File - /Users/ziadelsarrih/Desktop/Labs/OOP/oop4/src/main/java/com/Demo.java

```
1  package com;
2
3
4  import com.fasterxml.jackson.dataformat.xml.XmlMapper;
5  import com.filehandeling.FileReader;
6  import com.filehandeling.FileWriter;
7  import javafx.application.Application;
8  import javafx.scene.Scene;
9  import javafx.scene.control.Button;
10 import javafx.scene.control.TextField;
11 import javafx.scene.control.ToggleButton;
12 import javafx.scene.control.ToggleGroup;
13 import javafx.scene.layout.BorderPane;
14 import javafx.scene.layout.HBox;
15 import javafx.scene.layout.Pane;
16 import javafx.scene.layout.VBox;
17 import javafx.scene.shape.Circle;
18 import javafx.scene.shape.Rectangle;
19 import javafx.stage.Stage;
20 import net.openhft.compiler.CompilerUtils;
21
22 import java.io.*;
23 import java.nio.file.Files;
24 import java.nio.file.Paths;
25 import java.util.ArrayList;
26
27
28 public class Demo extends Application {
29
30     private static final String ACTION_1 = "data.xml";
31     private static final String A_Z_A_Z = "[a-zA-Z]+";
32     private static final String NUMERIC = "[0-9]+";
33     private boolean chooseItRectangle = false;
34     private boolean getContentsCreated = false;
35     private boolean init = false;
36     private boolean contentsCreated = false;
37     private String object;
38     private String object2;
39     private int layoutX = 100;
40
41     private int layoutY = 70;
42     private int degree = 5;
43     private int rotateCounter = 0;
44     private ToggleGroup toggleGroup;
45     private ToggleButton tbCircle;
46     private ToggleButton tbRectan;
47     private Pane paneContents;
48     private BorderPane pane;
49     private ArrayList<Rectangle> rectangles;
50     private ArrayList<Circle> circles;
51     private ArrayList<ObjectClass> rectanglesObjects;
52     private ArrayList<ObjectClass> circlesObjects;
53     private ObjectListHandler objectListHandler;
54     private String priority;
55     private Button clear;
56     private Button rotateObject;
57     private Button createObject;
58     private Button saveObject;
59     private Button edit;
60     private Button classLoader;
61     private TextField recWidth;
62     private TextField recHeight;
63     private TextField rCircle;
64     private TextField index;
65     private Class aClass;
66     private Class aClass2;
67
68
69     @Override
70     public void start(Stage primaryStage) throws IllegalAccessException, InstantiationException {
71         pane = new BorderPane();
72         paneContents = new Pane();
73         pane.setPrefWidth(450);
74         pane.setPrefHeight(450);
75         pane.setLayoutY(60);
76
77         createSceneContents();
78
79         createObject.setOnAction(event -> {
```

```
80              if (!contentsCreated) {
81                  createOtherSceneContents();
82                  if (!getContentsCreated) {
83                      getSources();
84                  }
85                  getObject('r');
86                  getObject('c');
87              }
88              try {
89                  createObject();
90              } catch (IllegalAccessException | InstantiationException e) {
91                  e.printStackTrace();
92              }
93          });
94
95          classLoader.setOnAction(event -> {
96              if (!getContentsCreated) {
97                  getSources();
98              }
99          });
100
101         rotateObject.setOnAction(event -> rotateObject());
102
103         saveObject.setOnAction(event -> {
104             try {
105                 serializableToXML();
106             } catch (Exception e) {
107                 System.out.println(e.getMessage());
108             }
109         });
110
111         clear.setOnAction(event -> clearScene());
112         edit.setOnAction(event -> editObject());
113
114
115         Scene scene = new Scene(paneContents, 650, 650);
116         primaryStage.setScene(scene);
117         primaryStage.show();
118
119     }
120
121     private void editObject() {
122         String inputIndex = index.getText();
123         if ((!inputIndex.isEmpty()) && (!inputIndex.matches(A_Z_A_Z))) {
124             int indexObject = Integer.valueOf(index.getText());
125             if (toggleGroup.getSelectedToggle() == tbRectan) {
126
127                 if ((indexObject < rectangles.size()) && (checkInputRectangle())) {
128                     int height = Integer.valueOf(recWidth.getText());
129                     int width = Integer.valueOf(recHeight.getText());
130                     rectangles.get(indexObject).setHeight(height);
131                     rectangles.get(indexObject).setWidth(width);
132                     rectanglesObjects.get(indexObject).setValueHeight(width);
133                     rectanglesObjects.get(indexObject).setValueWidth(height);
134                 }
135             } else if (toggleGroup.getSelectedToggle() == tbCircle) {
136                 if ((indexObject < circles.size()) && (checkInputCircle())) {
137                     int rCircl = Integer.valueOf(rCircle.getText());
138                     circles.get(indexObject).setRadius(rCircl);
139                     circlesObjects.get(indexObject).setRadius(rCircl);
140                 }
141             }
142
143         }
144
145     }
146
147     private boolean checkInputRectangle() {
148         String width = recWidth.getText();
149         String height = recHeight.getText();
150         if ((width.isEmpty()) || (height.isEmpty())
151                 || (width.matches(A_Z_A_Z))
152                 || (height.matches(A_Z_A_Z))
153                 || (width.matches(NUMERIC) && width.length() > 3)
154                 || (height.matches(NUMERIC) && height.length() > 3)) {
155             return false;
156         } else {
157             return true;
158         }
```

```
159     }
160
161     private boolean checkInputCircle() {
162         String rCircl = rCircle.getText();
163         if ((rCircl.isEmpty())
164                 || (rCircl.matches(A_Z_A_Z))
165                 || (rCircl.matches(NUMERIC) && rCircl.length() > 3)) {
166             return false;
167         } else {
168             return true;
169         }
170     }
171
172     private void createSceneContents() throws InstantiationException, IllegalAccessException {
173
174         createObject = new Button("Create");
175         rotateObject = new Button("Rotate");
176         saveObject = new Button("Save!!");
177         clear = new Button("Clear  ");
178         classLoader = new Button("Class.L");
179         edit = new Button("Edit ");
180
181         VBox buttons = new VBox(10);
182         buttons.setLayoutX(570);
183         buttons.setLayoutY(470);
184         buttons.getChildren().addAll(classLoader, createObject, rotateObject, clear, saveObject);
185
186         if (init) {
187             createOtherSceneContents();
188             contentsCreated = true;
189             getContentsCreated = true;
190             getSources();
191             getObject('r');
192             getObject('c');
193             generateObjects(objectListHandler);
194         }
195         paneContents.getChildren().add(buttons);
196     }
197
198     private void createOtherSceneContents() {
199
200
201         contentsCreated = true;
202         HBox toggleBox = new HBox(20);
203         toggleBox.setLayoutX(10);
204         toggleBox.setLayoutY(10);
205
206         VBox rectangleSize = new VBox(3);  // to put the rectangle toggle and his size field
207         HBox rectangleSizeField = new HBox(4);
208         VBox circleRadius = new VBox(3); // to put the Circule toggle and his radius field
209         VBox editVBox = new VBox(3);
210
211         recWidth = new TextField();
212         recWidth.setPrefSize(60, 20);
213
214         recHeight = new TextField();
215         recHeight.setPrefSize(60, 20);
216
217         rCircle = new TextField();
218         rCircle.setPrefSize(60, 20);
219
220         index = new TextField();
221         index.setPrefSize(60, 20);
222         index.setPrefSize(124, 20);
223
224
225         toggleGroup = new ToggleGroup();
226         tbCircle = new ToggleButton("Circle");
227         tbCircle.setPrefSize(60, 20);
228         tbRectan = new ToggleButton("Rectangle");
229         tbRectan.setPrefSize(124, 20);
230
231         rectangleSizeField.getChildren().addAll(recHeight, recWidth);
232         rectangleSize.getChildren().addAll(tbRectan, rectangleSizeField);
233         circleRadius.getChildren().addAll(tbCircle, rCircle);
234         editVBox.getChildren().addAll(edit, index);
235
236         tbCircle.setToggleGroup(toggleGroup);
237         tbRectan.setToggleGroup(toggleGroup);
```

```
238
239         toggleBox.getChildren().addAll(rectangleSize, circleRadius, editVBox);
240         paneContents.getChildren().addAll(toggleBox, pane);
241     }
242
243     private void generateObjects(ObjectListHandler objectClasses) throws IllegalAccessException,
    InstantiationException {
244         rectangles = new ArrayList<>();
245         circles = new ArrayList<>();
246         int counter = priority.length();
247         int k = 0;
248         int c = 0;
249         int r = 0;
250         int height;
251         int width;
252         ObjectClass objectClass = null;
253
254         while (counter != 0) {
255             if (priority.charAt(k) == 'r') {
256
257                 objectClass = (ObjectClass) aClass.newInstance();//getObject('r');
258
259                 height = objectClasses.getRectangles().get(r).getValueHeight();
260                 width = objectClasses.getRectangles().get(r).getValueWidth();
261
262                 Rectangle rectangle = (Rectangle) objectClass.createObject(height, width);
263                 rectangle.setLayoutX(objectClasses.getRectangles().get(r).getLayoutX());
264                 rectangle.setLayoutY(objectClasses.getRectangles().get(r).getLayoutY());
265                 rectangles.add(rectangle);
266                 pane.getChildren().add(rectangle);
267                 rotateCounter++;
268                 chooseItRectangle = true;
269                 r++;
270             } else {
271
272                 objectClass = (ObjectClass) aClass2.newInstance();//getObject('c');
273
274
275                 Circle circle = (Circle) objectClass.createObject(objectClasses.getCircles().get(c).getRadius());
276                 circles.add(circle);
277                 pane.getChildren().add(circle);
278                 c++;
279             }
280             k++;
281             counter--;
282         }
283     }
284
285     private ObjectClass getObject(char type) {
286
287
288         try {
289             if (type == 'r') {
290
291                 aClass = CompilerUtils.CACHED_COMPILER.loadFromJava("com.RectangleObject", object);
292                 return (ObjectClass) aClass.newInstance();
293             } else if (type == 'c') {
294
295                 aClass2 = CompilerUtils.CACHED_COMPILER.loadFromJava("com.CirclObject", object2);
296                 return (ObjectClass) aClass2.newInstance();
297             }
298         } catch (InstantiationException | IllegalAccessException | ClassNotFoundException e) {
299             e.printStackTrace();
300         }
301         return null;
302     }
303
304     private void setRectangleLayouts(Rectangle rectangle, ObjectClass objectClass) {
305
306         rectangle.setLayoutX(layoutX);
307         rectangle.setLayoutY(layoutY);
308         objectClass.setLayoutX(layoutX);
309         objectClass.setLayoutY(layoutY);
310
311     }
312
313     private void generateLayouts() {
314
315         layoutX += ((layoutX * 5) / 100);
```

```
316         layoutY += ((layoutY * 15) / 100);
317     }
318
319     private void setcirculeLayouts(Circle circle) {
320
321         circle.setCenterX(250);
322         circle.setCenterY(250);
323     }
324
325
326     private void createObject() throws IllegalAccessException, InstantiationException {
327         ObjectClass objectClass;
328
329         if (toggleGroup.getSelectedToggle() == tbRectan) {
330             chooseItRectangle = true;
331             priority += 'r';
332             objectClass = (ObjectClass) aClass.newInstance();//getObject('r');
333
334             addObject(objectClass, checkInputRectangle(), 'r');//
335
336         } else if (toggleGroup.getSelectedToggle() == tbCircle) {
337             priority += 'c';
338
339
340             objectClass = (ObjectClass) aClass2.newInstance();//getObject('c');
341
342             addObject(objectClass, checkInputCircle(), 'c');
343         }
344     }
345
346     private void getSources() {
347         FileReader fileReader = new FileReader();
348         String pathRes = "/Users/ziadelsarrih/Desktop/Labs/OOP/oop4/src/main/java/outResources/";
349         byte[] circlObject = fileReader.readFile(pathRes + "Obj1.txt");
350         byte[] rectangleObject = fileReader.readFile(pathRes + "Obj2.txt");
351         FileWriter fileWriter = new FileWriter();
352         object2 = new String(circlObject);
353         object = new String(rectangleObject);
354         String pathDest = "/Users/ziadelsarrih/Desktop/Labs/OOP/oop4/src/main/java/com/";
355         fileWriter.writeByte(circlObject, pathDest + "CirclObject.java");
356         fileWriter.writeByte(rectangleObject, pathDest + "RectangleObject.java");
357
358     }
359
360     private void addObject(ObjectClass objectClass, boolean b, char type) {
361
362         if (((rectanglesObjects == null) && (type == 'r'))) {
363
364             Rectangle rectangle;
365             if (!b) {
366                 rectangle = (Rectangle) objectClass.createObject(300, 200);
367             } else {
368                 rectangle = (Rectangle) objectClass.createObject(Integer.valueOf(recHeight.getText()), Integer.valueOf(recWidth.
    getText()));
369             }
370             setRectangleLayouts(rectangle, objectClass);
371             pane.getChildren().add(rectangle);
372             rectangles = new ArrayList<>();
373             rectanglesObjects = new ArrayList<>();
374             rectangles.add(rectangle);
375             rectanglesObjects.add(objectClass);
376             generateLayouts();
377             rotateCounter++;
378         } else if ((circlesObjects == null) && (type == 'c')) {
379
380             Circle circle;
381             if (!b) {
382                 circle = (Circle) objectClass.createObject(200);
383             } else {
384                 circle = (Circle) objectClass.createObject(Integer.valueOf(rCircle.getText()));
385             }
386             setcirculeLayouts(circle);
387             pane.getChildren().add(circle);
388             circles = new ArrayList<>();
389             circles.add(circle);
390             circlesObjects = new ArrayList<>();
391             circlesObjects.add(objectClass);
392         } else {
393             if (type == 'r') {
```

```
394
395                 regenerateLastObject(objectClass, b, type);
396                 rotateCounter++;
397             } else {
398
399                 regenerateLastObject(objectClass, b, type);
400
401             }
402
403         }
404     }
405
406     private void regenerateLastObject(ObjectClass objectClass, boolean size, char type) {
407
408
409         int width = 0;
410         int height = 0;
411         int radius = 0;
412         if (!size) {
413             objectClass = generateNew(objectClass, type);
414         } else {
415
416             if (type == 'r') {
417                 width = Integer.valueOf(recWidth.getText());
418                 height = Integer.valueOf(recHeight.getText());
419                 objectClass.setValueHeight(height);
420                 objectClass.setValueWidth(width);
421             } else {
422                 radius = Integer.valueOf(rCircle.getText());
423                 objectClass.setRadius(radius);
424             }
425         }
426         if (type == 'r') {
427             Rectangle rectangle = (Rectangle) objectClass.createObject();
428             setRectangleLayouts(rectangle, objectClass);
429             pane.getChildren().add(rectangle);
430             rectangles.add(rectangle);
431             rectanglesObjects.add(objectClass);
432             generateLayouts();
433         } else {
434             Circle circle = (Circle) objectClass.createObject();
435             setcirculeLayouts(circle);
436             pane.getChildren().add(circle);
437             circles.add(circle);
438             circlesObjects.add(objectClass);
439         }
440
441
442     }
443
444     public ObjectClass generateNew(ObjectClass objectClass, char type) {
445         ObjectClass o;
446         if (type == 'r') {
447             o = rectanglesObjects.get(rectangles.size() - 1);
448             int height = o.getValueHeight() - ((o.getValueHeight() * 10) / 100);
449             int width = o.getValueWidth() - ((o.getValueWidth() * 10) / 100);
450             objectClass.setValueHeight(height);
451             objectClass.setValueWidth(width);
452             return objectClass;
453         } else {
454             o = circlesObjects.get(circles.size() - 1);
455             int radius = (o.getRadius() - (o.getRadius() * 10 / 100));
456             objectClass.setRadius(radius);
457             return objectClass;
458
459         }
460     }
461
462
463     private void rotateObject() {
464         if (chooseItRectangle) {
465             if (rotateCounter > 0) {
466                 for (int i = rotateCounter - 1; i < rectangles.size(); i++) {
467                     rectangles.get(i).setRotate(degree);
468                     degree += 5;
469                 }
470                 rotateCounter--;
471             } else {
472                 rotateCounter = rectangles.size() - 1;
```

```
473             }
474
475         }
476     }
477
478     private void clearScene() {
479         pane.getChildren().clear();
480         rectangles = null;
481         circles = null;
482         rectanglesObjects = null;
483         circlesObjects = null;
484         priority = "";
485         layoutX = 100;
486         layoutY = 70;
487
488     }
489
490     private void serializableToXML() throws IOException {
491         ObjectListHandler objectListHandlerTem = new ObjectListHandler();
492         objectListHandlerTem.setRectangles(rectanglesObjects);
493         objectListHandlerTem.setCircles(circlesObjects);
494         objectListHandlerTem.setPriorety(priority);
495         //  if ((rectanglesObjects != null) && (circlesObjects != null)) {
496         try {
497             XmlMapper xmlMapper = new XmlMapper();
498             xmlMapper.writeValue(new File(ACTION_1), objectListHandlerTem);
499         } catch (FileNotFoundException e) {
500             System.out.println(e.getMessage());
501         }
502         //  }
503     }
504
505     public void init() throws IOException {
506
507         priority = "";
508         XmlMapper xmlMapper = new XmlMapper();
509         File file = new File(ACTION_1);
510         // ObjectListHandler objectListHandler = new ObjectListHandler();
511         if (file.exists()) {
512             String xml = new String(Files.readAllBytes(Paths.get(ACTION_1)));
513             objectListHandler = xmlMapper.readValue(xml, ObjectListHandler.class);
514             init = objectListHandler != null;
515             if (objectListHandler != null) {
516                 circlesObjects = objectListHandler.getCircles();
517                 rectanglesObjects = objectListHandler.getRectangles();
518                 priority = objectListHandler.getPriorety();
519             }
520
521         }
522
523     }
524
525
526     public static void main(String[] args) {
527         launch(args);
528     }
529 }
530
```

File - /Users/ziadelsarrih/Desktop/Labs/OOP/oop4/src/main/java/com/Controller.java

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

File - /Users/ziadelsarrih/Desktop/Labs/OOP/oop4/src/main/java/com/ObjectClass.java

```java
package com;

import javafx.scene.paint.Color;
import javafx.scene.paint.Paint;
import javafx.scene.shape.Shape;

import java.io.Serializable;
import java.util.Random;

public class ObjectClass implements Serializable {
    public int red;
    public int green;
    public int blue;
    public int valueHeight;
    public int valueWidth;
    public int radius;
    public int layoutX;
    public int layoutY;


    public ObjectClass(int radius) {
        this.radius = radius;
    }

    public ObjectClass() {
    }

    public ObjectClass(int valueHeight, int valueWidth) {

    }

    public Shape createObject(int radius) {
        return null;
    }

    public Shape createObject() {

        return null;
    }

    public Shape createObject(int valueHeight, int valueWidth) {

        return null;
    }

    public void randomColour() {
        Random random = new Random();
        this.red = random.nextInt(255);
        this.green = random.nextInt(255);
        this.blue = random.nextInt(255);

    }

    protected Paint getPaint() {
        return Color.rgb(red, green, blue);
    }

    public void setColor(int red, int green, int blue) {
        this.red = red;
        this.green = green;
        this.blue = blue;
    }

    public Shape Drawable() {
        return null;
    }

    public int getRed() {
        return red;
    }

    public int getGreen() {
        return green;
    }

    public int getBlue() {
        return blue;
    }

```

```
80      public int getValueHeight() {
81          return valueHeight;
82      }
83
84      public void setValueHeight(int valueHeight) {
85          this.valueHeight = valueHeight;
86      }
87
88      public int getValueWidth() {
89          return valueWidth;
90      }
91
92      public void setValueWidth(int valueWidth) {
93          this.valueWidth = valueWidth;
94      }
95
96      public int getRadius() {
97          return radius;
98      }
99
100     public void setRadius(int radius) {
101         this.radius = radius;
102     }
103
104     public int getLayoutX() {
105         return layoutX;
106     }
107
108     public void setLayoutX(int layoutX) {
109         this.layoutX = layoutX;
110     }
111
112     public int getLayoutY() {
113         return layoutY;
114     }
115
116     public void setLayoutY(int layoutY) {
117         this.layoutY = layoutY;
118     }
119 }
120
121
```

File - /Users/ziadelsarrih/Desktop/Labs/OOP/oop4/src/main/java/com/ObjectListHandler.java

```java
package com;


import java.io.Serializable;
import java.util.ArrayList;

public class ObjectListHandler implements Serializable {
    private ArrayList<ObjectClass> rectangles;
    private ArrayList<ObjectClass> circles;
    private String priorety;

    public String getPriorety() {
        return priorety;
    }

    public void setPriorety(String priorety) {
        this.priorety = priorety;
    }

    public ArrayList<ObjectClass> getRectangles() {
        return rectangles;
    }

    public void setRectangles(ArrayList<ObjectClass> rectangles) {
        this.rectangles = rectangles;
    }

    public ArrayList<ObjectClass> getCircles() {
        return circles;
    }

    public void setCircles(ArrayList<ObjectClass> circles) {
        this.circles = circles;
    }


}

```