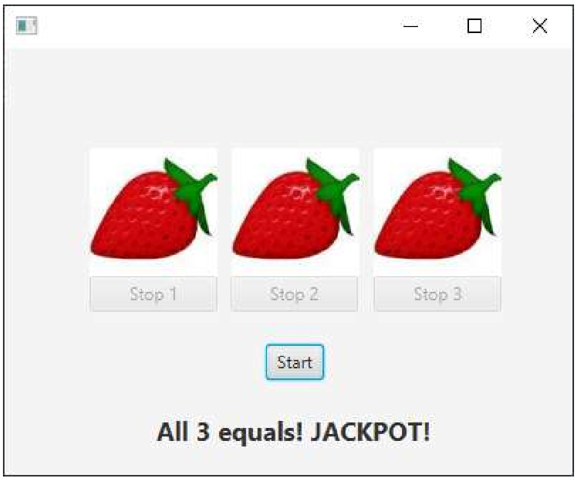
Update readme.md
Jonas Solhaug Kaad authored 2 months ago

readme.md 4.00 KiB

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:

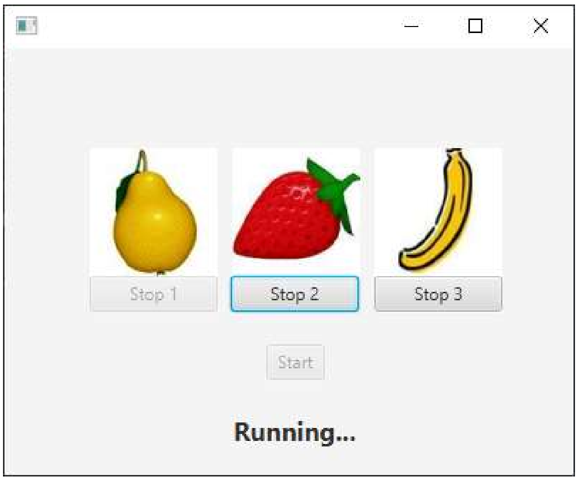
- Each button stops one sequence (e.g. with an interrupt of the corresponding Thread).
- And disable the stop button which called the `ActionHandler` .
    - *i.e. if the 2nd button called the ``ActionHandler , `the 2nd` ImageView` should be stopped (by interrupting the thread) and the 2nd button should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

In the same `ActionHandler` , when all 3 Threads are stopped:

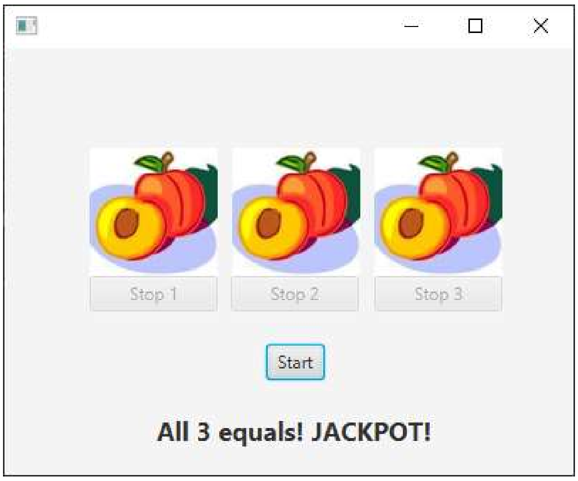
- Enable the Start button.

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

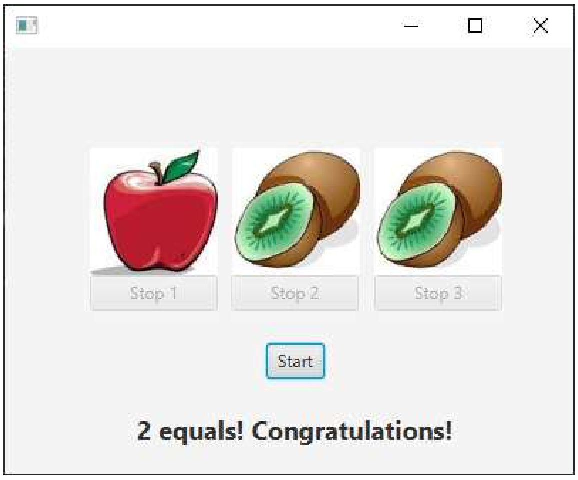
For reference here is an example of how the UI could look while running:



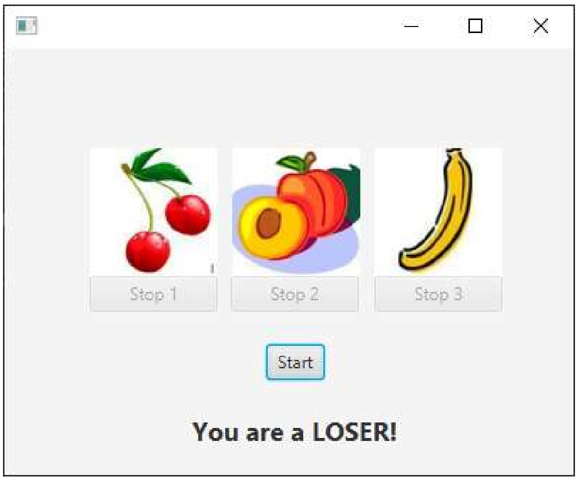
For reference here is an example of how the UI could look for all 3 results:



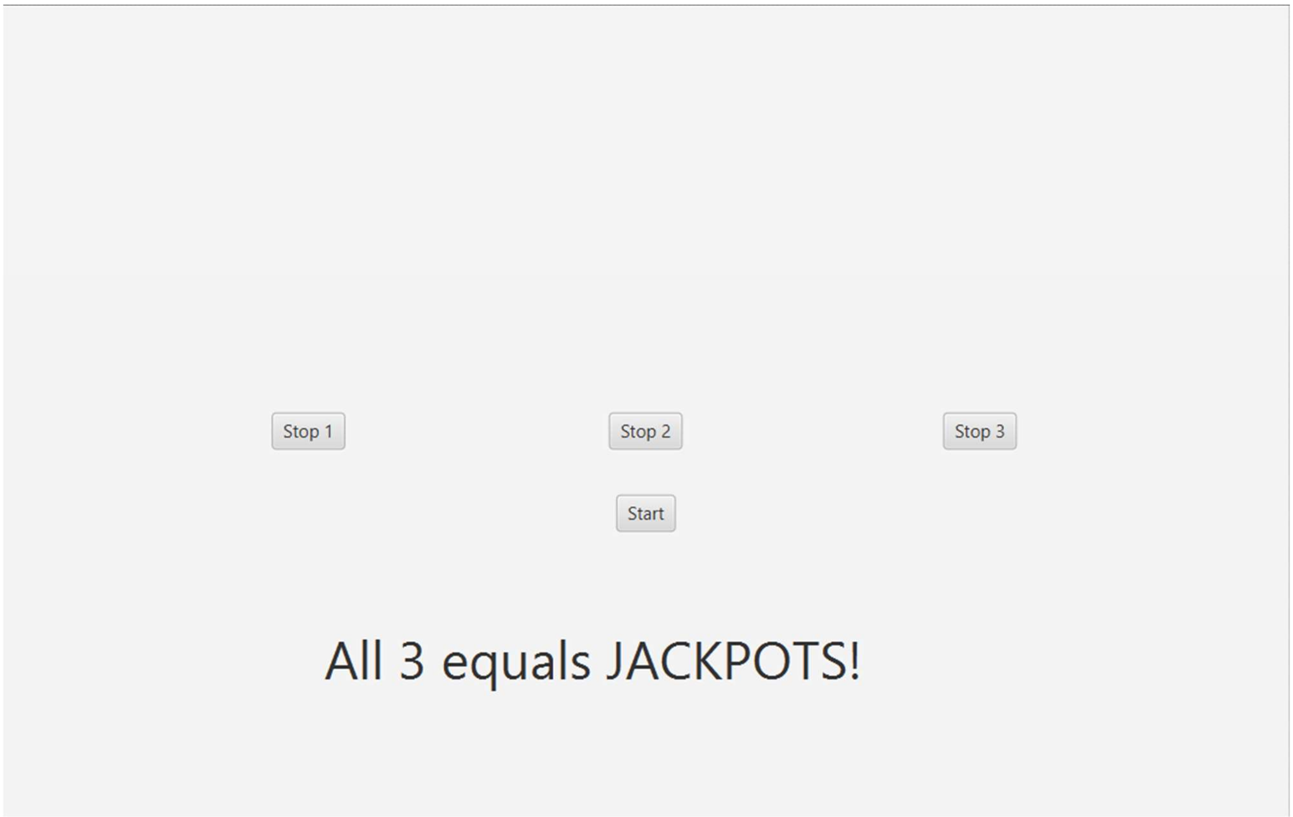
All 3 equals! JACKPOT!



2 equals! Congratulations!



You are a LOSER!



All 3 equals JACKPOTS!

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
package vop;


import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

import java.net.URISyntaxException;

public class PrimaryController {
    //Disse navnet er defineret under Code og kan findes ved Skeleton under View-Menu.
    @FXML
    private Label resultLabel;

    @FXML
    private ImageView spin1;

    @FXML
    private ImageView spin2;

    @FXML
    private ImageView spin3;

    @FXML
    private Button startButton;

    @FXML
    private Button stop1;

    @FXML
    private Button stop2;

    @FXML
    private Button stop3;

    //Her har vi lavet arrays for images og dannet
```

```java
threads.
    Image[] images;
    Thread t1, t2, t3;
    int spinsAlive;

    //Her har vi defineret en array index, hvor billederrne er indsat.
    //Vi har sørget for at referere til filerne under for-lykken.
    @FXML
    public void initialize(){
        images = new Image[10];
        spin1.setImage(images[1]);
        spin1.setImage(images[2]);
        spin1.setImage(images[3]);
        spin1.setImage(images[4]);
        spin1.setImage(images[5]);
        spin1.setImage(images[6]);
        spin1.setImage(images[7]);
        spin1.setImage(images[8]);
        spin1.setImage(images[9]);
        stop1.setDisable(true);
        stop2.setDisable(true);
        stop3.setDisable(true);
        for(int i = 0; i< images.length; i++){
            String filename = "fruits" + i + ".png";
            try{
                images[i]=new Image(getClass().getResource(filename).toURI().toString());
            } catch (URISyntaxException e) {
                throw new RuntimeException(e);
            }
        }
    }


    private synchronized void aliveCount(boolean up) {
        if (up) {
            spinsAlive++;
        } else {
            spinsAlive--;
        }

        System.out.println("Alive: " + spinsAlive);
        if (spinsAlive == 0) {
            startButton.setDisable(false);
            Platform.runLater(new Runnable() {
                @Override
                public void run() {
                    if (spin1.getImage() == spin2.getImage() && spin1.getImage() == spin3.getImage()) {
                        resultLabel.setText("All 3 equals! JACKPOT!");
                    } else if (spin1.getImage() == spin2.getImage()
                            || spin1.getImage() == spin3.getImage()
                            || spin2.getImage() == spin3.getImage()) {
                        resultLabel.setText("2 equals! Congratulations!");
                    } else {
                        resultLabel.setText("You are a LOSER!");

                    }
                }
            });
        }
    }


    public class BanditRunnable implements Runnable {

        private int i;
        private long sleepTime;
        private boolean running;
        private ImageView iw;

        public BanditRunnable(int i, long sleepTime, ImageView iw) {
            this.i = i;
            this.sleepTime = sleepTime;
            this.iw = iw;
        }

        @Override
        public void run() {
            running = true;
            aliveCount(true);
            System.out.println("Thread started: " + Thread.currentThread());

            while (running) {
                Platform.runLater(new Runnable() {
                    @Override
                    public void run() {

                        iw.setImage(images[i]);
                        i = (i + 1) % images.length;
                    }
                });
                synchronized (this) {
                    try {
                        //Thread.sleep(sleepTime);
                        wait(sleepTime);
                    } catch (InterruptedException ex) {
                        System.out.println("Interrupted: " + Thread.currentThread());
                        running = false;
                        aliveCount(false);
                    }
                }
            }
        }
    }

    //Her har vi lavet Threads fra BanditRunnable-Klassen.
    //Her har vi fået vores Threads til at køre.
    @FXML
    void startButtonmethod(ActionEvent event) {
        BanditRunnable banditrunnable1 = new BanditRunnable(120,100,spin1);
        BanditRunnable banditrunnable2 = new BanditRunnable(120,120,spin2);
        BanditRunnable banditrunnable3 = new BanditRunnable(120,140,spin3);

        Thread thread1 = new Thread(banditrunnable1);
        Thread thread2 = new Thread(banditrunnable2);
        Thread thread3 = new Thread(banditrunnable3);

        thread1.setDaemon(true);
        thread2.setDaemon(true);
        thread3.setDaemon(true);

        thread1.start();
        thread2.start();
        thread3.start();

        //Vi har også stoppet knapperne for billederne og derefter tillodt start-knappen til at køre når billederne er stoppet.
        if(event.getSource()==spin1){
            stop1.setDisable(false);
        }
        if (event.getSource()==spin2){
            stop2.setDisable(false);
        }
        if (event.getSource()==spin3){
            stop3.setDisable(false);
        }
        else if (event.getSource()!=spin3 & event.getSource()!=spin2 & event.getSource()!=spin1){
            startButton.setDisable(true);
        }
    }
}

```