Code Review of The Software Project: Minimalistic Sketch Software

Course Title: Software Development Project
Course NO: CSE- 3106

Project By:

Sourav Shome
Student ID: 210217
Mst. Eshrat Jahan Esha
Student ID: 210233
Puspita Chakma
Student ID: 210240

Reviewed By:

Md. Mursalin Biswas
Student ID: 210209
Md. Shuhan Hossen
Student ID: 210237

Submitted To:

Amit Kumar Mondal
Assistant Professor
Computer Science & Engineering Discipline
Khulna University, Khulna.

Introduction

Code reviews mean that every change effectively has shared authorship. The review identifies areas for improvement, adherence to best practices, and suggestions for enhancing maintainability, security, and overall code quality. In this code review, bad smells of the code, architecture evaluation, modularity check, condition statements of the code & other related sections are evaluated.

Code Smells

1. Large or complex methods:

Large or complex methods are not seen in this project. All the function that are used in this project are Readable and simple. In gui.java class the constructor is too big(62 lines) that could be separated by Button_Creation() and Logic_Creation() method .

2. Long parameter lists:

No long parameters are used in this project .

3. Excessive comments:

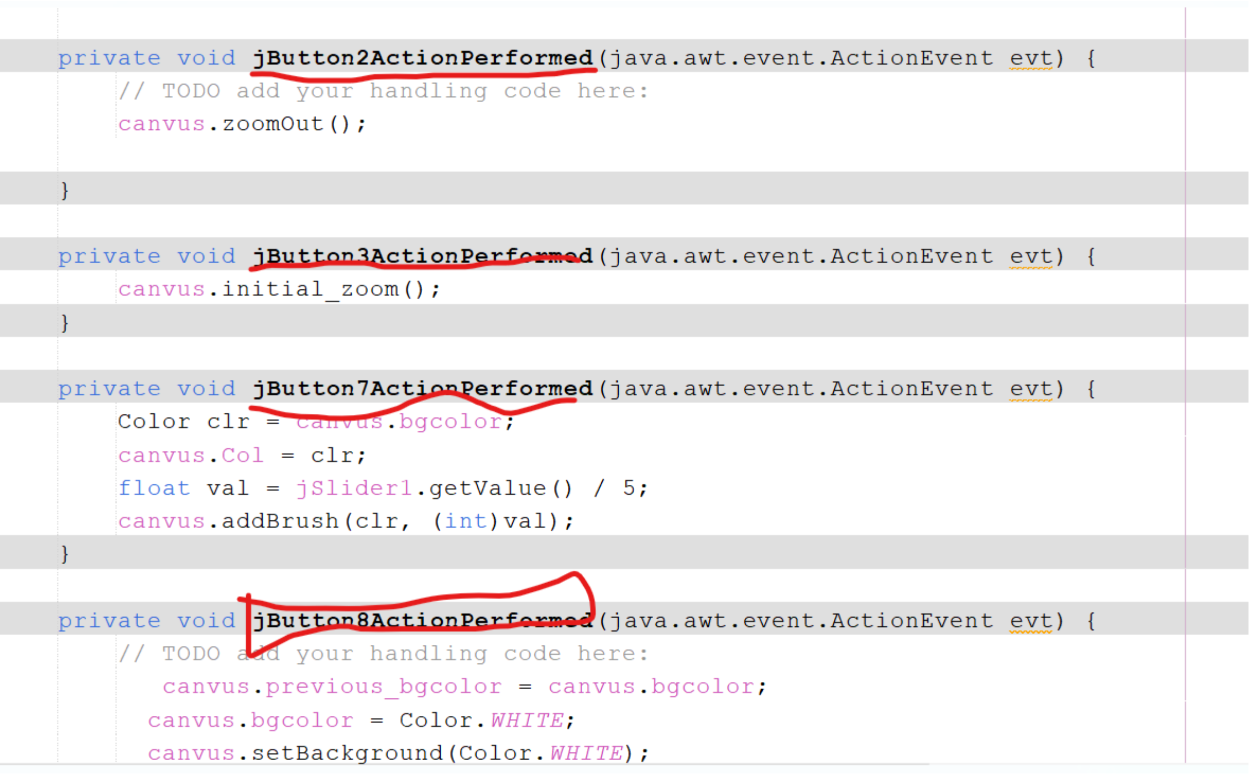
Many unnecessary comments are seen. unnecessary comment in brush.java class in **getColor()** and **setSize()** method . Necessary comments are kept before initializing excepts methods. comments are needed before creating menuBar and new components in GUI1 constructors.

4. Duplicate code:

No duplicate code are not used in code . So Code reusibility has been done quite effectively in most of the modules.

5. Inconsistent naming conventions:

Inconsistent naming conventions are seen in panelMenu.java for zoom in , zoomout , reset and eraser button .In the code there variables name are JButton1,JButton2,JButton3,JButton4 . Also confusing for undo ,redo and clear button variable name.



```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    canvus.zoomOut();

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    canvus.initial_zoom();
}

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    Color clr = canvus.bgcolor;
    canvus.Col = clr;
    float val = jSlider1.getValue() / 5;
    canvus.addBrush(clr, (int)val);
}

private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
       canvus.previous_bgcolor = canvus.bgcolor;
      canvus.bgcolor = Color.WHITE;
      canvus.setBackground(Color.WHITE);
```

6. Incomplete error handling:

Most of the cases errors are handled.As a result codes are easy to maintain and test and radiable .

7. Too many if/else statements:

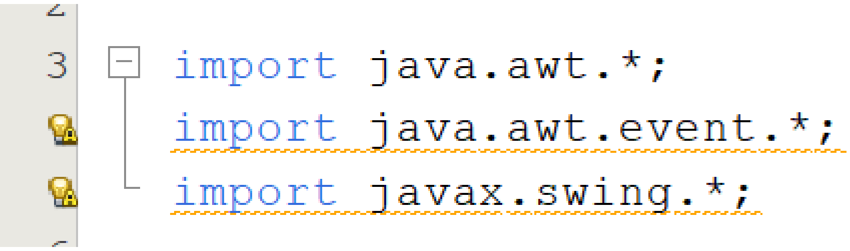
No if-else statements are used for running the code . As a result code is easy to read and maintain .It shows standard sign of code properties .

8. Poor use of inheritance:

There is no poor use of inheritance in this project which is particularly causing any problem.

9. Unnecessary dependencies:

There are many unnecessary dependencies are used and imported unused functionalities in the code .

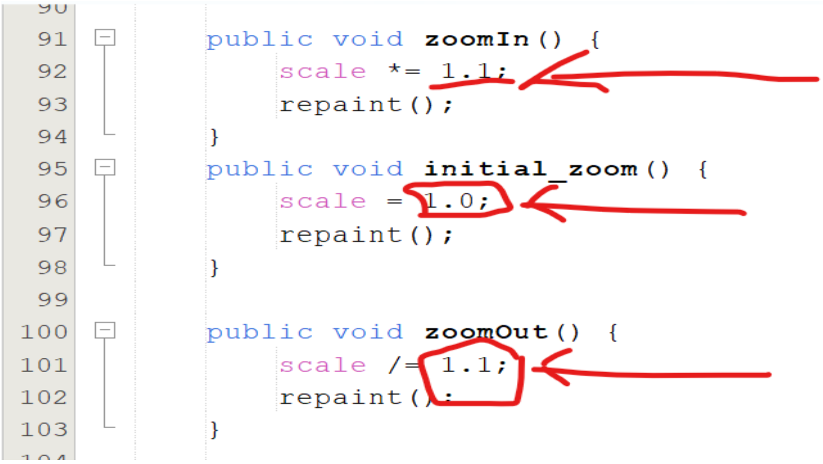


```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

These libraries/ frameworks are not being used but still imported unnecessary

10. Magic numbers or hard-coded values:

Some of the case magic number in the code . In MyCanvas.java class magic number are used .



```
    public void zoomIn() {
        scale *= 1.1;
        repaint();
    }
    public void initial_zoom() {
        scale = 1.0;
        repaint();
    }

    public void zoomOut() {
        scale /= 1.1;
        repaint();
    }
```

Proposed Architecture Evaluation

The proposed architecture for the project is the "**Model View Controller Architecture**." Within the project, we observe reflections of this architecture. Different layers or modules are evident.

For instance, there's **GUI1.java**, which can be likened to the "**View**" component of this architectural pattern. GUI1.java displays the main user interface, serving as the home page.

The **saveImage()** method within **GUI1.java** functions as the "Model" aspect of this architecture. This method saves the image as a .png file in storage.

The **Brush.java** and **Eraser.java** classes act as the "Controller," handling sketching and erasing within the project.

In summary, it may be stated that this project approximately follows the MVC architecture.

Modularity Check

In this project, the Brush.java and Eraser.java classes stand out for their compact and dynamic nature. On the other hand, the Canvas and GUI1.java classes are substantial, containing a considerable amount of code that could potentially be modularized into more subclasses.

These classes have the potential to be broken down into smaller, more specialized subclasses, which could enhance code organization and maintainability.

This approach aligns with the principles of object-oriented design, promoting encapsulation, reusability, and easier maintenance of the codebase

If/else Condition to Switch statement

No if/else condition are used in the project that can converted into switch statements.