O **One Armed Bandit** 🔒

Project ID: 5984

Forked from an inaccessible project.

Updated name of startButton in PrimaryController.java to match README
Jonas Solhaug Kaad authored 1 week ago

---

Name	Last commit	Last update
📁 assets	One Armed Bandit exercise	2 weeks ago
📁 src/main	Updated name of startButton in PrimaryCon...	1 week ago
.gitignore	One Armed Bandit exercise	2 weeks ago
pom.xml	One Armed Bandit exercise	2 weeks ago
readme.md	Update readme.md	2 weeks ago

📄 **readme.md**

One Armed Bandit

The term "one-armed bandit" is slang for the old-fashioned machines, where you could stand for hours, stuffing money in with one hand and pulling a handle with the other. In this task, such a bandit will be implemented, but without the betting and winning part. A possible layout is shown below:
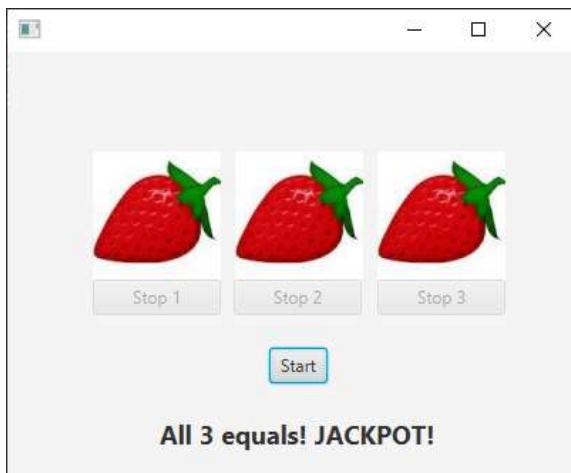
The system consists of:

- 3 `ImageViews` for changing sequences of images.
- 3 `Buttons` to stop the sequences individually.
- 1 start `Button`, which starts the sequences in all 3 image fields.
- 1 `Label`, which shows the result after all 3 sequences are finished

The easiest choice would be to use an implementation of javafx.animation.AnimationTimer, but that is forbidden in this task, as it is Threads in the context of JavaFX that we are training.

Resource files: contains 10 images of fruit. The images are 90*90 pixels.

Task 1

Define the user interface of the application. It is recommended to use SceneBuilder.

- Create 3 `ImageViews`
  - Name them `spin1`, `spin2`, `spin3`.
- Create 3 `Buttons` to stop the image sequence in each `ImageView`

- Create 1 `Button` to start the image sequences in all `ImageViews`
  - Name it `startButton`.
- Create a `Label` to display the result after all the image sequences are finished.
  - Name it `resultLabel`.

In the `PrimaryController`:

- Declare an Array of type `Image`, as a `javafx.scene.image.Image`, as follows `Image[] images`
- Declare 3 variables of type `Thread` as follows: `Thread t1, t2, t3;`:
- Declare a variable `spinsALive` of type `int`

Implement the `initialize()` method:

- Initialize the array declared earlier. The size of the array should be 10, as follows: `images= new Image[10]`;
- Use a for loop and load the 10 supplied images into this array.
- Each image can be inserted into the array with the following statement `images[i]=new Image(getClass().getResource(filename).toURI().toString());`
  - **Hint:** *Remember to declare the filename, which is always "fruits" + some number + ".png"*
  - (Remember to catch relevant exceptions, using a `try-catch`)
- Outside the for loop, set any 3 images in the 3 `ImageViews` as follows: `spin1.setImage(images[1])`;
- Disable the three stop buttons.
  - **Hint:** *You can use the `setDisable()` method*

Task 2

- A synchronized method `aliveCount` is already implemented. but it is commented. Uncomment it and study the code.

- An inner class `public class BanditRunnable implements Runnable` is already implemented, but it is commented. Uncomment it and study the code

Create an `ActionHandler` for the Start button:

- Create 3 instances of `BanditRunnable`. The constructor takes 3 arguments. Pass the appropriate arguments for each instance. (Remember to use all 3 `ImageViews`) **Hint:** *Different waiting times must be inserted in the three Threads between each image switch. For example 120, 100 and 140 milliseconds*
- Initialize the 3 threads you created earlier and pass each thread a different instance of `BanditRunnable`.
- Set each thread as a Daemon thread as follows: `t1.setDaemon(true)`.
- Start/execute each thread.
- Disable the Start button.
- Enable Stop buttons for each `ImageView`.
- Change the label to `"Running..."`

Create an `ActionHandler` for the Stop buttons:

- Each button stops one sequence (e.g. with an interrupt of the corresponding Thread).
- And disable the stop button which called the `ActionHandler`.
  - *i.e. if the 2nd button called the ``ActionHandler`, `the 2nd` ImageView` should be stopped (by interrupting the thread) and the 2nd button should be disabled.*

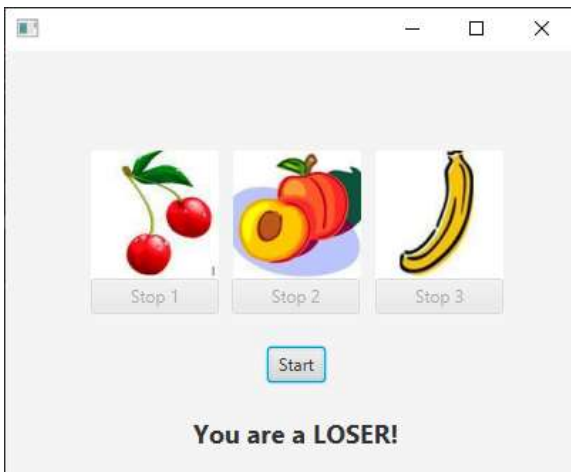In the same `ActionHandler`, when all 3 Threads are stopped:
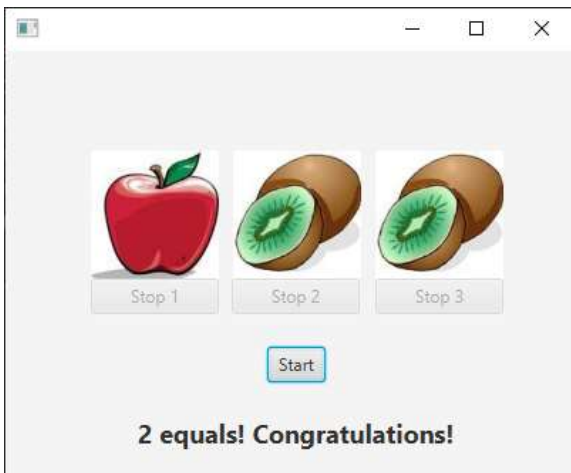
- Enable the Start button.

Hint: *You can use `event.getSource()`, to determine which button called the `ActionHandler`*

For reference here is an example of how the UI could look while running:
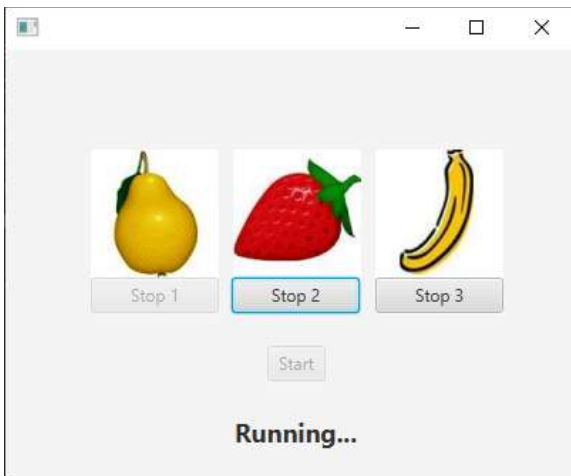
For reference here is an example of how the UI could look for all 3 results:
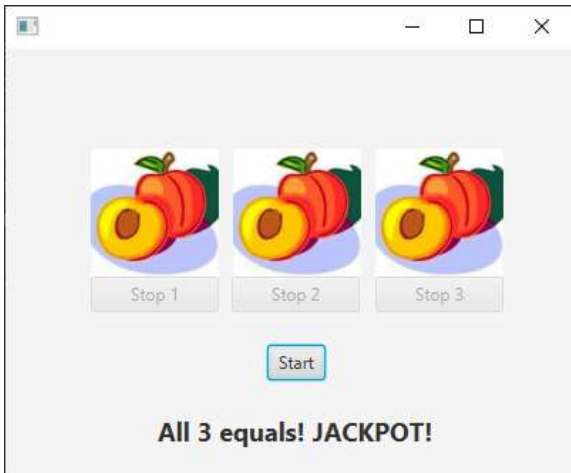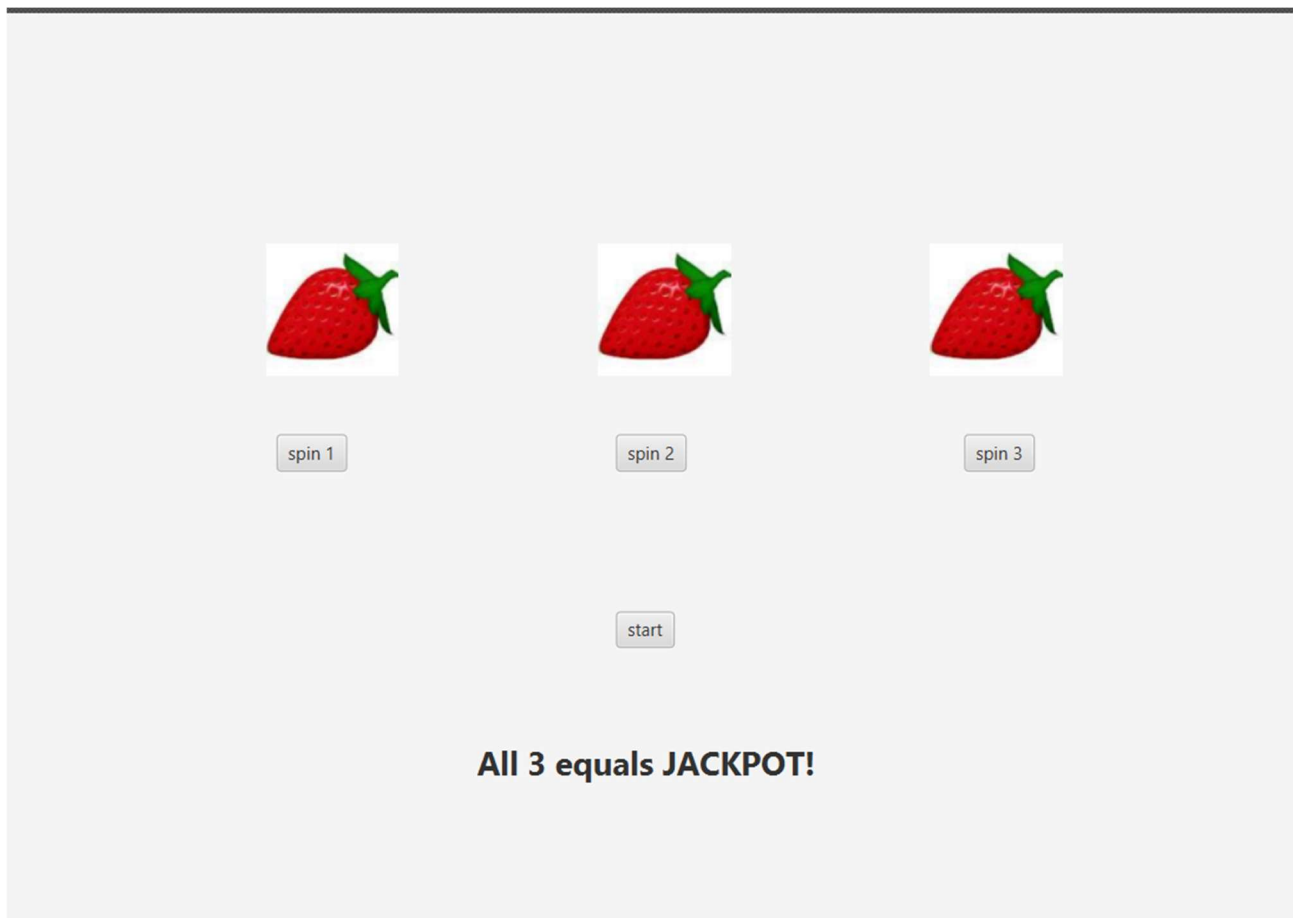
Ovenpå, kan det ses at jeg har lavet de forskellige komponenter for JavaFX i SceneBuilder.

I kan ignorere størrelsen på scenen, det vigtigste er bare at indsætte de rigtige knapper, figurer og labeler med tekst og id.

```java
 1  package vop;
 2
 3
 4  import javafx.application.Platform;
 5  import javafx.event.ActionEvent;
 6  import javafx.fxml.FXML;
 7  import javafx.scene.control.Button;
 8  import javafx.scene.control.Label;
 9  import javafx.scene.image.Image;
10  import javafx.scene.image.ImageView;
11
12  import java.io.FileNotFoundException;
13  import java.io.IOException;
14  import java.net.URISyntaxException;
15
16  public class PrimaryController {
17      //Hele tegningen og figuropsætningen skal foregå
    på Scenebuilder.
18      //Jeg har brugt Skeleton under View i
    Scenebuilder, til at få overblik over Controls,
    Labels og ActionEvents.
19      Image[] images;
20      Thread t1;
21      Thread t2;
22      Thread t3;
23      int spinsAlive;
24      @FXML
25      private Label resultLabel;
26
27      @FXML
28      private ImageView spin1;
29
30      @FXML
31      private ImageView spin2;
32
33      @FXML
34      private ImageView spin3;
35
36      @FXML
37      private Button startButton;
38
```

```java
39         @FXML
40         private Button stop1;
41
42         @FXML
43         private Button stop2;
44
45         @FXML
46         private Button stop3;
47
48
49         //Her har vi løst den anden del og sidste del af
   Opgave 1.
50         @FXML
51         public void initialize() {
52             images = new Image[10];
53             try{
54                 for(int i = 0; i < images.length; i++){
55                     images[i]=new Image(getClass().
   getResource("fruits" + i + ".png").toURI().toString
   ());
56                 }
57                 spin1.setImage(images[1]);
58                 spin2.setImage(images[2]);
59                 spin3.setImage(images[3]);
60                 stop1.setDisable(true);
61                 stop2.setDisable(true);
62                 stop3.setDisable(true);
63             }catch (URISyntaxException e){
64                 System.out.println("Images not found");
65             }
66
67         }
68
69         //Her er Opgave 2 løst, hvor kommanterene fra
   toppen og bunden er blevet fjernet.
70         private synchronized void aliveCount(boolean up
   ) {
71             if (up) {
72                 spinsAlive++;
73             } else {
74                 spinsAlive--;
```

```java
 75                }
 76                //Kode herværket for Opgave 2 er lavet her.
 77                System.out.println("Alive: " + spinsAlive);
 78                if (spinsAlive == 0) {
 79                    startButton.setDisable(false);
 80                    Platform.runLater(new Runnable() {
 81                        @Override
 82                        public void run() {
 83                            if (spin1.getImage() == spin2.
    getImage() && spin1.getImage() == spin3.getImage
    ()) {
 84                                resultLabel.setText("All 3
    equals! JACKPOT!");
 85                            } else if (spin1.getImage() ==
    spin2.getImage()
 86                                    || spin1.getImage() ==
    spin3.getImage()
 87                                    || spin2.getImage() ==
    spin3.getImage()) {
 88                                resultLabel.setText("2
    equals! Congratulations!");
 89                            } else {
 90                                resultLabel.setText("You are
     a LOSER!");
 91
 92                            }
 93                        }
 94                    });
 95                }
 96            }
 97        //Det her er en af de sidste dele af Opgaven.
 98        //Vi har lavet Threads som der blev snakket i
    Opgave 2.
 99        @FXML
100        void startButton(ActionEvent event) {
101            BanditRunnable runnable1 = new
    BanditRunnable(1,100,spin1);
102            BanditRunnable runnable2 = new
    BanditRunnable(2,120,spin2);
103            BanditRunnable runnable3 = new
    BanditRunnable(3,140,spin3);
```

```java
104            t1 = new Thread(runnable1);
105            t2 = new Thread(runnable2);
106            t3 = new Thread(runnable3);
107            t1.setDaemon(true);
108            t2.setDaemon(true);
109            t3.setDaemon(true);
110            t1.start();
111            t2.start();
112            t3.start();
113            startButton.setDisable(true);
114            stop1.setDisable(false);
115            stop2.setDisable(false);
116            stop3.setDisable(false);
117            resultLabel.setText("Running....");
118
119        }
120
121        //Vi har reduceret Kode herhenne, hvor vi bare
       lavet kode til at stoppe 1 billede med den samme
       knap mens de andre fortsætter.
122        @FXML
123        void stop1method(ActionEvent event) {
124            if(event.getSource()==stop1){
125                stop1.setDisable(true);
126                t1.interrupt();
127            }else if(event.getSource()==stop2){
128                stop2.setDisable(true);
129                t2.interrupt();
130            }else if(event.getSource()==stop3){
131                stop3.setDisable(true);
132                t3.interrupt();
133            }
134            if(stop1.isDisabled()&stop2.isDisabled()&
       stop3.isDisabled()){
135                startButton.setDisable(false);
136            }
137
138        }
139
140        //Kodet under stop1method er brugt til at undgå
       skrivning af lang kode i stop2method og stop3method.
```

```java
141        @FXML
142        void stop2method(ActionEvent event) {
143
144        }
145
146        @FXML
147        void stop3method(ActionEvent event) {
148
149        }
150
151
152
153
154        public class BanditRunnable implements Runnable
    {
155
156            private int i;
157            private long sleepTime;
158            private boolean running;
159            private ImageView iw;
160
161            public BanditRunnable(int i, long sleepTime
    , ImageView iw) {
162                this.i = i;
163                this.sleepTime = sleepTime;
164                this.iw = iw;
165            }
166
167            @Override
168            public void run() {
169                running = true;
170                aliveCount(true);
171                System.out.println("Thread started: " +
    Thread.currentThread());
172
173                while (running) {
174                    Platform.runLater(new Runnable() {
175                        @Override
176                        public void run() {
177
178                            iw.setImage(images[i]);
```

```java
179                              i = (i + 1) % images.length;
180                          }
181                      });
182                      synchronized (this) {
183                          try {
184                              //Thread.sleep(sleepTime);
185                              wait(sleepTime);
186                          } catch (InterruptedException ex
    ) {
187                              System.out.println("
    Interrupted: " + Thread.currentThread());
188                              running = false;
189                              aliveCount(false);
190                          }
191                      }
192                  }
193              }
194          }
195
196
197 }
198
```