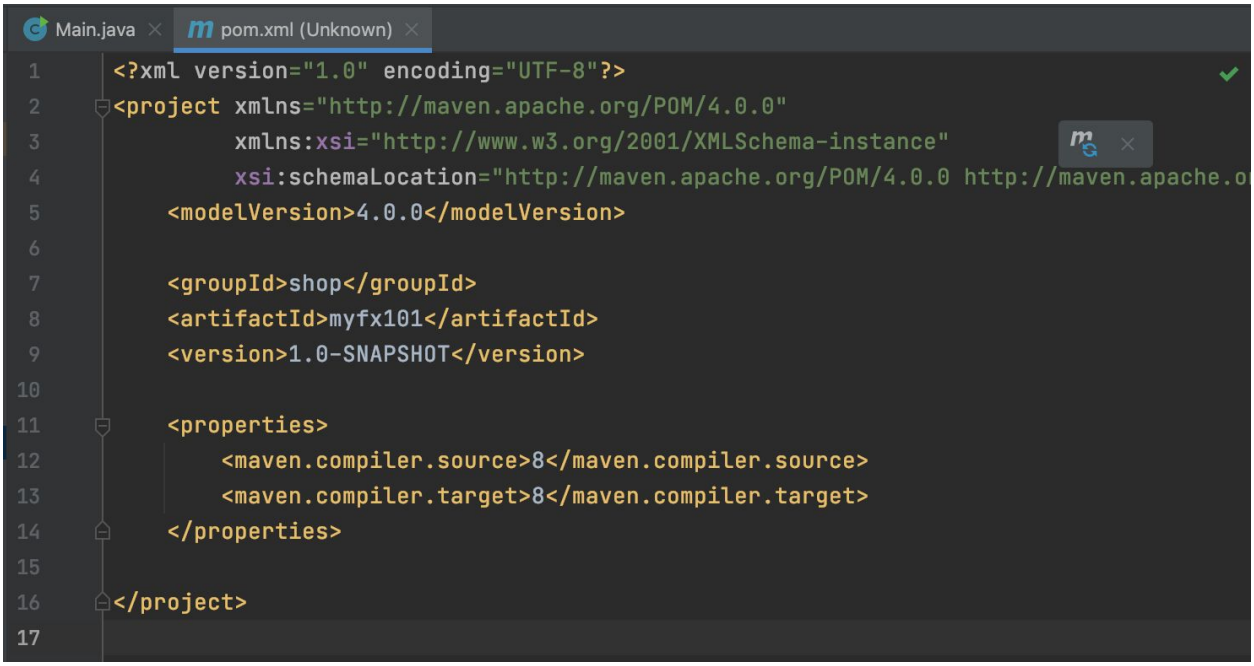
Maven framework ใช้จัดการ library ต่าง ๆ จากภายนอกโปรเจค มาใช้ในโปรเจค และช่วยในการ build jar file

9. ที่ Project panel คลิกขวาที่ชื่อโปรเจค แล้วเลือก Add Framework Support…
   เลื่อนหา Maven เลือก แล้วกดปุ่ม OK

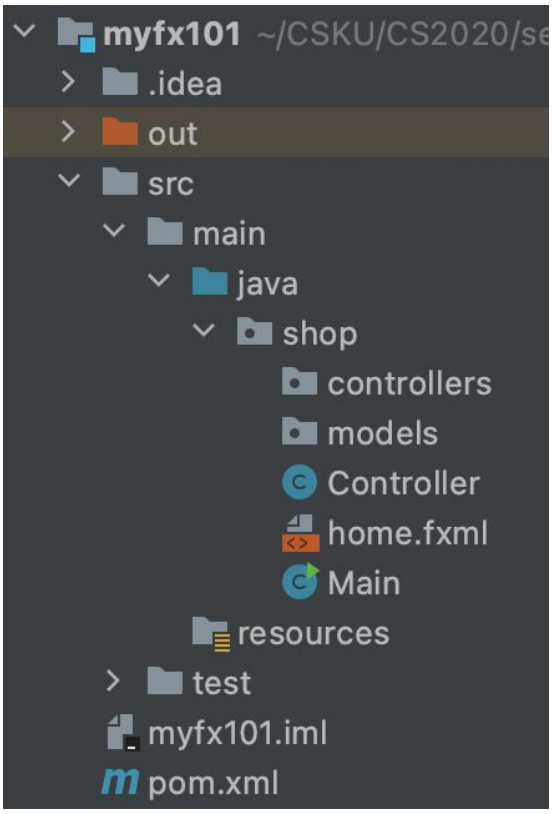
10. สังเกตว่า มีไฟล์ pom.xml และการจัดการ package ใน folder src เปลี่ยนแปลงไปจากเดิม



11. แก้ไขไฟล์ pom.xml โดยเปลี่ยน groupId เป็น ชื่อ package เดิม ก่อนเพิ่ม maven (shop)
จากนั้นคลิกขวาที่ไฟล์ pom.xml เลือก Maven > Reload project (หรือกด icon ตัว m ที่มีรูป reload)



12. สังเกตที่ Project Panel โครงสร้างไฟล์มีการเปลี่ยนแปลงไปเล็กน้อย



- Source Root (สัญลักษณ์โฟลเดอร์สีฟ้า)
  เปลี่ยนจากที่ src/ ไปอยู่ที่ src/main/java/

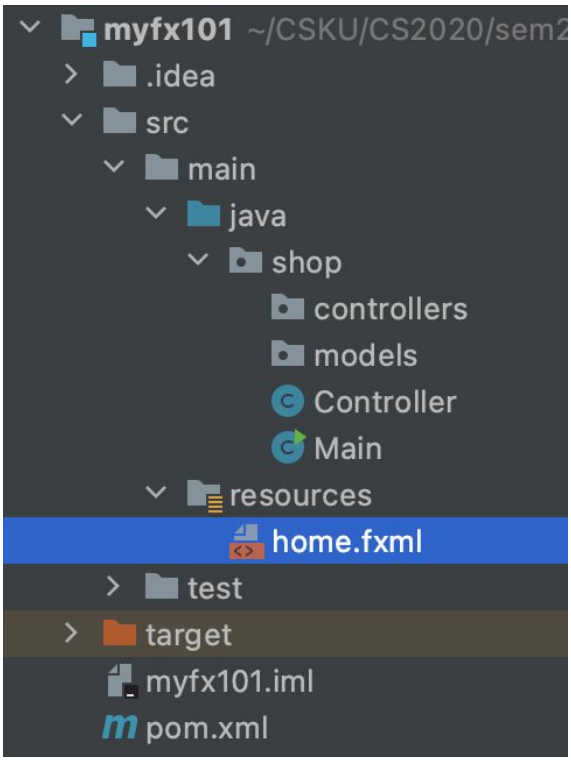
- มี folder src/main/resources/ เพิ่มขึ้นมา
  สำหรับเก็บ view

- มี folder src/test/ เพิ่มขึ้นมา สำหรับเก็บ unit test

13. ลองรันอีกครั้ง จะพบปัญหา InvocationTargetException

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_271.jdk/Contents/Home/bin/java ...
Exception in Application start method
java.lang.reflect.InvocationTargetException <4 internal calls>
    at com.sun.javafx.application.LauncherImpl.launchApplicationWithArgs(LauncherImpl.java:389)
    at com.sun.javafx.application.LauncherImpl.launchApplication(LauncherImpl.java:328) <4 internal calls>
    at sun.launcher.LauncherHelper$FXHelper.main(LauncherHelper.java:873)
Caused by: java.lang.RuntimeException Create breakpoint : Exception in Application start method
    at com.sun.javafx.application.LauncherImpl.launchApplication1(LauncherImpl.java:917)
    at com.sun.javafx.application.LauncherImpl.lambda$launchApplication$1(LauncherImpl.java:182) <1 internal call>
Caused by: java.lang.NullPointerException Create breakpoint : Location is required.
```

14. ให้ย้ายไฟล์ .fxml ทั้งหมด ไปอยู่ใน folder src/main/resources/ โดยการลากวางแล้วกด Refactor



- สังเกตว่า หลังจากรันโปรแกรม
  folder out/ จะหายไป
  และมี folder target/ เพิ่มขึ้นมา

15. แก้ไข Main Class การอ้างอิงไฟล์ fxml จาก resources จะต้องอ้างจาก path /

```
package ku.cs.classroom;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("/home.fxml"));
        primaryStage.setTitle("Student ID");
        primaryStage.setScene(new Scene(root, 1280, 1024));
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
```

```
    }
}
```

16. ลองรัน Main class อีกครั้ง จะต้องได้หน้าต่างของ Application

3. การ build JavaFX Project ที่ใช้ Maven Framework

17. แก้ไข pom.xml ดังนี้
    a. เพิ่ม properties เพื่อกำหนดค่าต่าง ๆ ดังนี้
        `out.folder` คือชื่อ folder ที่บรรจุ jar file ที่ build เสร็จแล้ว
        `out.filename` คือ ชื่อของ jar file ที่ build เสร็จแล้ว
        `project.mainClass` คือ Main Class ของโปรเจค ต้องระบุ package ด้วย
        properties ใน pom.xml จะเป็นเหมือนตัวแปรที่ใช้ในไฟล์ pom.xml
        สามารถเพิ่ม properties ชื่ออื่นได้

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>shop</groupId>
    <artifactId>myfx101</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <out.folder>62104xxxxx</out.folder>
        <out.filename>62104xxxxx-jar</out.filename>
        <project.mainClass>shop.Main</project.mainClass>
    </properties>
</project>
```

    b. เพิ่ม plugin ที่ช่วยเหลือในการ build jar file ดังนี้

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
    <modelVersion>4.0.0</modelVersion>

    <groupId>shop</groupId>
    <artifactId>myfx101</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <out.folder>62104xxxxx</out.folder>
        <out.filename>62104xxxxx-jar</out.filename>
        <project.mainClass>shop.Main</project.mainClass>
    </properties>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>3.2.0</version>
                <configuration>
                    <archive>
                        <manifest>
                            <addClasspath>true</addClasspath>
                            <classpathPrefix>lib/</classpathPrefix>
                            <mainClass>${project.mainClass}</mainClass>
                        </manifest>
                        <manifestEntries>
                            <Class-Path>.</Class-Path>
                        </manifestEntries>
                    </archive>
                    <finalName>${out.folder}/${out.filename}</finalName>
                </configuration>
            </plugin>

            <plugin>
                <artifactId>maven-resources-plugin</artifactId>
                <version>3.2.0</version>
                <executions>
                    <execution>
                        <id>copy-resources</id>
                        <phase>validate</phase>
                        <goals>
                            <goal>copy-resources</goal>
                        </goals>
                        <configuration>
```

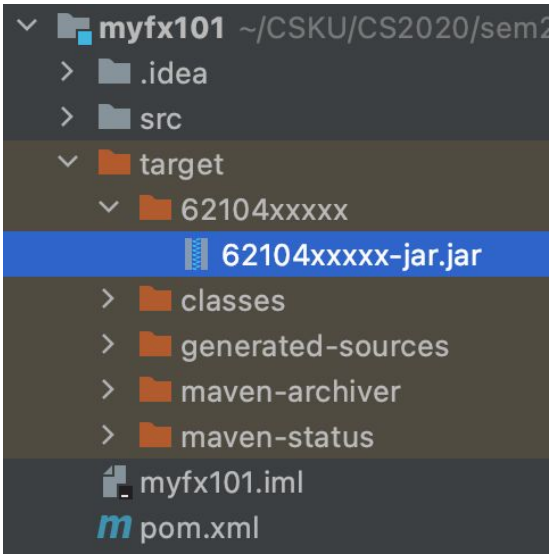
```
<outputDirectory>${basedir}/target/${out.folder}</outputDirectory>
                            <resources>
                                <resource>
                                    <directory>resources</directory>
                                    <filtering>true</filtering>
                                </resource>
                            </resources>
                        </configuration>
                    </execution>
                </executions>
            </plugin>

            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-dependency-plugin</artifactId>
                <executions>
                    <execution>
                        <id>copy-dependencies</id>
                        <phase>prepare-package</phase>
                        <goals>
                            <goal>copy-dependencies</goal>
                        </goals>
                        <configuration>

<outputDirectory>${project.build.directory}/${out.folder}/lib</outputDirectory>
                            <overWriteReleases>false</overWriteReleases>
                            <overWriteSnapshots>false</overWriteSnapshots>
                            <overWriteIfNewer>true</overWriteIfNewer>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>

</project>
```

maven-jar-plugin ช่วยในการสร้าง jar file
maven-resource-plugin ช่วย copy resource (เช่น fxml, static image, static file) ไปไว้ใน jar file
maven-dependency-plugin ช่วย copy library ภายนอก ฟ่วงไปกับ jar file
(โดยจะมี folder lib อยู่ข้าง jar file ที่ build แล้ว)

c. reload maven project

18. เปิด panel Maven (ทางขวาของ Intellij IDEA) แล้วเลือก Lifecycle > install จากนั้นกดปุ่ม Run Maven Build (สามเหลี่ยมสีเขียว)

จะได้ jar file ที่ build เสร็จแล้ว ใน folder target



ลอง double click jar file นอก Intellij IDEA จะได้โปรแกรมที่เขียน
และลองย้าย jar file ไปไว้ที่ folder อื่น แล้ว double click จะต้องได้โปรแกรมที่เขียนเช่นกัน

4. MVC Architecture

1. สร้าง class MemberCard ใน package shop.models

```
package shop.models;

public class MemberCard {

    private String name;
    private String phone;
    private double cumulativePurchase;
    private int stamp;

    public MemberCard(String name, String phone, int stamp) {
        this.name = name;
        this.phone = phone;
        this.stamp = stamp;
        cumulativePurchase = 0;
    }

    public String getName() {
        return name;
    }

    public String getPhone() {
        return phone;
    }

    public double getCumulativePurchase() {
        return cumulativePurchase;
    }

    public int getStamp() {
        return stamp;
    }


    public void addPurchase(double purchase) {
        cumulativePurchase += purchase;
        stamp += purchase / 50;
    }

    public boolean useStamp(int stamp) {
        if (this.stamp >= stamp) {
            this.stamp -= stamp;
            return true;
        }
        return false;
    }
```

```
    public MemberCard(String name, String phone) {
        this(name, phone, 0);
    }

}
```

2. สร้าง class MemberCardDetailController ใน package shop.controllers

```
package shop.controllers;

import javafx.fxml.FXML;
import shop.models.MemberCard;

public class MemberCardDetailController {

    // controller เชื่อมต่อกับ model เพื่อเก็บข้อมูลและเรียกข้อมูลมาแสดงผลที่ view
    private MemberCard memberCard;

    @FXML
    public void initialize() {
        // initialize จะถูกเรียกให้ทำงานเมื่อมีการ load Controller นี้
        System.out.println("initialize MemberCardDetailController");
        memberCard = new MemberCard("John Smith", "081-222-8888");
    }
}
```

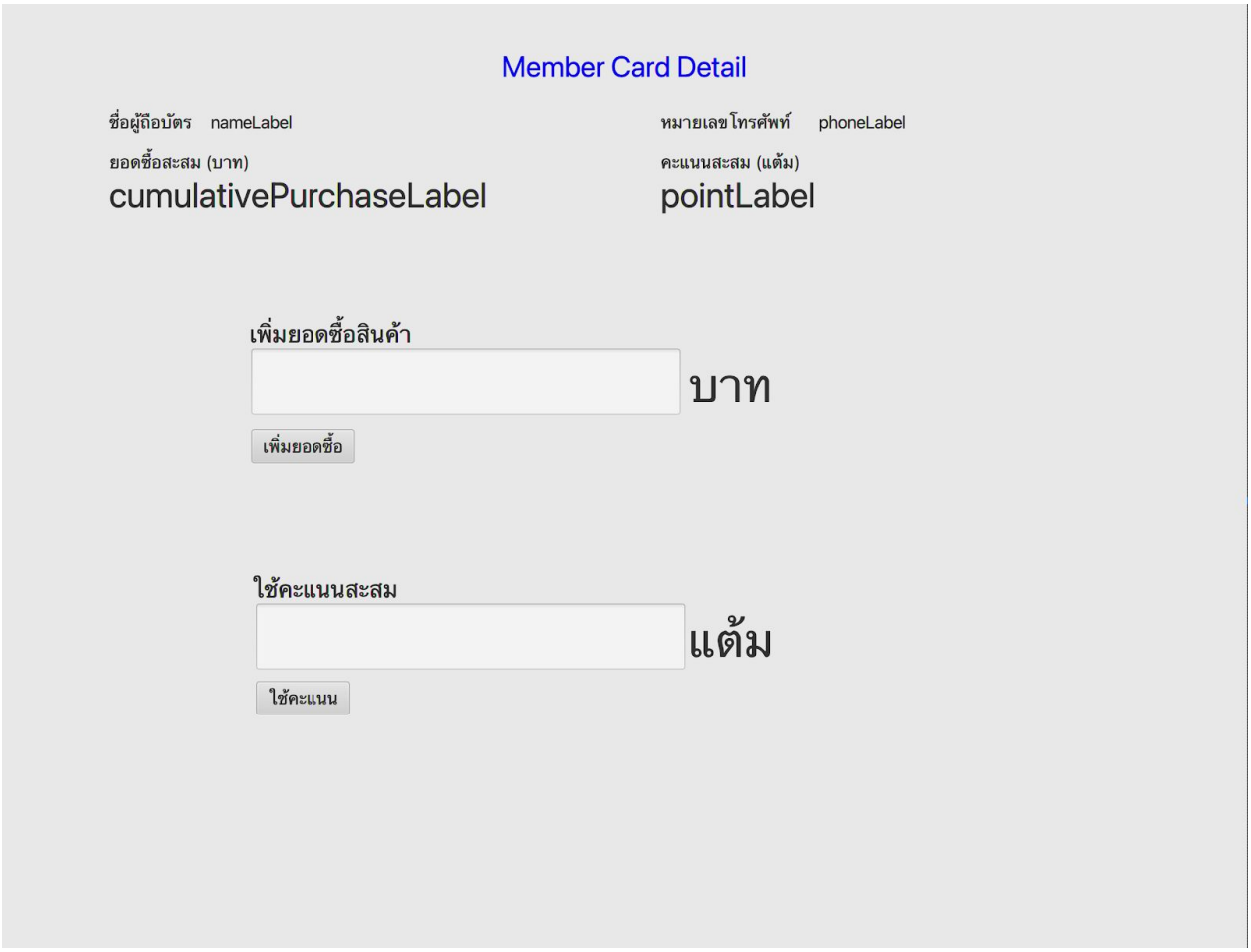
3. สร้างไฟล์ member_card_detail.fxml ใน folder src/resources/ แก้ไข fx:controller ให้ไปอ้างอิงคลาส MemberCardDetailController โดยระบุ package ด้วย

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<AnchorPane xmlns="http://javafx.com/javafx"
            xmlns:fx="http://javafx.com/fxml"
            fx:controller="shop.controllers.MemberCardDetailController"
            prefHeight="1024" prefWidth="1280">

</AnchorPane>
```

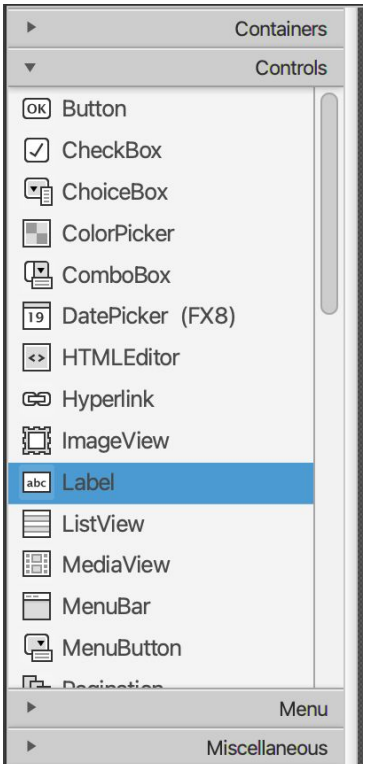
4. download program Scene Builder for Java 8 https://gluonhq.com/products/scene-builder/

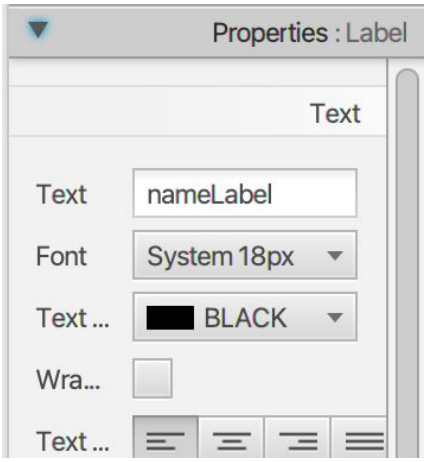
5. https://www.jetbrains.com/help/idea/opening-fxml-files-in-javafx-scene-builder.html

6. ใช้ Scene Builder เลือกไฟล์ view ที่ชื่อ member_card_detail.fxml ออกแบบหน้า UI ให้มีลักษณะนี้

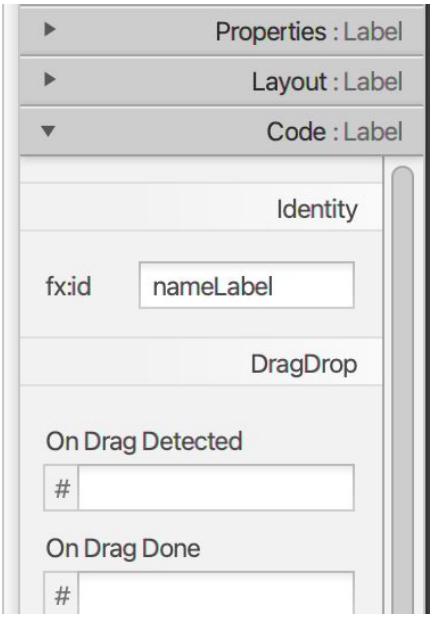


7. ส่วนของข้อความใน view ใช้ Controls > Label (เลือกจากซ้ายบนของ Scene Builder) กำหนดขนาดข้อความจาก Properties (เลือกจากทางขวาของ Scene Builder)

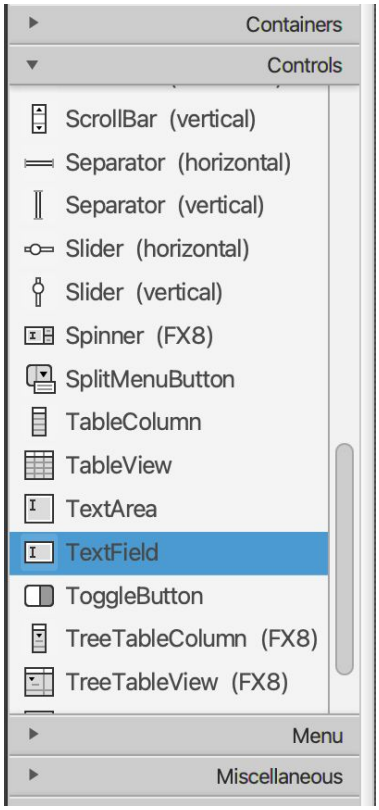




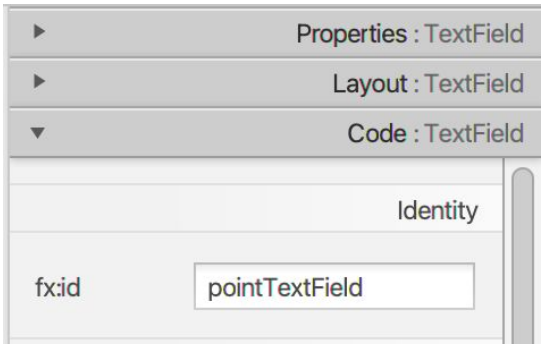
กำหนด identify fx:id (เลือกจากขวามือ) ให้กับ Label ที่มีข้อความ nameLabel, phoneLabel, cumulativePurchaseLabel, pointLabel โดยกำหนดชื่อ fx:id ให้ตรงกับข้อความ (คลิกเลือก Label ของข้อความนั้นก่อน แล้วค่อยกำหนดชื่อ) Label ข้อความอื่น ไม่ต้องกำหนด fx:id



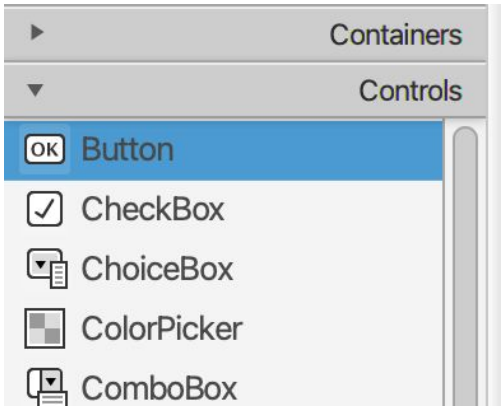
8. ส่วนของกล่องใส่ข้อความใน view ใช้ Controls > TextField
แนะนำให้กำหนดความใหญ่ของกล่องข้อความจาก Properties > Font



กำหนด fx:id ของแต่ละ TextField เป็น purchaseTextField และ pointTextField ตามลำดับ



9. ส่วนของปุ่มใน view ใช้ Controls > Button

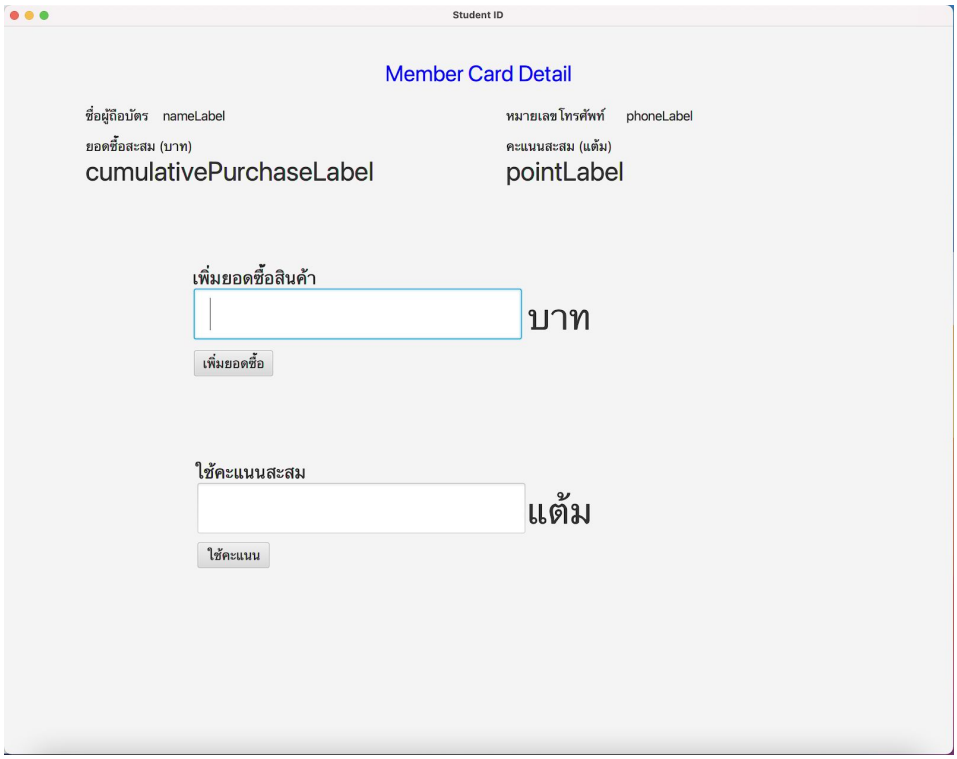


ไม่ต้องกำหนด fx:id ของปุ่ม

10. save file ที่แก้ไขใน SceneBuilder

11. แก้ไข Main Class ให้ load view member_card_detail.fxml
ทดลองรัน Main Class ควรจะได้หน้าโปรแกรม

12. แก้ไข Main Class ให้ load view member_card_detail.fxml
ทดลองรัน Main Class ควรจะได้หน้าโปรแกรม



13. แก้ไข MemberCardDetailController ให้เชื่อมข้อมูลจาก Model MemberCard มาแสดงผลที่ view

```
package shop.controllers;

import javafx.fxml.FXML;
import javafx.scene.control.Label;
import shop.models.MemberCard;

public class MemberCardDetailController {

    // ระบุชื่อ field ตามชื่อ fx:id และใส่ @FXML ไว้ด้านหน้าทุก field ที่เชื่อมกับ view
    // ตอนพิมพ์ ให้เลือก import จาก javafx.scene.xxxx
    @FXML private Label nameLabel;
    @FXML private Label phoneLabel;
    @FXML private Label cumulativePurchaseLabel;
    @FXML private Label pointLabel;

    // controller เชื่อมต่อกับ model เพื่อเก็บข้อมูลและเรียกข้อมูลมาแสดงผลที่ view
    private MemberCard memberCard;

    @FXML
    public void initialize() {
        // initialize จะถูกเรียกให้ทำงานเมื่อมีการ load Controller นี้
        System.out.println("initialize MemberCardDetailController");
        memberCard = new MemberCard("John Smith", "081-222-8888");
        showMemberCardData();
    }

    private void showMemberCardData() {
        nameLabel.setText(memberCard.getName());
        phoneLabel.setText(memberCard.getPhone());
        String cumulativePurchase =
                           String.format("%.2f", memberCard.getCumulativePurchase());
        cumulativePurchaseLabel.setText(cumulativePurchase);
        String point = "" + memberCard.getStamp();
        pointLabel.setText(point);
    }
}
```

ลองรันโปรแกรมที่ Main Class จะเห็นว่า ข้อมูลใน MemberCard แสดงผลได้ถูกต้อง

14. แก้ไข MemberCardDetailController ให้รับข้อมูลจาก view ในส่วนการเพิ่มยอดซื้อสะสม เพื่อมาเปลี่ยนสถานะของ Model MemberCard และแสดงผล

```
package shop.controllers;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import shop.models.MemberCard;

public class MemberCardDetailController {

    // ระบุชื่อ field ตามชื่อ fx:id และใส่ @FXML ไว้ด้านหน้าทุก field ที่เชื่อมกับ view
    // ตอนพิมพ์ ให้เลือก import จาก javafx.scene.xxxx
    @FXML private Label nameLabel;
    @FXML private Label phoneLabel;
    @FXML private Label cumulativePurchaseLabel;
    @FXML private Label pointLabel;
    @FXML private TextField purchaseTextField;

    // controller เชื่อมต่อกับ model เพื่อเก็บข้อมูลและเรียกข้อมูลมาแสดงผลที่ view
    private MemberCard memberCard;

    @FXML
    public void initialize() {
        // initialize จะถูกเรียกให้ทำงานเมื่อมีการ load Controller นี้
        System.out.println("initialize MemberCardDetailController");
        memberCard = new MemberCard("John Smith", "081-222-8888");
        showMemberCardData();
    }

    @FXML
    public void handleAddPurchaseButton(ActionEvent actionEvent) {
        // รับข้อมูลจาก TextField ข้อมูลที่รับเป็น String เสมอ
        String purchaseString = purchaseTextField.getText();
        // แปลงชนิดข้อมูล String เป็น double ด้วย Double.parseDouble()
        double purchase = Double.parseDouble(purchaseString);
        // เรียกการคำนวณต่าง ๆ จาก model
        memberCard.addPurchase(purchase);
        // แสดงผลข้อมูล
        showMemberCardData();
        // clear ช่อง TextField
        purchaseTextField.clear();
    }

    private void showMemberCardData() {
        nameLabel.setText(memberCard.getName());
```

```
        phoneLabel.setText(memberCard.getPhone());
        String cumulativePurchase =
                            String.format("%.2f", memberCard.getCumulativePurchase());
        cumulativePurchaseLabel.setText(cumulativePurchase);
        String point = "" + memberCard.getStamp();
        pointLabel.setText(point);
    }
}
```

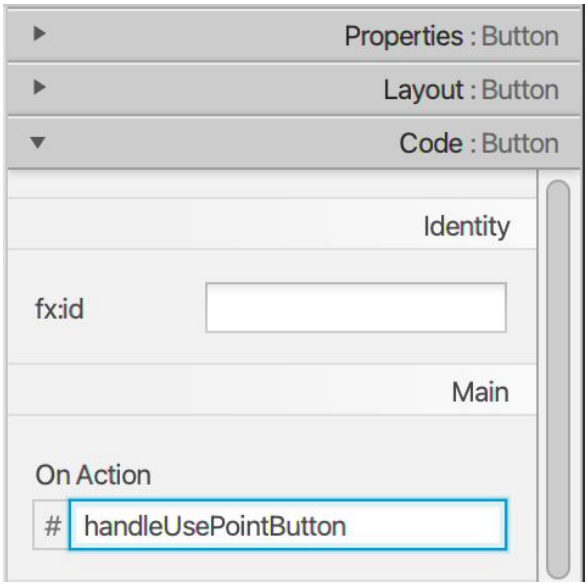
15. แก้ไข view member_card_detail.fxml โดยคลิกเลือกปุ่มเพิ่มยอดซื้อ
    จากนั้นเพิ่ม Code: > On Action เป็นชื่อ method handleAddPurchaseButton
16. รัน Main Class
    โปรแกรมควรจะรับค่า เพิ่มยอดซื้อสินค้า ได้

17. แก้ไข MemberCardDetailController ให้รับข้อมูลจาก view ในส่วนการใช้คะแนนสะสม
    เพื่อมาเปลี่ยนสถานะของ Model MemberCard และแสดงผล
    ทำคล้ายกับข้อ 14

```
package shop.controllers;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import shop.models.MemberCard;

public class MemberCardDetailController {

    // ระบุชื่อ field ตามชื่อ fx:id และใส่ @FXML ไว้ด้านหน้าทุก field ที่เชื่อมกับ view
    // ตอนพิมพ์ ให้เลือก import จาก javafx.scene.xxxx
    @FXML private Label nameLabel;
    @FXML private Label phoneLabel;
    @FXML private Label cumulativePurchaseLabel;
    @FXML private Label pointLabel;
    @FXML private TextField purchaseTextField;
    @FXML private TextField pointTextField;

    // controller เชื่อมต่อกับ model เพื่อเก็บข้อมูลและเรียกข้อมูลมาแสดงผลที่ view
    private MemberCard memberCard;

    @FXML
    public void initialize() {
        // initialize จะถูกเรียกให้ทำงานเมื่อมีการ load Controller นี้
        System.out.println("initialize MemberCardDetailController");
        memberCard = new MemberCard("John Smith", "081-222-8888");
        showMemberCardData();
```

```
    }

    // method ที่จะให้ปุ่มทำงาน จะต้องมี @FXML นำหน้า
    // ActionEvent เลือก import จาก javafx.event.ActionEvent
    @FXML
    public void handleAddPurchaseButton(ActionEvent actionEvent) {
        // รับข้อมูลจาก TextField ข้อมูลที่รับเป็น String เสมอ
        String purchaseString = purchaseTextField.getText();
        // แปลงชนิดข้อมูล String เป็น double ด้วย Double.parseDouble()
        double purchase = Double.parseDouble(purchaseString);
        // เรียกการคำนวณต่าง ๆ จาก model
        memberCard.addPurchase(purchase);
        // แสดงผลข้อมูล
        showMemberCardData();
        // clear ช่อง TextField
        purchaseTextField.clear();
    }

    @FXML
    public void handleUsePointButton(ActionEvent actionEvent) {
        String pointString = pointTextField.getText();
        // แปลงชนิดข้อมูล String เป็น int ด้วย Integer.parseInt()
        int point = Integer.parseInt(pointString);
        memberCard.useStamp(point);
        showMemberCardData();
        pointTextField.clear();
    }

    private void showMemberCardData() {
        nameLabel.setText(memberCard.getName());
        phoneLabel.setText(memberCard.getPhone());
        String cumulativePurchase = String.format("%.2f",
memberCard.getCumulativePurchase());
        cumulativePurchaseLabel.setText(cumulativePurchase);
        String point = "" + memberCard.getStamp();
        pointLabel.setText(point);
    }
}
```

18. แก้ไข view member_card_detail.fxml โดยคลิกเลือกปุ่มใช้คะแนน
    จากนั้นเพิ่ม Code: > On Action เป็นชื่อ method handleUsePointButton



19. รัน Main Class
    โปรแกรมควรจะรับค่า ใช้คะแนนสะสม ได้

20. Build jar file ผ่าน menu Maven (Lifecycle > Install)

21. ทดลอง double click jar file ที่ build ได้