

```

1  package com;
2
3
4  import com.fasterxml.jackson.dataformat.xml.XmlMapper;
5  import javafx.application.Application;
6  import javafx.scene.Scene;
7  import javafx.scene.control.Button;
8  import javafx.scene.control.TextField;
9  import javafx.scene.control.ToggleButton;
10 import javafx.scene.control.ToggleGroup;
11 import javafx.scene.layout.BorderPane;
12 import javafx.scene.layout.HBox;
13 import javafx.scene.layout.Pane;
14 import javafx.scene.layout.VBox;
15 import javafx.scene.shape.Circle;
16 import javafx.scene.shape.Rectangle;
17 import javafx.stage.Stage;
18
19 import java.io.*;
20 import java.nio.file.Files;
21 import java.nio.file.Paths;
22 import java.util.ArrayList;
23
24
25 public class Main extends Application {
26
27     private static final String ACTION_1 = "data.xml";
28     private final String alphabet = "[a-zA-Z]+";
29     private final String numeric = "[0-9]+";
30     private boolean chooseltRectangle = false;
31     private boolean init = false;
32     private int layoutX = 100;
33     private int layoutY = 70;
34     private int degree = 5;
35     private int rotateCounter = 0;
36     private ToggleGroup toggleGroup;
37     private ToggleButton tbCircle;
38     private ToggleButton tbRectan;
39     private Pane paneContents;
40     private BorderPane pane;
41     private ArrayList<Rectangle> rectangles;
42     private ArrayList<Circle> circles;
43     private ArrayList<RectangleObject> rectanglesObjects;
44     private ArrayList<CircleObject> circlesObjects;
45     private ObjectListHandler objectListHandler;
46     private String priority;
47     private Button clear;
48     private Button rotateObject;
49     private Button createObject;
50     private Button saveObject;
51     private Button edit;
52     private TextField recWidth;
53     private TextField recHeight;
54     private TextField rCircle;
55     private TextField index;
56
57     @Override
58     public void start(Stage primaryStage) {
59         pane = new BorderPane();
60         paneContents = new Pane();
61         pane.setPrefWidth(450);
62         pane.setPrefHeight(450);
63         pane.setLayoutY(60);
64
65         createSceneContents();
66
67         createObject.setOnAction(event -> createObject());
68
69         rotateObject.setOnAction(event -> rotateObject());
70
71         saveObject.setOnAction(event -> {
72             try {
73                 serializableToXML();
74             } catch (Exception e) {
75                 System.out.println(e.getMessage());
76             }
77         });
78
79         clear.setOnAction(event -> clearScene());

```

```

80     edit.setOnAction(event -> editObject());
81
82     Scene scene = new Scene(paneContents, 650, 650);
83     primaryStage.setScene(scene);
84     primaryStage.show();
85
86 }
87
88 private void editObject() {
89     int indexObject = Integer.valueOf(index.getText());
90     if (toggleGroup.getSelectedToggle() == tbRectan) {
91
92         if ((indexObject < rectangles.size()) && (checkInputRectangle())) {
93             int height = Integer.valueOf(recHeight.getText());
94             int width = Integer.valueOf(recWidth.getText());
95             rectangles.get(indexObject).setHeight(height);
96             rectangles.get(indexObject).setWidth(width);
97             rectanglesObjects.get(indexObject).setValueHeight(height);
98             rectanglesObjects.get(indexObject).setValueWidth(width);
99         }
100     } else if (toggleGroup.getSelectedToggle() == tbCircle) {
101         if ((indexObject < circles.size()) && (checkInputCircle())) {
102             int rCircI = Integer.valueOf(rCircle.getText());
103             circles.get(indexObject).setRadius(rCircI);
104             circlesObjects.get(indexObject).setRadius(rCircI);
105         }
106     }
107 }
108
109 private boolean checkInputRectangle() {
110     String width = recWidth.getText();
111     String height = recHeight.getText();
112     if ((width.isEmpty()) || (height.isEmpty())
113         || (width.matches(alphabet))
114         || (height.matches(alphabet))
115         || (width.matches(numeric) && width.length() > 3)
116         || (height.matches(numeric) && height.length() > 3)) {
117         return false;
118     } else {
119         return true;
120     }
121 }
122
123 private boolean checkInputCircle() {
124     String rCircI = rCircle.getText();
125     if ((rCircI.isEmpty())
126         || (rCircI.matches(alphabet))
127         || (rCircI.matches(numeric) && rCircI.length() > 3)) {
128         return false;
129     } else {
130         return true;
131     }
132 }
133
134 private void createSceneContents() {
135     createObject = new Button("Create");
136     rotateObject = new Button("Rotate");
137     saveObject = new Button("Save!!");
138     clear = new Button("Clear ");
139     edit = new Button("Edit ");
140
141     VBox buttons = new VBox(10);
142     HBox toggleBox = new HBox(20);
143
144     toggleBox.setLayoutX(10);
145     toggleBox.setLayoutY(10);
146     buttons.setLayoutX(570);
147     buttons.setLayoutY(510);
148
149     VBox rectangleSize = new VBox(3); // to put the rectangle toggle and his size field
150     HBox rectangleSizeField = new HBox(4);
151     VBox circleRadius = new VBox(3); // to put the Circle toggle and his radius field
152     VBox editVBox = new VBox(3);
153
154     recWidth = new TextField();
155     recWidth.setPrefSize(60, 20);
156
157     recHeight = new TextField();

```

```

159     recHeight.setPrefSize(60, 20);
160
161     rCircle = new TextField();
162     rCircle.setPrefSize(60, 20);
163
164     index = new TextField();
165     index.setPrefSize(60, 20);
166     index.setPrefSize(124, 20);
167
168
169     toggleGroup = new ToggleGroup();
170     tbCircle = new ToggleButton("Circle");
171     tbCircle.setPrefSize(60, 20);
172     tbRectan = new ToggleButton("Triangle");
173     tbRectan.setPrefSize(124, 20);
174     tbRectan.setSelected(true);
175
176     rectangleSizeField.getChildren().addAll(recHeight, recWidth);
177     rectangleSize.getChildren().addAll(tbRectan, rectangleSizeField);
178     circleRadius.getChildren().addAll(tbCircle, rCircle);
179     editVBox.getChildren().addAll(edit, index);
180
181     tbCircle.setToggleGroup(toggleGroup);
182     tbRectan.setToggleGroup(toggleGroup);
183
184     toggleBox.getChildren().addAll(rectangleSize, circleRadius, editVBox);
185     buttons.getChildren().addAll(createObject, rotateObject, clear, saveObject);
186
187     paneContents.getChildren().addAll(toggleBox, buttons, pane);
188
189     if (init) {
190         generateObjects(objectListHandler);
191     }
192 }
193
194 private void generateObjects(ObjectListHandler objectClasses) {
195     rectangles = new ArrayList<>();
196     circles = new ArrayList<>();
197     int counter = priority.length();
198     int k = 0;
199     int c = 0;
200     int r = 0;
201
202     while (counter != 0) {
203         if (priority.charAt(k) == 'r') {
204             Rectangle rectangle = objectClasses.getRectangles().get(r).rectangleObject();
205             rectangles.add(rectangle);
206             pane.getChildren().add(rectangle);
207             rotateCounter++;
208             chooseRectan = true;
209             r++;
210         } else {
211             Circle circle = objectClasses.getCircles().get(c).circleObject();
212             circles.add(circle);
213             pane.getChildren().add(circle);
214             c++;
215         }
216         k++;
217         counter--;
218     }
219 }
220
221 private void setRectangleLayouts(Rectangle rectangle, RectangleObject rectangleObject) {
222
223     rectangle.setLayoutX(layoutX);
224     rectangle.setLayoutY(layoutY);
225     rectangleObject.setLayoutX(layoutX);
226     rectangleObject.setLayoutY(layoutY);
227
228 }
229
230 private void generateLayouts() {
231
232     layoutX += ((layoutX * 5) / 100);
233     layoutY += ((layoutY * 15) / 100);
234 }
235
236 private void setcircleLayouts(Circle circle) {
237

```

```

238     circle.setCenterX(250);
239     circle.setCenterY(250);
240 }
241
242
243 private void createObject() {
244
245     if (toggleGroup.getSelectedToggle() == tbRectan) {
246         chooseltRectangle = true;
247         // String width = recWidth.getText();
248         // String height = recHeight.getText();
249
250         if (!checkInputRectangle()) {
251             priority += 'R';
252             addObject(new RectangleObject(), false);
253         } else {
254             priority += 'R';
255             addObject(new RectangleObject(), true);
256         }
257     } else if (toggleGroup.getSelectedToggle() == tbCircle) {
258
259         if (!checkInputCircle()) {
260             priority += 'C';
261             addObject(new CirclObject(), false);
262         } else {
263             priority += 'C';
264             addObject(new CirclObject(), true);
265         }
266     }
267 }
268
269 private void addObject(ObjectClass objectClass, boolean b) {
270
271     if (((rectangles.isEmpty()) && (objectClass instanceof RectangleObject)) {
272         Rectangle rectangle;
273         RectangleObject rectangleObject = new RectangleObject();
274         if (!b) {
275             rectangle = rectangleObject.rectangleObject(300, 200);
276         } else {
277             rectangle = rectangleObject.rectangleObject(Integer.valueOf(recHeight.getText()), Integer.valueOf(recWidth.
278 getText()));
279         }
280         setRectangleLayouts(rectangle, rectangleObject);
281         pane.getChildren().add(rectangle);
282         rectangles = new ArrayList<>();
283         rectanglesObjects = new ArrayList<>();
284         rectangles.add(rectangle);
285         rectanglesObjects.add(rectangleObject);
286         generateLayouts();
287         rotateCounter++;
288     } else if ((circles.isEmpty()) && (objectClass instanceof CirclObject)) {
289         Circle circle;
290         CirclObject circlObject = new CirclObject();
291         if (!b) {
292             circle = circlObject.circlObject(200);
293         } else {
294             circle = circlObject.circlObject(Integer.valueOf(rCircle.getText()));
295         }
296         setcirculeLayouts(circle);
297         pane.getChildren().add(circle);
298         circles = new ArrayList<>();
299         circlesObjects = new ArrayList<>();
300         circles.add(circle);
301         circlesObjects.add(circlObject);
302     } else {
303         if (objectClass instanceof RectangleObject) {
304             regenerateLastObject(rectanglesObjects.get(rectangles.size() - 1), b);
305             rotateCounter++;
306         } else {
307             regenerateLastObject(circlesObjects.get(circles.size() - 1), b);
308         }
309     }
310 }
311 }
312 }
313
314 private void regenerateLastObject(ObjectClass objectClass, boolean size) {
315

```

```

316   ObjectClass thisObjectClass;
317   int width = 0;
318   int height = 0;
319   int radius = 0;
320   if (!size) {
321       thisObjectClass = ObjectGenerating.generateNew(objectClass);
322   } else {
323
324       if (ObjectGenerating.getObjectType(objectClass)) {
325           width = Integer.valueOf(recWidth.getText());
326           height = Integer.valueOf(recHeight.getText());
327           thisObjectClass = new RectangleObject(height, width);
328       } else {
329           radius = Integer.valueOf(rCircle.getText());
330           thisObjectClass = new CircleObject(radius);
331       }
332   }
333   if (ObjectGenerating.getObjectType(objectClass)) {
334       Rectangle rectangle = ((RectangleObject) thisObjectClass).rectangleObject();
335       setRectangleLayouts(rectangle, (RectangleObject) thisObjectClass);
336       pane.getChildren().add(rectangle);
337       rectangles.add(rectangle);
338       rectanglesObjects.add((RectangleObject) thisObjectClass);
339       generateLayouts();
340   } else {
341       Circle circle = ((CircleObject) thisObjectClass).circleObject();
342       setCircleLayouts(circle);
343       pane.getChildren().add(circle);
344       circles.add(circle);
345       circlesObjects.add((CircleObject) thisObjectClass);
346   }
347
348
349 }
350
351
352 private void rotateObject() {
353     if (chooseRect) {
354         if (rotateCounter > 0) {
355             for (int i = rotateCounter - 1; i < rectangles.size(); i++) {
356                 rectangles.get(i).setRotate(degree);
357                 degree += 5;
358             }
359             rotateCounter--;
360         } else {
361             rotateCounter = rectangles.size() - 1;
362         }
363     }
364 }
365
366
367 private void clearScene() {
368     pane.getChildren().clear();
369     rectangles = null;
370     circles = null;
371     priority = "";
372     layoutX = 100;
373     layoutY = 70;
374
375 }
376
377 private void serializableToXML() throws IOException {
378     ObjectListHandler objectListHandlerTem = new ObjectListHandler();
379     objectListHandlerTem.setRectangles(rectanglesObjects);
380     objectListHandlerTem.setCircles(circlesObjects);
381     objectListHandlerTem.setPriority(priority);
382     try {
383         XmlMapper xmlMapper = new XmlMapper();
384         xmlMapper.writeValue(new File(ACTION_1), objectListHandlerTem);
385     } catch (FileNotFoundException e) {
386         System.out.println(e.getMessage());
387     }
388 }
389
390 public void init() throws IOException {
391     priority = "";
392     XmlMapper xmlMapper = new XmlMapper();
393     File file = new File(ACTION_1);

```

```
395 // ObjectListHandler objectListHandler = new ObjectListHandler();
396 if (file.exists()) {
397     String xml = new String(Files.readAllBytes(Paths.get(ACTION_1)));
398     objectListHandler = xmlMapper.readValue(xml, ObjectListHandler.class);
399     init = objectListHandler != null;
400     if (objectListHandler != null) {
401         circlesObjects = objectListHandler.getCircles();
402         rectanglesObjects = objectListHandler.getRectangles();
403         priority = objectListHandler.getPriorety();
404     }
405 }
406 }
407 }
408 }
409 }
410 }
411 public static void main(String[] args) {
412     launch(args);
413 }
414 }
415 }
```

```
1 package com;  
2  
3 public class Controller {  
4 }  
5
```

```
1 package com;
2
3 import javafx.scene.paint.Color;
4 import javafx.scene.shape.Circle;
5 import java.io.Serializable;
6
7
8 public class CirclObject extends ObjectClass implements Serializable {
9
10     public int radius;
11
12     public Circle circlObject(int radius) {
13
14         this.radius = radius;
15         randomColour();
16         return getDrawwable();
17     }
18
19     public CirclObject() {
20         // setColor(red,green,blue);
21     }
22
23     public Circle circlObject() {
24         randomColour();
25         return getDrawwable();
26     }
27
28
29     private Circle getDrawwable() {
30         Circle circle = new Circle();
31         circle.setRadius(radius);
32         circle.setFill(getPaint());
33         circle.setStroke(Color.DARKGRAY);
34         circle.setStrokeWidth(2);
35         circle.setCenterX(250);
36         circle.setCenterY(250);
37         return circle;
38     }
39
40
41
42
43     public void setRadius(int radius) {
44
45         this.radius = radius;
46     }
47
48     public CirclObject(int radius) {
49         this.radius = radius;
50     }
51
52     public int getRadius() {
53
54         return radius;
55     }
56
57 }
58
59
```



```
1 package com;
2
3 import javafx.scene.paint.Color;
4 import javafx.scene.paint.Paint;
5
6 import java.io.Serializable;
7 import java.util.Random;
8
9 public class ObjectClass implements Serializable {
10     int red;
11     int green;
12     int blue;
13
14     public void randomColour() {
15         Random random = new Random();
16         this.red = random.nextInt(255);
17         this.green = random.nextInt(255);
18         this.blue = random.nextInt(255);
19     }
20
21     Paint getPaint() {
22         return Color.rgb(red, green, blue);
23     }
24
25     public void setColor(int red, int green, int blue) {
26         this.red = red;
27         this.green = green;
28         this.blue = blue;
29     }
30
31     public int getRed() {
32         return red;
33     }
34
35     public int getGreen() {
36         return green;
37     }
38
39     public int getBlue() {
40         return blue;
41     }
42 }
43
44
```

```

1  package com;
2
3  import com.ObjectClass;
4  import javafx.scene.paint.Color;
5  import javafx.scene.shape.Rectangle;
6
7  import java.io.Serializable;
8
9  class RectangleObject extends ObjectClass implements Serializable {
10
11     public int valueHeight;
12     public int valueWidth;
13     public int layoutX;
14     public int layoutY;
15
16
17     public Rectangle rectangleObject(int valueHeight, int valueWidth) {
18
19         this.valueHeight = valueHeight;
20         this.valueWidth = valueWidth;
21         randomColour();
22         return getDrawwable();
23     }
24
25     public Rectangle rectangleObject() {
26
27         return getDrawwable();
28     }
29
30     private Rectangle getDrawwable() {
31         Rectangle rectangle = new Rectangle(valueHeight, valueWidth);
32         rectangle.setFill(getPaint());
33         rectangle.setStroke(Color.DARKGRAY);
34         rectangle.setStrokeWidth(2);
35         rectangle.setArcHeight(10);
36         rectangle.setArcWidth(10);
37         rectangle.setLayoutX(layoutX);
38         rectangle.setLayoutY(layoutY);
39         return rectangle;
40     }
41
42
43     public RectangleObject() {
44     }
45
46     public RectangleObject(int valueHeight, int valueWidth) {
47         this.valueHeight = valueHeight;
48         this.valueWidth = valueWidth;
49         randomColour();
50     }
51
52
53     public int getValueHeight() {
54         return valueHeight;
55     }
56
57     public void setValueHeight(int valueHeight) {
58         this.valueHeight = valueHeight;
59     }
60
61     public int getValueWidth() {
62         return valueWidth;
63     }
64
65     public void setValueWidth(int valueWidth) {
66         this.valueWidth = valueWidth;
67     }
68
69     public int getLayoutX() {
70         return layoutX;
71     }
72
73     public void setLayoutX(int layoutX) {
74         this.layoutX = layoutX;
75     }
76
77     public int getLayoutY() {
78         return layoutY;
79     }

```

```
80
81  public void setLayoutY(int layoutY) {
82      this.layoutY = layoutY;
83  }
84
85 }
86
```

```

1  package com;
2
3
4  /**
5   * this class made it for
6   * generate rectangle with
7   * new size minus 10%
8   */
9  class ObjectGenerating {
10
11
12
13     /**
14      * set the old value minus
15      * 10% from original size
16      */
17
18     static ObjectClass generateNew(ObjectClass objectClass){
19
20         if (getObjectType(objectClass)){
21             RectangleObject rectangleObject = (RectangleObject) objectClass;
22             int height = rectangleObject.getValueHeight()-(rectangleObject.getValueHeight()*10/100);
23             int width = rectangleObject.getValueWidth()-(rectangleObject.getValueWidth()*10/100);
24             return new RectangleObject(height,width);
25         }else{
26             CircleObject circleObject = (CircleObject) objectClass;
27             int radius = (circleObject.getRadius()-(circleObject.getRadius()*10/100));
28             return new CircleObject(radius);
29         }
30     }
31 }
32 //
33 // static com.ObjectClass generateNewSize(com.ObjectClass objectClass , int Height, int Width, int Radius){
34 //
35 //     if (getObjectType(objectClass)){
36 //         RectangleObject ro = new RectangleObject();
37 //
38 //         ro.setValueHeight(Height);
39 //         ro.setValueWidth (Width);
40 //         return ro;
41 //     }else{
42 //         com.CircleObject circle = new com.CircleObject();
43 //         circle.setRadius(Radius);
44 //         return circle;
45 //     }
46 // }
47 // }
48 // }
49
50 static boolean getObjectType(ObjectClass objectClass){
51
52     return (objectClass instanceof RectangleObject);
53 }
54 }
55

```

```
1 package com;
2
3 import com.CirclObject;
4
5 import java.io.Serializable;
6 import java.util.ArrayList;
7
8 public class ObjectListHandler implements Serializable {
9     private ArrayList<RectangleObject> rectangles ;
10    private ArrayList<CirclObject> circles ;
11    private String priorety ;
12
13    public String getPriorety() {
14        return priorety;
15    }
16
17    public void setPriorety(String priorety) {
18        this.priorety = priorety;
19    }
20
21    public ArrayList<RectangleObject> getRectangles() {
22        return rectangles;
23    }
24
25    public void setRectangles(ArrayList<RectangleObject> rectangles) {
26        this.rectangles = rectangles;
27    }
28
29    public ArrayList<CirclObject> getCircles() {
30        return circles;
31    }
32
33    public void setCircles(ArrayList<CirclObject> circles) {
34        this.circles = circles;
35    }
36
37
38
39 }
40
```