EBU4201 Introductory Java Programming 2022/23

Mini Project

Task 1 [30 marks]

`SumItUp` is a simple application for children where they can practise their counting and adding skills (see Figure 1).

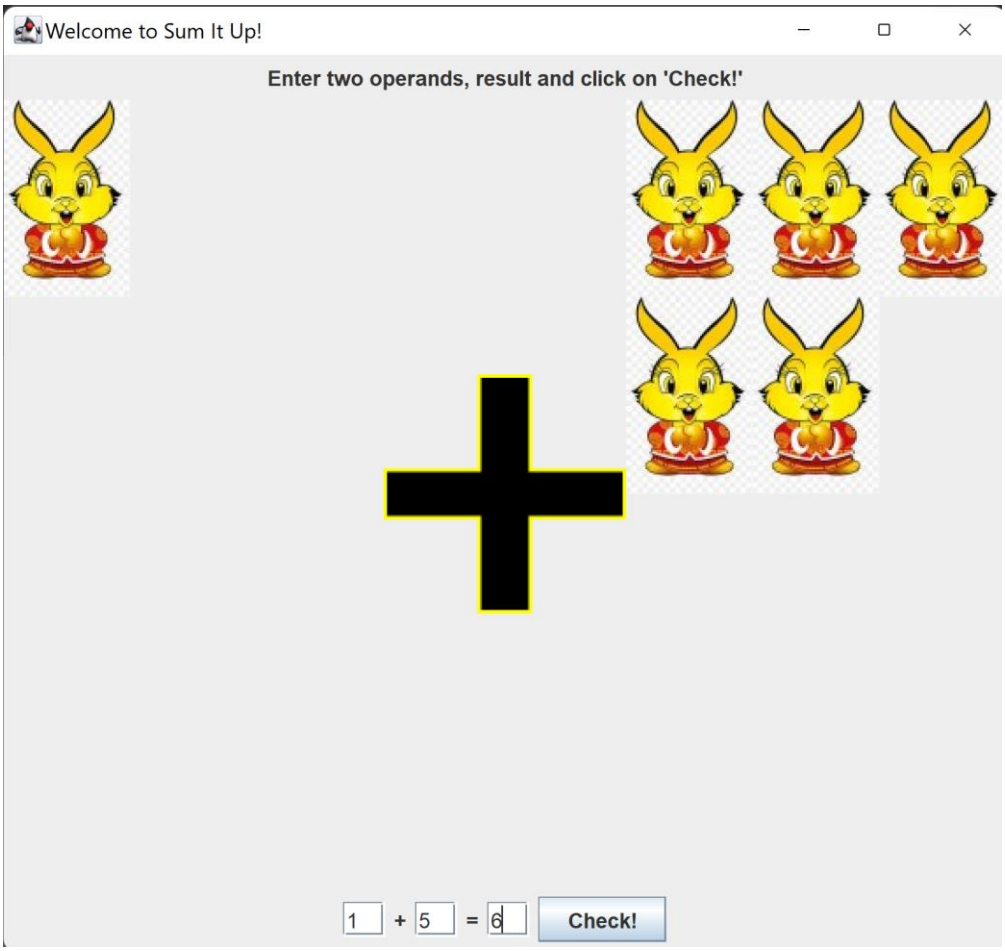

Figure 1 First launch with some input values entered

A random number of Rabbit images ranging from 1 to 10 are displayed for each operand and the user is expected to enter the values of the two operands and the result of adding the two operands, in the given text fields. When the user clicks on the button 'Check!', one of two things can happen:

Case 1: all three input values are correct
i) the text changes to '"Correct! Have another go?"'.
ii) the number of Rabbit images displayed for each of the two operands changes. See Figure 2 for an example.
iii) the three text fields are reset (i.e. they are made empty).

Case 2: at least one of the input values entered is incorrect
i) the text changes to 'Wrong! Try again!'.
ii) the number of Rabbit images displayed does NOT change.
iii) the text fields do NOT change.

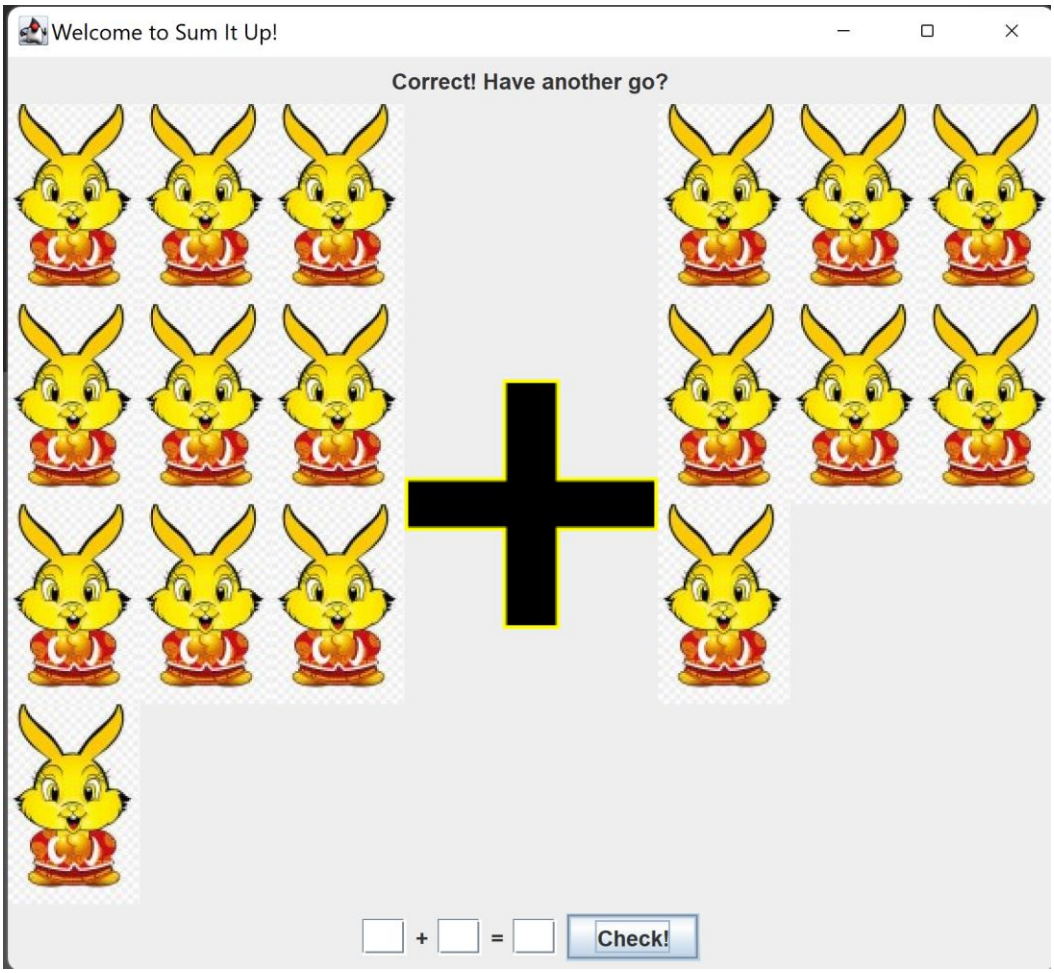

Figure 2 Action following Case 1

Implement `SumItUp` as a Java application. You application must satisfy ALL the specific requirements given below:

a) The title of the top-level container must be 'Welcome to SumItUp!'.
b) The initial text should be 'Enter two operands, result and click on 'Check!''. See Figure 1.
c) There should be no more than 4 Rabbit images per row. See Hint 1.
d) The text fields should be wide enough to display at least TWO characters.
e) The button 'Check!' must not resize when the GUI is resized. See Hint 2 and Figure 3.
f) The 'plus sign' icon should appear vertically centered between the two sets of Rabbit images and must not resize when the GUI is resized. See Hint 2 and Figure 3.
g) When first launched and whenever a correct answer is given, the number of displayed Rabbit images for each operand should change to any number between 1 and 10 (inclusive). See Hint 3 and Hint 4.
Note: It is possible for the next number(s) to be the same as the current number(s).
h) Nothing should happen if the user clicks the 'Check!' button while at least one of the text fields are empty, i.e. no errors should be thrown in this case.
Note: You can assume that only a numeric value will be entered into the text fields.

Hint 1: Use an array of `JLabel` components for the Rabbit images. The following constructor may be helpful for the 'plus sign' icon.

```
public JLabel(Icon image)
```

A Rabbit image and 'plus sign' image are provided. You must use these images.

Hint 2: Consider using containers within other containers and using layouts intelligently.

Hint 3: Suggested approach for displaying images: look up the following method in the class `javax.swing.JLabel`.

```
public void setIcon(Icon icon)
```

Hint 4: Suggested approaches for displaying a variable number of images: classes `java.util.Random` OR `java.lang.Math`.

Note: All the necessary files should be placed in a directory called `Task1`. You can choose whether to place the image files directly under `Task1` or within a sub-directory. Whichever approach you take, the images must be displayed on the GUI without having to move the image files to different locations within your directory structure. ***Also note that your application must run as expected from the command line on OpenJDK 19.0.2.***

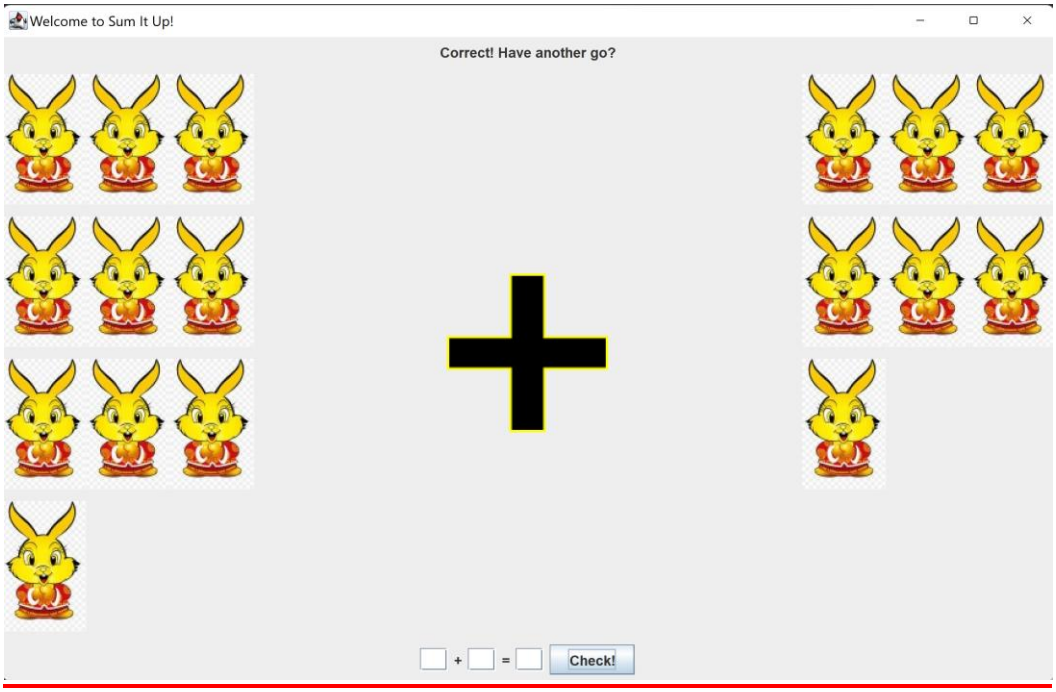

Figure 3 GUI Resized

Task 2 [10 marks]

You may notice that entering a non-numeric value and clicking the 'Check!' button will cause a run-time error on the console. Therefore, your second task is to improve the application developed in Task 1 to ensure the user is only allowed to supply valid input values, i.e. a

number between 1 and 10 (inclusive). The application must still function as specified in Task 1 although you may remove the 'Check!' button if you wish.

Hint: Use another appropriate component in place of the text field.

Note: All the necessary files (including any reused ones from Task 1) should be placed in a directory called `Task2`.

Documentation [10 marks]

You must include:

a. Generated Javadocs
b. Internal comments in your code.
c. User Manual. This should be no more than 2 pages and include how to run the program (both how to start and how to use it).

Note: All documentation files should be placed in a directory called `Documentation`.

Extra Credit [5 marks]

Extra marks from this section can be used to top up your final grade for this project. Maximum mark is still 50.

Further improve your application such that the maximum number of Rabbit images displayed for each operand can be any number between 10 and MAX (inclusive), specified as a command line argument. E.g. assuming your class is called `SumItUpExtra`, the command

```
Java SumItUpExtra 20
```

will launch a GUI similar to Figure 1 where the maximum number of Rabbit images displayed per operand will be 20. Whenever a correct answer is entered, the number of Rabbit images per operand will change to any number between 1 and 20 (inclusive). There should be no more than 5 Rabbit images per row.

MAX must be calculated as follows.
If the last digit of your QM student number is 0-4, MAX = 20
If the last digit of your QM student number is 5-9, MAX = 25

If no command line argument is given OR a number outside the valid range is given, the program must terminate printing out an appropriate error message to the console. You can assume only a numeric value will be given as the argument.

Note: All the necessary files (including any reused ones from Task 1 and Task 2) should be placed in a directory called `ExtraCredit`.

Important notes:

1. All three directories must be included in a zip file. The filename must be your **QM Student Number**.

2. You should design your classes properly, following object oriented principles. E.g. do NOT write everything in the main method, keep code repetition to a minimum (i.e. use methods), do NOT use static methods unless there is a good reason. There will be marks allocated for good design.