2. De creat 3 primitive grafice ecran, care apar din diferite parti ale ferestrei, şi sa se alinieze unul după altul în centru. Procesul trebuie să fie repetat ciclic.

Codul programului:

```
import javafx.animation.Timeline;
import javafx.application.Application;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.*;
import javafx.stage.Stage;
import javafx.animation.PathTransition;
import javafx.scene.Scene;
import javafx.util.Duration;

public class Main extends Application {
        public static void main(String[] args) {
                launch(args);
        }
        @Override
        public void start(Stage primaryStage){
                try {
                        Pane root = new Pane();
                        final Rectangle rect1 = new Rectangle(10, 10, 100, 100);
                        rect1.setArcHeight(20);
                        rect1.setArcWidth(20);
                        rect1.setFill(Color.PINK);

                        final Rectangle rect2 = new Rectangle(10, 10, 100, 100);
                        rect2.setArcHeight(20);
                        rect2.setArcWidth(20);
                        rect2.setFill(Color.YELLOW);

                        final Rectangle rect3 = new Rectangle(10, 10, 100, 100);
                        rect3.setArcHeight(20);
                        rect3.setArcWidth(20);
                        rect3.setFill(Color.ORANGE);

                        Path path = new Path();
                        path.getElements().add(new MoveTo(771, 248));
                        path.getElements().add(new CubicCurveTo(7,5,120,136,191,159));
                        path.getElements().add(new CubicCurveTo(257,168,291,280,353,326));
                        path.getElements().add(new CubicCurveTo(506,310,607,247,771,248));

                        Path path2 = new Path(); path2.getElements().add(new MoveTo(771, 248));
                        path2.getElements().add(new CubicCurveTo(1830,53,1550,117,1433,235));
                        path2.getElements().add(new CubicCurveTo(1327,325,1194,372,1131,339));
                        path2.getElements().add(new CubicCurveTo(1123,334,1062,309,771,248));

                        Path path3 = new Path(); path3.getElements().add(new MoveTo(771, 248));
                        path3.getElements().add(new CubicCurveTo(1170,788,1110,712,894,687));
                        path3.getElements().add(new CubicCurveTo(798,718,595,762,551,622));
                        path3.getElements().add(new CubicCurveTo(763,534,930,392,771,248));
```

```
            PathTransition pathT = new PathTransition();
            pathT.setDuration(Duration.millis(4000)); pathT.setPath(path);
            pathT.setNode(rect1); pathT.setCycleCount(Timeline.INDEFINITE);
        pathT.setOrientation(PathTransition.OrientationType.ORTHOGONAL_TO_TANGENT);
            pathT.setAutoReverse(true);
            pathT.play();

            PathTransition pathT2 = new PathTransition();
            pathT2.setDuration(Duration.millis(4000));
            pathT2.setPath(path2);
            pathT2.setNode(rect2);
            pathT2.setCycleCount(Timeline.INDEFINITE);
        pathT2.setOrientation(PathTransition.OrientationType.ORTHOGONAL_TO_TANGENT);
            pathT2.setAutoReverse(true);
            pathT2.play();

            PathTransition pathT3 = new PathTransition();
            pathT3.setDuration(Duration.millis(4000));
            pathT3.setPath(path3);
            pathT3.setNode(rect3);
            pathT3.setCycleCount(Timeline.INDEFINITE);
        pathT3.setOrientation(PathTransition.OrientationType.ORTHOGONAL_TO_TANGENT);
            pathT3.setAutoReverse(true);
            pathT3.play();

            Scene scene = new Scene(root, 300, 400);

            root.getChildren().add(rect1);

            root.getChildren().add(rect2);

            root.getChildren().add(rect3);

            primaryStage.setScene(scene);

            primaryStage.show();

            } catch (Exception e){

            e.printStackTrace();

            }

        }

}
```