1. Implementation

1.1. Functionality approach/justification: Component 1

I used the Arc class object to create semi-circle.

```
ShapeButton.setOnAction(e -> {
                    /* validate input and draw shape
                    draw circle*/
                    if (ShapeField.getText().equals("semi-circle")) {

                          Arc arc = new Arc(250, 300, 70, 70, 0, -180);
                          arc.setFill(Color.BLACK);
                          arc.setType(ArcType.OPEN);

                          // show semi-circle
                          root.getChildren().add(arc);
```

I used the rectangle class to create rectangle.

```
// draw rectangle
                    } else if (ShapeField.getText().equals("rectangle")) {

                          Rectangle rectangle = new Rectangle(400, 200);
                          rectangle.setX(125.0f);
                          rectangle.setY(125.0f);

                          // show rectangle
                                       root.getChildren().add(rectangle);
```

I used the polygon class to create pentagon.

```
else if (ShapeField.getText().equals("pentagon")) {
                          Polygon pentagon = new Polygon();
                          pentagon.getPoints().addAll(new Double[] {70.0, 140.0,
                                       70.0, 80.0,
                                       120.0, 40.0,
                                       160.0, 80.0,
                                       160.0, 140.0});

                          // show pentagon
                                       root.getChildren().add(pentagon);
```

I used the Alert class to create error prompt.

```
else {
                          Alert ErrorAlert = new Alert(AlertType.ERROR);
                          ErrorAlert.setHeaderText("Invalid shape.");
                          ErrorAlert.setContentText("Please input semi-circle,
rectangle, or pentagon");
                          ErrorAlert.showAndWait();
                                 }
```

1.2. Functionality approach/justification: Component 2

I used the .setOnAction to change the colour. I used the Alert class to create error prompt.

```
      // colour shape
                          ColorButton.setOnAction(e1 -> {
                                 if (ColorField.getText().equals("orange")) {
                                       pentagon.setFill(Color.ORANGE);
                                 } else if (ColorField.getText().equals("blue")) {
                                       pentagon.setFill(Color.BLUE);
                                 } else if (ColorField.getText().equals("grey")) {
                                        pentagon.setFill(Color.GREY);
                                } else {
                                        Alert ErrorAlert = new
Alert(AlertType.ERROR);
                                        ErrorAlert.setHeaderText("Invalid
colour.");
                                        ErrorAlert.setContentText("Please input
orange, blue or grey.");
                                                        ErrorAlert.showAndWait();
```

1.3. GUI design approach/justification: Component 1

I created a text field, a label, a button and a Vbox to contain all these to show the interactive component 1

```
        // create a text field for text input
                TextField ShapeField = new TextField();
                ShapeField.setMaxWidth(300);

                // add a label for users
                Label ShapeLabel = new Label("Enter a shape (semi-circle, rectangle,
or pentagon):");
                ShapeLabel.setTextFill(Color.BLACK);
                ShapeLabel.setFont(Font.font("Verdana", 25));

                // add a button to input shape
                Button ShapeButton = new Button();
                ShapeButton.setText("Draw the shape");

                // create a VBox to contain components
                VBox root = new VBox(30);
                root.setAlignment(Pos.CENTER);
                root.getChildren().addAll(ShapeLabel, ShapeField, ShapeButton);

                // create a scene and add it to stage
                Scene scene = new Scene(root, 700, 700);
                stage.setScene(scene);
                stage.setTitle("Shapes");
                                stage.show();
```

1.4. GUI design approach/justification: Component 2

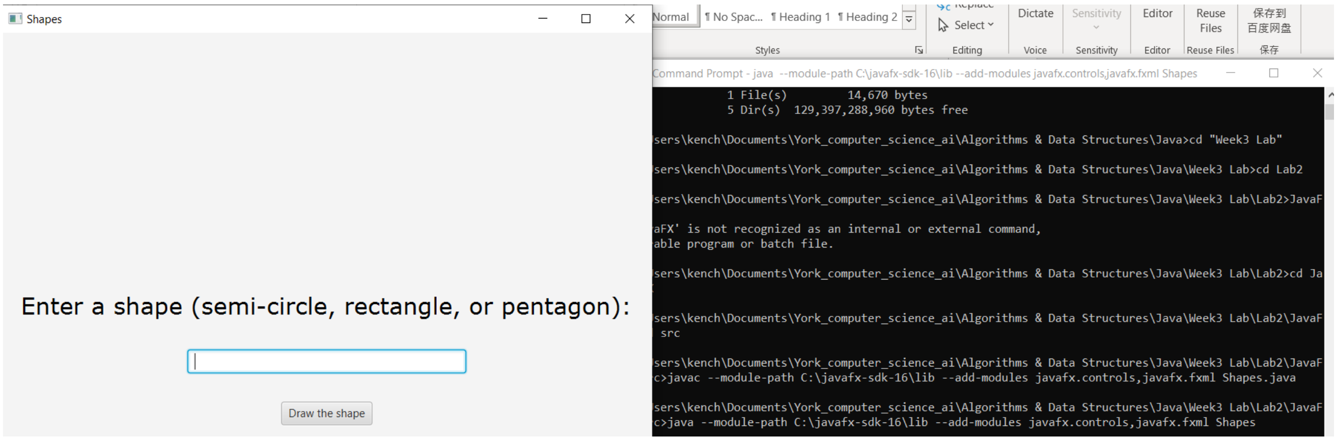
I created a text field, a label, a button to show the interactive component 1

```
                /* inform user to enter a colour
                                add a label for users*/
                                Label ColorLabel = new Label("Enter a colour (orange,
blue or grey):");

                                ColorLabel.setTextFill(Color.BLUE);
                                ColorLabel.setFont(Font.font("Verdana", 25));

                                // create a text field for text input
                                TextField ColorField = new TextField();
                                ColorField.setMaxWidth(300);

                                // add a button to input shape
                                Button ColorButton = new Button();
                                                ColorButton.setText("Change Colour");
```

2. Functionality and Testing

2.1. Overview of strategy for testing/demonstration of functionality

First I show that the Shapes class can be compiled and run from the command line.



Enter a shape (semi-circle, rectangle, or pentagon):

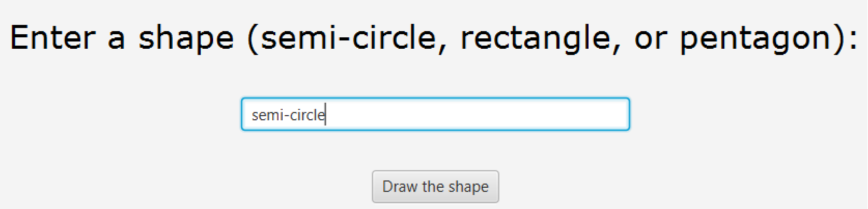
Command Prompt - java --module-path C:\javafx-sdk-16\lib --add-modules javafx.controls,javafx.fxml Shapes

```
          1 File(s)         14,670 bytes
          5 Dir(s)  129,397,288,960 bytes free

sers\kench\Documents\York_computer_science_ai\Algorithms & Data Structures\Java>cd "Week3 Lab"

sers\kench\Documents\York_computer_science_ai\Algorithms & Data Structures\Java\Week3 Lab>cd Lab2

sers\kench\Documents\York_computer_science_ai\Algorithms & Data Structures\Java\Week3 Lab\Lab2>JavaF

aFX' is not recognized as an internal or external command,
able program or batch file.

sers\kench\Documents\York_computer_science_ai\Algorithms & Data Structures\Java\Week3 Lab\Lab2>cd Ja


sers\kench\Documents\York_computer_science_ai\Algorithms & Data Structures\Java\Week3 Lab\Lab2\JavaF
 src

sers\kench\Documents\York_computer_science_ai\Algorithms & Data Structures\Java\Week3 Lab\Lab2\JavaF
c>javac --module-path C:\javafx-sdk-16\lib --add-modules javafx.controls,javafx.fxml Shapes.java

sers\kench\Documents\York_computer_science_ai\Algorithms & Data Structures\Java\Week3 Lab\Lab2\JavaF
c>java --module-path C:\javafx-sdk-16\lib --add-modules javafx.controls,javafx.fxml Shapes
```

Then I can input something into the text field, to enter a shape (semi-circle, rectangle, or pentagon) to test the functionality of component 1. I can see a black semi-circle when semi-circle is input. I can see a black rectangle when rectangle is input. I can see a black pentagon when pentagon is input. Also, I can enter a shape that is not one of semi-circle, rectangle, or pentagon, and a feedback is given when invalid input is entered, i.e. Invalid shape. Please input semi-circle, rectangle, or pentagon
After that I can test Component 2, I can enter something into the text field, to enter a colour (orange, blue or grey). I can see the shape change to orange when orange is input. I can see the shape change to blue when blue is input. I can see the shape change to grey when grey is input. Also, I can enter a colour that is not one of orange, blue or grey, and a feedback is given when invalid input is entered, i.e. Invalid colour. Please input orange, blue or grey.

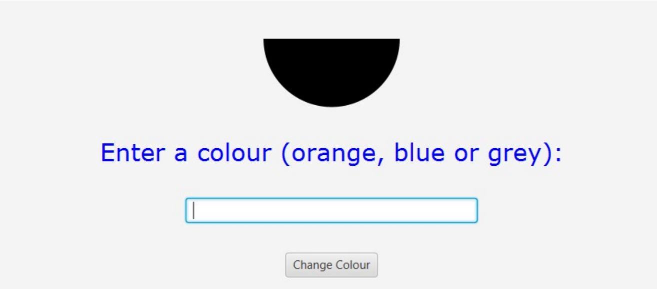
2.2. Testing/evidence of functionality: Component 1

I can see a black semi-circle when semi-circle is input.

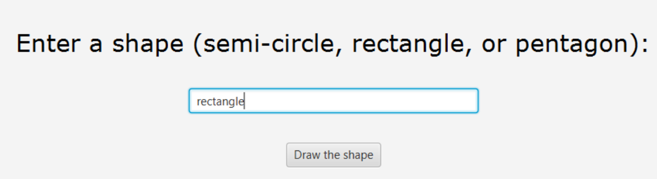


Enter a shape (semi-circle, rectangle, or pentagon):

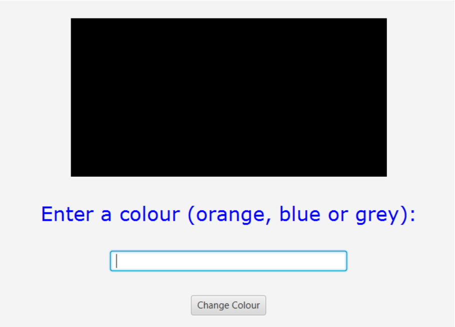
semi-circle



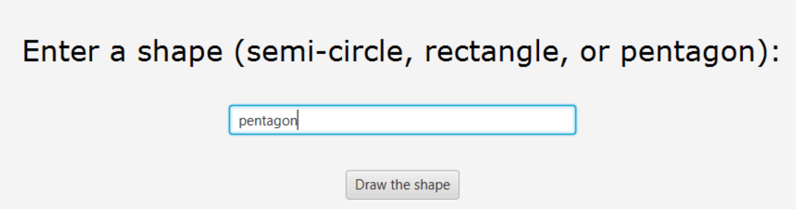
I can see a black rectangle when rectangle is input.



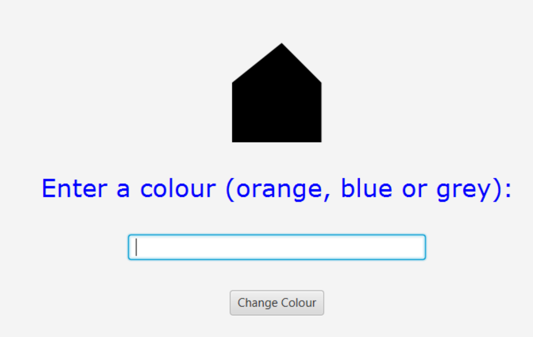
rectangle



I can see a black pentagon when pentagon is input.

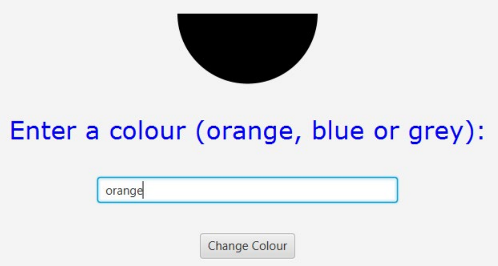


pentagon

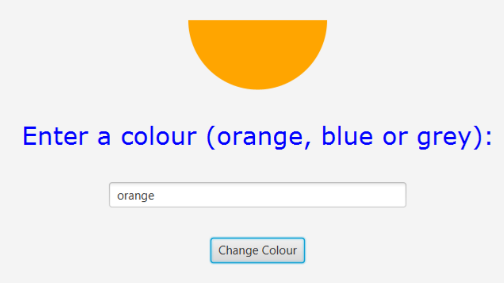


2.3. Testing/evidence of functionality: Component 2

I can see the shape change to orange when orange is input.



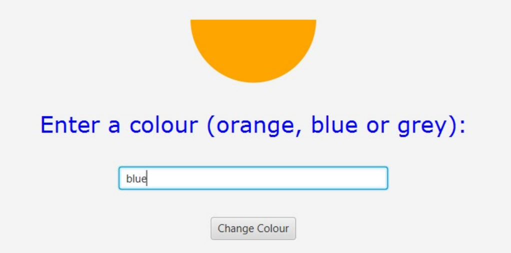
orange



Enter a colour (orange, blue or grey):

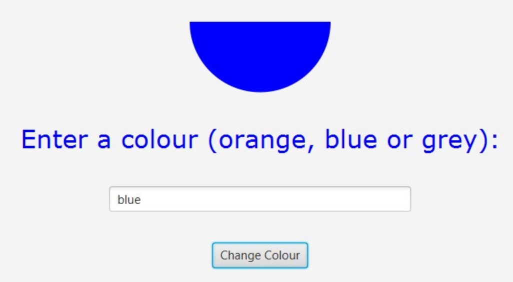
orange

I can see the shape change to blue when blue is input.



Enter a colour (orange, blue or grey):

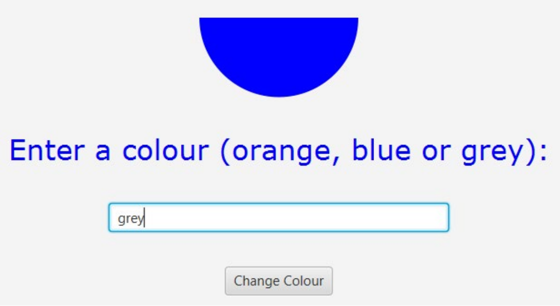
blue



Enter a colour (orange, blue or grey):

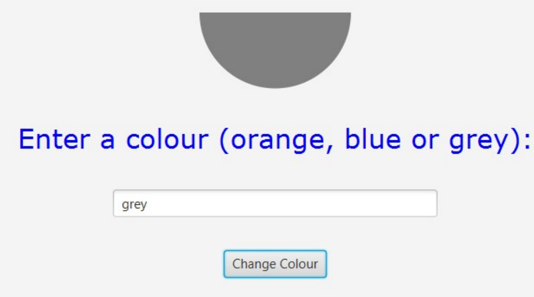
blue

I can see the shape change to grey when grey is input.



Enter a colour (orange, blue or grey):

grey

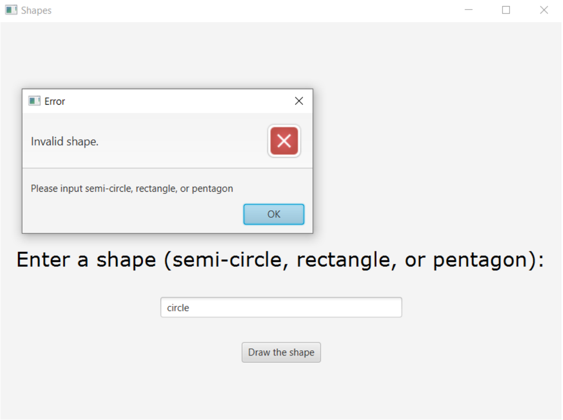


Enter a colour (orange, blue or grey):

grey

2.4. Testing/evidence of error handling: Component 1

I can enter a shape that is not one of semi-circle, rectangle, or pentagon, e.g. circle and a feedback is given when invalid input is entered, i.e. Invalid shape. Please input semi-circle, rectangle, or pentagon.



Shapes

Error

Invalid shape.

Please input semi-circle, rectangle, or pentagon

OK

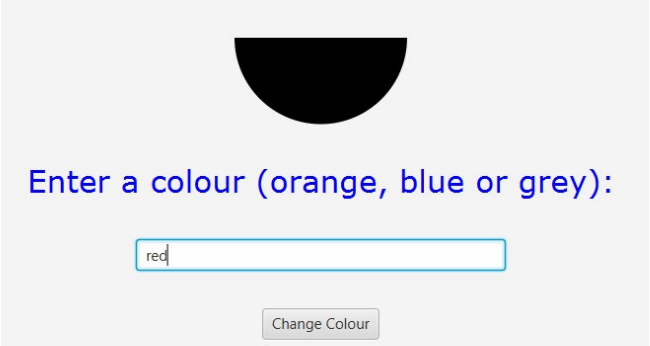
Enter a shape (semi-circle, rectangle, or pentagon):

circle

Draw the shape

2.5. Testing/evidence of error handling: Component 2

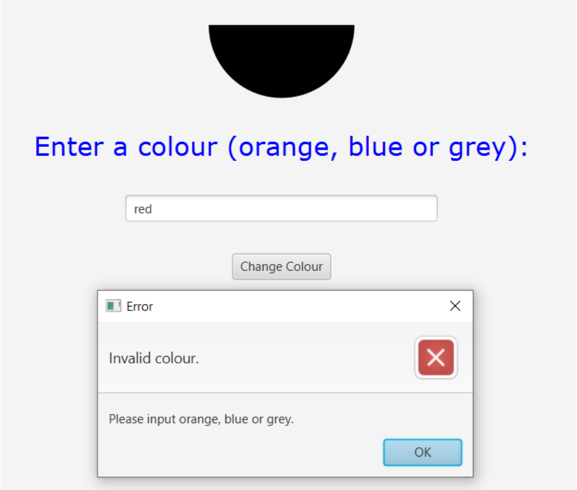
I can enter a colour that is not one of orange, blue or grey, e.g. red and a feedback is given when invalid input is entered, i.e. Invalid colour. Please input orange, blue or grey.



Enter a colour (orange, blue or grey):

red

Change Colour



Enter a colour (orange, blue or grey):

red

Change Colour

Error

Invalid colour.

Please input orange, blue or grey.

OK

References

[1] Q. Charatan and A. Kans, Java in Two Semesters. Switzerland: Springer Nature, 2019.

[2] Oracle. 39 Using Text in JavaFX, docs.oracle.com. [Online]. Available: https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/text-settings.htm [Accessed: May. 23, 2021].