Report of LAB 2.2: JAVAFX

Mariem Ben Rhouma

1. Objectives:

o Create clock:

- Use the ClockPane class
- Set a random time
- Control the clock
- Animate the clock

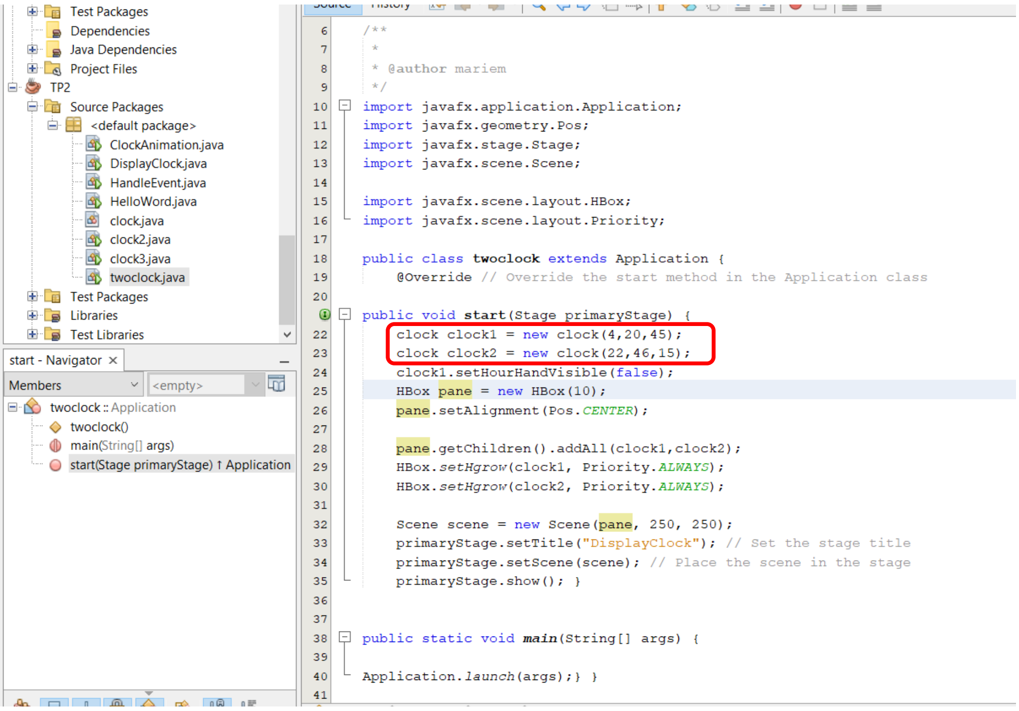
o Develop a racing car.

- pause/resume the animation with a button press/release
- increase/decrease the car speed by pressing the UP and DOWN arrow keys.

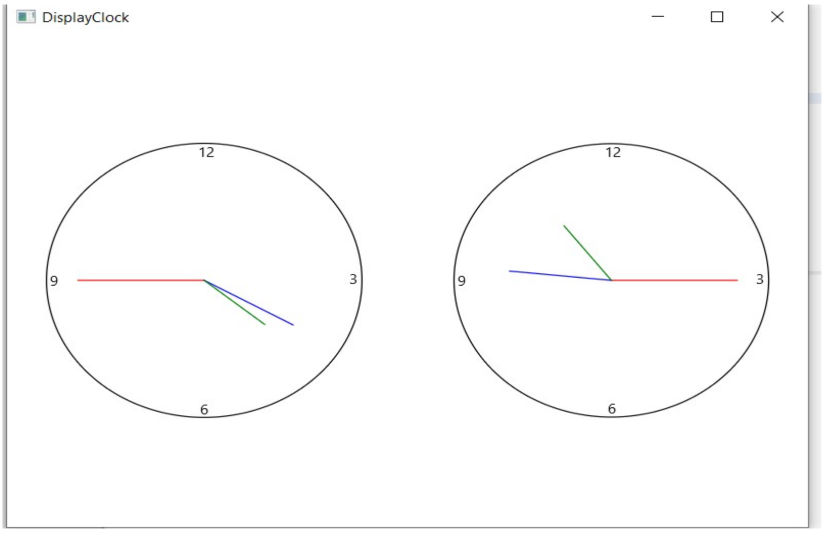
2.Activity List:

Activity 1: Clock

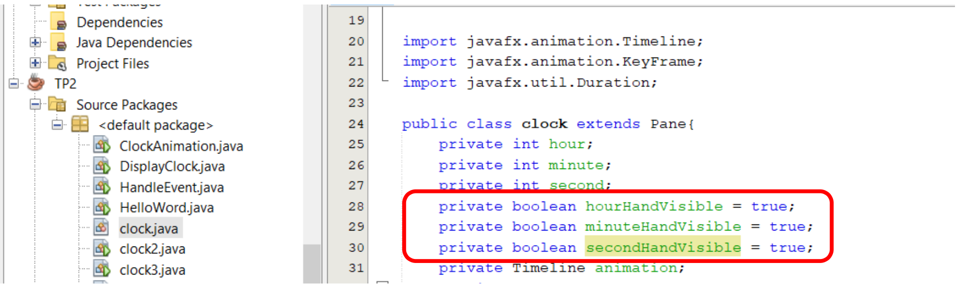
1. Use the Clock class to write a program that displays two clocks. The hour, minute, and second values are 4, 20, 45 for the first clock and 22, 46, 15 for the second clock, as shown in the following figure



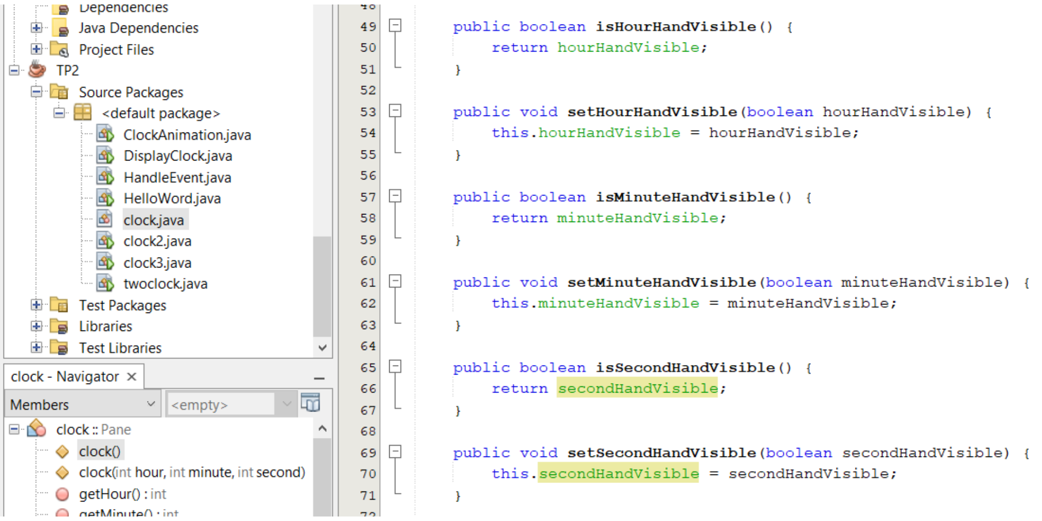
```java
/**
 *
 * @author mariem
 */
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.stage.Stage;
import javafx.scene.Scene;

import javafx.scene.layout.HBox;
import javafx.scene.layout.Priority;

public class twoclock extends Application {
    @Override // Override the start method in the Application class

public void start(Stage primaryStage) {
    clock clock1 = new clock(4,20,45);
    clock clock2 = new clock(22,46,15);
    clock1.setHourHandVisible(false);
    HBox pane = new HBox(10);
    pane.setAlignment(Pos.CENTER);

    pane.getChildren().addAll(clock1,clock2);
    HBox.setHgrow(clock1, Priority.ALWAYS);
    HBox.setHgrow(clock2, Priority.ALWAYS);

    Scene scene = new Scene(pane, 250, 250);
    primaryStage.setTitle("DisplayClock"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); }


public static void main(String[] args) {

Application.launch(args);} }
```



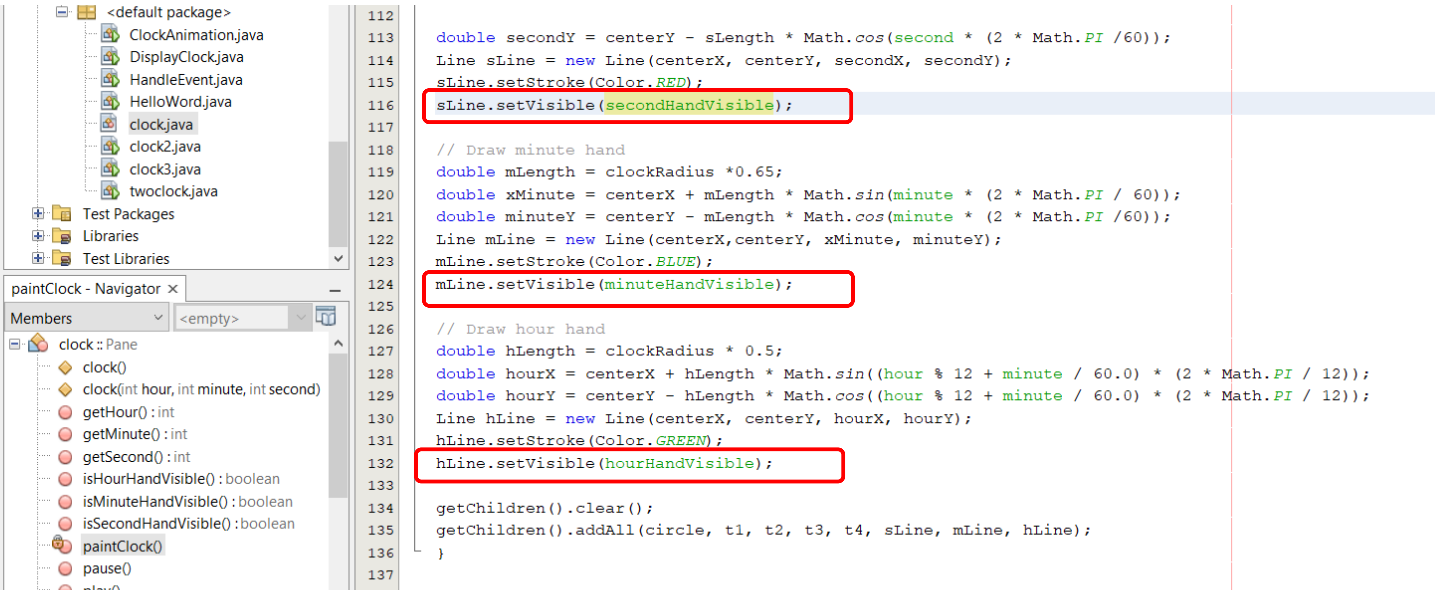
2. Modify the Clock class with three new Boolean properties
- hourHandVisible
- minuteHandVisible
- secondHandVisible
and their associated accessor and mutator methods.
You can use the set methods to make a hand visible or invisible



```
import javafx.animation.Timeline;
import javafx.animation.KeyFrame;
import javafx.util.Duration;

public class clock extends Pane{
    private int hour;
    private int minute;
    private int second;
    private boolean hourHandVisible = true;
    private boolean minuteHandVisible = true;
    private boolean secondHandVisible = true;
    private Timeline animation;
```

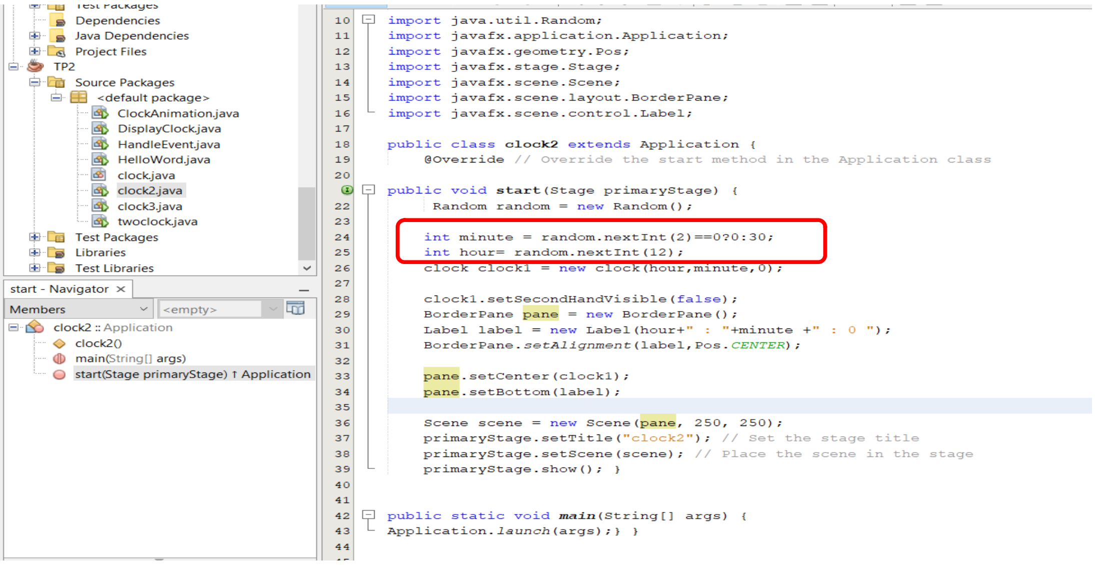


```
    public boolean isHourHandVisible() {
        return hourHandVisible;
    }

    public void setHourHandVisible(boolean hourHandVisible) {
        this.hourHandVisible = hourHandVisible;
    }

    public boolean isMinuteHandVisible() {
        return minuteHandVisible;
    }

    public void setMinuteHandVisible(boolean minuteHandVisible) {
        this.minuteHandVisible = minuteHandVisible;
    }

    public boolean isSecondHandVisible() {
        return secondHandVisible;
    }

    public void setSecondHandVisible(boolean secondHandVisible) {
        this.secondHandVisible = secondHandVisible;
    }
```

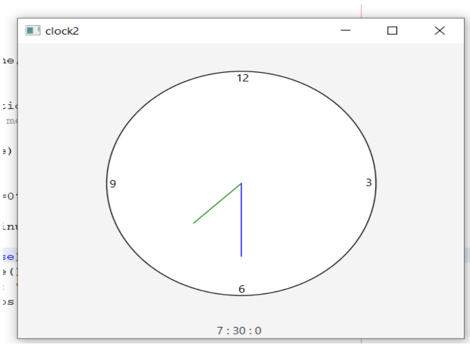


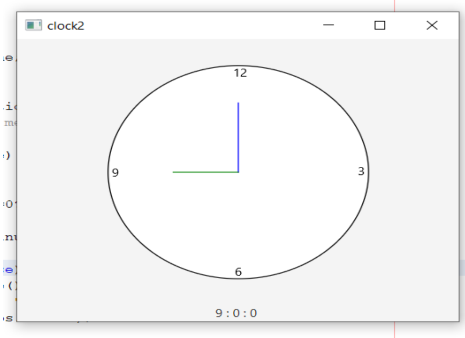
```
double secondY = centerY - sLength * Math.cos(second * (2 * Math.PI /60));
Line sLine = new Line(centerX, centerY, secondX, secondY);
sLine.setStroke(Color.RED);
sLine.setVisible(secondHandVisible);

// Draw minute hand
double mLength = clockRadius *0.65;
double xMinute = centerX + mLength * Math.sin(minute * (2 * Math.PI / 60));
double minuteY = centerY - mLength * Math.cos(minute * (2 * Math.PI /60));
Line mLine = new Line(centerX,centerY, xMinute, minuteY);
mLine.setStroke(Color.BLUE);
mLine.setVisible(minuteHandVisible);

// Draw hour hand
double hLength = clockRadius * 0.5;
double hourX = centerX + hLength * Math.sin((hour % 12 + minute / 60.0) * (2 * Math.PI / 12));
double hourY = centerY - hLength * Math.cos((hour % 12 + minute / 60.0) * (2 * Math.PI / 12));
Line hLine = new Line(centerX, centerY, hourX, hourY);
hLine.setStroke(Color.GREEN);
hLine.setVisible(hourHandVisible);

getChildren().clear();
getChildren().addAll(circle, t1, t2, t3, t4, sLine, mLine, hLine);
}
```

3. Write a test program that displays only the hour and minute hands. The hour and minute values are randomly generated. The hour is between 0 and 11, and the minute is either 0 or 30, as shown in the figure below

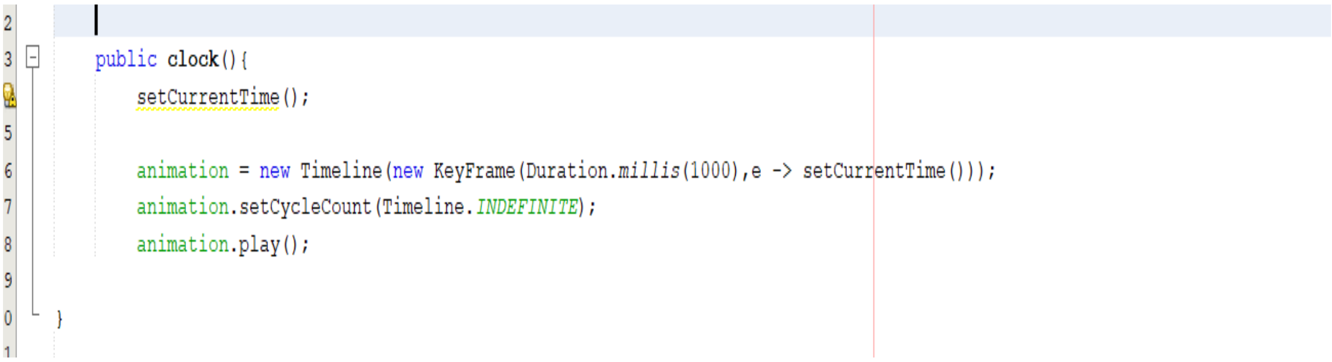


```java
import java.util.Random;
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.scene.control.Label;

public class clock2 extends Application {
    @Override // Override the start method in the Application class

public void start(Stage primaryStage) {
     Random random = new Random();

    int minute = random.nextInt(2)==0?0:30;
    int hour= random.nextInt(12);
    clock clock1 = new clock(hour,minute,0);

    clock1.setSecondHandVisible(false);
    BorderPane pane = new BorderPane();
    Label label = new Label(hour+" : "+minute +" : 0 ");
    BorderPane.setAlignment(label,Pos.CENTER);

    pane.setCenter(clock1);
    pane.setBottom(label);

    Scene scene = new Scene(pane, 250, 250);
    primaryStage.setTitle("clock2"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); }


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

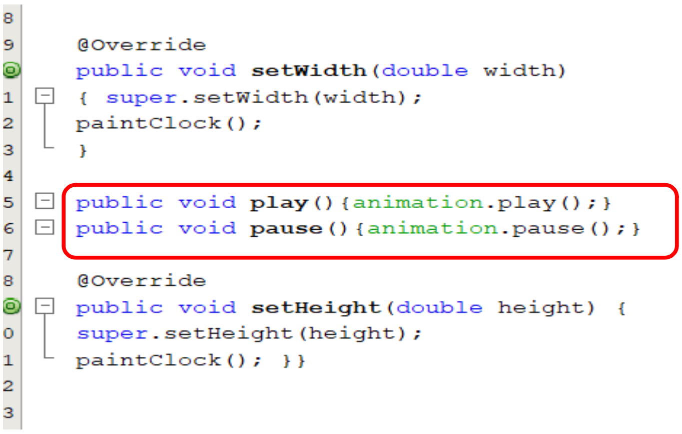




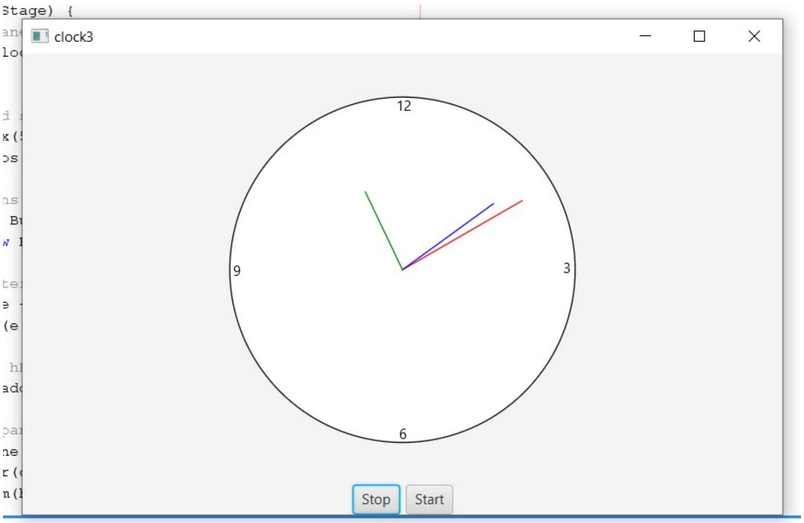
4. Modify Clock.java, to add the animation into this class and add two methods start() and stop() to start and stop the clock.



```java

    public clock(){
        setCurrentTime();

        animation = new Timeline(new KeyFrame(Duration.millis(1000),e -> setCurrentTime()));
        animation.setCycleCount(Timeline.INDEFINITE);
        animation.play();

    }
```



```
    @Override
    public void setWidth(double width)
    { super.setWidth(width);
    paintClock();
    }

    public void play(){animation.play();}
    public void pause(){animation.pause();}

    @Override
    public void setHeight(double height)  {
    super.setHeight(height);
    paintClock();  }}
```


Stage) {
clock3
12
3
6
9
Stop
Start