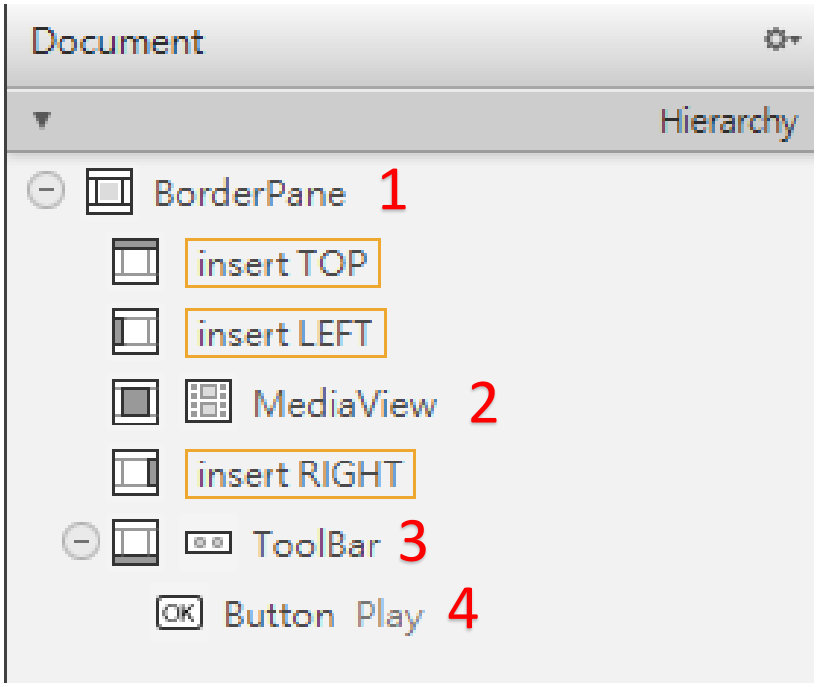
Homework 7

Please implement following GUI by Scene Builder and complete the application with given codes. Study the codes carefully and make sure you get a best understanding of what/how/when the programs do.

1. Video Player

GUI Description:

Hierarchy:



0) File Name: VideoPlayer.fxml
Controller Class: VideoPlayerController

1) BorderPane

a) Min Width: USE_COMPUTED_SIZE
b) Min Height: USE_COMPUTED_SIZE
c) Pref Width: 600
d) Pref Height: 400
e) Max Width: USE_COMPUTED_SIZE
f) Max Height: USE_COMPUTED_SIZE
g) Style: -fx-background-color: black

2) MediaView

a) fx:id: mediaView
b) BorderPane.alignment: CENTER

3) ToolBar

a) Pref Height: 40.0
b) Pref Width: 200.0
c) BorderPane.alignment: CENTER

4) **Button**
   a) fx:id: playPauseButton
   b) onAction: playPauseButtonPressed
   c) Pref Height: 25.0
   d) Pref Width: 60.0
   e) Text: Play

Note:
1) You should put the file “controlsfx-8.40.12.jar” and other files in the same folder.
2) When you compile the java files, you should add “javafx.media” in the command.

VideoPlayer.java

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

/**
 *
 * @author PaulDeitel
 */
public class VideoPlayer extends Application {

   @Override
   public void start(Stage stage) throws Exception {
      Parent root =
FXMLLoader.load(getClass().getResource("VideoPlayer.fxml"));

      Scene scene = new Scene(root);
      stage.setTitle("Video Player");
      stage.setScene(scene);
      stage.show();
   }

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
```

```
        launch(args);
    }

}
```

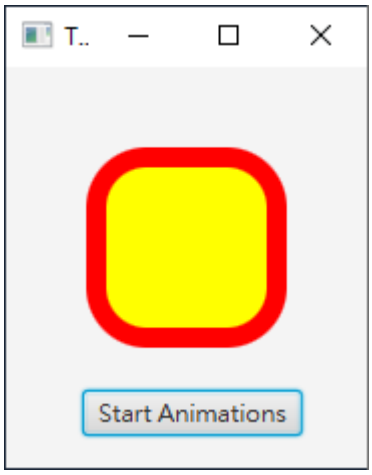
VideoPlayerController.java

```
import java.net.URL;
import javafx.beans.binding.Bindings;
import javafx.beans.property.DoubleProperty;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
import javafx.util.Duration;
import org.controlsfx.dialog.ExceptionDialog;

public class VideoPlayerController {
    @FXML private MediaView mediaView;
    @FXML private Button playPauseButton;
    private MediaPlayer mediaPlayer;
    private boolean playing = false;

    public void initialize() {
        // get URL of the video file
        URL url = VideoPlayerController.class.getResource("sts117.mp4");

        // create a Media object for the specified URL
        Media media = new Media(url.toExternalForm());

        // create a MediaPlayer to control Media playback
        mediaPlayer = new MediaPlayer(media);

        // specify which MediaPlayer to display in the MediaView
        mediaView.setMediaPlayer(mediaPlayer);

        // set handler to be called when the video completes playing
```

```
        mediaPlayer.setOnEndOfMedia(
            new Runnable() {
                public void run() {
                    playing = false;
                    playPauseButton.setText("Play");
                    mediaPlayer.seek(Duration.ZERO);
                    mediaPlayer.pause();
                }
            }
        );

        // set handler that displays an ExceptionDialog if an error occurs
        mediaPlayer.setOnError(
            new Runnable() {
                public void run() {
                    ExceptionDialog dialog =
                        new ExceptionDialog(mediaPlayer.getError());
                    dialog.showAndWait();
                }
            }
        );

        // set handler that resizes window to video size once ready to play
        mediaPlayer.setOnReady(
            new Runnable() {
                public void run() {
                    DoubleProperty width = mediaView.fitWidthProperty();
                    DoubleProperty height = mediaView.fitHeightProperty();
                    width.bind(Bindings.selectDouble(
                        mediaView.sceneProperty(), "width"));
                    height.bind(Bindings.selectDouble(
                        mediaView.sceneProperty(), "height"));
                }
            }
        );
    }

    // toggle media playback and the text on the playPauseButton
    @FXML
    private void playPauseButtonPressed(ActionEvent e) {
```

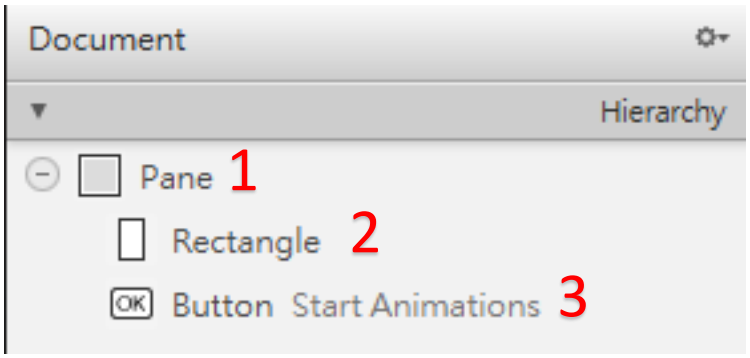
```
        playing = !playing;

        if (playing) {
            playPauseButton.setText("Pause");
            mediaPlayer.play();
        }
        else {
            playPauseButton.setText("Play");
            mediaPlayer.pause();
        }
    }
}
```

2. Transition Animations



GUI Description:

Hierarchy:



0) File Name: TransitionAnimations.fxml
Controller Class: TransitionAnimationsController

1) Pane

a) Stylesheets: TransitionAnimations.css
b) Min Width: USE_COMPUTED_SIZE
c) Min Height: USE_COMPUTED_SIZE
d) Pref Width: 180
e) Pref Height: 200
f) Max Width: USE_COMPUTED_SIZE
g) Max Height: USE_COMPUTED_SIZE
h) Id: Pane

2) Rectangle

a) fx:id: rectangle
b) height: 90.0
c) width: 90.0
d) layoutX: 45.0
e) layoutY: 45.0

3) Button

a) fx:id: startButton
b) layoutX: 38.0
c) layoutY: 161.0
d) mnemonicParsing: false
e) onAction: startButtonPressed
f) text: Start Animations

TransitionAnimations.java

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class TransitionAnimations extends Application {
   @Override
   public void start(Stage stage) throws Exception {
      Parent root =
         FXMLLoader.load(getClass().getResource("TransitionAnimations.fxml"));

      Scene scene = new Scene(root);
      stage.setTitle("TransitionAnimations");
      stage.setScene(scene);
      stage.show();
   }

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

TransitionAnimationsController.java

```
import javafx.animation.FadeTransition;
import javafx.animation.FillTransition;
import javafx.animation.Interpolator;
import javafx.animation.ParallelTransition;
import javafx.animation.PathTransition;
import javafx.animation.RotateTransition;
import javafx.animation.ScaleTransition;
import javafx.animation.SequentialTransition;
import javafx.animation.StrokeTransition;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.paint.Color;
import javafx.scene.shape.LineTo;
```

```
import javafx.scene.shape.MoveTo;
import javafx.scene.shape.Path;
import javafx.scene.shape.Rectangle;
import javafx.util.Duration;

public class TransitionAnimationsController {
   @FXML private Rectangle rectangle;

   // configure and start transition animations
   @FXML
   private void startButtonPressed(ActionEvent event) {
      // transition that changes a shape's fill
      FillTransition fillTransition =
         new FillTransition(Duration.seconds(1));
      fillTransition.setToValue(Color.CYAN);
      fillTransition.setCycleCount(2);

      // each even cycle plays transition in reverse to restore original
      fillTransition.setAutoReverse(true);

      // transition that changes a shape's stroke over time
      StrokeTransition strokeTransition =
         new StrokeTransition(Duration.seconds(1));
      strokeTransition.setToValue(Color.BLUE);
      strokeTransition.setCycleCount(2);
      strokeTransition.setAutoReverse(true);

      // parallelizes multiple transitions
      ParallelTransition parallelTransition =
         new ParallelTransition(fillTransition, strokeTransition);

      // transition that changes a node's opacity over time
      FadeTransition fadeTransition =
         new FadeTransition(Duration.seconds(1));
      fadeTransition.setFromValue(1.0); // opaque
      fadeTransition.setToValue(0.0); // transparent
      fadeTransition.setCycleCount(2);
      fadeTransition.setAutoReverse(true);

      // transition that rotates a node
```

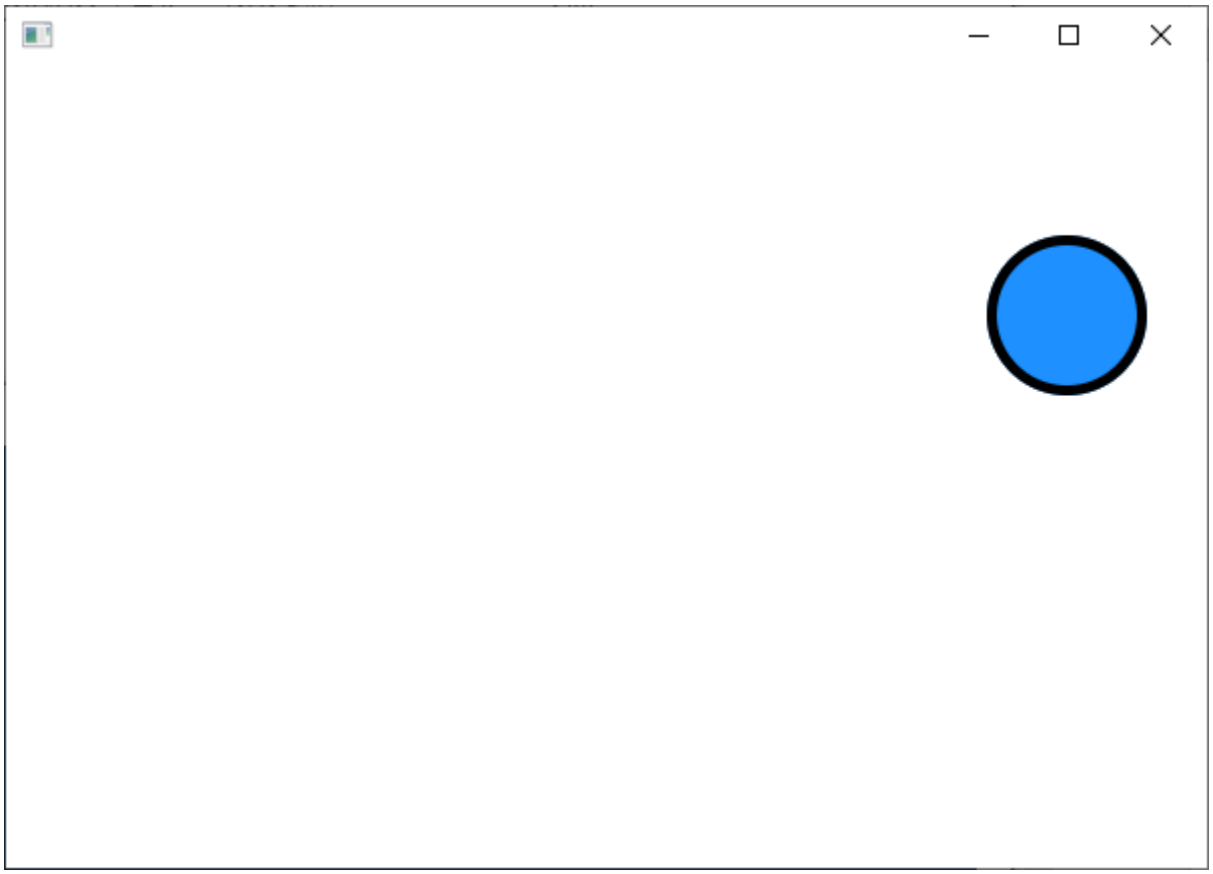
```
        RotateTransition rotateTransition =
            new RotateTransition(Duration.seconds(1));
        rotateTransition.setByAngle(360.0);
        rotateTransition.setCycleCount(2);
        rotateTransition.setInterpolator(Interpolator.EASE_BOTH);
        rotateTransition.setAutoReverse(true);

        // transition that moves a node along a Path
        Path path = new Path(new MoveTo(45, 45), new LineTo(45, 0),
            new LineTo(90, 0), new LineTo(90, 90), new LineTo(0, 90));
        PathTransition translateTransition =
            new PathTransition(Duration.seconds(2), path);
        translateTransition.setCycleCount(2);
        translateTransition.setInterpolator(Interpolator.EASE_IN);
        translateTransition.setAutoReverse(true);

        // transition that scales a shape to make it larger or smaller
        ScaleTransition scaleTransition =
            new ScaleTransition(Duration.seconds(1));
        scaleTransition.setByX(0.75);
        scaleTransition.setByY(0.75);
        scaleTransition.setCycleCount(2);
        scaleTransition.setInterpolator(Interpolator.EASE_OUT);
        scaleTransition.setAutoReverse(true);

        // transition that applies a sequence of transitions to a node
        SequentialTransition sequentialTransition =
            new SequentialTransition (rectangle, parallelTransition,
                fadeTransition, rotateTransition, translateTransition,
                scaleTransition);
        sequentialTransition.play(); // play the transition
    }
}
```

TransitionAnimations.css

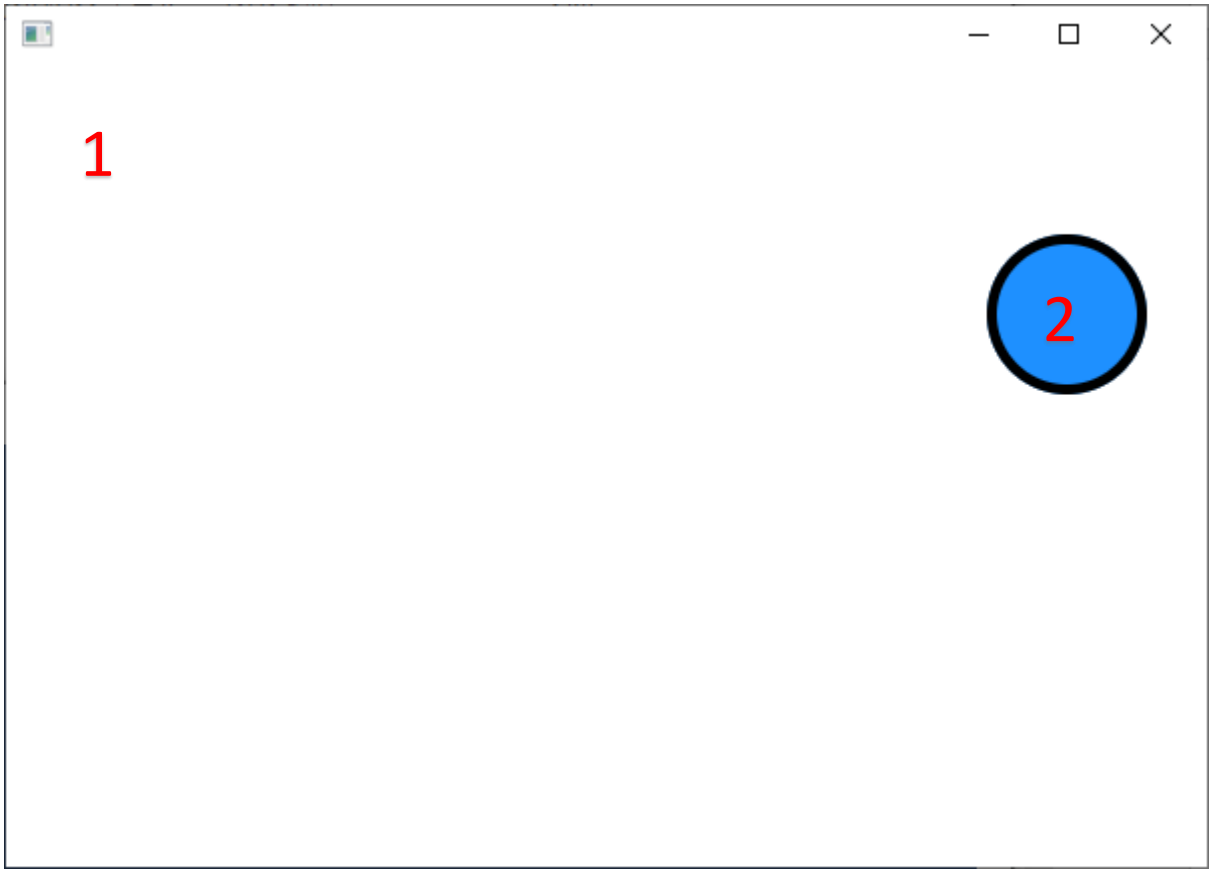
```
Rectangle {
    -fx-stroke-width: 10;
    -fx-stroke: red;
```

```
   -fx-arc-width: 50;
   -fx-arc-height: 50;
   -fx-fill: yellow;
}
```

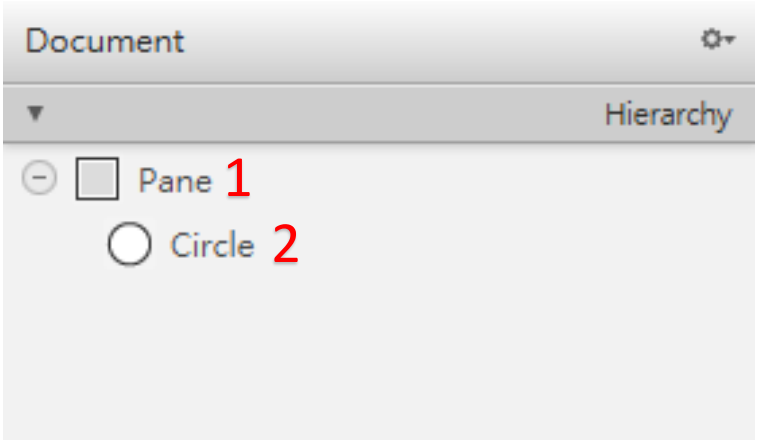
3. Timeline Animation



GUI Description:



Hierarchy:



0) File Name: TimelineAnimation.fxml
Controller Class: TimelineAnimationController

1) Pane

a) Min Width: USE_COMPUTED_SIZE
b) Min Height: USE_COMPUTED_SIZE
c) Pref Width: 600
d) Pref Height: 400
e) Max Width: USE_COMPUTED_SIZE
f) Max Height: USE_COMPUTED_SIZE
g) fx:id: pane
h) id: Pane

2) Circle

a) Fill: DODGERBLUE
b) layoutX: 142.0
c) layoutY: 143.0
d) radius: 40.0
e) stroke: BLACK
f) strokeType: INSIDE
g) strokeWidth: 5.0
h) fx:id="c"

TimelineAnimation.java

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class TimelineAnimation extends Application {
   @Override
   public void start(Stage stage) throws Exception {
      Parent root =
FXMLLoader.load(getClass().getResource("TimelineAnimation.fxml"));

      Scene scene = new Scene(root);

      stage.setScene(scene);
      stage.show();
   }

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

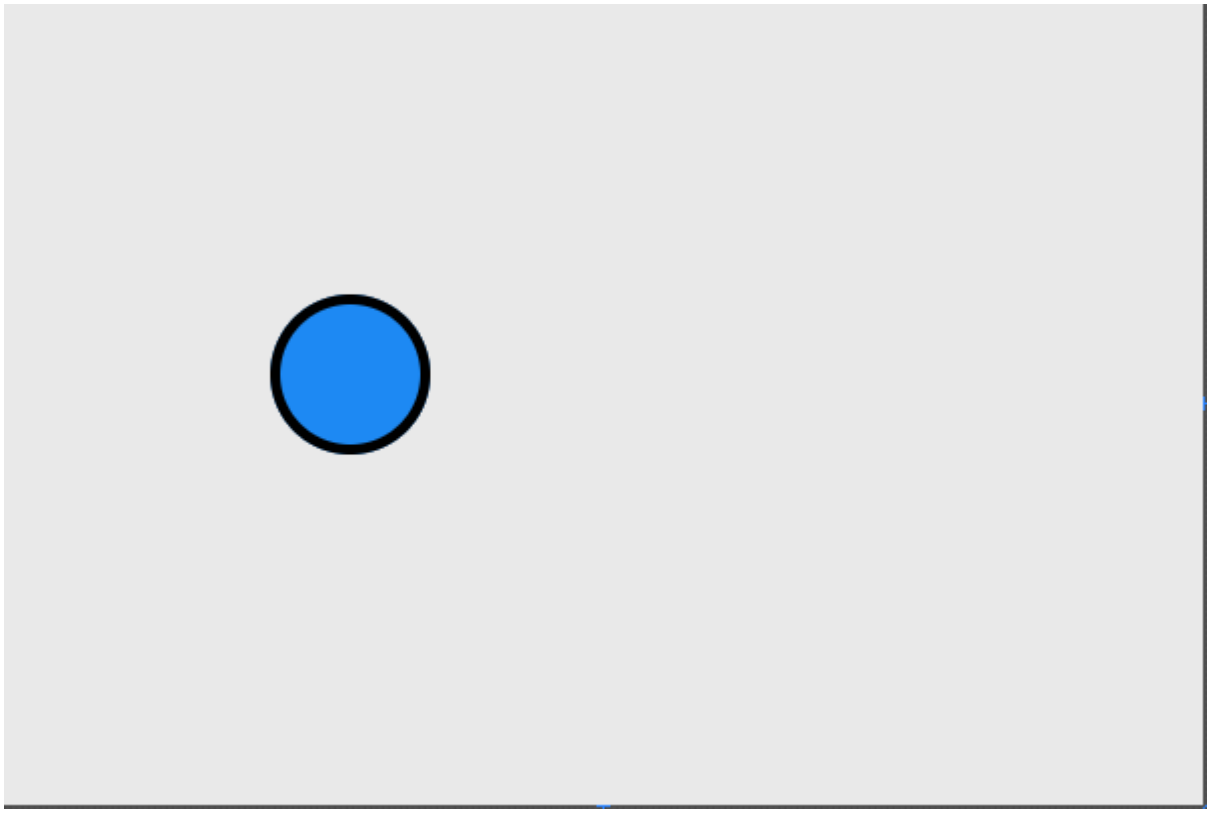
TimelineAnimationController.java

```
import java.security.SecureRandom;
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.geometry.Bounds;
import javafx.scene.layout.Pane;
import javafx.scene.shape.Circle;
import javafx.util.Duration;

public class TimelineAnimationController {
   @FXML Circle c;
```

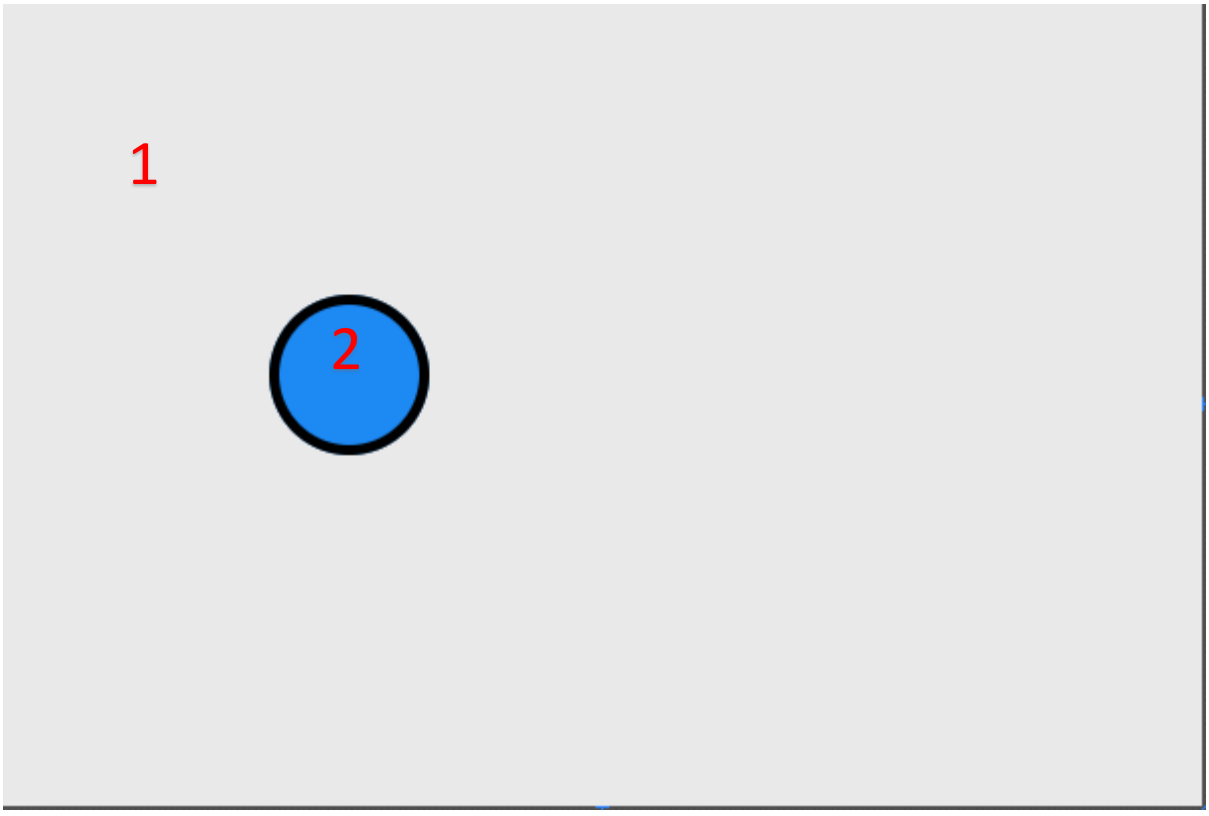
```
@FXML Pane pane;

public void initialize() {
   SecureRandom random = new SecureRandom();

   // define a timeline animation
   Timeline timelineAnimation = new Timeline(
      new KeyFrame(Duration.millis(10),
         new EventHandler<ActionEvent>() {
            int dx = 1 + random.nextInt(5);
            int dy = 1 + random.nextInt(5);

            // move the circle by the dx and dy amounts
            @Override
            public void handle(final ActionEvent e) {
               c.setLayoutX(c.getLayoutX() + dx);
               c.setLayoutY(c.getLayoutY() + dy);
               Bounds bounds = pane.getBoundsInLocal();

               if (hitRightOrLeftEdge(bounds)) {
                  dx *= -1;
               }

               if (hitTopOrBottom(bounds)) {
                  dy *= -1;
               }
            }
         }
      )
   );

   // indicate that the timeline animation should run indefinitely
   timelineAnimation.setCycleCount(Timeline.INDEFINITE);
   timelineAnimation.play();
}

// determines whether the circle hit the left or right of the window
private boolean hitRightOrLeftEdge(Bounds bounds) {
   return (c.getLayoutX() <= (bounds.getMinX() + c.getRadius())) ||
      (c.getLayoutX() >= (bounds.getMaxX() - c.getRadius()));
```

```
    }

    // determines whether the circle hit the top or bottom of the window
    private boolean hitTopOrBottom(Bounds bounds) {
        return (c.getLayoutY() <= (bounds.getMinY() + c.getRadius())) ||
            (c.getLayoutY() >= (bounds.getMaxY() - c.getRadius()));
    }
}
```

4. Ball Animation



GUI Description:



Hierarchy:

0) File Name: BallAnimationTimer.fxml
Controller Class: TipCalculatorController

1) Pane

a) fx:id: Pane
b) Pref Width: 600
c) Pref Height: 400

2) Circle

a) fx:id: c
b) Stroke Width: 5

BallAnimationTimer.java

```
// BallAnimationTimer.java
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class BallAnimationTimer extends Application {
   @Override
   public void start(Stage stage) throws Exception {
      Parent root =
FXMLLoader.load(getClass().getResource("BallAnimationTimer.fxml"));

      Scene scene = new Scene(root);

      stage.setScene(scene);
      stage.show();
   }
```

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

BallAnimationTimerController.java

```
// BallAnimationTimerController.java
// Bounce a circle around a window using an AnimationTimer subclass.
import java.security.SecureRandom;
import javafx.animation.AnimationTimer;
import javafx.fxml.FXML;
import javafx.geometry.Bounds;
import javafx.scene.layout.Pane;
import javafx.scene.shape.Circle;
import javafx.util.Duration;

public class BallAnimationTimerController {
   @FXML private Circle c;
   @FXML private Pane pane;

   public void initialize() {
      SecureRandom random = new SecureRandom();

      // define a timeline animation
      AnimationTimer timer = new AnimationTimer() {
         int dx = 1 + random.nextInt(5);
         int dy = 1 + random.nextInt(5);
         int velocity = 60; // used to scale distance changes
         long previousTime = System.nanoTime(); // time since app launch

         // specify how to move Circle for current animation frame
         @Override
         public void handle(long now) {
            double elapsedTime = (now - previousTime) / 1000000000.0;
            previousTime = now;
            double scale = elapsedTime * velocity;
```

```
            Bounds bounds = pane.getBoundsInLocal();
            c.setLayoutX(c.getLayoutX() + dx * scale);
            c.setLayoutY(c.getLayoutY() + dy * scale);

            if (hitRightOrLeftEdge(bounds)) {
               dx *= -1;
            }

            if (hitTopOrBottom(bounds)) {
               dy *= -1;
            }
         }
      };

      timer.start();
   }

   // determines whether the circle hit left/right of the window
   private boolean hitRightOrLeftEdge(Bounds bounds) {
      return (c.getLayoutX() <= (bounds.getMinX() + c.getRadius())) ||
         (c.getLayoutX() >= (bounds.getMaxX() - c.getRadius()));
   }

   // determines whether the circle hit top/bottom of the window
   private boolean hitTopOrBottom(Bounds bounds) {
      return (c.getLayoutY() <= (bounds.getMinY() + c.getRadius())) ||
         (c.getLayoutY() >= (bounds.getMaxY() - c.getRadius()));
   }
}
```