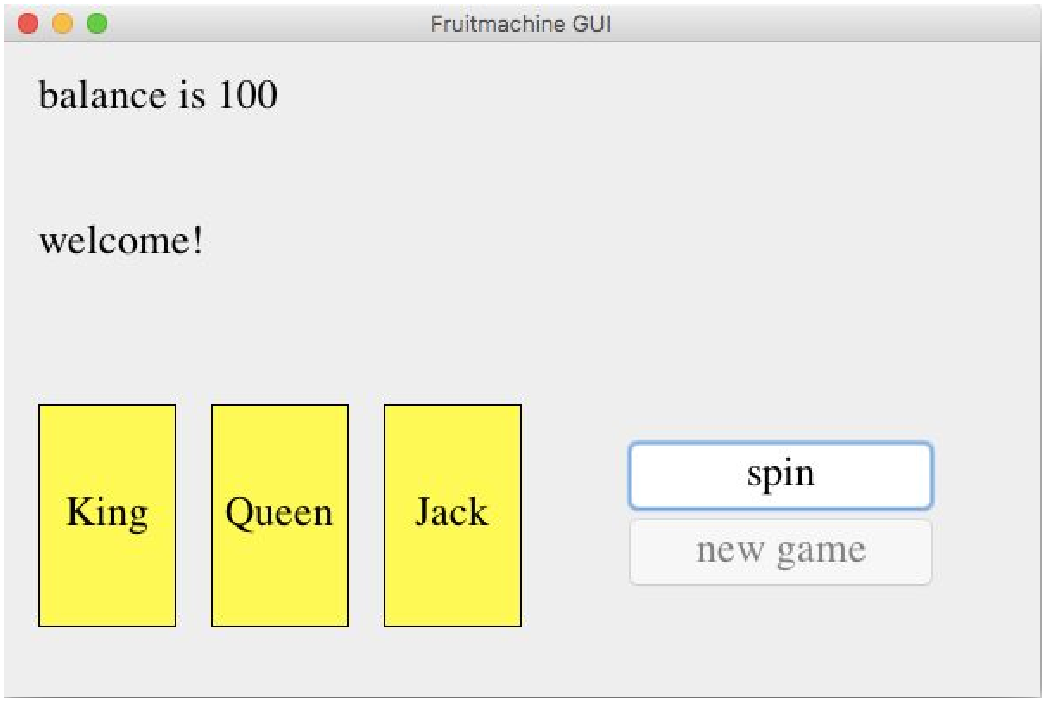
Programming IT 2019-2020

Assessed Exercise 3

Dr. Simon Rogers & Prof. Alice Miller

Fruit Machine

In this exercise you will implement a fruit machine game and user interface using Java and Java Swing. The game is illustrated below. Note that this image was generated from a Mac computer - it may look slightly different on yours.



The user interface contains two **labels** (top left hand quarter of the window). The first indicates the total balance of points held by the player (which is initially 100), and the second displays a message, which is initially set to “welcome!”

On the bottom left there are 3 **panels** which denote playing cards. These each contain a label indicating which card they represent. The labels on the cards can be any of “Ace”, “King”, “Queen”, “Jack” or “Joker”. The cards initially represent a King, Queen and Jack respectively.

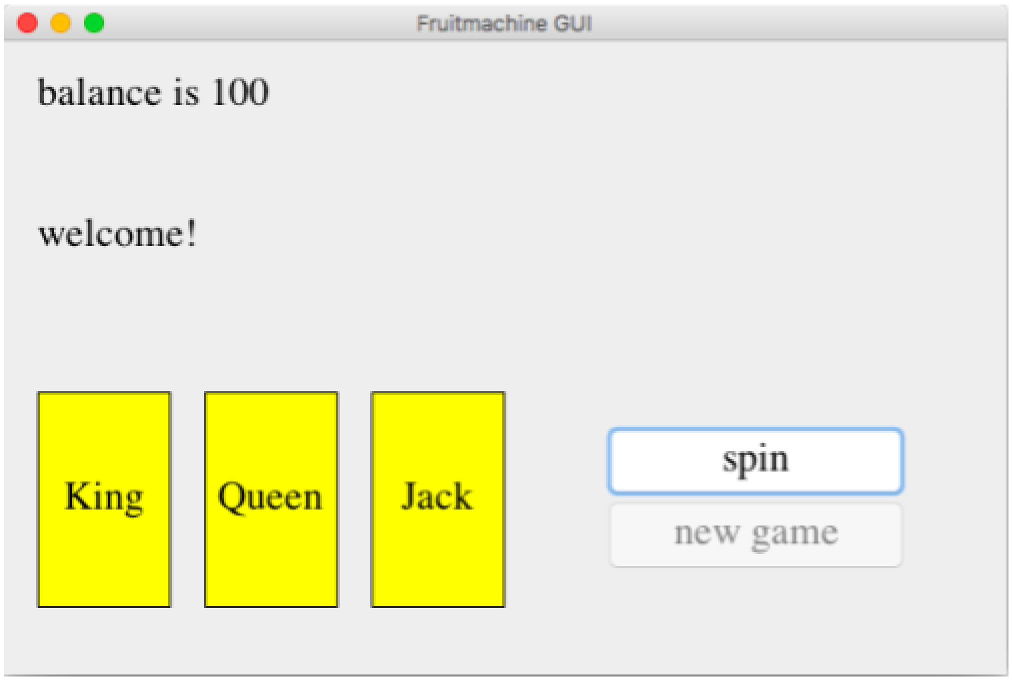
On the bottom right of the screen are two buttons. One is used to *spin* the cards. This entails setting each of the cards randomly to any card from the available set: “Ace”, “King”, “Queen”, “Jack” or “Joker” (each of the four outcomes is equally likely). The other button is used to start a new game (in which case the labels and cards are all set to their initial values).

When the cards have been *spun* a number of points are either added or subtracted from the current balance as follows:

- If at least one joker, 25 points are subtracted from the balance for every joker
- Otherwise (ie. if cards contain no jokers):
    - Increase the balance by 50 points if all three cards are the same
    - Increase the balance by 20 points if two of the cards are the same (but not all three)
    - Leave the balance unchanged if all three cards are different

The game continues until the player has less than 0 points, in which case the player loses; or the player has at least 150 points, in which case they win.

The following is a (rather unlucky!) sequence of screen shots showing a possible execution of the game:

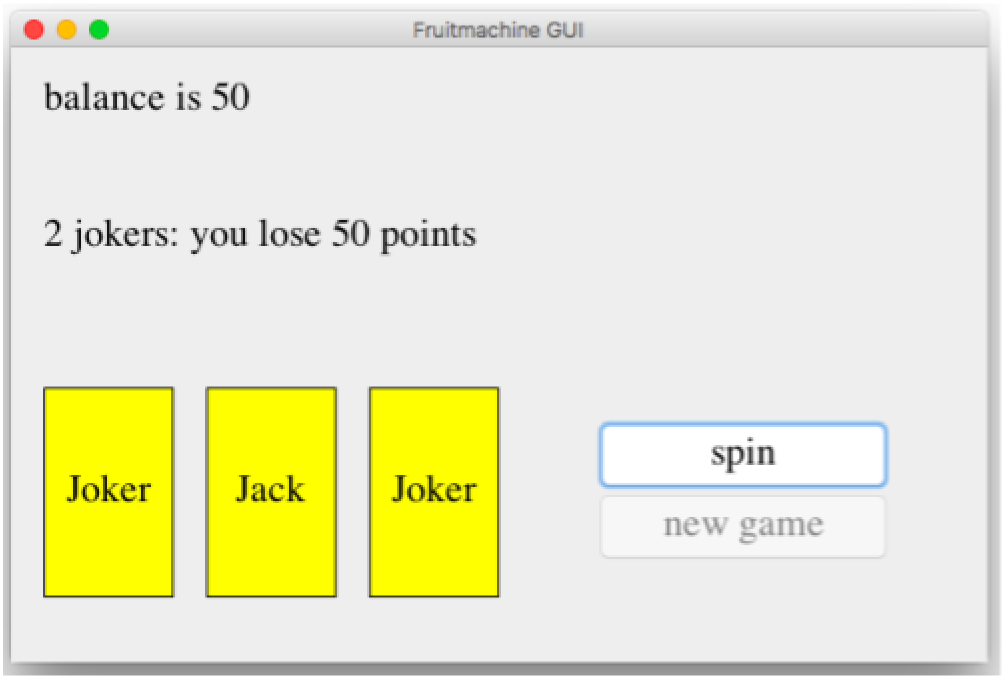


balance is 100

welcome!

King Queen Jack

1

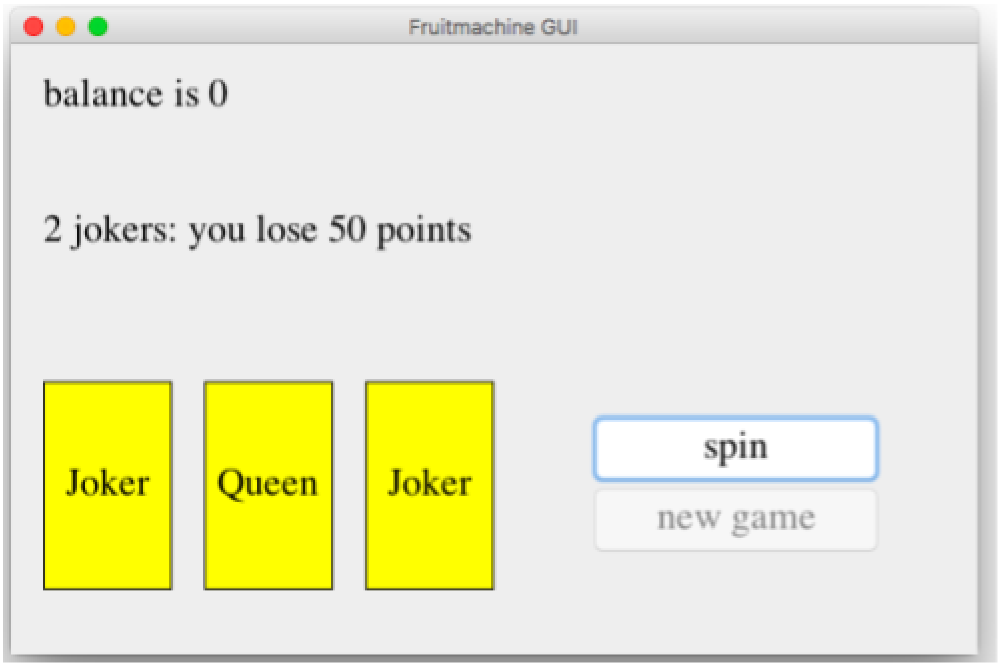


balance is 50

2 jokers: you lose 50 points

Joker Jack Joker

2

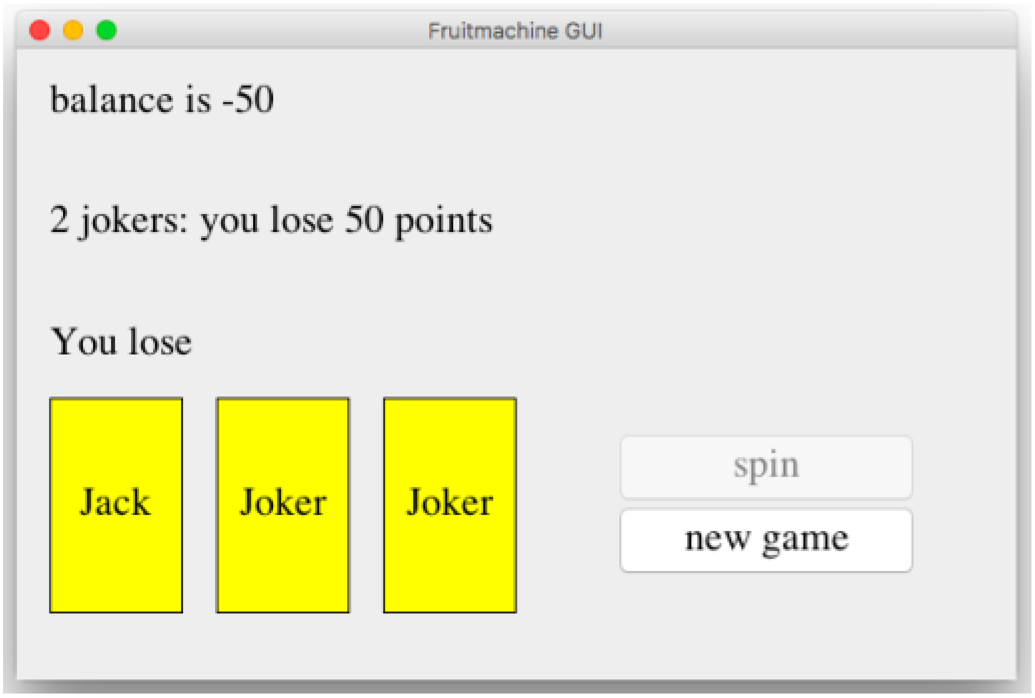


balance is 0

2 jokers: you lose 50 points

Joker Queen Joker

3



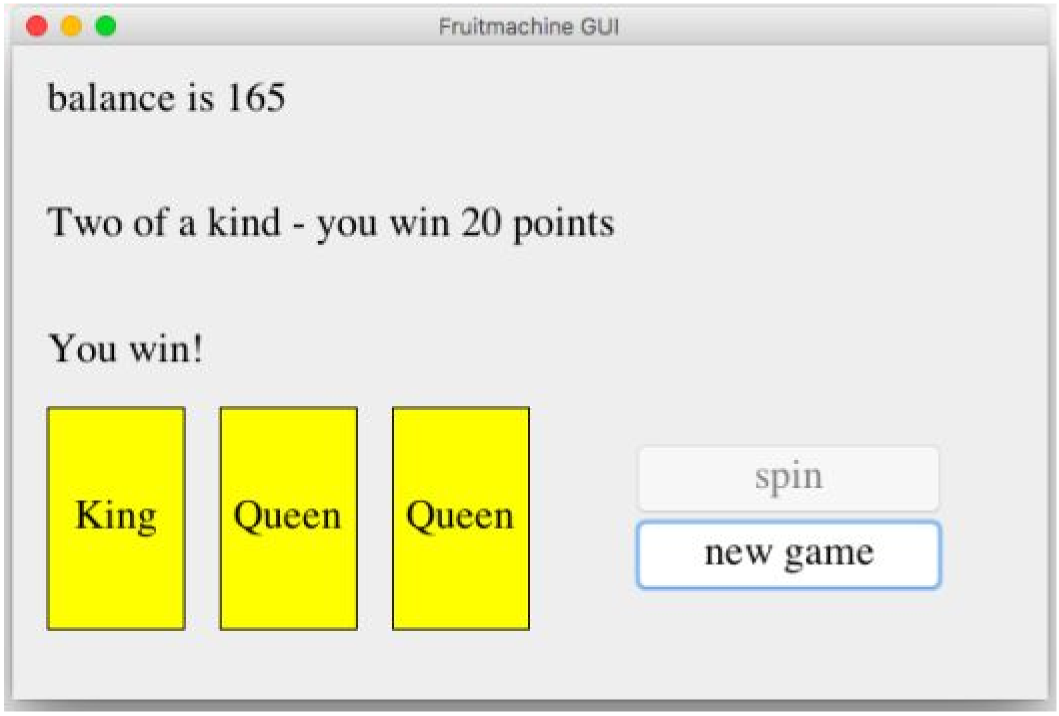
balance is -50

2 jokers: you lose 50 points

You lose

Jack Joker Joker

A screenshot denoting the final screen in a winning game is shown below:



balance is 165

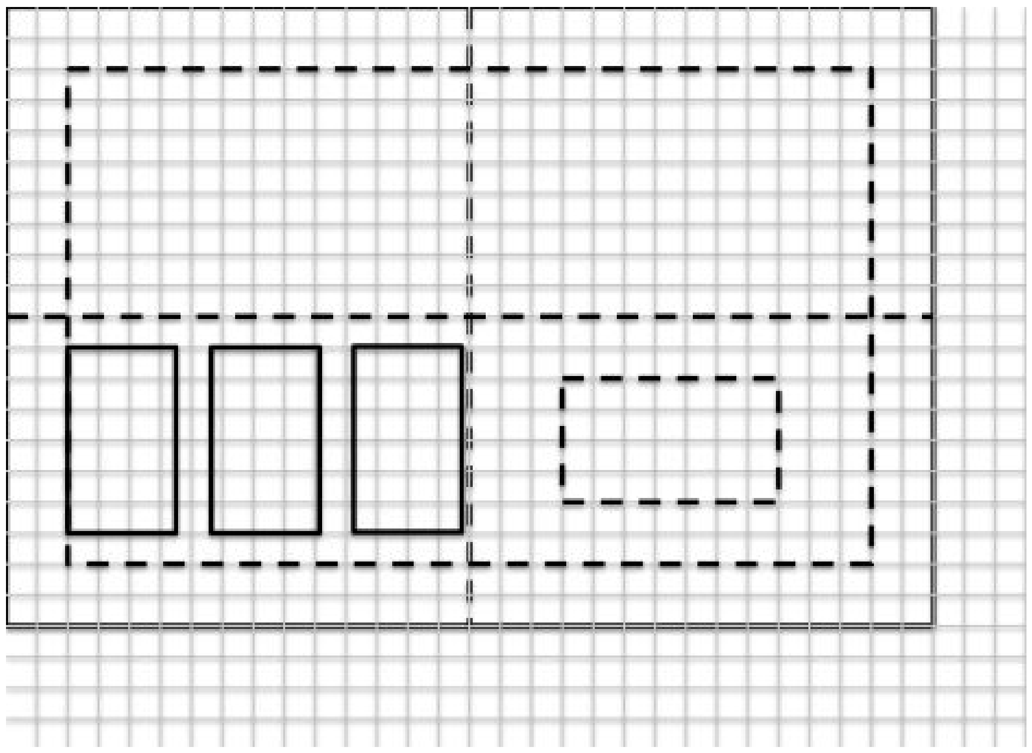
Two of a kind - you win 20 points

You win!

King Queen Queen

Note that when the game ends in each case the spin button becomes disabled and the new game button becomes enabled.

A sketch on squared paper which will help you with your layout (including borders) is provided below.



Task

You should create your own classes to implement the game and the user interface. Your class containing your main method must be called `FruitMachine.java` and you should submit all of your own classes that are required to run your program (this does not include classes that you import).You will also submit a short document (report.pdf), the content of which is explained below.

Note that the only components that should generate events are the two buttons.

Ensure your name and matric are given within comments in **every class you submit** and that your **report** also contains your name and matric. Classes should also contain comments indicating their purpose.

The marking scheme for this exercise is as follows.

1. If your code runs completely as expected [up to 22 marks]

   a. 16 marks for successful implementation of program
   b. Up to 6 marks for good programming style (e.g. appropriate use of multiple component classes, model-view-controller architecture, commenting etc.)

In this case, your report document should simply state “Program runs as required”.

2. If your code produces the correct output (layout) but the game cannot be played correctly [up to 16 marks = B2]
   a. Up to 5 marks for correct layout
   b. Up to 4 marks if ActionListeners are implemented correctly
   c. Up to 3 marks for well-designed class structure
   d. Up to 4 marks for your report which should state honestly the state of your program (which parts run, and which don’t) and what you think is wrong.

3. If your main method fails to produce even a correct layout [up to 8 marks = E1]
   a. Up to 5 marks for a reflective report indicating what attempts you made to implement your program.
   b. Up to 3 marks for well-designed class structure.

So….get the program working and you’ll get a B2. Get it working with excellent style, clear commenting, etc, and you’ll get more than a B2.
If your program doesn’t play correctly, the maximum you can be awarded is a B2.

Hints

- You might find it useful to think about what constitutes the **model**. At any point in the program, what is the state? I.e. what gets updated? Another way of thinking of this: what would I **need** to know to start up the program in the middle of a game.
- We think you probably need a minimum of three classes. Although, more would likely make the solution neater.
- You don’t need to match our layout exactly. Layout needs to be clear, but no extra marks for beauty.

Submission

Submit a .zip file including your .java files and your .pdf report via moodle. Deadline 09/12/2019 @ 17:30