```java
package com;


import com.fasterxml.jackson.dataformat.xml.XmlMapper;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.control.ToggleButton;
import javafx.scene.control.ToggleGroup;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;

import java.io.*;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;


public class Main extends Application {

    private static final String ACTION_1 = "data.xml";
    private final String alphabet = "[a-zA-Z]+";
    private final String numeric = "[0-9]+";
    private boolean chooseItRectangle = false;
    private boolean init = false;
    private int layoutX = 100;
    private int layoutY = 70;
    private int degree = 5;
    private int rotateCounter = 0;
    private ToggleGroup toggleGroup;
    private ToggleButton tbCircle;
    private ToggleButton tbRectan;
    private Pane paneContents;
    private BorderPane pane;
    private ArrayList<Rectangle> rectangles;
    private ArrayList<Circle> circles;
    private ArrayList<RectangleObject> rectanglesObjects;
    private ArrayList<CirclObject> circlesObjects;
    private ObjectListHandler objectListHandler;
    private String priority;
    private Button clear;
    private Button rotateObject;
    private Button createObject;
    private Button saveObject;
    private Button edit;
    private TextField recWidth;
    private TextField recHeight;
    private TextField rCircle;
    private TextField index;

    @Override
    public void start(Stage primaryStage) {
        pane = new BorderPane();
        paneContents = new Pane();
        pane.setPrefWidth(450);
        pane.setPrefHeight(450);
        pane.setLayoutY(60);

        createSceneContents();

        createObject.setOnAction(event -> createObject());

        rotateObject.setOnAction(event -> rotateObject());

        saveObject.setOnAction(event -> {
            try {
                serializableToXML();
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
        });

        clear.setOnAction(event -> clearScene());
        edit.setOnAction(event -> editObject());

        Scene scene = new Scene(paneContents, 650, 650);
        primaryStage.setScene(scene);
        primaryStage.show();

    }

    private void editObject() {
        int indexObject = Integer.valueOf(index.getText());
        if (toggleGroup.getSelectedToggle() == tbRectan) {

            if ((indexObject < rectangles.size()) && (checkInputRectangle())) {
                int height = Integer.valueOf(recWidth.getText());
                int width = Integer.valueOf(recHeight.getText());
                rectangles.get(indexObject).setHeight(height);
                rectangles.get(indexObject).setWidth(width);
                rectanglesObjects.get(indexObject).setValueHeight(height);
                rectanglesObjects.get(indexObject).setValueWidth(width);
            }
        } else if (toggleGroup.getSelectedToggle() == tbCircle) {
            if ((indexObject < circles.size()) && (checkInputCircle())) {
                int rCircl = Integer.valueOf(rCircle.getText());
                circles.get(indexObject).setRadius(rCircl);
                circlesObjects.get(indexObject).setRadius(rCircl);
            }
        }

    }

    private boolean checkInputRectangle() {
        String width = recWidth.getText();
        String height = recHeight.getText();
        if ((width.isEmpty()) || (height.isEmpty())
                || (width.matches(alphabet))
                || (height.matches(alphabet))
                || (width.matches(numeric) && width.length() > 3)
                || (height.matches(numeric) && height.length() > 3)) {
            return false;
        } else {
            return true;
        }
    }

    private boolean checkInputCircle() {
        String rCircl = rCircle.getText();
        if ((rCircl.isEmpty())
                || (rCircl.matches(alphabet))
                || (rCircl.matches(numeric) && rCircl.length() > 3)) {
            return false;
        } else {
            return true;
        }
    }

    private void createSceneContents() {
        createObject = new Button("Create");
        rotateObject = new Button("Rotate");
        saveObject = new Button("Save!!");
        clear = new Button("Clear  ");
        edit = new Button("Edit ");

        VBox buttons = new VBox(10);
        HBox toggleBox = new HBox(20);

        toggleBox.setLayoutX(10);
        toggleBox.setLayoutY(10);
        buttons.setLayoutX(570);
        buttons.setLayoutY(510);

        VBox rectangleSize = new VBox(3);  // to put the rectangle toggle and his size field
        HBox rectangleSizeField = new HBox(4);
        VBox circleRadius = new VBox(3); // to put the Circule toggle and his radius field
        VBox editVBox = new VBox(3);

        recWidth = new TextField();
        recWidth.setPrefSize(60, 20);

        recHeight = new TextField();
        recHeight.setPrefSize(60, 20);

        rCircle = new TextField();
        rCircle.setPrefSize(60, 20);

        index = new TextField();
        index.setPrefSize(60, 20);
        index.setPrefSize(124, 20);


        toggleGroup = new ToggleGroup();
        tbCircle = new ToggleButton("Circle");
        tbCircle.setPrefSize(60, 20);
        tbRectan = new ToggleButton("Triangle");
        tbRectan.setPrefSize(124, 20);
        tbRectan.setSelected(true);

        rectangleSizeField.getChildren().addAll(recHeight, recWidth);
        rectangleSize.getChildren().addAll(tbRectan, rectangleSizeField);
        circleRadius.getChildren().addAll(tbCircle, rCircle);
        editVBox.getChildren().addAll(edit, index);

        tbCircle.setToggleGroup(toggleGroup);
        tbRectan.setToggleGroup(toggleGroup);

        toggleBox.getChildren().addAll(rectangleSize, circleRadius, editVBox);
        buttons.getChildren().addAll(createObject, rotateObject, clear, saveObject);

        paneContents.getChildren().addAll(toggleBox, buttons, pane);

        if (init) {
            generateObjects(objectListHandler);
        }
    }

    private void generateObjects(ObjectListHandler objectClasses) {
        rectangles = new ArrayList<>();
        circles = new ArrayList<>();
        int counter = priority.length();
        int k = 0;
        int c = 0;
        int r = 0;

        while (counter != 0) {
            if (priority.charAt(k) == 'r') {
                Rectangle rectangle = objectClasses.getRectangles().get(r).rectangleObject();
                rectangles.add(rectangle);
                pane.getChildren().add(rectangle);
                rotateCounter++;
                chooseItRectangle = true;
                r++;
            } else {
                Circle circle = objectClasses.getCircles().get(c).circlObject();
                circles.add(circle);
                pane.getChildren().add(circle);
                c++;
            }
            k++;
            counter--;
        }
    }

    private void setRectangleLayouts(Rectangle rectangle, RectangleObject rectangleObject) {

        rectangle.setLayoutX(layoutX);
        rectangle.setLayoutY(layoutY);
        rectangleObject.setLayoutX(layoutX);
        rectangleObject.setLayoutY(layoutY);

    }

    private void generateLayouts() {

        layoutX += ((layoutX * 5) / 100);
        layoutY += ((layoutY * 15) / 100);
    }

    private void setcirculeLayouts(Circle circle) {

        circle.setCenterX(250);
        circle.setCenterY(250);
    }


    private void createObject() {

        if (toggleGroup.getSelectedToggle() == tbRectan) {
            chooseItRectangle = true;
            //  String width = recWidth.getText();
            // String height = recHeight.getText();

            if (!checkInputRectangle()) {
                priority += 'r';
                addObject(new RectangleObject(), false);
            } else {
                priority += 'r';
                addObject(new RectangleObject(), true);
            }
        } else if (toggleGroup.getSelectedToggle() == tbCircle) {

            if (!checkInputCircle()) {
                priority += 'c';
                addObject(new CirclObject(), false);
            } else {
                priority += 'c';
                addObject(new CirclObject(), true);
            }
        }
    }

    private void addObject(ObjectClass objectClass, boolean b) {

        if (((rectangles.isEmpty()) && (objectClass instanceof RectangleObject))) {
            Rectangle rectangle;
            RectangleObject rectangleObject = new RectangleObject();
            if (!b) {
                rectangle = rectangleObject.rectangleObject(300, 200);
            } else {
                rectangle = rectangleObject.rectangleObject(Integer.valueOf(recHeight.getText()), Integer.valueOf(recWidth.
getText()));
            }
            setRectangleLayouts(rectangle, rectangleObject);
            pane.getChildren().add(rectangle);
            rectangles = new ArrayList<>();
            rectanglesObjects = new ArrayList<>();
            rectangles.add(rectangle);
            rectanglesObjects.add(rectangleObject);
            generateLayouts();
            rotateCounter++;
        } else if ((circles.isEmpty()) && (objectClass instanceof CirclObject)) {
            Circle circle;
            CirclObject circlObject = new CirclObject();
            if (!b) {
                circle = circlObject.circlObject(200);
            } else {
                circle = circlObject.circlObject(Integer.valueOf(rCircle.getText()));
            }
            setcirculeLayouts(circle);
            pane.getChildren().add(circle);
            circles = new ArrayList<>();
            circles.add(circle);
            circlesObjects = new ArrayList<>();
            circlesObjects.add(circlObject);
        } else {
            if (objectClass instanceof RectangleObject) {

                regenerateLastObject(rectanglesObjects.get(rectangles.size() - 1), b);
                rotateCounter++;
            } else {
                regenerateLastObject(circlesObjects.get(circles.size() - 1), b);

            }

        }
    }

    private void regenerateLastObject(ObjectClass objectClass, boolean size) {

        ObjectClass thisObjectClass;
        int width = 0;
        int height = 0;
        int radius = 0;
        if (!size) {
            thisObjectClass = ObjectGenerating.generateNew(objectClass);
        } else {

            if (ObjectGenerating.getObjectType(objectClass)) {
                width = Integer.valueOf(recWidth.getText());
                height = Integer.valueOf(recHeight.getText());
                thisObjectClass = new RectangleObject(height, width);
            } else {
                radius = Integer.valueOf(rCircle.getText());
                thisObjectClass = new CirclObject(radius);
            }
        }
        if (ObjectGenerating.getObjectType(objectClass)) {
            Rectangle rectangle = ((RectangleObject) thisObjectClass).rectangleObject();
            setRectangleLayouts(rectangle, (RectangleObject) thisObjectClass);
            pane.getChildren().add(rectangle);
            rectangles.add(rectangle);
            rectanglesObjects.add((RectangleObject) thisObjectClass);
            generateLayouts();
        } else {
            Circle circle = ((CirclObject) thisObjectClass).circlObject();
            setcirculeLayouts(circle);
            pane.getChildren().add(circle);
            circles.add(circle);
            circlesObjects.add((CirclObject) thisObjectClass);
        }


    }


    private void rotateObject() {
        if (chooseItRectangle) {
            if (rotateCounter > 0) {
                for (int i = rotateCounter - 1; i < rectangles.size(); i++) {
                    rectangles.get(i).setRotate(degree);
                    degree += 5;
                }
                rotateCounter--;
            } else {
                rotateCounter = rectangles.size() - 1;
            }

        }
    }

    private void clearScene() {
        pane.getChildren().clear();
        rectangles = null;
        circles = null;
        priority = "";
        layoutX = 100;
        layoutY = 70;

    }

    private void serializableToXML() throws IOException {
        ObjectListHandler objectListHandlerTem = new ObjectListHandler();
        objectListHandlerTem.setRectangles(rectanglesObjects);
        objectListHandlerTem.setCircles(circlesObjects);
        objectListHandlerTem.setPrioretу(priority);
        try {
            XmlMapper xmlMapper = new XmlMapper();
            xmlMapper.writeValue(new File(ACTION_1), objectListHandlerTem);
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        }
    }

    public void init() throws IOException {

        priority = "";
        XmlMapper xmlMapper = new XmlMapper();
        File file = new File(ACTION_1);
        // ObjectListHandler objectListHandler = new ObjectListHandler();
        if (file.exists()) {
            String xml = new String(Files.readAllBytes(Paths.get(ACTION_1)));
            objectListHandler = xmlMapper.readValue(xml, ObjectListHandler.class);
            init = objectListHandler != null;
            if (objectListHandler != null) {
                circlesObjects = objectListHandler.getCircles();
                rectanglesObjects = objectListHandler.getRectangles();
                priority = objectListHandler.getPriorety();
            }

        }

    }


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

```java
package com;

public class Controller {
}

```

```java
package com;

import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import java.io.Serializable;


public class CirclObject extends ObjectClass implements Serializable {

    public int radius;

    public Circle circlObject(int radius) {

        this.radius = radius;
        randomColour();
        return getDrawwable();
    }

    public CirclObject() {
       // setColor(red,green,blue);
    }

    public Circle circlObject() {
        randomColour();
        return getDrawwable();
    }


    private Circle getDrawwable() {
        Circle circle = new Circle();
        circle.setRadius(radius);
        circle.setFill(getPaint());
        circle.setStroke(Color.DARKGRAY);
        circle.setStrokeWidth(2);
        circle.setCenterX(250);
        circle.setCenterY(250);
        return circle;
    }




    public void setRadius(int radius) {

        this.radius = radius;
    }

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

    public int getRadius() {

        return radius;
    }


}

```

```java
package com;

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

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

public class ObjectClass implements Serializable {
    int red;
    int green;
    int blue;

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

    }

    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 int getRed() {
        return red;
    }

    public int getGreen() {
        return green;
    }

    public int getBlue() {
        return blue;
    }
}
```

```java
package com;

import com.ObjectClass;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;

import java.io.Serializable;

class RectangleObject extends ObjectClass implements Serializable {

    public int valueHeight;
    public int valueWidth;
    public int layoutX;
    public int layoutY;


    public Rectangle rectangleObject(int valueHeight, int valueWidth) {

        this.valueHeight = valueHeight;
        this.valueWidth = valueWidth;
        randomColour();
        return getDrawwable();
    }

    public Rectangle rectangleObject() {

        return getDrawwable();
    }

    private Rectangle getDrawwable() {
        Rectangle rectangle = new Rectangle(valueHeight, valueWidth);
        rectangle.setFill(getPaint());
        rectangle.setStroke(Color.DARKGRAY);
        rectangle.setStrokeWidth(2);
        rectangle.setArcHeight(10);
        rectangle.setArcWidth(10);
        rectangle.setLayoutX(layoutX);
        rectangle.setLayoutY(layoutY);
        return rectangle;

    }

    public RectangleObject() {
    }

    public RectangleObject(int valueHeight, int valueWidth) {
        this.valueHeight = valueHeight;
        this.valueWidth = valueWidth;
        randomColour();
    }


    public int getValueHeight() {
        return valueHeight;
    }

    public void setValueHeight(int valueHeight) {
        this.valueHeight = valueHeight;
    }

    public int getValueWidth() {
        return valueWidth;
    }

    public void setValueWidth(int valueWidth) {
        this.valueWidth = valueWidth;
    }

    public int getLayoutX() {
        return layoutX;
    }

    public void setLayoutX(int layoutX) {
        this.layoutX = layoutX;
    }

    public int getLayoutY() {
        return layoutY;
    }

    public void setLayoutY(int layoutY) {
        this.layoutY = layoutY;
    }

}

```

```java
package com;


/**
 * this class made it for
 * generate rectangle with
 * new size minus 10%
 */
 class ObjectGenerating {



    /**
     * set the old value minus
     * 10% from original size
     */


     static ObjectClass generateNew(ObjectClass objectClass){

         if (getObjectType(objectClass)){
             RectangleObject rectangleObject = (RectangleObject) objectClass;
             int height = rectangleObject.getValueHeight()-((rectangleObject.getValueHeight()*10)/100);
             int width = rectangleObject.getValueWidth()-((rectangleObject.getValueWidth()*10)/100);
             return new RectangleObject(height,width);
         }else{
                 CirclObject circleObject = (CirclObject) objectClass;
                 int radius = (circleObject.getRadius()-(circleObject.getRadius()*10/100));
             return new CirclObject(radius);

         }
    }
//
//     static com.ObjectClass generateNewSize(com.ObjectClass objectClass , int Height, int Width, int Radius){
//
//         if (getObjectType(objectClass)){
//             RectangleObject ro = new RectangleObject();
//
//             ro.setValueHeight(Height);
//             ro.setValueWidth (Width);
//             return ro;
//         }else{
//             com.CirclObject circle = new com.CirclObject();
//             circle.setRadius(Radius);
//             return circle;
//
//         }
//     }

    static  boolean getObjectType(ObjectClass objectClass){

        return (objectClass instanceof RectangleObject);
    }
}
```

```java
package com;

import com.CirclObject;

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

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

    public String getPriorety() {
        return priorety;
    }

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

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

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

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

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



}

```