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

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

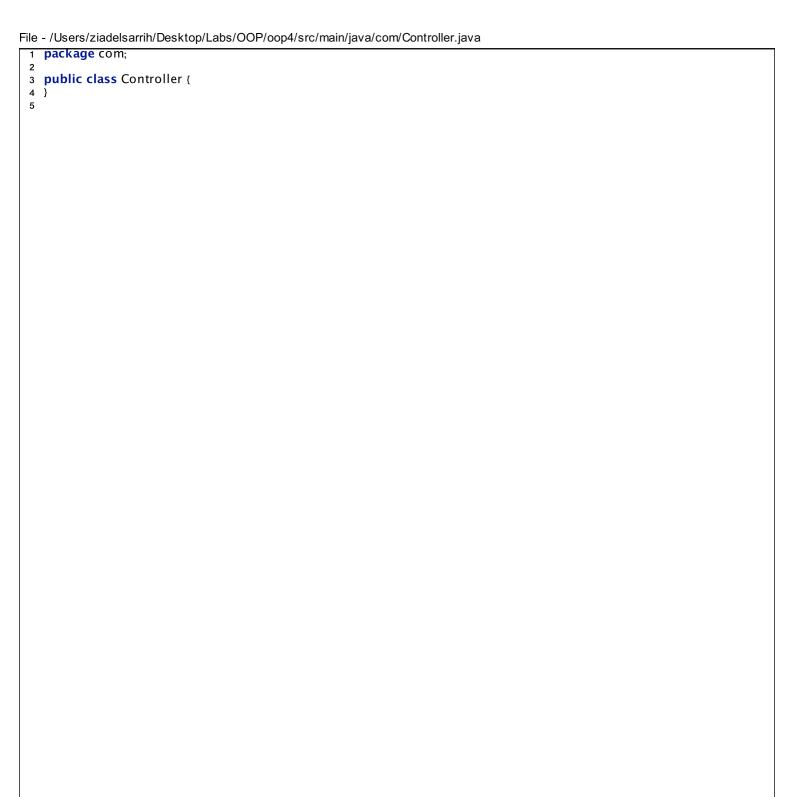
```
159
160
       private boolean checkInputCircle() {
161
162
          String rCircl = rCircle.getText();
          if ((rCircl.isEmpty())
163
                \parallel (rCircl.matches(A_Z_A_Z))
164
                || (rCircl.matches(NUMERIC) && rCircl.length() > 3)) {
165
166
             return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

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

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

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

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



```
package com;
   import javafx.scene.paint.Color;
3
   import javafx.scene.paint.Paint;
   import javafx.scene.shape.Shape;
   import java.io.Serializable;
7
   import java.util.Random;
8
   public class ObjectClass implements Serializable {
10
      public int red;
11
12
      public int green;
      public int blue;
13
      public int valueHeight;
      public int valueWidth;
15
      public int radius;
16
      public int layoutX;
17
      public int layoutY;
18
19
20
      public ObjectClass(int radius) {
21
         this.radius = radius;
22
23
24
      public ObjectClass() {
25
26
27
28
      public ObjectClass(int valueHeight, int valueWidth) {
29
30
31
      public Shape createObject(int radius) {
32
33
         return null;
34
35
36
      public Shape createObject() {
37
38
         return null;
39
      }
40
      public Shape createObject(int valueHeight, int valueWidth) {
41
42
43
         return null;
      }
44
45
46
      public void randomColour() {
         Random random = new Random();
47
48
         this.red = random.nextInt(255);
         this green = random.nextInt(255);
49
50
         this.blue = random.nextInt(255);
51
52
      }
53
      protected Paint getPaint() {
54
55
         return Color.rgb(red, green, blue);
56
57
      public void setColor(int red, int green, int blue) {
58
         this.red = red;
59
60
         this.green = green;
         this.blue = blue;
61
62
63
      public Shape Drawable() {
64
         return null,
65
      }
66
67
      public int getRed() {
68
69
         return red;
70
71
      public int getGreen() {
72
73
         return green;
74
75
      public int getBlue() {
76
77
         return blue;
78
```

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

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

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

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