GUI Basics

The current Java technology for developing GUI programs is JavaFX. In this document, we will introduce the technology in a relatively practical way, via a series of projects with increasing complexity.

Project 1.

Here is a relatively simple JavaFX program. It does some very fundamental things: create a GUI window and display a button that can be clicked. But the GUI does not respond to those clicks. (We will see how to do such things shortly.)

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class Project1 extends Application {
   @Override
   public void start(Stage primaryStage) {
      Button button = new Button("OK");
      Scene scene = new Scene(button, 200, 200);
      primaryStage.setTitle("Project 1");
      primaryStage.setScene(scene);
      primaryStage.show();
   }

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

The first thing to note here is that to create a GUI program, the class needs to extend `Application`. The GUI is started by the call `Application.launch(args);`

Within the class that extends `Application`, override the `start()` method. This method has a single parameter of type `Stage`. A stage can have a `Scene` and the scene can change. This is similar to the stage of a play, where you can have multiple scenes. Similarly, we have widgets like `Button` objects, which are like actors in a scene.

The `Stage` object passed to the `start()` method is created by JavaFX itself. You can see that the code creates a `Button` object with the label `OK` and puts the button into a `Scene` object with a size of 200 by 200 pixels. The `Stage` represents the frame of the GUI window and its title is set to `Project 1`. The `Scene` object is inserted into the `Stage` object and displayed.

Project 2

This example creates an application with two widgets: a button `okButton` (that can be clicked) and a text field named `field` (where the user can type in data). Again, we don't deal with button clicks or process the data entered in `field`.

```
import javafx.application.Application;
import javafx.scene.Scene;
```

```
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class Project2 extends Application {
   private Button okButton = new Button("OK");
   private TextField field = new TextField();

   @Override
   public void start(Stage primaryStage) throws Exception {
      GridPane pane = new GridPane();
      pane.add(field, 0, 0);
      pane.add(okButton, 0, 1);
      Scene scene = new Scene(pane);
      primaryStage.setTitle("Project 2");
      primaryStage.setScene(scene);
      primaryStage.show();
   }

   public static void main(String[] args) {
      Application.launch(null);
   }

}
```

The button object and text field are stored in the Scene via what is called a GridPane, which allows us arrange the widgets in a grid. The lines

```
      GridPane pane = new GridPane();
      pane.add(field, 0, 0);
      pane.add(okButton, 0, 1);
```

creates a `GridPane` object `pane`, adds the text field at column 0, row 0, and adds the button object at column 0, row 1 of `pane`.

Project 3

In this project, we extend Project 2 in three important ways:

1. Clicking on buttons now makes the program do things. The program has two buttons (`okButton` and `decrementButton`) and a text field. The text field displays the message

   ```
   Clicked <n> times
   ```

   Where `<n>` is a number. This number increases with each click of the `okButton` and decrements with each click of the `decrementButton`.

2. It shows how to do some graphics.

3. It shows how to create a new stage.

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
```

```
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class Project3 extends Application implements EventHandler<ActionEvent> {
   private Button okButton = new Button("OK");
   private Button decrementButton = new Button("Decrement");
   private TextField field = new TextField();
   private int counter = 1;
   Canvas canvas = new Canvas(100, 200);

   @Override
   public void start(Stage primaryStage) throws Exception {
      GridPane pane = new GridPane();
      pane.add(field, 0, 0);
      pane.add(okButton, 0, 1);
      pane.add(decrementButton, 0, 2);
      pane.add(canvas, 0, 3);
      Scene scene = new Scene(pane);
      primaryStage.setTitle("Project 3");
      primaryStage.setScene(scene);
      okButton.setOnAction(this);
      decrementButton.setOnAction(this);
      primaryStage.show();
   }

   public static void main(String[] args) {
      Application.launch(null);
   }

   @Override
   public void handle(ActionEvent event) {
      if (event.getSource() == okButton) {
         field.setText("Clicked " + counter + " times");
         counter++;
         GraphicsContext gc = canvas.getGraphicsContext2D();
         gc.strokeLine(20, 30, 40, 50);
         new S2();
      } else if (event.getSource() == decrementButton) {
         counter--;
         field.setText("Decremented to " + counter);
      }
   }
}
```

Execute the program and see how it behaves. Then go through the code (the fields and lines in stage() through the line

```
      primaryStage.setScene(scene);
```

The field

```
   Canvas canvas = new Canvas(100, 200);
```

Creates a `Canvas` object of size 100 by 200 pixels. On this object, we ca draw stuff like lines and circles. The canvas gets added as the fourth row of `pane`.

The line

```
okButton.setOnAction(this);
```

tells the `okButton` object that when that button is clicked, this object (that is, the instance of `Project3`) should be notified. In order for this to happen, the Project3 class (which wants to be notified) must implement the interface `EventHandler<ActionEvent>`, which means implementing the method `public void handle(ActionEvent event).`

Read the implementation of the `handle()` method. When `okButton` is clicked, this method gets called with a parameter that represents the event of clicking the button. The event object is of type `ActionEvent` and contains information about the event, including the "source" of the event, which could be `okButton`. So we compare the event source against `okButton` in the line

```
if (event.getSource() == okButton) {
```

If this condition is satisfied, the code sets the text field to the text

```
Clicked <n> times
```

in the code

```
field.setText("Clicked " + counter + " times");
```

It then draws a line on `canvas`. For this, we first get what is called a `GraphicsContext` using the following code.

```
GraphicsContext gc = canvas.getGraphicsContext2D();
```

Think of a `GraphicsContext` like a brush. A brush has a certain thickness can be given a certain color, etc., and using it one can draw lines, circles, write text, etc. Similar functionality exists for a `GraphicsContext` object as well.

The following line draws a line from (x = 20, y = 30) to (x = 40, y = 50). *The y-coordinate values increase as we go down the canvas.*

```
gc.strokeLine(20, 30, 40, 50);
```

The following line creates a `Stage` of type `S2`.

```
new S2();
```

The code for `S2`is given below. The three widgets (`b1`, `t1`, and `canvas`) are arranged vertically in a container called a `VBox`. Note that the way you add widgets to the `VBox` object. Note that the rest of the code in `S2` and the code for `deceremenButton` in `Project3` can be followed from the information already given.

```
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Button;
```

```
import javafx.scene.control.TextField;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class S2 extends Stage implements EventHandler<ActionEvent> {
   private Button b1 = new Button("try");
   private TextField t1 = new TextField();
   Canvas canvas = new Canvas(800, 400);

   public S2() {
      VBox box = new VBox();
      box.getChildren().add(b1);
      box.getChildren().add(t1);
      box.getChildren().add(canvas);
      Scene s = new Scene(box);
      b1.setOnAction(this);
      this.setScene(s);
      show();
   }

   @Override
   public void handle(ActionEvent event) {
      t1.setText("HAHA");
      GraphicsContext gc = canvas.getGraphicsContext2D();
      gc.strokeLine(20, 30, 40, 50);
   }
}
```

Project 4

This project introduces a new type called a `TextArea`, which can contain text with multiple columns and rows.

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class Project4 extends Application implements EventHandler<ActionEvent> {
   private Button okButton = new Button("OK");
   private TextField field = new TextField();
   private TextArea textArea = new TextArea();
   private int counter = 1;

   @Override
   public void start(Stage primaryStage) throws Exception {
      GridPane pane = new GridPane();
      pane.add(textArea, 0, 0);
      pane.add(field, 0, 1);
      pane.add(okButton, 0, 2);
      Scene scene = new Scene(pane);
      primaryStage.setTitle("Project 4");
      primaryStage.setScene(scene);
      okButton.setOnAction(this);
      primaryStage.show();
   }
```

```
   public static void main(String[] args) {
      Application.launch(null);
   }

   @Override
   public void handle(ActionEvent event) {
      textArea.appendText(field.getText() + "\n");
   }
}
```

This program allows us to concatenate the text in the text field and store the concatenated text in the text area. The code itself is not hard to follow, provided you have understood the previous discussion.

Project 5

This program shows how to process mouse clicks and draw filled circles and rectangles in different colors. It shows how to store a pane in another pane, which allows us create more complicated GUI interfaces in a systematic way.

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Button;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.GridPane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class Project5 extends Application implements EventHandler<ActionEvent> {
   private Button rectangleButton = new Button("Rectangle");
   private Button circleButton = new Button("Circle");
   private Button greenButton = new Button("Green");
   private Button redButton = new Button("Red");
   private Canvas canvas = new Canvas(300, 300);
   private Color color = Color.RED;
   private int shape = RECTANGLE;
   private static final int RECTANGLE = 1;
   private static final int CIRCLE = 2;

   @Override
   public void start(Stage primaryStage) throws Exception {
      GridPane pane = new GridPane();
      pane.add(canvas, 0, 0);
      GridPane buttonPane = new GridPane();
      buttonPane.add(rectangleButton, 0, 0);
      buttonPane.add(circleButton, 1, 0);
      buttonPane.add(greenButton, 0, 1);
      buttonPane.add(redButton, 1, 1);
      pane.add(buttonPane, 0, 1);
      Scene scene = new Scene(pane);
      primaryStage.setTitle("Project 5");
      primaryStage.setScene(scene);
      rectangleButton.setOnAction(this);
      greenButton.setOnAction(this);
      circleButton.setOnAction(this);
```

```
      redButton.setOnAction(this);
      canvas.setOnMouseClicked(new EventHandler<MouseEvent>() {

         @Override
         public void handle(MouseEvent event) {
            GraphicsContext gc = canvas.getGraphicsContext2D();
            gc.setFill(color);
            if (shape == CIRCLE) {
               gc.fillOval(event.getSceneX(), event.getSceneY(), 50, 50);
            } else if (shape == RECTANGLE) {
               gc.fillRect(event.getSceneX(), event.getSceneY(), 80, 40);
            }
         }
      });
      primaryStage.show();
   }

   public static void main(String[] args) {
      Application.launch(null);
   }

   @Override
   public void handle(ActionEvent event) {
      if (event.getSource() == redButton) {
         color = Color.RED;
      } else if (event.getSource() == greenButton) {
         color = Color.GREEN;
      } else if (event.getSource() == rectangleButton) {
         shape = RECTANGLE;
      } else if (event.getSource() == circleButton) {
         shape = CIRCLE;
      }
   }
}
```

This code

```
canvas.setOnMouseClicked(new EventHandler<MouseEvent>() {

         @Override
         public void handle(MouseEvent event) {
            GraphicsContext gc = canvas.getGraphicsContext2D();
            gc.setFill(color);
            if (shape == CIRCLE) {
               gc.fillOval(event.getSceneX(), event.getSceneY(), 50, 50);
            } else if (shape == RECTANGLE) {
               gc.fillRect(event.getSceneX(), event.getSceneY(), 80, 40);
            }
         }
      });
```

creates an anonymous class that processes mouse clicks. When the mouse is clicked, the `handle()` method gets called. It gets the `GraphicsContext` object and sets its fill color to a `Color` object (like `Color.RED`) and draws a circle or a rectangle. Refer to JDK documentation for detailed descriptions of the `fillOval()` and `fillRect()` methods. The `color` field is set in the `handle()` method.

Project 6

This project does not introduce any new concepts. It alters the functionality of Project 5 slightly. Execute it to see the difference and see where the difference comes from.

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;

public class Project6 extends Application implements EventHandler<ActionEvent> {
   private Button rectangleButton = new Button("Rectangle");
   private Button circleButton = new Button("Circle");
   private Button greenButton = new Button("Green");
   private Button redButton = new Button("Red");
   private Color color = Color.RED;
   private int shape = RECTANGLE;
   private static final int RECTANGLE = 1;
   private static final int CIRCLE = 2;
   Pane pane = new Pane();
   GridPane bigPane = new GridPane();

   @Override
   public void start(Stage primaryStage) throws Exception {
      pane.setPrefWidth(400);
      pane.setPrefHeight(400);
      bigPane.add(pane, 0, 0);
      GridPane buttonPane = new GridPane();
      buttonPane.add(rectangleButton, 0, 0);
      buttonPane.add(circleButton, 1, 0);
      buttonPane.add(greenButton, 0, 1);
      buttonPane.add(redButton, 1, 1);
      bigPane.add(buttonPane, 0, 1);
      Scene scene = new Scene(bigPane);
      primaryStage.setTitle("Project 6");
      primaryStage.setScene(scene);
      rectangleButton.setOnAction(this);
      greenButton.setOnAction(this);
      circleButton.setOnAction(this);
      redButton.setOnAction(this);
      pane.setOnMouseClicked(new EventHandler<MouseEvent>() {

         @Override
         public void handle(MouseEvent event) {
            if (shape == CIRCLE) {
               Circle circle = new Circle();
               circle.setCenterX(event.getSceneX());
               circle.setCenterY(event.getSceneY());
               circle.setFill(color);
               circle.setRadius(50);
               pane.getChildren().add(circle);

            } else if (shape == RECTANGLE) {
               Rectangle rectangle = new Rectangle();
               rectangle.setX(event.getSceneX());
```

```
            rectangle.setY(event.getSceneY());
            rectangle.setWidth(80);
            rectangle.setHeight(40);
            rectangle.setFill(color);
            pane.getChildren().add(rectangle);
         }
      }
   });
   primaryStage.show();

}

public static void main(String[] args) {
   Application.launch(null);
}

@Override
public void handle(ActionEvent event) {
   if (event.getSource() == redButton) {
      color = Color.RED;
   } else if (event.getSource() == greenButton) {
      color = Color.GREEN;
   } else if (event.getSource() == rectangleButton) {
      shape = RECTANGLE;
   } else if (event.getSource() == circleButton) {
      shape = CIRCLE;
   }
}

}
```

Installation of JavaFX

With the more recent releases of Java, FX is no longer downloaded as part of the JDK. Visit https://openjfx.io/openjfx-docs/ to see how to install FX.