Tecnológico de Monterrey

SLOT MACHINE

Project Programming Languages

Autor:
Pablo Emanuel Pozos Aguilar
A01700854

Professor:
Benjamín Valdés Aguirre

May 29, 2020

Contents

Abstract

The fun in different casinos is used by different machines, which are based on a sequence of plays to determine or return a given value, all slot machines have the same mission so that the player cannot know the combination that will come out and so Anticipate, there are dozens of different operations so that the result is different and thus catching the user, both in time and in money.

1 Context of the problem

The games that are presented in a casino are a lot but at the moment we will focus on generating a slot machine through concurrent programming and use of multithreads for the operation of this, it is a simple game where through combinations we will try to hit figures that are the same to get the desired result.

1.1 Java Concurrency

Concurrency is the ability of your program to deal (not doing) with many things at once and is achieved through multithreading. Do not confuse concurrency with parallelism which is about doing many things at once.

Multithreading is a technique that allows for concurrent (simultaneous) execution of two or more parts of a program for maximum utilization of a CPU, Programs are made up of processes and threads («Multithreading and Concurrency Fundamentals», 2019).

Thread is the smallest executable unit of a process. A process can have multiple threads with one main thread. In the example, a single thread could be displaying the current tab you're in, and a different thread could be another tab.

Why use concurrency and multithreading?

With multiple cores, applications have the option to take more advantage of the hardware, to run individual threads through different cores, to make the application more responsive and efficient. This allows you to take more advantage of the CPU and the nuclei that the computer has.

Some reasons why multiple threads are used are:

- *Higher performance*
- *Responsive applications that give the illusion of multitasking.*
- *Efficient utilization of resources.*
    - The creation of threads is easier and faster compared to generating a completely new process and also when processing different requests, much fewer resources are consumed.

1.2 Slot Machine Function

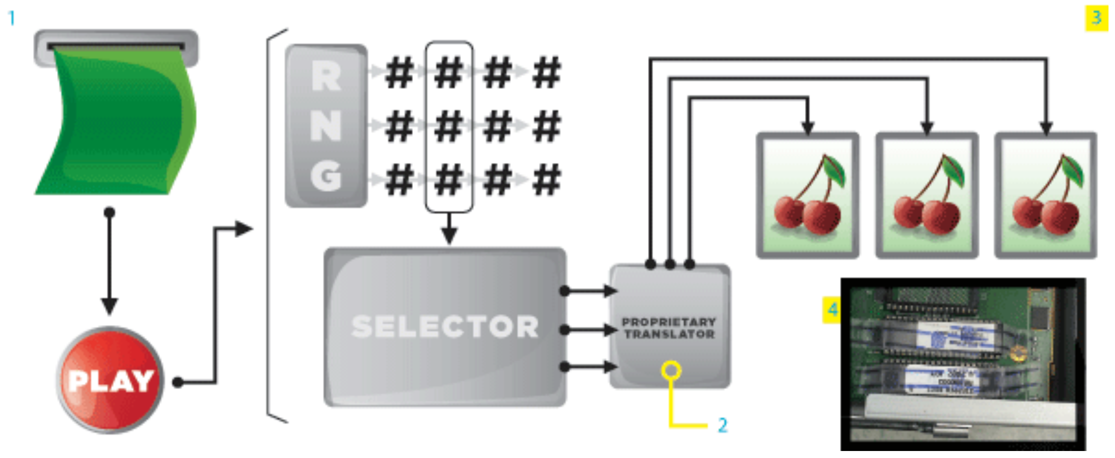
Slot machines have existed since 1895 when Charles Fey invented the Liberty Bell made up of 3 reels that allowed customers to try to align 3 identical symbols.

Slot Machines usually have three reels, but in other cases, they can be made of five reels. These reels have different symbols depending on the theme of the game and other factors.

The more unlikely you are to set a specific group of symbols, the greater the payout for creating a winning combination, this by turning the reels, which are operated either by a lever, a button, or different actions depending on the type of machine to play.

All slot machines are controlled by a computer and reproduce randomness thanks to the Random Number Generator (RNG) system that guarantees a safe and reliable game. However, this random number generator offers an advantage for the casino.

The random number generator (RNG) is the only element on which the algorithm is based. The random number generator works continuously and when a player spins the slot will automatically receive the numbers that the RNG has produced at that time.



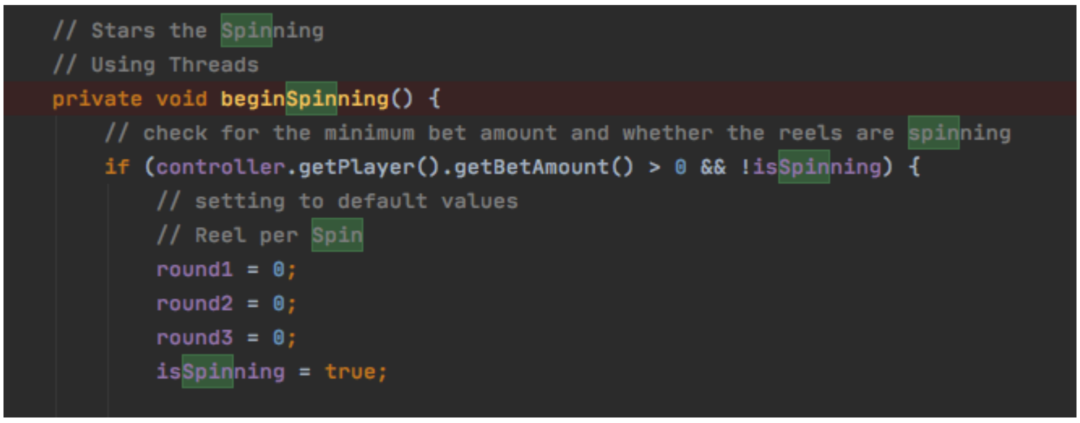
2 Solution

For the solution of this problem, I considered creating a slot machine through a video game where the operation of this machine is shown a bit as well as having some fun.

For the realization of this video game use multi-threads for the wheels that rotate simultaneously as well as for them to stop, verification is also reeling out to check if the images that come out agree with each other, thus it is determined if was won, lost or tied.

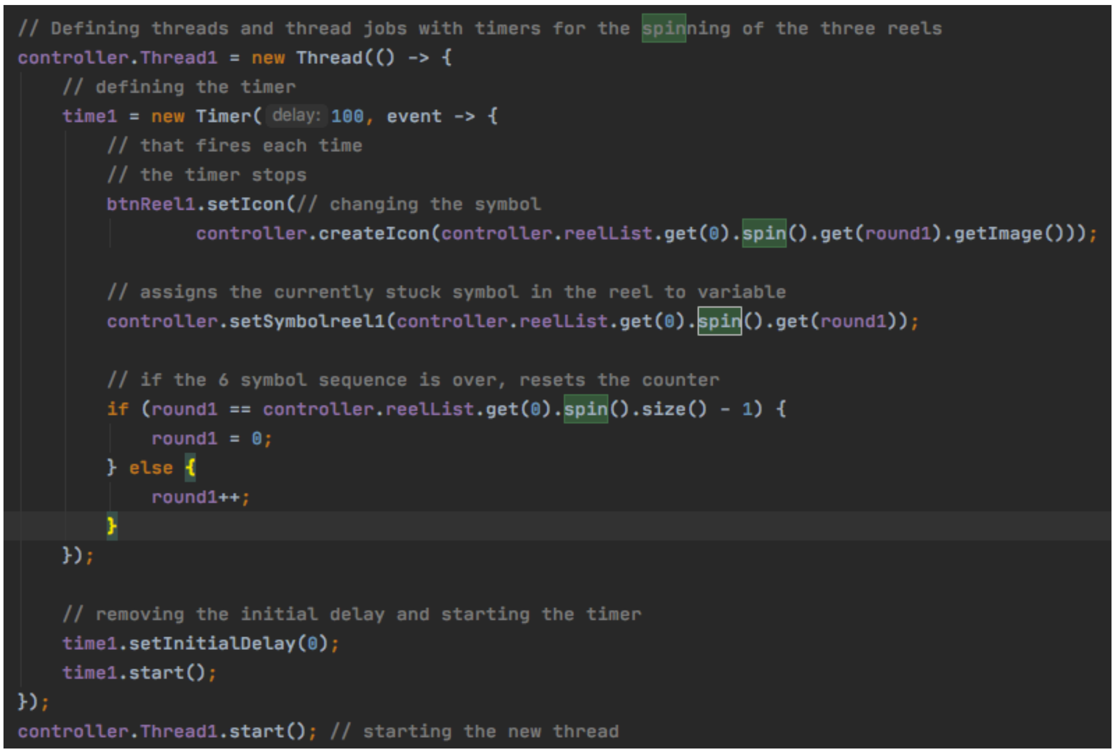
2.1 Start the spinning

In order for the wheel to start turning first, it is verified if the player has the minimum bet to start rolling them, the rounds that help to obtain the values of the images that are displayed are initialized.



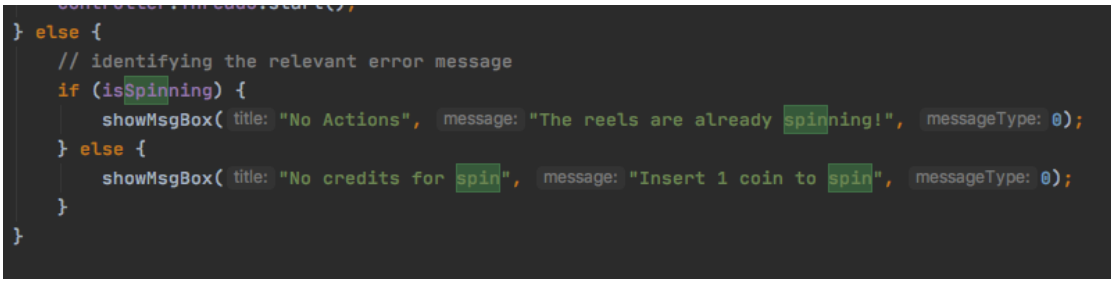
```
// Stars the Spinning
// Using Threads
private void beginSpinning() {
    // check for the minimum bet amount and whether the reels are spinning
    if (controller.getPlayer().getBetAmount() > 0 && !isSpinning) {
        // setting to default values
        // Reel per Spin
        round1 = 0;
        round2 = 0;
        round3 = 0;
        isSpinning = true;
```

A timer is used to give you the simulation that they rotate, the previously declared rounds work to obtain the values of the images they come from since only 6 images are used, those values are obtained so that the wheel is rotating, for later in the stop the value is obtained and is compared with the other reels, this is done 3 times since there are 3 reels working.



```
// Defining threads and thread jobs with timers for the spinning of the three reels
controller.Thread1 = new Thread(() -> {
    // defining the timer
    time1 = new Timer( delay: 100, event -> {
        // that fires each time
        // the timer stops
        btnReel1.setIcon(// changing the symbol
                controller.createIcon(controller.reelList.get(0).spin().get(round1).getImage()));

        // assigns the currently stuck symbol in the reel to variable
        controller.setSymbolreel1(controller.reelList.get(0).spin().get(round1));

        // if the 6 symbol sequence is over, resets the counter
        if (round1 == controller.reelList.get(0).spin().size() - 1) {
            round1 = 0;
        } else {
            round1++;
        }
    });

    // removing the initial delay and starting the timer
    time1.setInitialDelay(0);
    time1.start();
});
controller.Thread1.start(); // starting the new thread
```

In the end, only messages come out, both when the wheel is spinning and you don't spin it again, as well as if you have the minimum bet to be able to spin it.

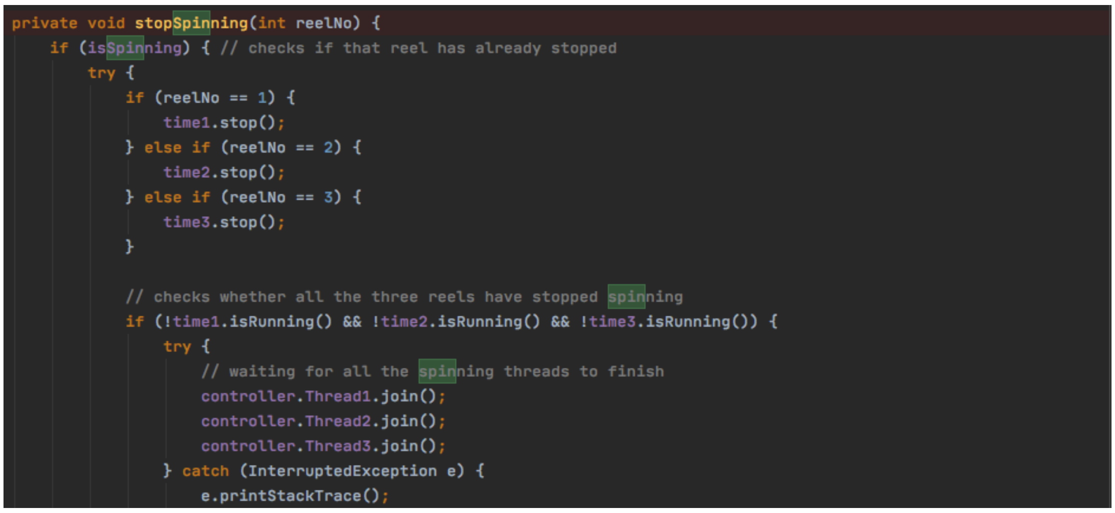


```
} else {
    // identifying the relevant error message
    if (isSpinning) {
        showMsgBox( title: "No Actions",  message: "The reels are already spinning!",  messageType: 0);
    } else {
        showMsgBox( title: "No credits for spin",  message: "Insert 1 coin to spin",  messageType: 0);
    }
}
```

2.2 Stop the spinning

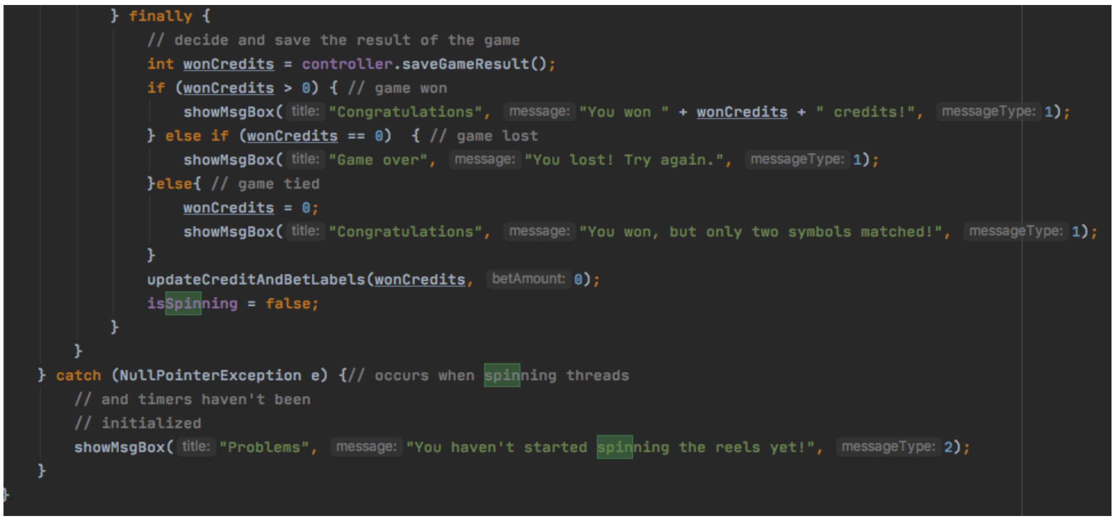
In this section, it is verified if the reels continue to rotate one by one when you click on one the timer that worked on each reel stops so that they are stopped one by one in some random image.

They are verified if the reels continue to rotate or are already stopped, but the three reels are expected to have finished giving the result.



```
private void stopSpinning(int reelNo) {
    if (isSpinning) { // checks if that reel has already stopped
        try {
            if (reelNo == 1) {
                time1.stop();
            } else if (reelNo == 2) {
                time2.stop();
            } else if (reelNo == 3) {
                time3.stop();
            }

            // checks whether all the three reels have stopped spinning
            if (!time1.isRunning() && !time2.isRunning() && !time3.isRunning()) {
                try {
                    // waiting for all the spinning threads to finish
                    controller.Thread1.join();
                    controller.Thread2.join();
                    controller.Thread3.join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
```

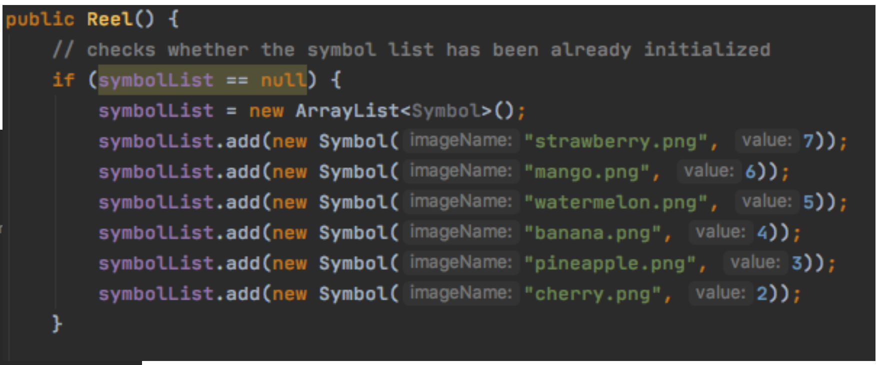
Once the turn of these is compared, some of the final results are, It was won, It was lost and if you get 2 equal images you are tied and you can play again,



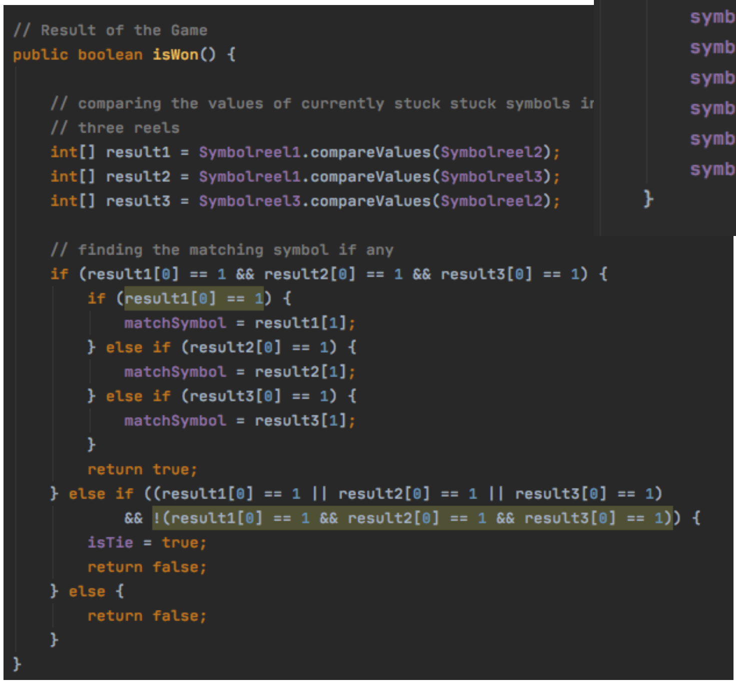
```
            } finally {
                // decide and save the result of the game
                int wonCredits = controller.saveGameResult();
                if (wonCredits > 0) { // game won
                    showMsgBox( title: "Congratulations",  message: "You won " + wonCredits + " credits!",  messageType: 1);
                } else if (wonCredits == 0)  { // game lost
                    showMsgBox( title: "Game over",  message: "You lost! Try again.",  messageType: 1);
                }else{ // game tied
                    wonCredits = 0;
                    showMsgBox( title: "Congratulations",  message: "You won, but only two symbols matched!",  messageType: 1);
                }
                updateCreditAndBetLabels(wonCredits,  betAmount: 0);
                isSpinning = false;
            }
        }
    } catch (NullPointerException e) {// occurs when spinning threads
        // and timers haven't been
        // initialized
        showMsgBox( title: "Problems",  message: "You haven't started spinning the reels yet!",  messageType: 2);
    }
}
```

2.3 Values and Obtain the Won

Each Image is a symbol, and a value is assigned to it to obtain it and to be able to make the turn with the counter when it begins to turn and also to make it possible to determine the winner by making comparisons between them, this is done by means of a list.



```
public Reel() {
    // checks whether the symbol list has been already initialized
    if (symbolList == null) {
        symbolList = new ArrayList<Symbol>();
        symbolList.add(new Symbol( imageName: "strawberry.png",  value: 7));
        symbolList.add(new Symbol( imageName: "mango.png",  value: 6));
        symbolList.add(new Symbol( imageName: "watermelon.png",  value: 5));
        symbolList.add(new Symbol( imageName: "banana.png",  value: 4));
        symbolList.add(new Symbol( imageName: "pineapple.png",  value: 3));
        symbolList.add(new Symbol( imageName: "cherry.png",  value: 2));
    }
```

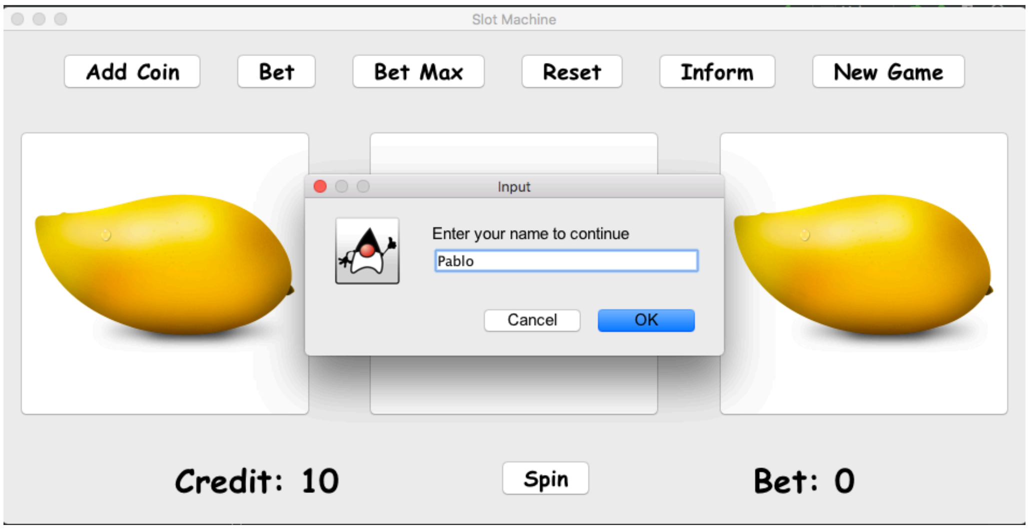


```
// Result of the Game
public boolean isWon() {

    // comparing the values of currently stuck stuck symbols ir
    // three reels
    int[] result1 = Symbolreel1.compareValues(Symbolreel2);
    int[] result2 = Symbolreel1.compareValues(Symbolreel3);
    int[] result3 = Symbolreel3.compareValues(Symbolreel2);

    // finding the matching symbol if any
    if (result1[0] == 1 && result2[0] == 1 && result3[0] == 1) {
        if (result1[0] == 1) {
            matchSymbol = result1[1];
        } else if (result2[0] == 1) {
            matchSymbol = result2[1];
        } else if (result3[0] == 1) {
            matchSymbol = result3[1];
        }
        return true;
    } else if ((result1[0] == 1 || result2[0] == 1 || result3[0] == 1)
            && !(result1[0] == 1 && result2[0] == 1 && result3[0] == 1)) {
        isTie = true;
        return false;
    } else {
        return false;
    }
}
```

3 Results

Here are some tests that were carried out, both of lost plays, won and tied as well as the operation of a file where the times you won or lost are saved. As well as a little user interface.

3.1 Test cases



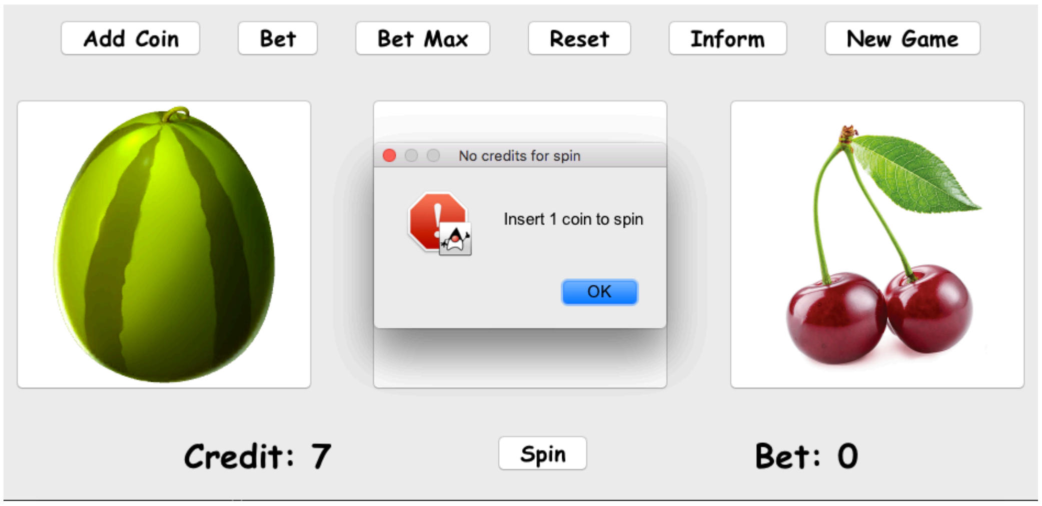
Enter your name to continue

Pablo

Credit: 10

Bet: 0

[Fig 1.1 Menu]

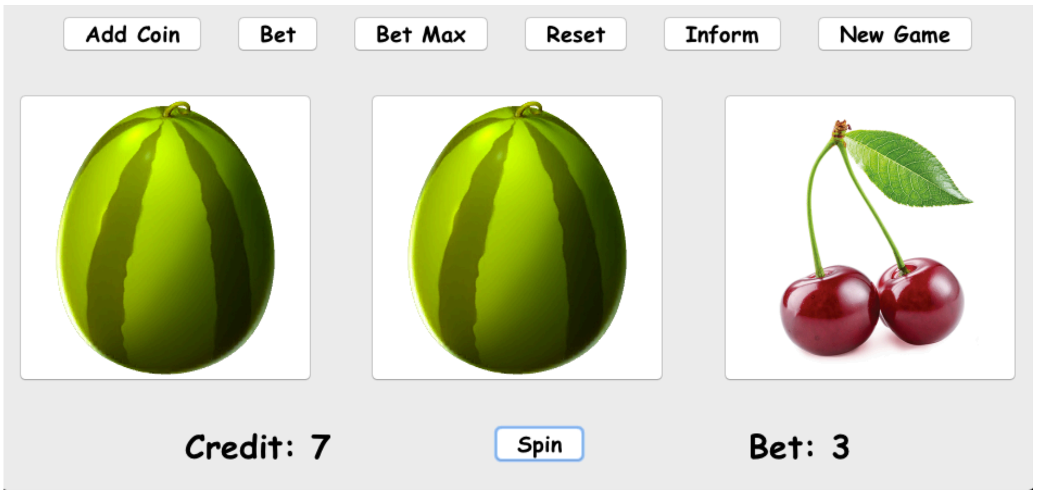


Insert 1 coin to spin

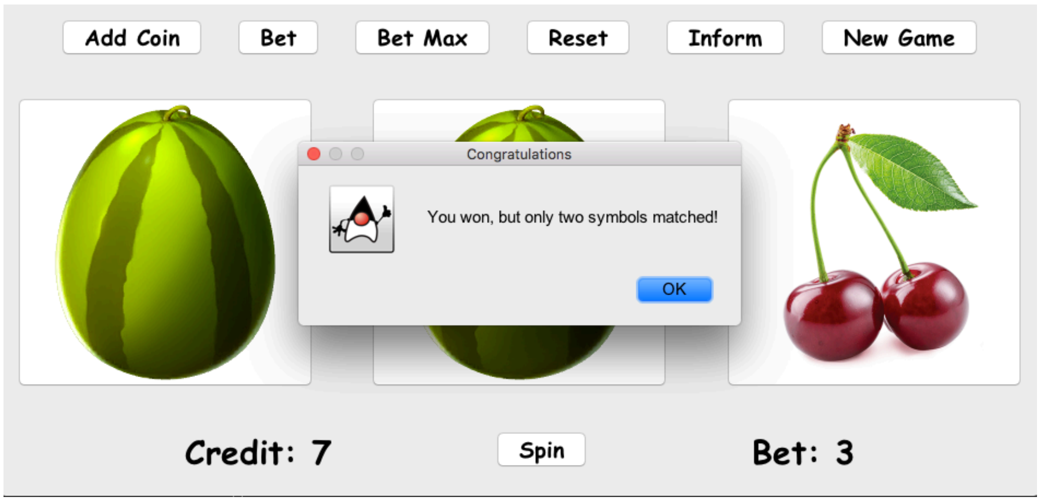
Credit: 7

Bet: 0

[Fig 1.2 Start but without bet]

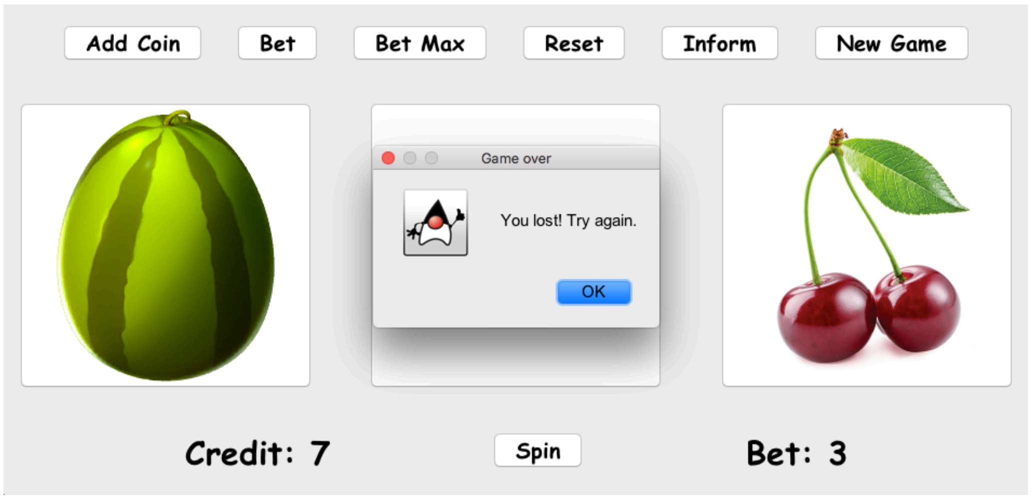


[Fig 1.3 You add the bet and start the Spin]



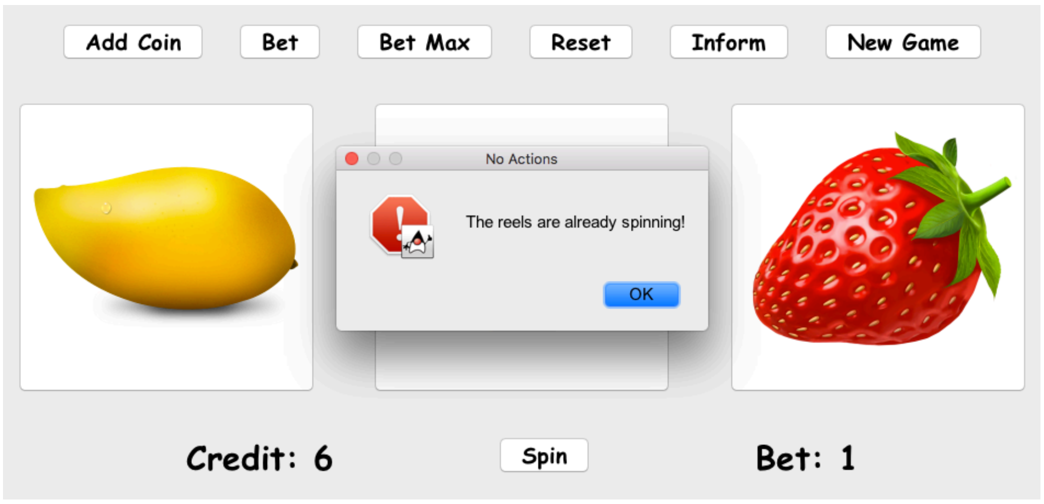
You won, but only two symbols matched!

[Fig 1.4 If you won but with two matches]



You lost! Try again.

[Fig 1.5 If you lose]



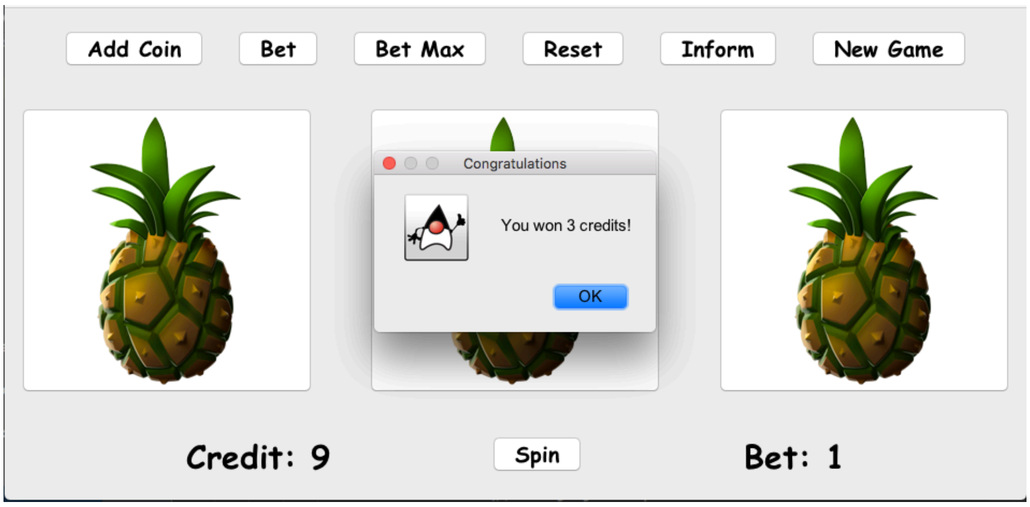
No Actions

The reels are already spinning!

Credit: 6

Bet: 1

[Fig 1.6 If you spin when the reels are spinning]



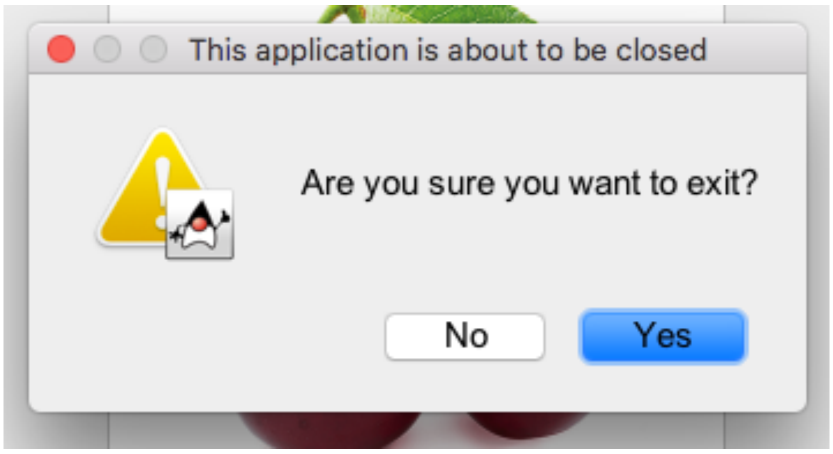
Congratulations

You won 3 credits!

Credit: 9

Bet: 1

[Fig 1.7 If you won]



This application is about to be closed

Are you sure you want to exit?

[Fig 1.8 Exit the game]

4 Conclusions

This project was carried out based on a saber on the use of multi threads and how they communicate with each other, I learned to use them more conveniently, and I wanted to show it through this video game where we can see their activity at the time of the wheels Spin as well as know more about the operation of slot machines. It is also for the purpose of having a little fun learning.

5 Setup instructions

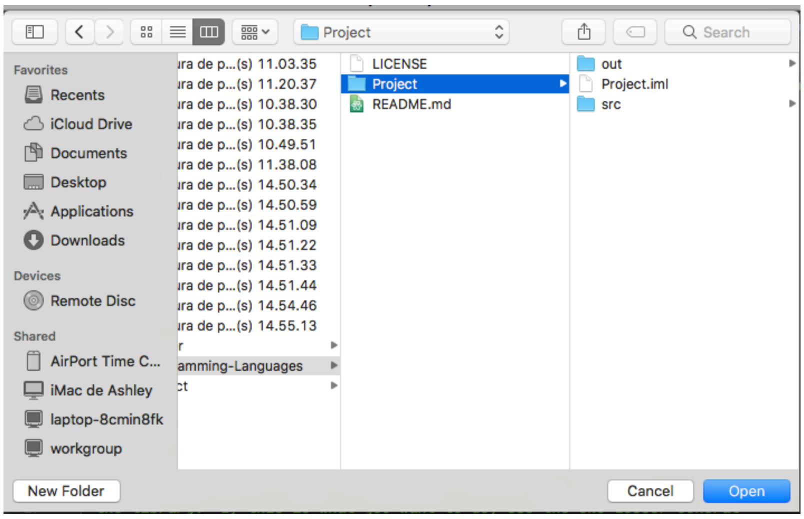
To start the program, it is necessary to clone or download the git found in the following link:

https://github.com/PPozos/Programming-Languages

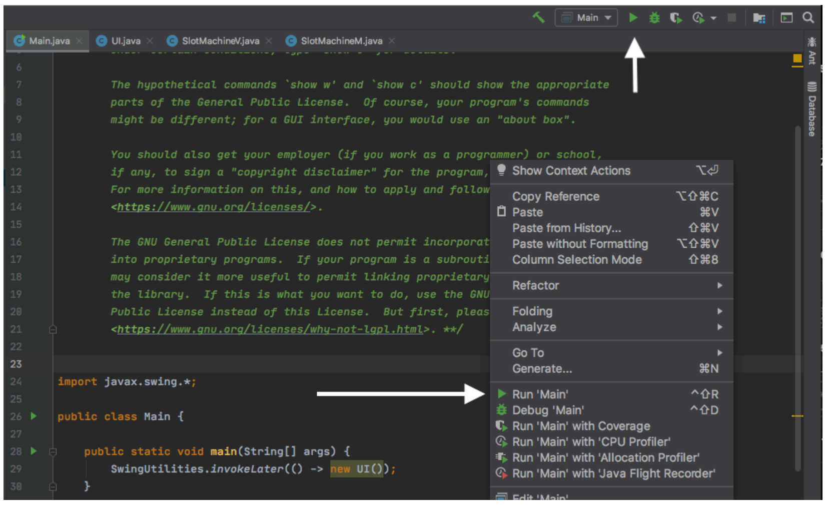
5.1 IDE

If you use some IDE like NetBeans, IntelliJ Idea, or other.

1. *Open app*
2. *Select open project*
3. *Select folder project*



4. *Go to the Main class*
5. *Right-click and click Run on the run arrow*

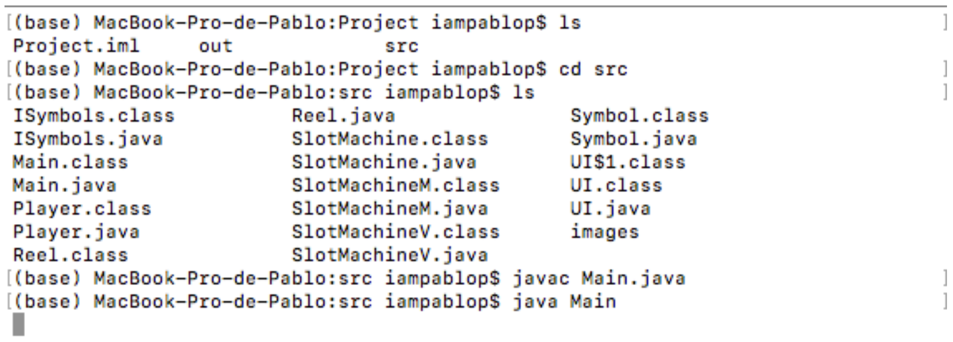


```
    The hypothetical commands `show w' and `show c' should show the appropriate
    parts of the General Public License.  Of course, your program's commands
    might be different; for a GUI interface, you would use an "about box".

    You should also get your employer (if you work as a programmer) or school,
    if any, to sign a "copyright disclaimer" for the program,
    For more information on this, and how to apply and follow
    <https://www.gnu.org/licenses/>.

    The GNU General Public License does not permit incorporat
    into proprietary programs.  If your program is a subrouti
    may consider it more useful to permit linking proprietary
    the library.  If this is what you want to do, use the GNU
    Public License instead of this License.  But first, pleas
    <https://www.gnu.org/licenses/why-not-lgpl.html>. **/


import javax.swing.*;

public class Main {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new UI());
    }
```

6. *Have fun*

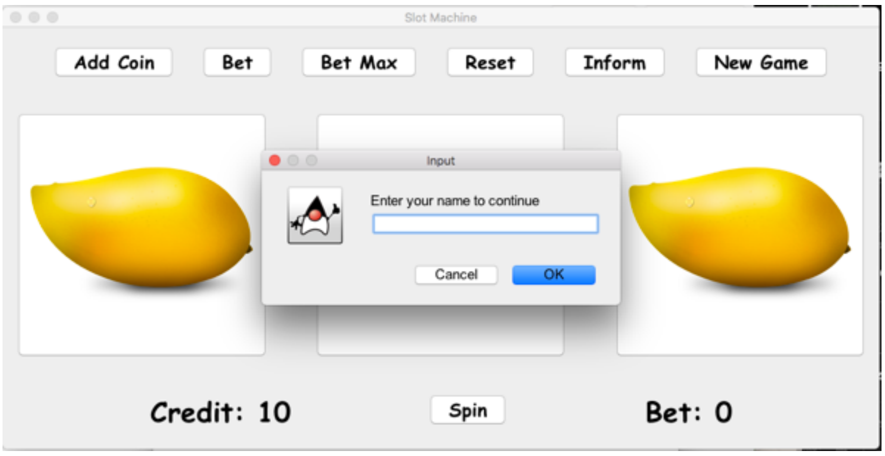
5.2 *Terminal*

1. *Open the terminal*
2. *Go to the path where is the folder of the project*
3. *Go to the folder src*
4. *Compile Main.java (javac Main.java)*
5. *Run (java Main)*



```
[(base) MacBook-Pro-de-Pablo:Project iampablop$ ls
 Project.iml      out             src
[(base) MacBook-Pro-de-Pablo:Project iampablop$ cd src
[(base) MacBook-Pro-de-Pablo:src iampablop$ ls
 ISymbols.class          Reel.java               Symbol.class
 ISymbols.java           SlotMachine.class       Symbol.java
 Main.class              SlotMachine.java        UI$1.class
 Main.java               SlotMachineM.class      UI.class
 Player.class            SlotMachineM.java       UI.java
 Player.java             SlotMachineV.class      images
 Reel.class              SlotMachineV.java
[(base) MacBook-Pro-de-Pablo:src iampablop$ javac Main.java
[(base) MacBook-Pro-de-Pablo:src iampablop$ java Main
```

6. *Enjoy*



6 References

6.1 References

[1]Multithreading and Concurrency Fundamentals. (2019, junio 24). Recuperado de https://www.educative.io/blog/multithreading-and-concurrency-fundamentals?aid=5082902844932096&utm_source=google&utm_medium=cpc&utm_campaign=blog-dynamic&gclid=EAIaIQobChMI5Nmm_4vV6QIVDdvACh2Z4w3OEAAYASAAEgI4g_D_BwE

[2] Harris, T. (2020, enero 27). How Slot Machines Work. Recuperado de https://entertainment.howstuffworks.com/slot-machine.htm

[3] Multithreading in Java. (2020, abril 26). Recuperado de https://www.geeksforgeeks.org/multithreading-in-java/