CS 602 Java, HW#2 – Group H (Team Work)

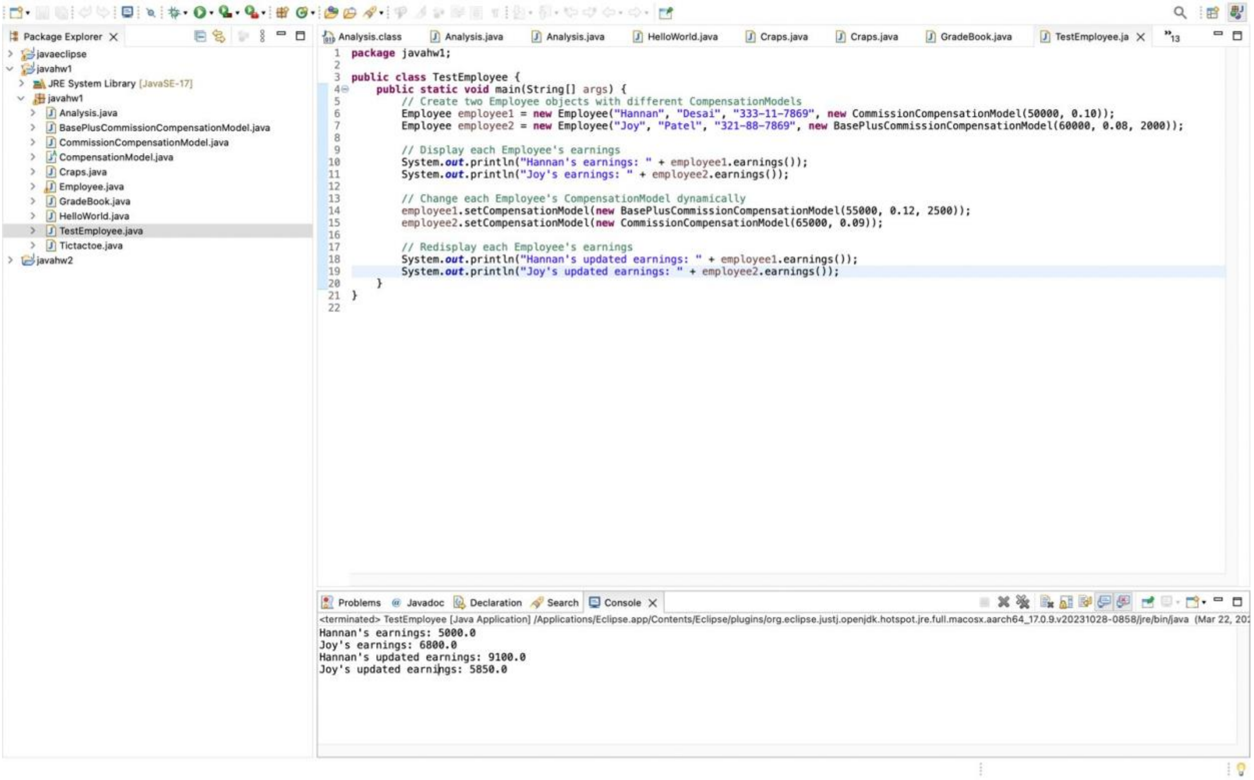
Group Members:

- **Charitha Kammari(ck369)**
- **Mohammed Hannan Desai(mmd76)**
- **Joy Patel(jp2267)**
- **Dhruval Dhameliya(dd548)**
- **Revanth Guntupalli(rg757)**

a. How 9.16 works, page 360 - Team (1 point)

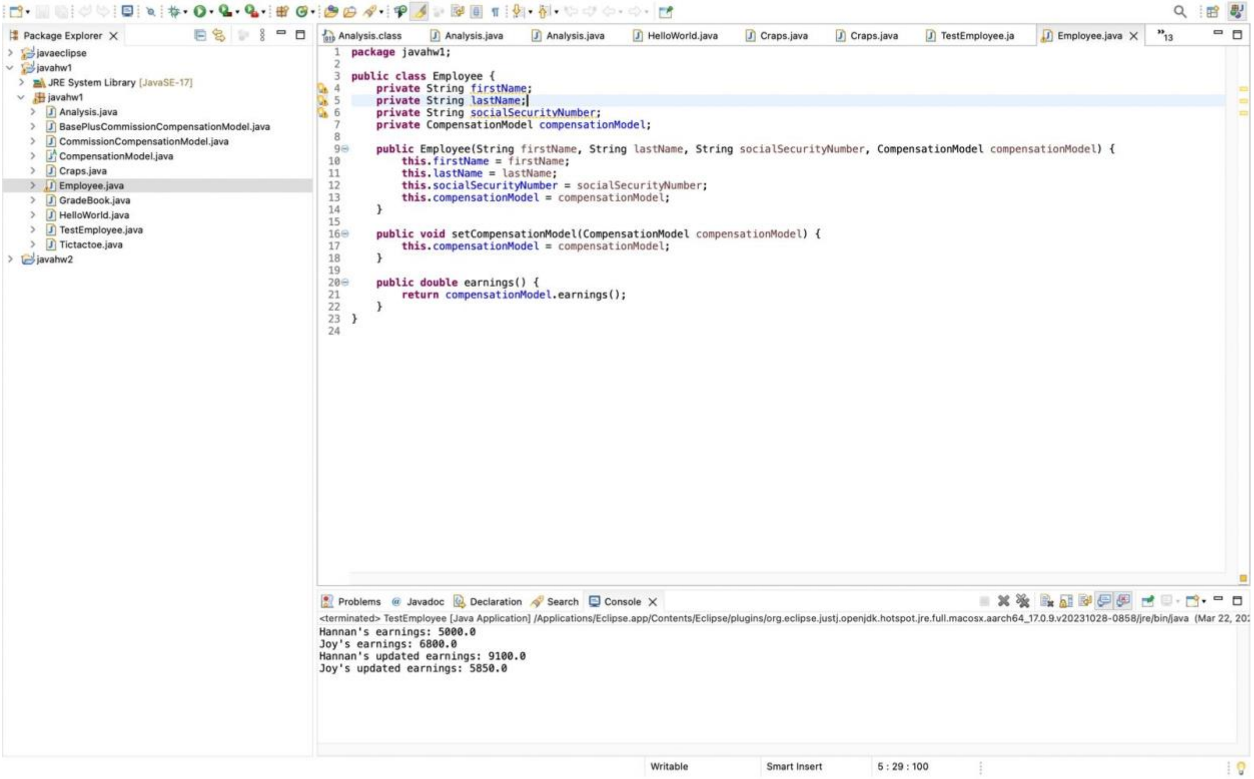
EXPLAINATION: In order to capture the common properties, such as firstName, lastName, and socialSecurityNumber, we constructed an Employee superclass in this query. We utilize a constructor in addition to the function toString to obtain the values getFirstName, getLastName, and getSocialSecurityNumber. As a subclass of the Employee class, we created the class CommisionEmployee. The constructor of the class CommisionEmployee calls the constructor of the Employee class, and the toString method calls the toString function of the Employee class to retrieve the necessary employee information. A subclass of the CommisionEmployee class, BasePlusCommisionEmployee inherits some of the CommisionEmployee class's characteristics. The BasePlusCommisionEmployee class receives information about the employee, including firstName, lastname, and SocialSecurityNumber. We also have a double-type variable in the class named baseSalary declared as private.

Following the creation of the classes, we execute the CommisionEmployeeTest.java and BaseCommisionEmployeeTest.java to acquire information using the get methods, followed by updated information using the toString method for CommisionEmployee where the commission and gross sales are updated. The toString value is used in BasePlusCommisionEmployeeTest to change the base salary value.



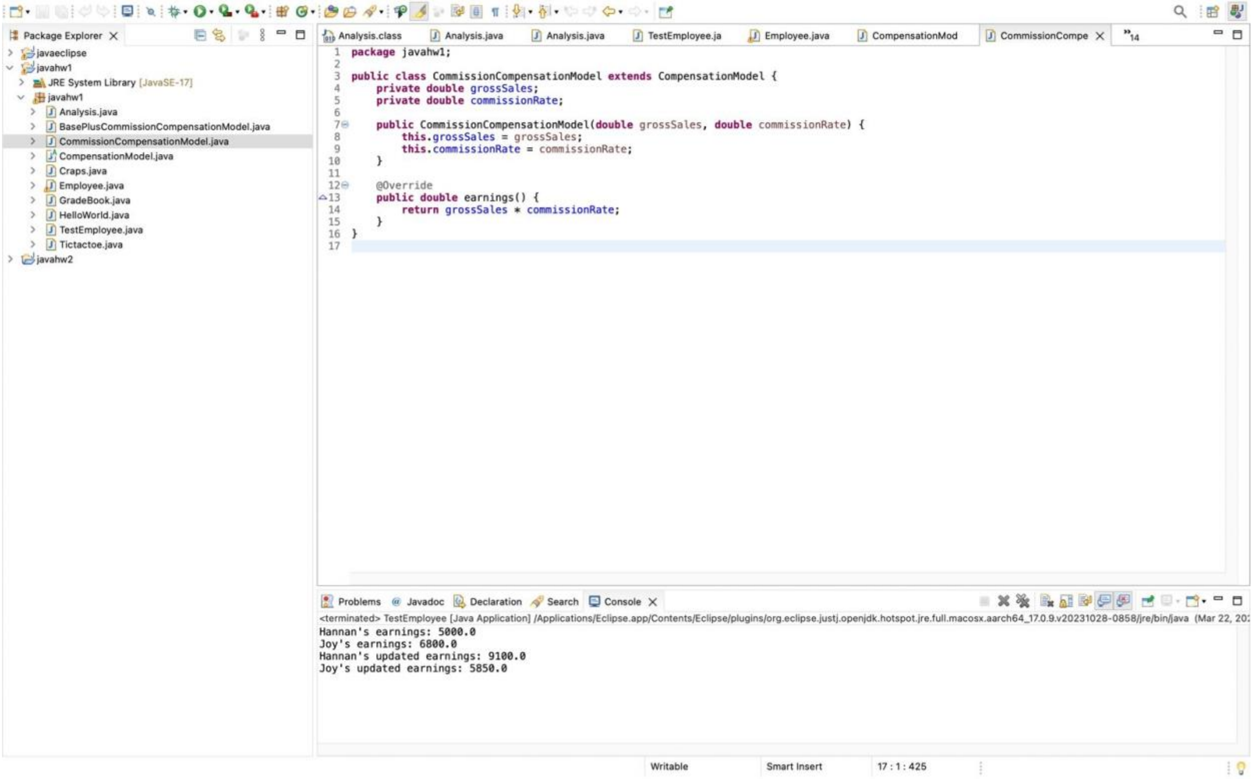
```java
package javahw1;

public class TestEmployee {
    public static void main(String[] args) {
        // Create two Employee objects with different CompensationModels
        Employee employee1 = new Employee("Hannan", "Desai", "333-11-7869", new CommissionCompensationModel(50000, 0.10));
        Employee employee2 = new Employee("Joy", "Patel", "321-88-7869", new BasePlusCommissionCompensationModel(60000, 0.08, 2000));

        // Display each Employee's earnings
        System.out.println("Hannan's earnings: " + employee1.earnings());
        System.out.println("Joy's earnings: " + employee2.earnings());

        // Change each Employee's CompensationModel dynamically
        employee1.setCompensationModel(new BasePlusCommissionCompensationModel(55000, 0.12, 2500));
        employee2.setCompensationModel(new CommissionCompensationModel(65000, 0.09));

        // Redisplay each Employee's earnings
        System.out.println("Hannan's updated earnings: " + employee1.earnings());
        System.out.println("Joy's updated earnings: " + employee2.earnings());
    }
}
```

```
<terminated> TestEmployee [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.9.v20231028-0858/jre/bin/java  (Mar 22, 202
Hannan's earnings: 5000.0
Joy's earnings: 6800.0
Hannan's updated earnings: 9100.0
Joy's updated earnings: 5850.0
```



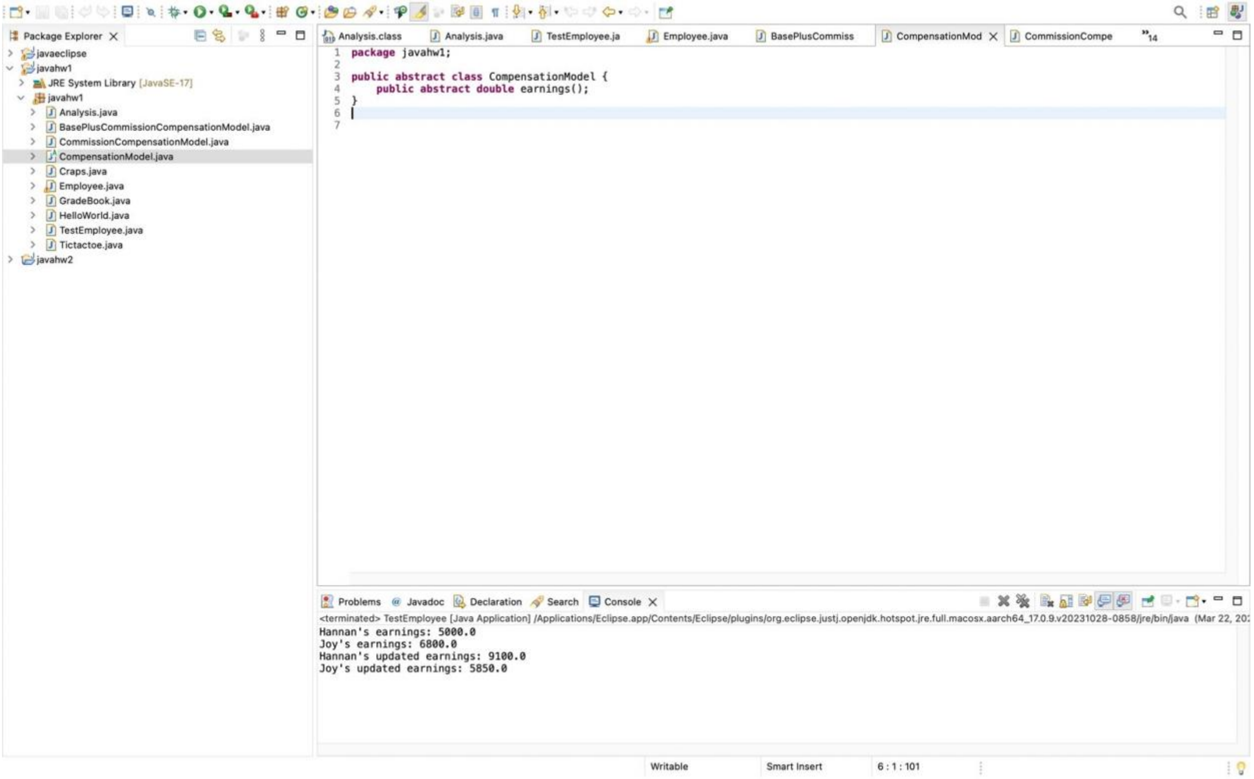
```java
package javahw1;

public class Employee {
    private String firstName;
    private String lastName;
    private String socialSecurityNumber;
    private CompensationModel compensationModel;

    public Employee(String firstName, String lastName, String socialSecurityNumber, CompensationModel compensationModel) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.socialSecurityNumber = socialSecurityNumber;
        this.compensationModel = compensationModel;
    }

    public void setCompensationModel(CompensationModel compensationModel) {
        this.compensationModel = compensationModel;
    }

    public double earnings() {
        return compensationModel.earnings();
    }
}

```

```
<terminated> TestEmployee [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.9.v20231028-0858/jre/bin/java  (Mar 22, 20
Hannan's earnings: 5000.0
Joy's earnings: 6800.0
Hannan's updated earnings: 9100.0
Joy's updated earnings: 5850.0
```



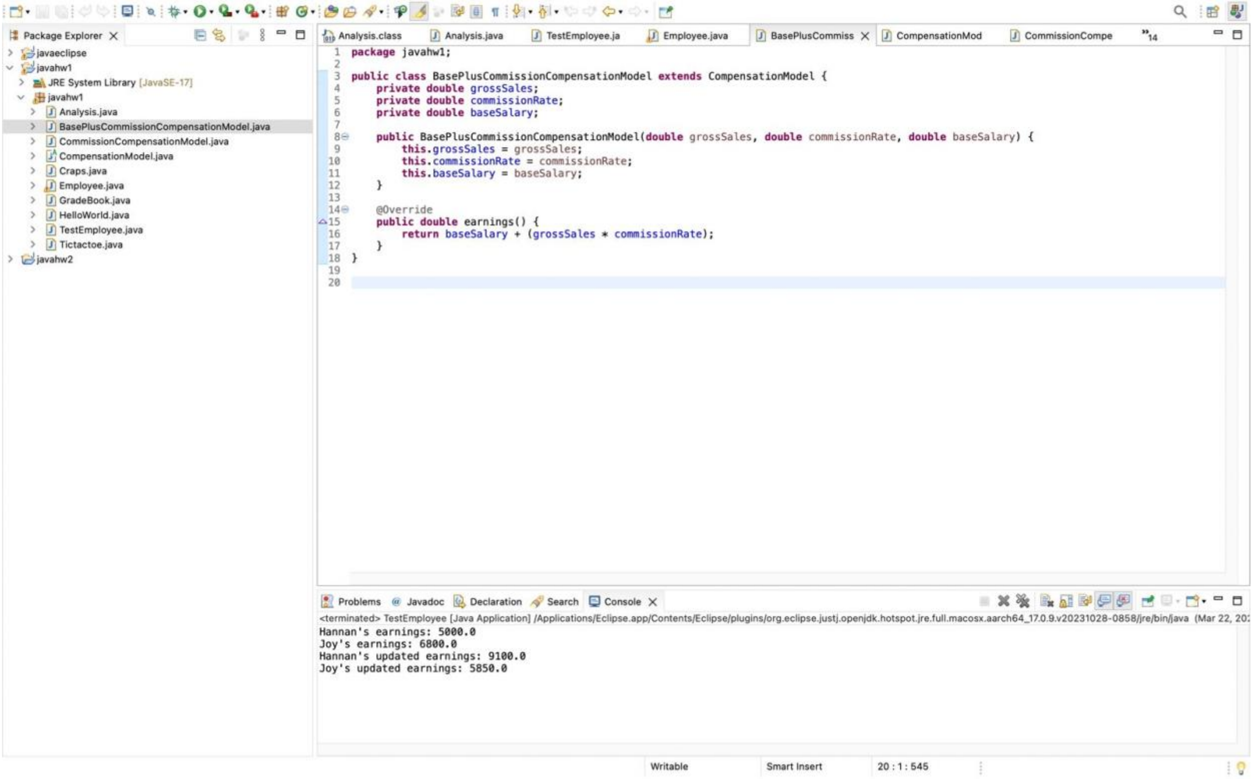
```java
package javahw1;

public class CommissionCompensationModel extends CompensationModel {
    private double grossSales;
    private double commissionRate;

    public CommissionCompensationModel(double grossSales, double commissionRate) {
        this.grossSales = grossSales;
        this.commissionRate = commissionRate;
    }

    @Override
    public double earnings() {
        return grossSales * commissionRate;
    }
}

```

```
<terminated> TestEmployee [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.9.v20231028-0858/jre/bin/java  (Mar 22, 20
Hannan's earnings: 5000.0
Joy's earnings: 6800.0
Hannan's updated earnings: 9100.0
Joy's updated earnings: 5850.0
```



```java
package javahw1;

public abstract class CompensationModel {
    public abstract double earnings();
}

```

```
<terminated> TestEmployee [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.9.v20231028-0858/jre/bin/java  (Mar 22, 20
Hannan's earnings: 5000.0
Joy's earnings: 6800.0
Hannan's updated earnings: 9100.0
Joy's updated earnings: 5850.0
```



```java
package javahw1;

public class BasePlusCommissionCompensationModel extends CompensationModel {
    private double grossSales;
    private double commissionRate;
    private double baseSalary;

    public BasePlusCommissionCompensationModel(double grossSales, double commissionRate, double baseSalary) {
        this.grossSales = grossSales;
        this.commissionRate = commissionRate;
        this.baseSalary = baseSalary;
    }

    @Override
    public double earnings() {
        return baseSalary + (grossSales * commissionRate);
    }
}

```

```
<terminated> TestEmployee [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.9.v20231028-0858/jre/bin/java  (Mar 22, 20
Hannan's earnings: 5000.0
Joy's earnings: 6800.0
Hannan's updated earnings: 9100.0
Joy's updated earnings: 5850.0
```

c. How 11.21 works, pages 439 - Team (1 point)

EXPLAINATION: Two try blocks in this program make an effort to carry out actions that could result in an exception. When the first try block tries to divide by zero, an ArithmeticException is raised. The second try block raises an ArrayIndexOutOfBoundsException when it tries to access an element of an array that doesn't exist.

The catch block that explicitly captures the exception type is used in both try blocks to catch the matching exception and manage it within the same try block. To catch any generic Exception, the program additionally has a catch block at the conclusion. Any exceptions that are thrown within the try blocks but are not caught by the specified catch blocks in those try blocks will be caught by this catch block.

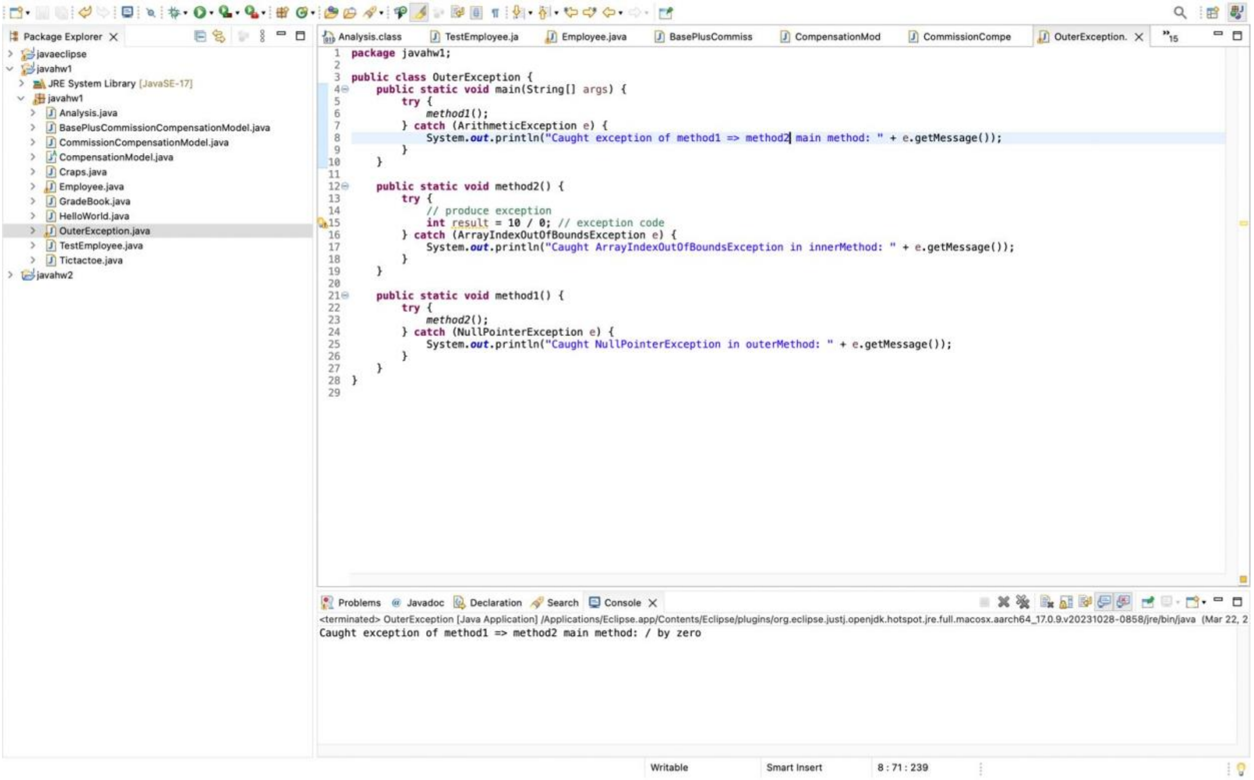
When you run this program, it will output:

Caught Arithmetic Exception:/by zero

Caught ArrayIndexOutOfBoundsException:5Programcompleted.

As you can see, despite not being caught in the appropriate catch blocks within the try blocks, the software managed to capture both exceptions and print out their contents.

The exceptions, however, "slipped through" to the outside scope and were discovered there.



```
package javahw1;

public class OuterException {
    public static void main(String[] args) {
        try {
            method1();
        } catch (ArithmeticException e) {
            System.out.println("Caught exception of method1 => method2 main method: " + e.getMessage());
        }
    }

    public static void method2() {
        try {
            // produce exception
            int result = 10 / 0; // exception code
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Caught ArrayIndexOutOfBoundsException in innerMethod: " + e.getMessage());
        }
    }

    public static void method1() {
        try {
            method2();
        } catch (NullPointerException e) {
            System.out.println("Caught NullPointerException in outerMethod: " + e.getMessage());
        }
    }
}
```

```
<terminated> OuterException [Java Application] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.9.v20231028-0858/jre/bin/java (Mar 22, 2
Caught exception of method1 => method2 main method: / by zero
```

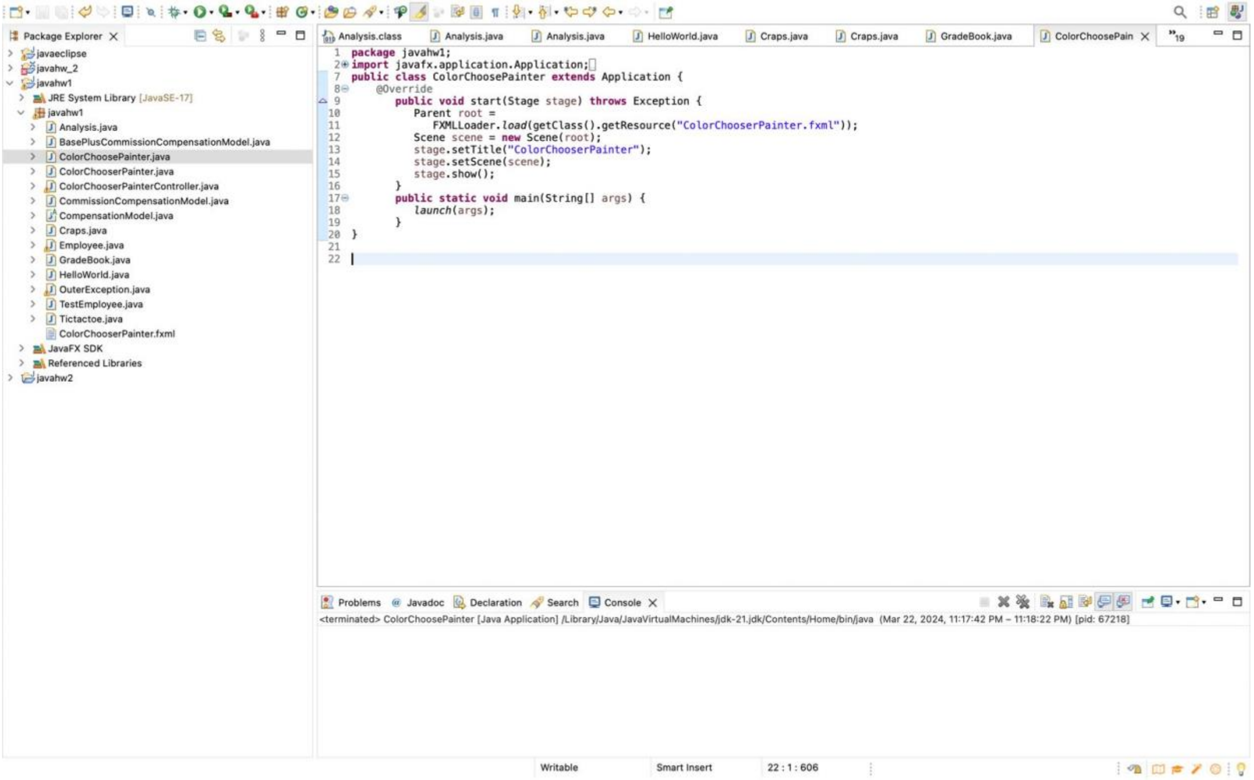
e. How 13.3 works, page 514 - Team (1 point)

EXPLAINATION: The current drawing color is now stored in a new instance variable called brushColor in this upgraded version. A new ColorChooserPanel component has also been added to the layout, and a brand-new ColorChooserChangeListener has been developed to update the brushColor instance variable whenever the color chooser's value changes.

Every time the value of the slider changes, the brushColor instance variable in the SizeSliderChangeListener is updated. This makes sure that the color of the artwork is always current with the color that was chosen in the color chooser.

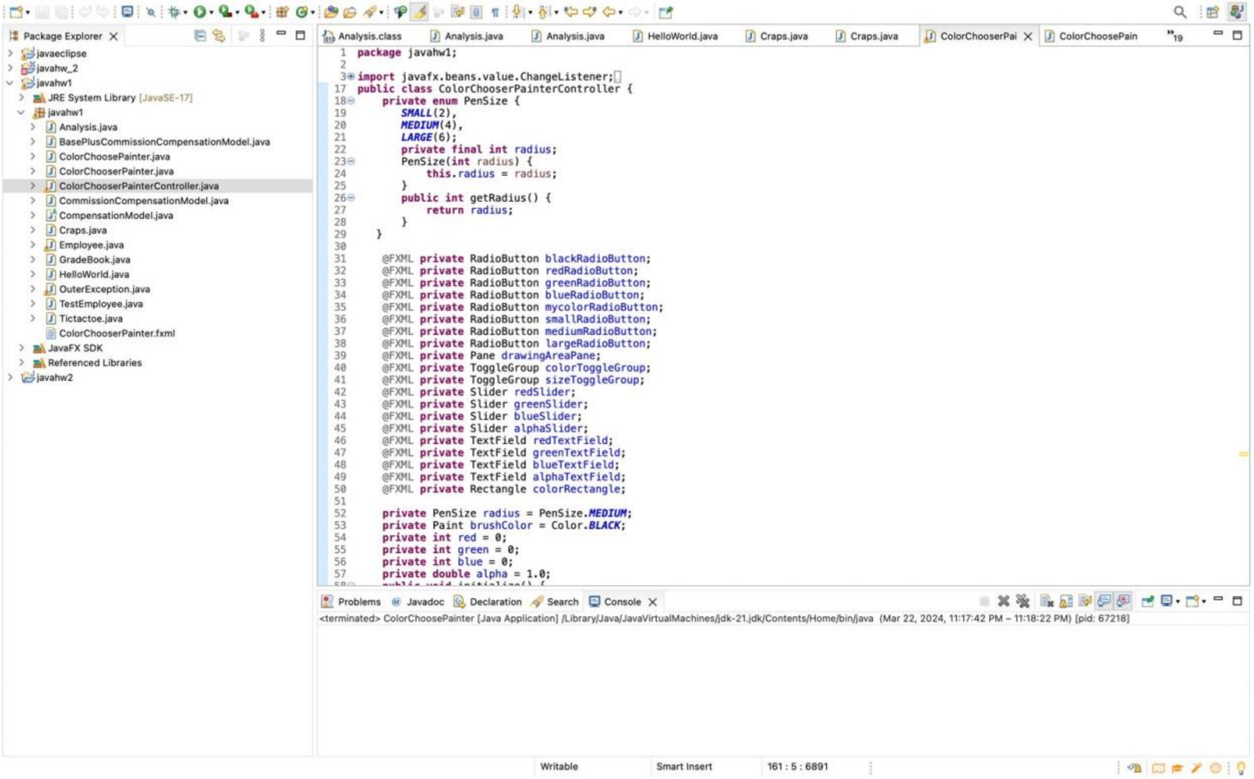
The brushColor instance variable is lastly used in the CanvasMouseListener to set the drawing color when a new oval is drawn on the canvas.

With these modifications, the user may now utilize the RGBA color chooser to select whatever drawing color they like, and the brush color will be adjusted accordingly.



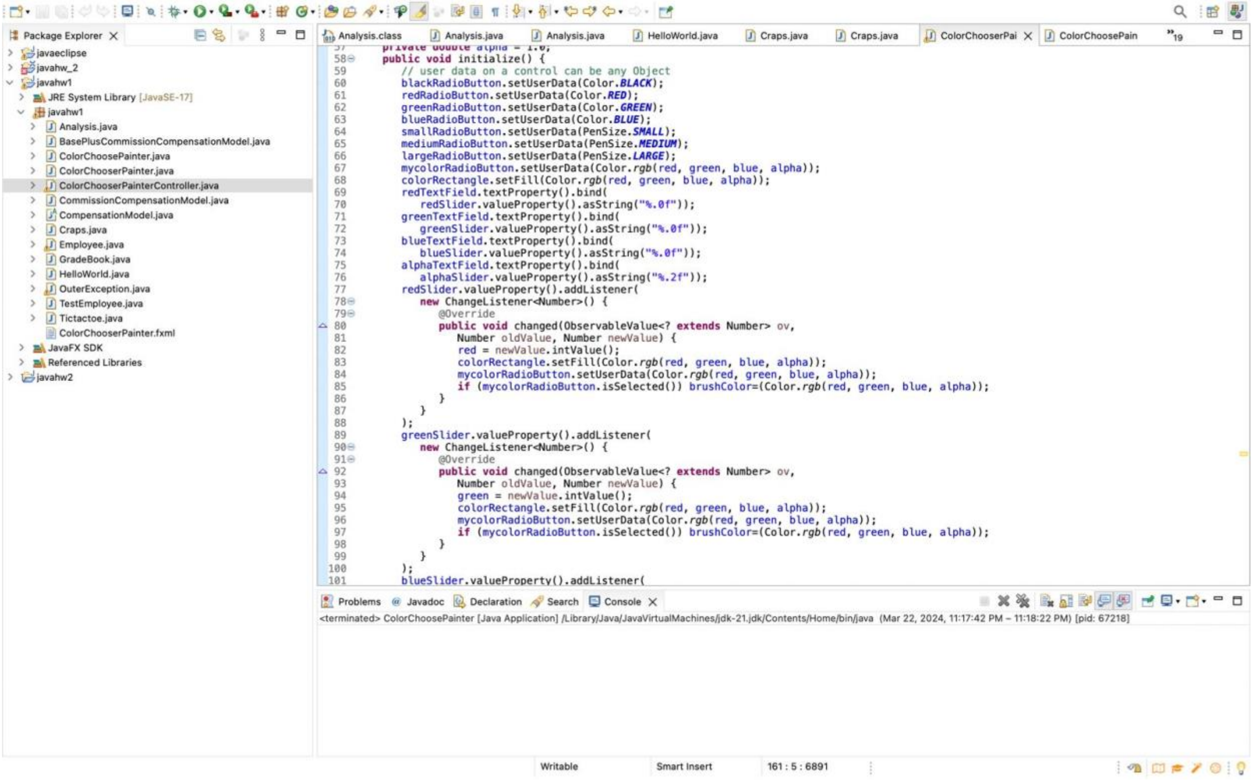
```java
package javahw1;
import javafx.application.Application;
public class ColorChoosePainter extends Application {
    @Override
        public void start(Stage stage) throws Exception {
           Parent root =
              FXMLLoader.load(getClass().getResource("ColorChooserPainter.fxml"));
           Scene scene = new Scene(root);
           stage.setTitle("ColorChooserPainter");
           stage.setScene(scene);
           stage.show();
        }
        public static void main(String[] args) {
           launch(args);
        }
}

```

```
<terminated> ColorChoosePainter [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (Mar 22, 2024, 11:17:42 PM – 11:18:22 PM) [pid: 67218]
```



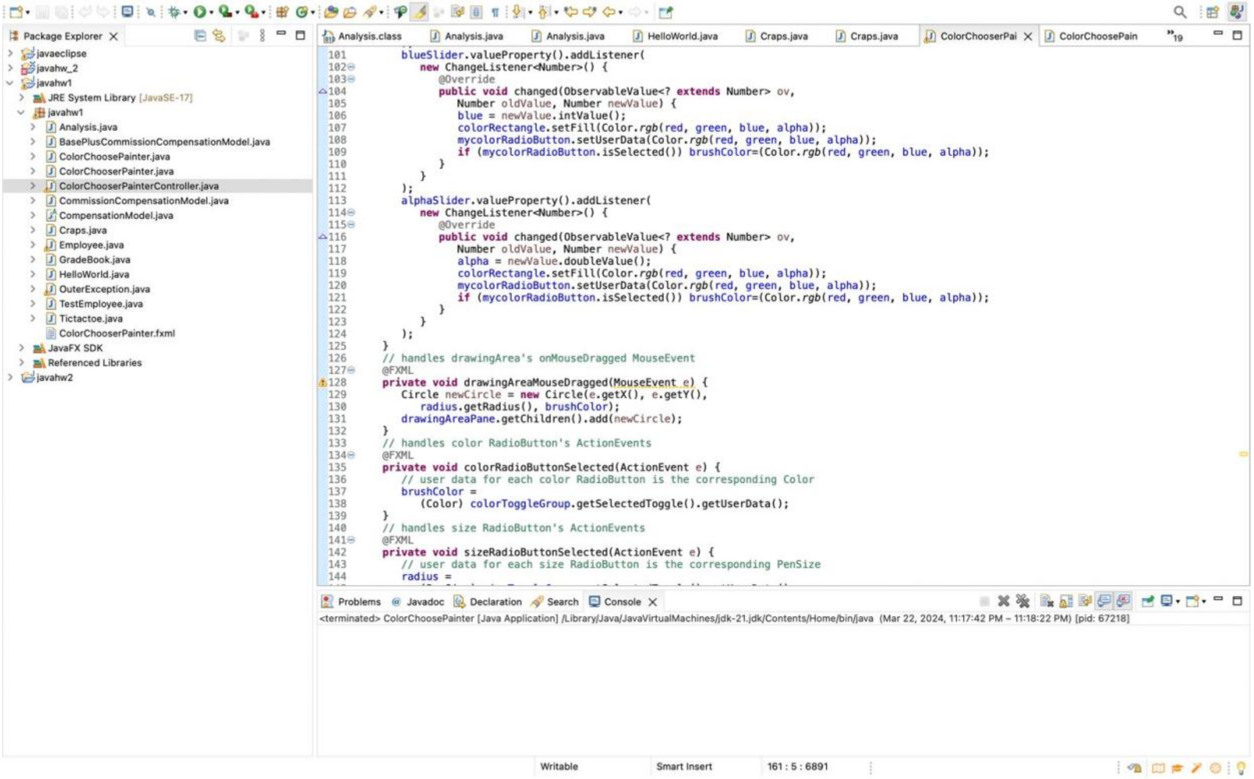
```java
package javahw1;

import javafx.beans.value.ChangeListener;
public class ColorChooserPainterController {
    private enum PenSize {
        SMALL(2),
        MEDIUM(4),
        LARGE(6);
        private final int radius;
        PenSize(int radius) {
            this.radius = radius;
        }
        public int getRadius() {
            return radius;
        }
    }

     @FXML private RadioButton blackRadioButton;
     @FXML private RadioButton redRadioButton;
     @FXML private RadioButton greenRadioButton;
     @FXML private RadioButton blueRadioButton;
     @FXML private RadioButton mycolorRadioButton;
     @FXML private RadioButton smallRadioButton;
     @FXML private RadioButton mediumRadioButton;
     @FXML private RadioButton largeRadioButton;
     @FXML private Pane drawingAreaPane;
     @FXML private ToggleGroup colorToggleGroup;
     @FXML private ToggleGroup sizeToggleGroup;
     @FXML private Slider redSlider;
     @FXML private Slider greenSlider;
     @FXML private Slider blueSlider;
     @FXML private Slider alphaSlider;
     @FXML private TextField redTextField;
     @FXML private TextField greenTextField;
     @FXML private TextField blueTextField;
     @FXML private TextField alphaTextField;
     @FXML private Rectangle colorRectangle;

     private PenSize radius = PenSize.MEDIUM;
     private Paint brushColor = Color.BLACK;
     private int red = 0;
     private int green = 0;
     private int blue = 0;
     private double alpha = 1.0;
     public void initialize() {
```

```
<terminated> ColorChoosePainter [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (Mar 22, 2024, 11:17:42 PM – 11:18:22 PM) [pid: 67218]
```



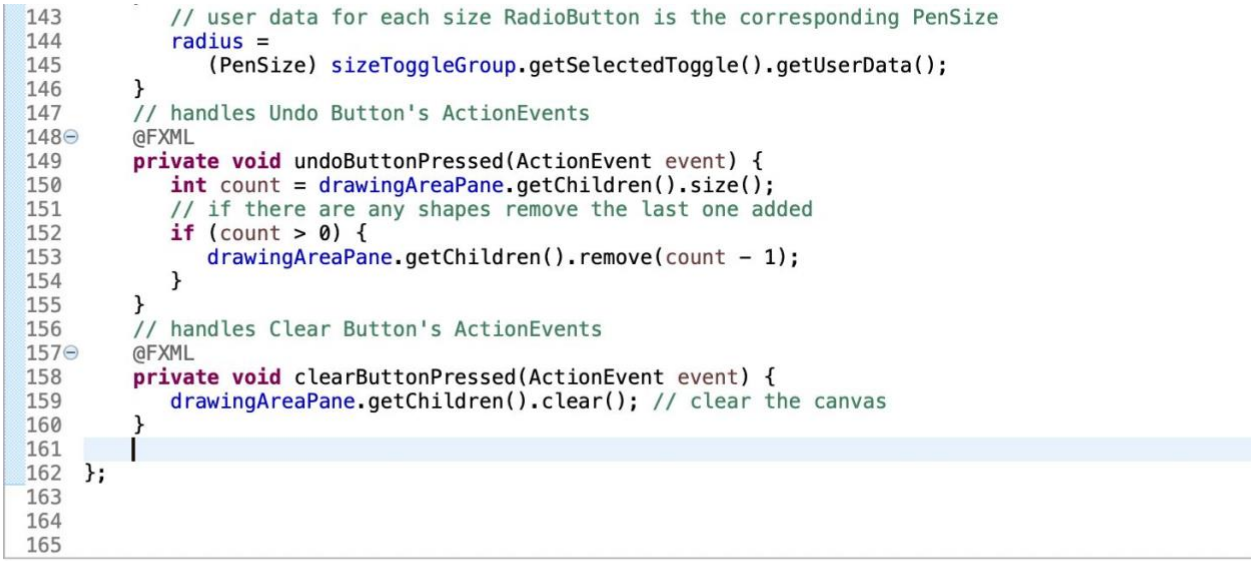
```java
     private double alpha = 1.0;
     public void initialize() {
        // user data on a control can be any Object
        blackRadioButton.setUserData(Color.BLACK);
        redRadioButton.setUserData(Color.RED);
        greenRadioButton.setUserData(Color.GREEN);
        blueRadioButton.setUserData(Color.BLUE);
        smallRadioButton.setUserData(PenSize.SMALL);
        mediumRadioButton.setUserData(PenSize.MEDIUM);
        largeRadioButton.setUserData(PenSize.LARGE);
        mycolorRadioButton.setUserData(Color.rgb(red, green, blue, alpha));
        colorRectangle.setFill(Color.rgb(red, green, blue, alpha));
        redTextField.textProperty().bind(
           redSlider.valueProperty().asString("%.0f"));
        greenTextField.textProperty().bind(
           greenSlider.valueProperty().asString("%.0f"));
        blueTextField.textProperty().bind(
           blueSlider.valueProperty().asString("%.0f"));
        alphaTextField.textProperty().bind(
           alphaSlider.valueProperty().asString("%.2f"));
        redSlider.valueProperty().addListener(
           new ChangeListener<Number>() {
              @Override
              public void changed(ObservableValue<? extends Number> ov,
                 Number oldValue, Number newValue) {
                 red = newValue.intValue();
                 colorRectangle.setFill(Color.rgb(red, green, blue, alpha));
                 mycolorRadioButton.setUserData(Color.rgb(red, green, blue, alpha));
                 if (mycolorRadioButton.isSelected()) brushColor=(Color.rgb(red, green, blue, alpha));
              }
           }
        );
        greenSlider.valueProperty().addListener(
           new ChangeListener<Number>() {
              @Override
              public void changed(ObservableValue<? extends Number> ov,
                 Number oldValue, Number newValue) {
                 green = newValue.intValue();
                 colorRectangle.setFill(Color.rgb(red, green, blue, alpha));
                 mycolorRadioButton.setUserData(Color.rgb(red, green, blue, alpha));
                 if (mycolorRadioButton.isSelected()) brushColor=(Color.rgb(red, green, blue, alpha));
              }
           }
        );
        blueSlider.valueProperty().addListener(
```

```
<terminated> ColorChoosePainter [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (Mar 22, 2024, 11:17:42 PM – 11:18:22 PM) [pid: 67218]
```

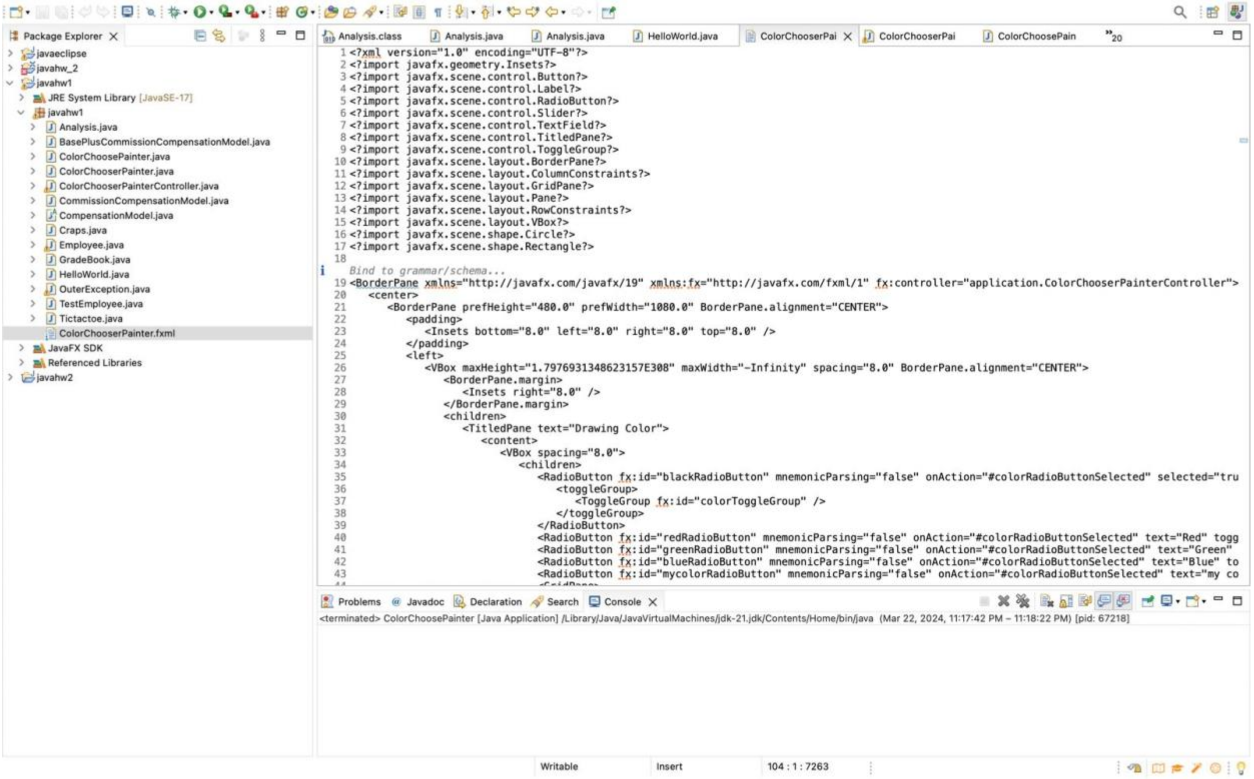


```java
      blueSlider.valueProperty().addListener(
         new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> ov,
               Number oldValue, Number newValue) {
               blue = newValue.intValue();
               colorRectangle.setFill(Color.rgb(red, green, blue, alpha));
               mycolorRadioButton.setUserData(Color.rgb(red, green, blue, alpha));
               if (mycolorRadioButton.isSelected()) brushColor=(Color.rgb(red, green, blue, alpha));
            }
         }
      );
      alphaSlider.valueProperty().addListener(
         new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> ov,
               Number oldValue, Number newValue) {
               alpha = newValue.doubleValue();
               colorRectangle.setFill(Color.rgb(red, green, blue, alpha));
               mycolorRadioButton.setUserData(Color.rgb(red, green, blue, alpha));
               if (mycolorRadioButton.isSelected()) brushColor=(Color.rgb(red, green, blue, alpha));
            }
         }
      );
   }
   // handles drawingArea's onMouseDragged MouseEvent
   @FXML
   private void drawingAreaMouseDragged(MouseEvent e) {
      Circle newCircle = new Circle(e.getX(), e.getY(),
         radius.getRadius(), brushColor);
      drawingAreaPane.getChildren().add(newCircle);
   }
   // handles color RadioButton's ActionEvents
   @FXML
   private void colorRadioButtonSelected(ActionEvent e) {
      // user data for each color RadioButton is the corresponding Color
      brushColor =
         (Color) colorToggleGroup.getSelectedToggle().getUserData();
   }
   // handles size RadioButton's ActionEvents
   @FXML
   private void sizeRadioButtonSelected(ActionEvent e) {
      // user data for each size RadioButton is the corresponding PenSize
      radius =
```

```
<terminated> ColorChoosePainter [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (Mar 22, 2024, 11:17:42 PM – 11:18:22 PM) [pid: 67218]
```

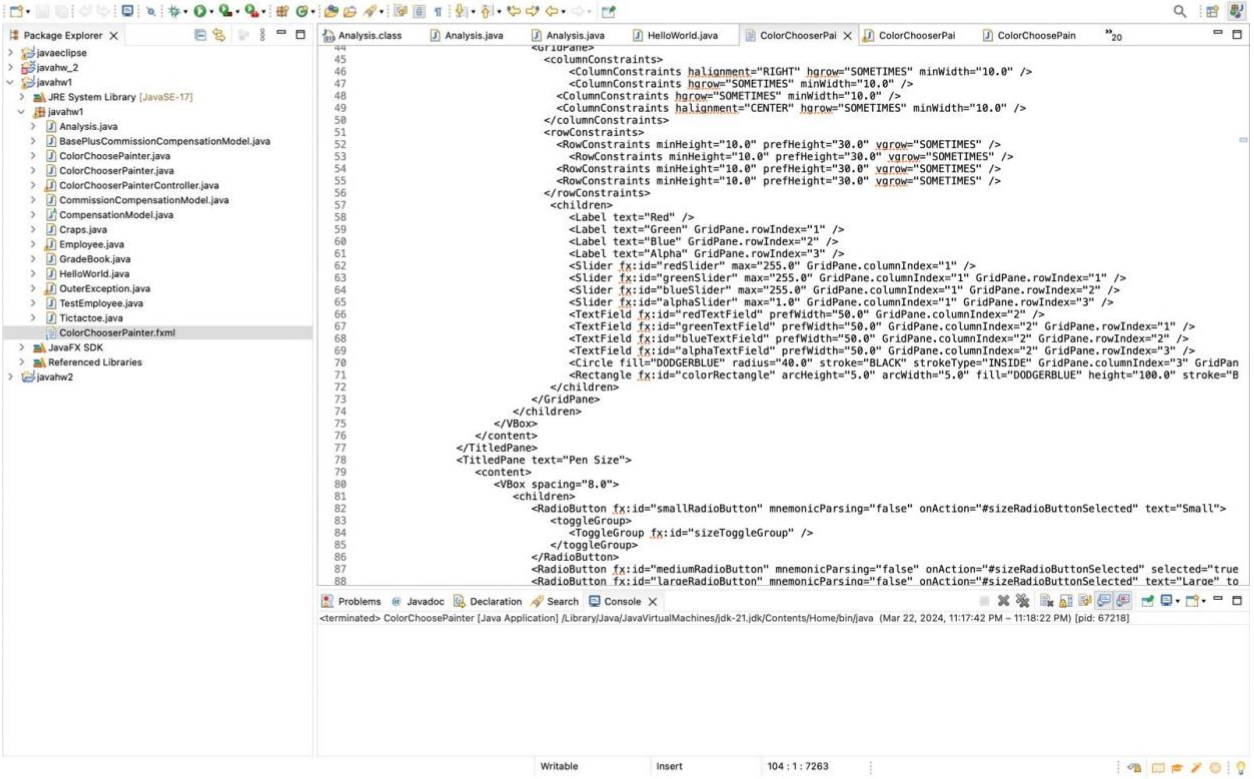


```java
      // user data for each size RadioButton is the corresponding PenSize
      radius =
         (PenSize) sizeToggleGroup.getSelectedToggle().getUserData();
   }
   // handles Undo Button's ActionEvents
   @FXML
   private void undoButtonPressed(ActionEvent event) {
      int count = drawingAreaPane.getChildren().size();
      // if there are any shapes remove the last one added
      if (count > 0) {
         drawingAreaPane.getChildren().remove(count - 1);
      }
   }
   // handles Clear Button's ActionEvents
   @FXML
   private void clearButtonPressed(ActionEvent event) {
      drawingAreaPane.getChildren().clear(); // clear the canvas
   }

};
```



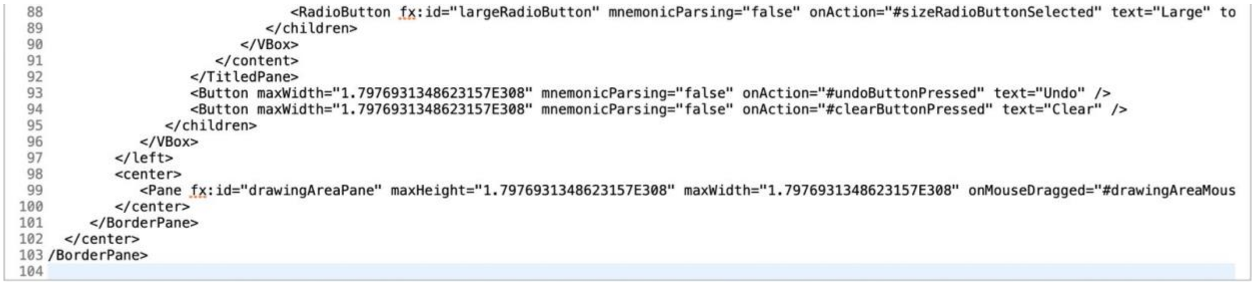
```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.RadioButton?>
<?import javafx.scene.control.Slider?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.TitledPane?>
<?import javafx.scene.control.ToggleGroup?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.shape.Circle?>
<?import javafx.scene.shape.Rectangle?>

  Bind to grammar/schema...
<BorderPane xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1" fx:controller="application.ColorChooserPainterController">
   <center>
      <BorderPane prefHeight="480.0" prefWidth="1080.0" BorderPane.alignment="CENTER">
         <padding>
            <Insets bottom="8.0" left="8.0" right="8.0" top="8.0" />
         </padding>
         <left>
            <VBox maxHeight="1.7976931348623157E308" maxWidth="-Infinity" spacing="8.0" BorderPane.alignment="CENTER">
               <BorderPane.margin>
                  <Insets right="8.0" />
               </BorderPane.margin>
               <children>
                  <TitledPane text="Drawing Color">
                     <content>
                        <VBox spacing="8.0">
                           <children>
                              <RadioButton fx:id="blackRadioButton" mnemonicParsing="false" onAction="#colorRadioButtonSelected" selected="tru
                                 <toggleGroup>
                                    <ToggleGroup fx:id="colorToggleGroup" />
                                 </toggleGroup>
                              </RadioButton>
                              <RadioButton fx:id="redRadioButton" mnemonicParsing="false" onAction="#colorRadioButtonSelected" text="Red" togg
                              <RadioButton fx:id="greenRadioButton" mnemonicParsing="false" onAction="#colorRadioButtonSelected" text="Green"
                              <RadioButton fx:id="blueRadioButton" mnemonicParsing="false" onAction="#colorRadioButtonSelected" text="Blue" to
                              <RadioButton fx:id="mycolorRadioButton" mnemonicParsing="false" onAction="#colorRadioButtonSelected" text="my co
```

```
<terminated> ColorChoosePainter [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (Mar 22, 2024, 11:17:42 PM – 11:18:22 PM) [pid: 67218]
```

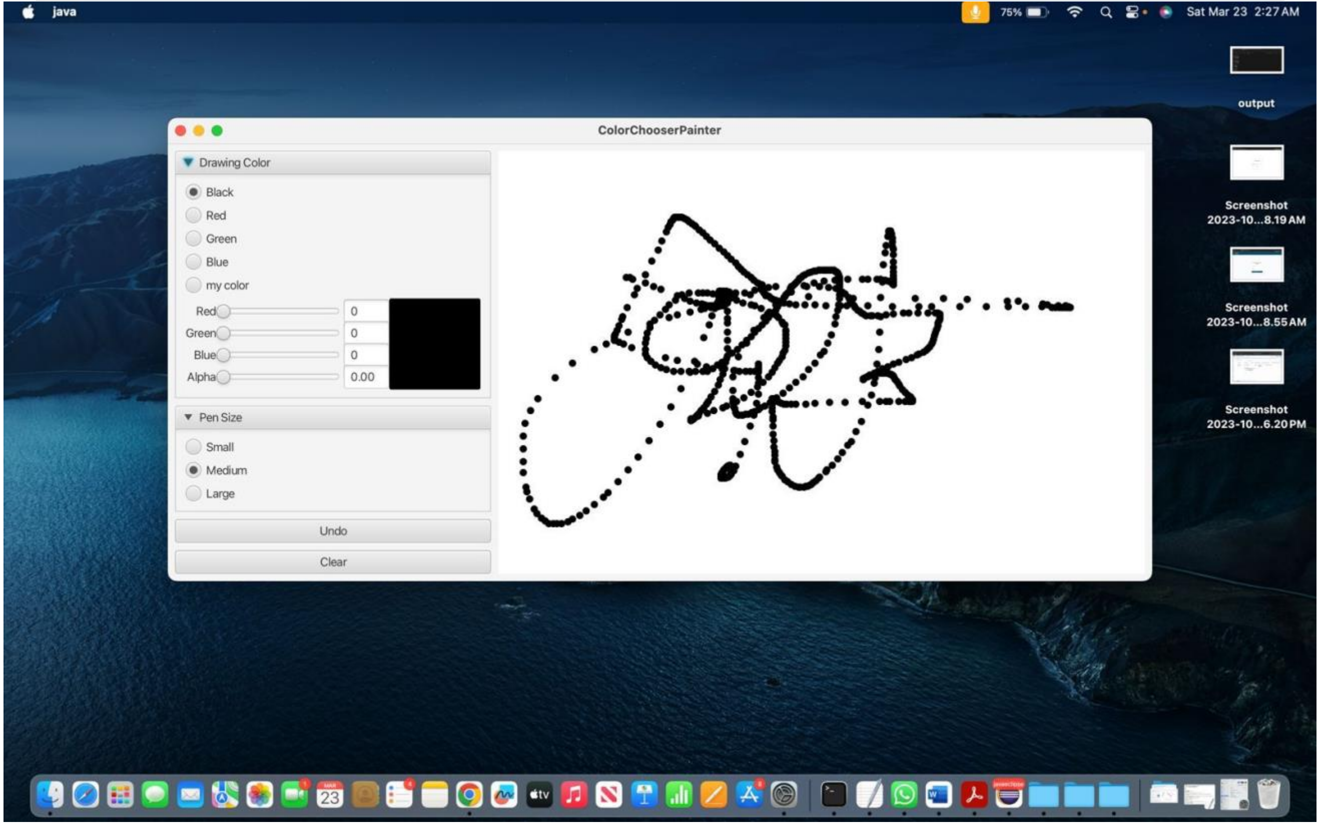


```
                             <GridPane>
                               <columnConstraints>
                                   <ColumnConstraints halignment="RIGHT" hgrow="SOMETIMES" minWidth="10.0" />
                                   <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" />
                                 <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" />
                                 <ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" />
                               </columnConstraints>
                               <rowConstraints>
                                 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                                   <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                                 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                                 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                               </rowConstraints>
                                <children>
                                   <Label text="Red" />
                                   <Label text="Green" GridPane.rowIndex="1" />
                                   <Label text="Blue" GridPane.rowIndex="2" />
                                   <Label text="Alpha" GridPane.rowIndex="3" />
                                   <Slider fx:id="redSlider" max="255.0" GridPane.columnIndex="1" />
                                   <Slider fx:id="greenSlider" max="255.0" GridPane.columnIndex="1" GridPane.rowIndex="1" />
                                   <Slider fx:id="blueSlider" max="255.0" GridPane.columnIndex="1" GridPane.rowIndex="2" />
                                   <Slider fx:id="alphaSlider" max="1.0" GridPane.columnIndex="1" GridPane.rowIndex="3" />
                                   <TextField fx:id="redTextField" prefWidth="50.0" GridPane.columnIndex="2" />
                                   <TextField fx:id="greenTextField" prefWidth="50.0" GridPane.columnIndex="2" GridPane.rowIndex="1" />
                                   <TextField fx:id="blueTextField" prefWidth="50.0" GridPane.columnIndex="2" GridPane.rowIndex="2" />
                                   <TextField fx:id="alphaTextField" prefWidth="50.0" GridPane.columnIndex="2" GridPane.rowIndex="3" />
                                   <Circle fill="DODGERBLUE" radius="40.0" stroke="BLACK" strokeType="INSIDE" GridPane.columnIndex="3" GridPan
                                   <Rectangle fx:id="colorRectangle" arcHeight="5.0" arcWidth="5.0" fill="DODGERBLUE" height="100.0" stroke="B
                                </children>
                             </GridPane>
                          </children>
                       </VBox>
                    </content>
                 </TitledPane>
                 <TitledPane text="Pen Size">
                    <content>
                       <VBox spacing="8.0">
                          <children>
                             <RadioButton fx:id="smallRadioButton" mnemonicParsing="false" onAction="#sizeRadioButtonSelected" text="Small">
                                <toggleGroup>
                                   <ToggleGroup fx:id="sizeToggleGroup" />
                                </toggleGroup>
                             </RadioButton>
                             <RadioButton fx:id="mediumRadioButton" mnemonicParsing="false" onAction="#sizeRadioButtonSelected" selected="true
                             <RadioButton fx:id="largeRadioButton" mnemonicParsing="false" onAction="#sizeRadioButtonSelected" text="Large" to
```

```
<terminated> ColorChoosePainter [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (Mar 22, 2024, 11:17:42 PM – 11:18:22 PM) [pid: 67218]
```



```
                             <RadioButton fx:id="largeRadioButton" mnemonicParsing="false" onAction="#sizeRadioButtonSelected" text="Large" to
                          </children>
                       </VBox>
                    </content>
                 </TitledPane>
                 <Button maxWidth="1.7976931348623157E308" mnemonicParsing="false" onAction="#undoButtonPressed" text="Undo" />
                 <Button maxWidth="1.7976931348623157E308" mnemonicParsing="false" onAction="#clearButtonPressed" text="Clear" />
              </children>
           </VBox>
        </left>
        <center>
           <Pane fx:id="drawingAreaPane" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" onMouseDragged="#drawingAreaMous
        </center>
     </BorderPane>
  </center>
/BorderPane>
```


java
75%
Sat Mar 23 2:27 AM
output
Screenshot 2023-10...8.19 AM
Screenshot 2023-10...8.55 AM
Screenshot 2023-10...6.20 PM
ColorChooserPainter
Undo
Clear

Drawing Color

● Black
○ Red
○ Green
○ Blue
○ my color

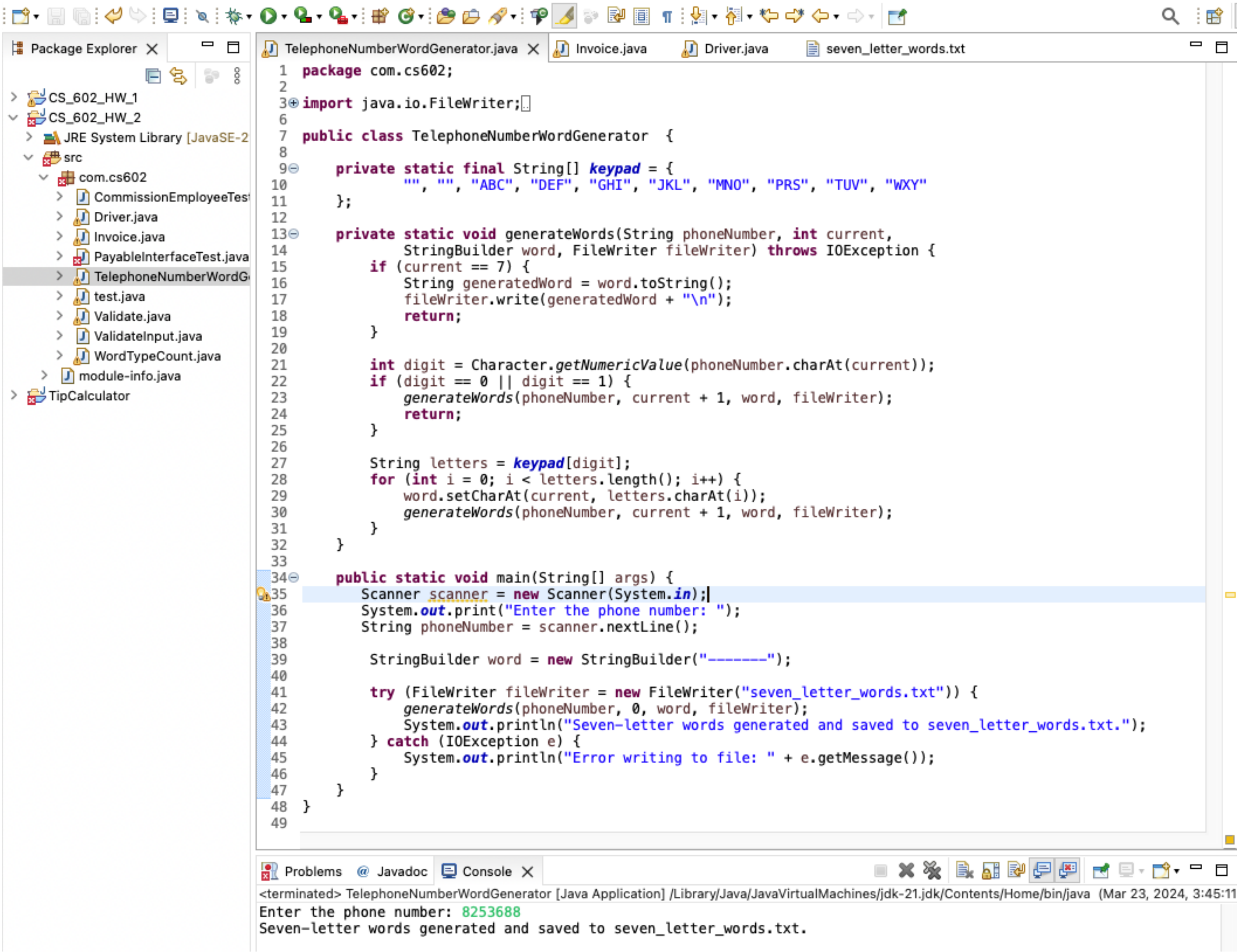
Red 0
Green 0
Blue 0
Alpha 0.00

Pen Size

○ Small
● Medium
○ Large

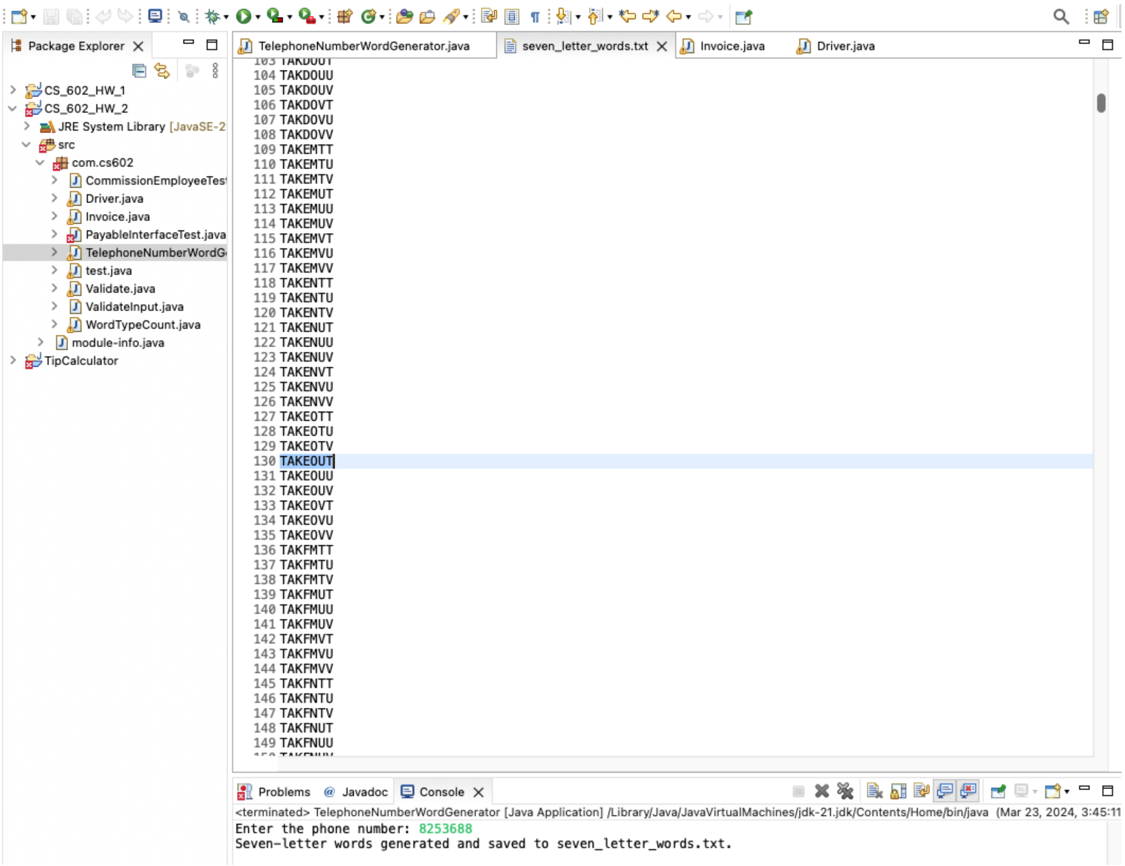
g. How 15.7 works, pages 602-603 - Team (1 point)

Explanation: The provided Java code is designed to facilitate the generation of seven-letter words based on a user-provided phone number input. It utilizes a keypad mapping, where each digit (except 0 and 1) corresponds to a set of letters. Upon receiving the phone number from the user, the program recursively explores all possible combinations of letters for each digit, constructing seven-letter words. This recursive process ensures that every potential arrangement of letters is considered, adhering to the constraints imposed by the phone keypad. The resulting words are then stored in a file named "seven_letter_words.txt" for easy reference and analysis. Additionally, the code is equipped to handle any potential errors that may occur during the file writing operation, ensuring a robust and reliable execution flow. Overall, this implementation provides a systematic and efficient means of generating seven-letter words based on a given phone number, leveraging the inherent structure of a phone keypad to guide the word formation process.



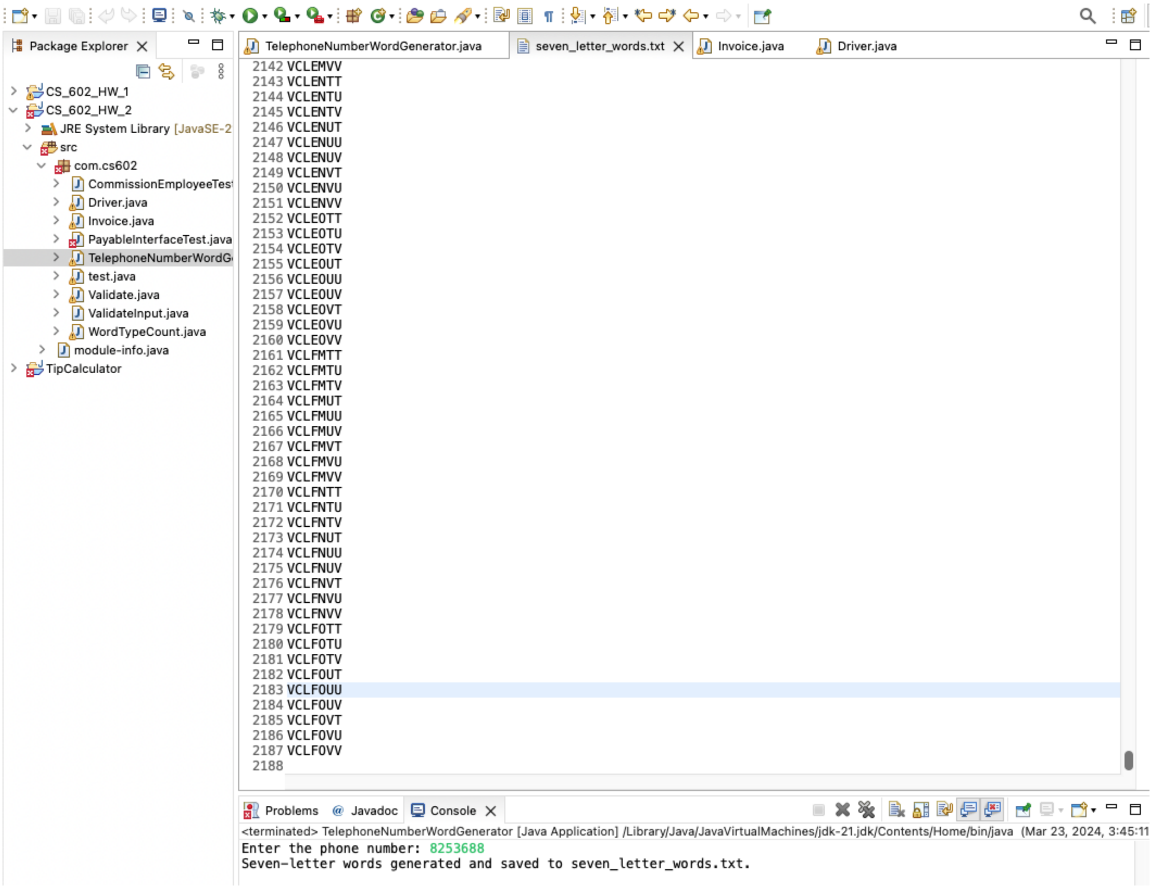
```java
package com.cs602;

import java.io.FileWriter;

public class TelephoneNumberWordGenerator  {

    private static final String[] keypad = {
            "", "", "ABC", "DEF", "GHI", "JKL", "MNO", "PRS", "TUV", "WXY"
    };

    private static void generateWords(String phoneNumber, int current,
            StringBuilder word, FileWriter fileWriter) throws IOException {
        if (current == 7) {
            String generatedWord = word.toString();
            fileWriter.write(generatedWord + "\n");
            return;
        }

        int digit = Character.getNumericValue(phoneNumber.charAt(current));
        if (digit == 0 || digit == 1) {
            generateWords(phoneNumber, current + 1, word, fileWriter);
            return;
        }

        String letters = keypad[digit];
        for (int i = 0; i < letters.length(); i++) {
            word.setCharAt(current, letters.charAt(i));
            generateWords(phoneNumber, current + 1, word, fileWriter);
        }
    }

    public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);
       System.out.print("Enter the phone number: ");
       String phoneNumber = scanner.nextLine();

        StringBuilder word = new StringBuilder("-------");

        try (FileWriter fileWriter = new FileWriter("seven_letter_words.txt")) {
            generateWords(phoneNumber, 0, word, fileWriter);
            System.out.println("Seven-letter words generated and saved to seven_letter_words.txt.");
        } catch (IOException e) {
            System.out.println("Error writing to file: " + e.getMessage());
        }
    }
}
```

```
<terminated> TelephoneNumberWordGenerator [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (Mar 23, 2024, 3:45:11
Enter the phone number: 8253688
Seven-letter words generated and saved to seven_letter_words.txt.
```



```
TAKDOUT
TAKDOUU
TAKDOUV
TAKDOVT
TAKDOVU
TAKDOVV
TAKEMTT
TAKEMTU
TAKEMTV
TAKEMUT
TAKEMUU
TAKEMUV
TAKEMVT
TAKEMVU
TAKEMVV
TAKENTT
TAKENTU
TAKENTV
TAKENUT
TAKENUU
TAKENUV
TAKENVT
TAKENVU
TAKENVV
TAKEOTT
TAKEOTU
TAKEOTV
TAKEOUT
TAKEOUU
TAKEOUV
TAKEOVT
TAKEOVU
TAKEOVV
TAKFMTT
TAKFMTU
TAKFMTV
TAKFMUT
TAKFMUU
TAKFMUV
TAKFMVT
TAKFMVU
TAKFMVV
TAKFNTT
TAKFNTU
TAKFNTV
TAKFNUT
TAKFNUU
```

```
<terminated> TelephoneNumberWordGenerator [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (Mar 23, 2024, 3:45:11
Enter the phone number: 8253688
Seven-letter words generated and saved to seven_letter_words.txt.
```

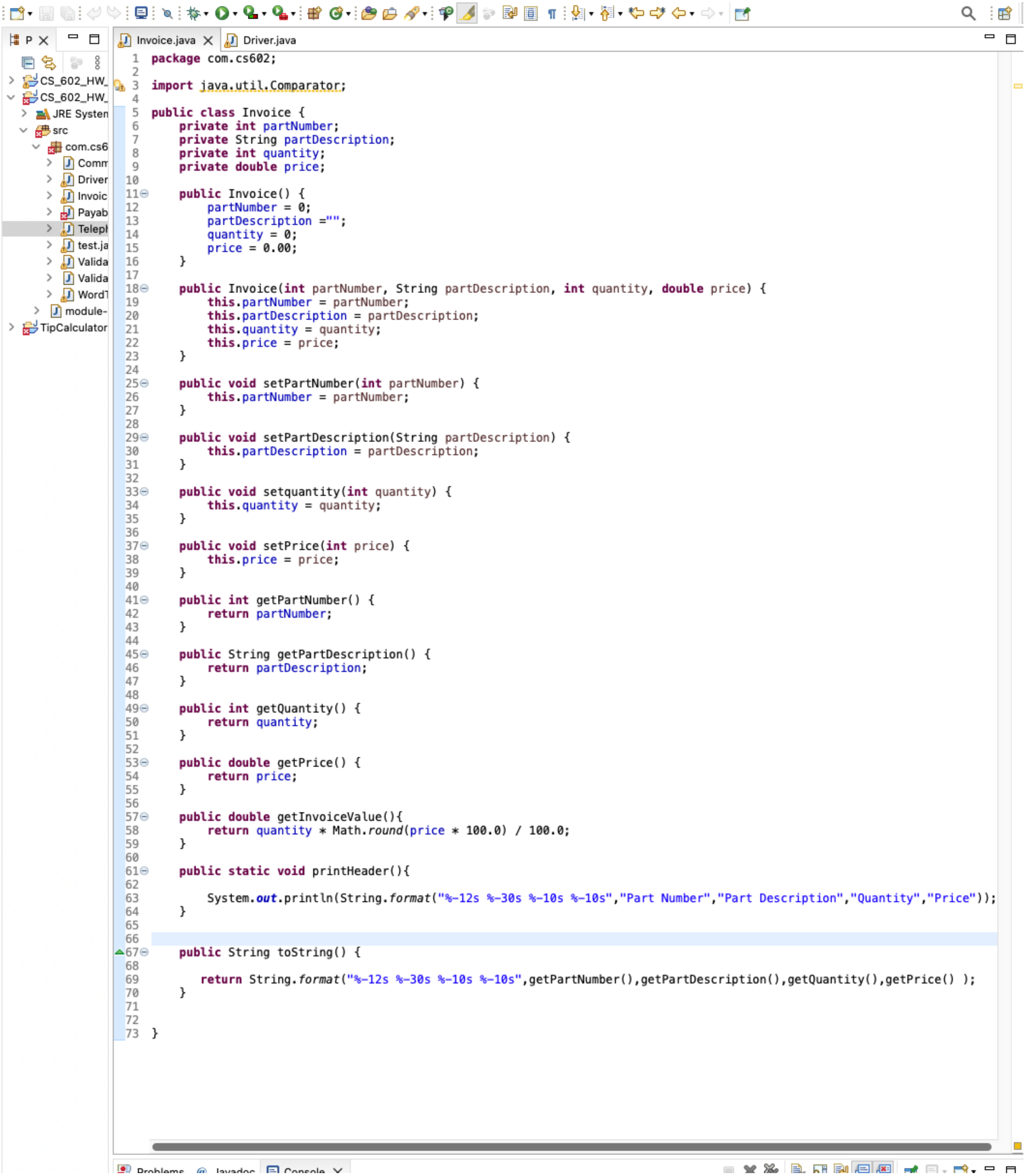


```
VCLEMVV
VCLENTT
VCLENTU
VCLENTV
VCLENUT
VCLENUU
VCLENUV
VCLENVT
VCLENVU
VCLENVV
VCLEOTT
VCLEOTU
VCLEOTV
VCLEOUT
VCLEOUU
VCLEOUV
VCLEOVT
VCLEOVU
VCLEOVV
VCLFMTT
VCLFMTU
VCLFMTV
VCLFMUT
VCLFMUU
VCLFMUV
VCLFMVT
VCLFMVU
VCLFMVV
VCLFNTT
VCLFNTU
VCLFNTV
VCLFNUT
VCLFNUU
VCLFNUV
VCLFNVT
VCLFNVU
VCLFNVV
VCLFOTT
VCLFOTU
VCLFOTV
VCLFOUT
VCLFOUU
VCLFOUV
VCLFOVT
VCLFOVU
VCLFOVV
```

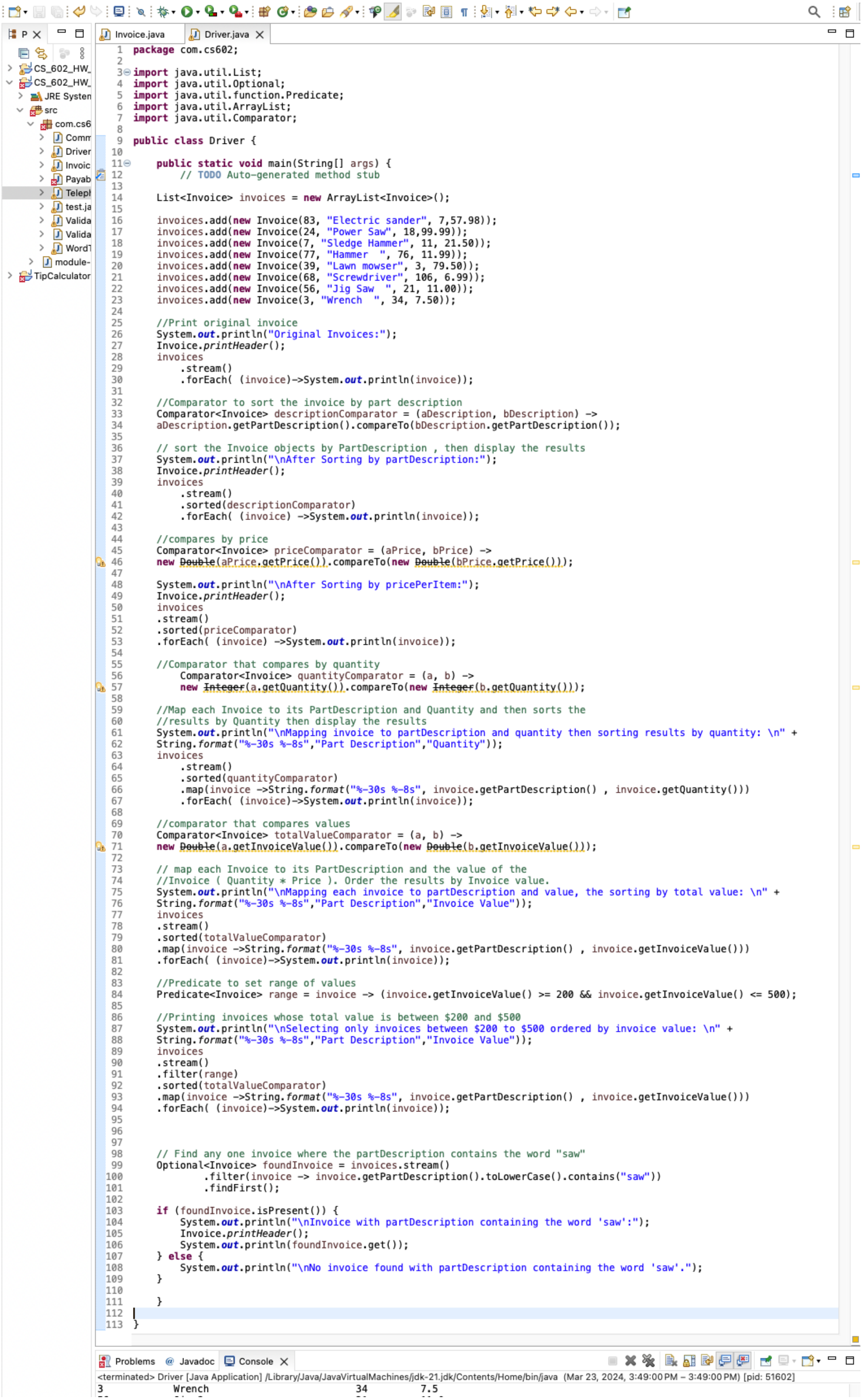
```
<terminated> TelephoneNumberWordGenerator [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (Mar 23, 2024, 3:45:11
Enter the phone number: 8253688
Seven-letter words generated and saved to seven_letter_words.txt.
```

i. How 17.12 works, pages 706 – Team (2 points)

Explanation: Manipulating a Stream<Invoice>
Making an array of invoice objects is the goal of this task. A partNumber, a partDescription, the number of the item being purchased, and a pricePerItem are the four instance variables used in this case. We employ a variety of methods in the class Invoice.java to obtain the part number, part description, quantity, set the price per item, obtain the price per item, and return the string representation of the invoice object. All of the Java utilities are imported into ProcessInvoice.java, and we use Listinvoices=new ArrayList to produce a list of invoices (). Finally, using invoices, we make a few invoices and add them to the invoices list. new invoice() is added.
a) Using getPartDescription, produce an invoice that is arranged by description ().
b) Using getprice to print the invoice price ().
c) Using getQuantity to arrange the data by quantity after mapping each invoice to its PartDescription and quantity ().
d) Mapping each invoice to partDescription by getQuantity()*invoice.getPrice().
e) Invoice values ranging between $200 and $500 by sorting, filtering and mapping.
f)invoice.getPartDescription().contains("saw")) to find any invoice containing the word "saw" in the partDescription.

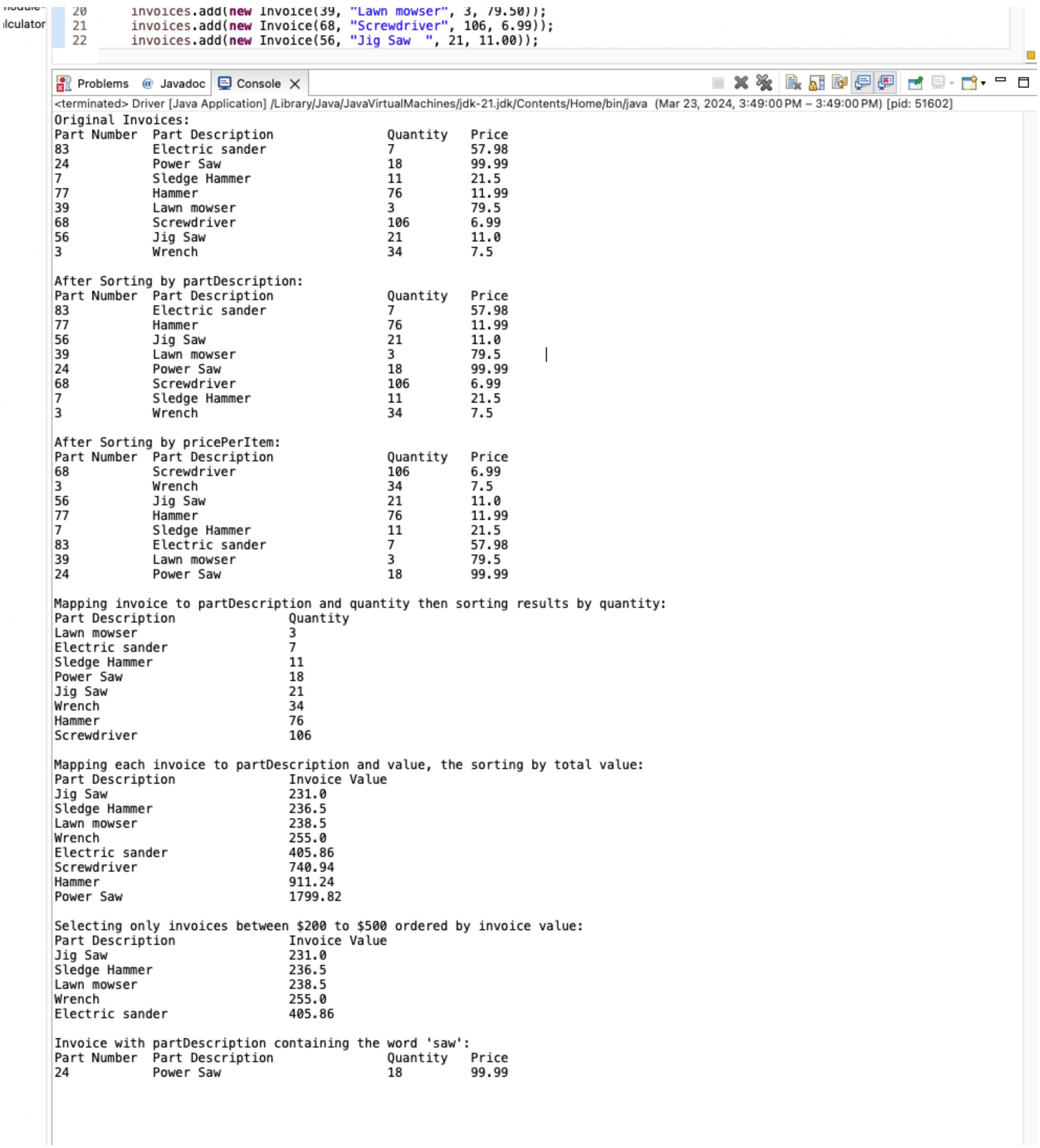


```java
package com.cs602;

import java.util.Comparator;

public class Invoice {
    private int partNumber;
    private String partDescription;
    private int quantity;
    private double price;

    public Invoice() {
        partNumber = 0;
        partDescription ="";
        quantity = 0;
        price = 0.00;
    }

    public Invoice(int partNumber, String partDescription, int quantity, double price) {
        this.partNumber = partNumber;
        this.partDescription = partDescription;
        this.quantity = quantity;
        this.price = price;
    }

    public void setPartNumber(int partNumber) {
        this.partNumber = partNumber;
    }

    public void setPartDescription(String partDescription) {
        this.partDescription = partDescription;
    }

    public void setquantity(int quantity) {
        this.quantity = quantity;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public int getPartNumber() {
        return partNumber;
    }

    public String getPartDescription() {
        return partDescription;
    }

    public int getQuantity() {
        return quantity;
    }

    public double getPrice() {
        return price;
    }

    public double getInvoiceValue(){
        return quantity * Math.round(price * 100.0) / 100.0;
    }

    public static void printHeader(){

        System.out.println(String.format("%-12s %-30s %-10s %-10s","Part Number","Part Description","Quantity","Price"));
    }


    public String toString() {

       return String.format("%-12s %-30s %-10s %-10s",getPartNumber(),getPartDescription(),getQuantity(),getPrice() );
    }


}
```



```java
package com.cs602;

import java.util.List;
import java.util.Optional;
import java.util.function.Predicate;
import java.util.ArrayList;
import java.util.Comparator;

public class Driver {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    List<Invoice> invoices = new ArrayList<Invoice>();

    invoices.add(new Invoice(83, "Electric sander", 7,57.98));
    invoices.add(new Invoice(24, "Power Saw", 18,99.99));
    invoices.add(new Invoice(7, "Sledge Hammer", 11, 21.50));
    invoices.add(new Invoice(77, "Hammer   ", 76, 11.99));
    invoices.add(new Invoice(39, "Lawn mowser", 3, 79.50));
    invoices.add(new Invoice(68, "Screwdriver", 106, 6.99));
    invoices.add(new Invoice(56, "Jig Saw  ", 21, 11.00));
    invoices.add(new Invoice(3, "Wrench  ", 34, 7.50));

    //Print original invoice
    System.out.println("Original Invoices:");
    Invoice.printHeader();
    invoices
        .stream()
        .forEach( (invoice)->System.out.println(invoice));

    //Comparator to sort the invoice by part description
    Comparator<Invoice> descriptionComparator = (aDescription, bDescription) ->
    aDescription.getPartDescription().compareTo(bDescription.getPartDescription());

    // sort the Invoice objects by PartDescription , then display the results
    System.out.println("\nAfter Sorting by partDescription:");
    Invoice.printHeader();
    invoices
        .stream()
        .sorted(descriptionComparator)
        .forEach( (invoice) ->System.out.println(invoice));

    //compares by price
    Comparator<Invoice> priceComparator = (aPrice, bPrice) ->
    new Double(aPrice.getPrice()).compareTo(new Double(bPrice.getPrice()));

    System.out.println("\nAfter Sorting by pricePerItem:");
    Invoice.printHeader();
    invoices
    .stream()
    .sorted(priceComparator)
    .forEach( (invoice) ->System.out.println(invoice));

    //Comparator that compares by quantity
        Comparator<Invoice> quantityComparator = (a, b) ->
        new Integer(a.getQuantity()).compareTo(new Integer(b.getQuantity()));

    //Map each Invoice to its PartDescription and Quantity and then sorts the
    //results by Quantity then display the results
    System.out.println("\nMapping invoice to partDescription and quantity then sorting results by quantity: \n" +
    String.format("%-30s %-8s","Part Description","Quantity"));
    invoices
        .stream()
        .sorted(quantityComparator)
        .map(invoice ->String.format("%-30s %-8s", invoice.getPartDescription() , invoice.getQuantity()))
        .forEach( (invoice)->System.out.println(invoice));

    //comparator that compares values
    Comparator<Invoice> totalValueComparator = (a, b) ->
    new Double(a.getInvoiceValue()).compareTo(new Double(b.getInvoiceValue()));

    // map each Invoice to its PartDescription and the value of the
    //Invoice ( Quantity * Price ). Order the results by Invoice value.
    System.out.println("\nMapping each invoice to partDescription and value, the sorting by total value: \n" +
    String.format("%-30s %-8s","Part Description","Invoice Value"));
    invoices
    .stream()
    .sorted(totalValueComparator)
    .map(invoice ->String.format("%-30s %-8s", invoice.getPartDescription() , invoice.getInvoiceValue()))
    .forEach( (invoice)->System.out.println(invoice));

    //Predicate to set range of values
    Predicate<Invoice> range = invoice -> (invoice.getInvoiceValue() >= 200 && invoice.getInvoiceValue() <= 500);

    //Printing invoices whose total value is between $200 and $500
    System.out.println("\nSelecting only invoices between $200 to $500 ordered by invoice value: \n" +
    String.format("%-30s %-8s","Part Description","Invoice Value"));
    invoices
    .stream()
    .filter(range)
    .sorted(totalValueComparator)
    .map(invoice ->String.format("%-30s %-8s", invoice.getPartDescription() , invoice.getInvoiceValue()))
    .forEach( (invoice)->System.out.println(invoice));



    // Find any one invoice where the partDescription contains the word "saw"
    Optional<Invoice> foundInvoice = invoices.stream()
            .filter(invoice -> invoice.getPartDescription().toLowerCase().contains("saw"))
            .findFirst();

    if (foundInvoice.isPresent()) {
        System.out.println("\nInvoice with partDescription containing the word 'saw':");
        Invoice.printHeader();
        System.out.println(foundInvoice.get());
    } else {
        System.out.println("\nNo invoice found with partDescription containing the word 'saw'.");
    }

    }

}
```

```
<terminated> Driver [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java  (Mar 23, 2024, 3:49:00 PM – 3:49:00 PM) [pid: 51602]
3            Wrench                            34         7.5
```



```java
    invoices.add(new Invoice(39, "Lawn mowser", 3, 79.50));
    invoices.add(new Invoice(68, "Screwdriver", 106, 6.99));
    invoices.add(new Invoice(56, "Jig Saw  ", 21, 11.00));
```

<terminated> Driver [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java (Mar 23, 2024, 3:49:00 PM – 3:49:00 PM) [pid: 51602]

```
Original Invoices:
Part Number  Part Description               Quantity   Price
83           Electric sander                7          57.98
24           Power Saw                      18         99.99
7            Sledge Hammer                  11         21.5
77           Hammer                         76         11.99
39           Lawn mowser                    3          79.5
68           Screwdriver                    106        6.99
56           Jig Saw                        21         11.0
3            Wrench                         34         7.5

After Sorting by partDescription:
Part Number  Part Description               Quantity   Price
83           Electric sander                7          57.98
77           Hammer                         76         11.99
56           Jig Saw                        21         11.0
39           Lawn mowser                    3          79.5
24           Power Saw                      18         99.99
68           Screwdriver                    106        6.99
7            Sledge Hammer                  11         21.5
3            Wrench                         34         7.5

After Sorting by pricePerItem:
Part Number  Part Description               Quantity   Price
68           Screwdriver                    106        6.99
3            Wrench                         34         7.5
56           Jig Saw                        21         11.0
77           Hammer                         76         11.99
7            Sledge Hammer                  11         21.5
83           Electric sander                7          57.98
39           Lawn mowser                    3          79.5
24           Power Saw                      18         99.99

Mapping invoice to partDescription and quantity then sorting results by quantity:
Part Description               Quantity
Lawn mowser                    3
Electric sander                7
Sledge Hammer                  11
Power Saw                      18
Jig Saw                        21
Wrench                         34
Hammer                         76
Screwdriver                    106

Mapping each invoice to partDescription and value, the sorting by total value:
Part Description               Invoice Value
Jig Saw                        231.0
Sledge Hammer                  236.5
Lawn mowser                    238.5
Wrench                         255.0
Electric sander                405.86
Screwdriver                    740.94
Hammer                         911.24
Power Saw                      1799.82

Selecting only invoices between $200 to $500 ordered by invoice value:
Part Description               Invoice Value
Jig Saw                        231.0
Sledge Hammer                  236.5
Lawn mowser                    238.5
Wrench                         255.0
Electric sander                405.86

Invoice with partDescription containing the word 'saw':
Part Number  Part Description               Quantity   Price
24           Power Saw                      18         99.99
```