**Family Name:** ..... **Other Names:** .....

**Student ID:** ..... **Signature** .....

# COMP 102 : Test 3

2022, 9 December ** WITH SOLUTIONS **

## Instructions

- Time allowed: **45 minutes**
- Attempt **all** the questions. There are 30 marks in total.
- Write your answers in this test paper and hand in all sheets. (We have different instructions for distance students)
- If you think a question is unclear, ask for clarification.
- Brief Java documentation is provided with the test.
- This test contributes 10% of your final grade.
- You may use dictionaries and calculators.
- You may write notes and working on this paper, but make sure your answers are clear.
- You may assume all the programs import the ecs100 library and other standard libraries.

| Questions | Marks | |
|---|---|---|
| 1. Counted **for** loops | [7] | |
| 2. Defining Objects | [9] | |
| 3. Files without headers | [8] | |
| 4. While loops with numbers | [6] | |
| | TOTAL: | |

**SPARE PAGE FOR EXTRA ANSWERS**

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

**Question 1. Counted for loops** **[7 marks]**

Complete the following addSequence(...) method. It has two integer parameters, first and last, and returns an int.

The method should use a **for** loop to add up and return the sum of all the integers from first to last (inclusive).

If there are no numbers in the sequence (ie, last is smaller than first), then it should return 0.

For example:

addSequence(10, 13) should return 46 (because 10+11+12+13 = 46).
addSequence(15, 8) should return 0.
addSequence(7, 7) should return 7.

**Hint**: use a counted **for** loop.

```
public int addSequence(int first, int last){
    int total = 0;
    for (int num = first; num <= last; num++){
        total = total + num;
    }
    return total;









}
```

**Question 2. Defining Objects** **[9 marks]**

Suppose you are writing a program for a factory simulation game which involves filling sacks with different kinds of grains, such as wheat, rice, and oats. Each sack only has one kind of grain.

Part of the program is a Sack class on the next page. Complete the definition of the Sack class, with fields, a constructor, and three methods:

- There are two pieces of information stored in a Sack object:
  - the kind of grain in the sack (a String),
  - the amount of grain in the sack (a double).
- Every new Sack object starts with 2.5 kg of grain in the sack.
- The kind of grain in the sack is set when a Sack object is constructed. The constructor should have one parameter for the kind of grain.
- The getKind() and getAmount() methods return the kind of grain in the sack and the current amount of grain in the sack.
- The add(double amt) method will add the specified amount of grain to the sack. amt must be positive. If it is negative, the amount of grain in the sack is not changed, and the method prints out "Can't add a negative amount".

(Question 2 continued on next page)

**(Question 2 continued)**

```
/** Represents a sack of grain in the simulation game */
public class Sack{
// fields
    private String kind;
    private double amount = 2.5;



// constructor
    public Sack(String kind){
        this.kind = kind;
    }



//methods
    public String getKind(){
        return this.kind;


    }
    public double getAmount(){
        return this.amount;


    }
    public void add(double amt){
        if (amt >= 0) {
            this.amount = this.amount + amt;
        }
        else {
            UI.println("Can't add a negative amount");
        }









    }
}
```

**Question 3. Files without headers** **[8 marks]**

Complete the drawNamesFromFile() method on the next page. The method should read data from the file called "`namecloud.txt`", and draw the names from the file on the graphics pane.

Each line of "`namecloud.txt`" contains the data about one name:
*first-name last-name fontsize x y*

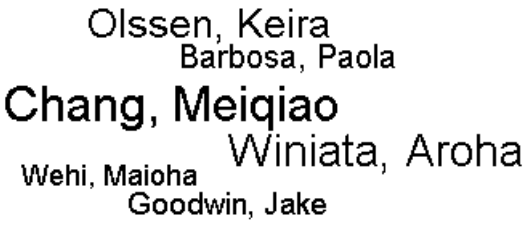
The method should draw the name in the format

*last-name, first-name*

at the specified (x,y) position with the specified font size.
(Note, the names do not contain any spaces)

An example of part of "`namecloud.txt`" is shown below, along with the diagram it should draw.

```
Keira Olssen 25 184 46
Maioha Wehi 18 143 138
Aroha Winiata 28 272 126
Jake Goodwin 19 209 156
Meiqiao Chang 30 132 98
Paola Barbosa 20 241 65
```




**Hint**: use a new Scanner on each line.

(Question 3 continued on next page)

**(Question 3 continued)**

```
public void drawNamesFromFile(){
    try{
        List<String> lines = Files.readAllLines(Path.of("namecloud.txt"));
        for (String line : lines){
            Scanner sc = new Scanner(line);
            String firstName = sc.next();
            String secndName = sc.next();
            double fontSize = sc.nextDouble();
            double x = sc.nextDouble();
            double y = sc.nextDouble();
            UI.setFontSize(fontSize);
            String name = secndName+", "+firstName;
            UI.drawString(name, x, y);
        }









    } catch(IOException e){UI.println("Fail: " + e);}
}
```

**Question 4. While loops with numbers** **[6 marks]**

Complete the findSpikes(..) method on the next page. The method is passed the name of a file containing a sequence of measurements of pressure in a cylinder that should be changing smoothly over time.

The method should read the sequence and identify "spikes" — places where a pressure value is more than 10 units larger than the value right before it and more than 10 units larger than the value right after it.

For example, given the file of numbers on the left, the method should print the output on the right:

```
14 13 56 33
32 45 43 38 34
37 15 17
29 14 32 11 10 25 24
```

```
Spike:  13 - 56 - 33
Spike:  17 - 29 - 14
Spike:  14 - 32 - 11
```

**Hint:** Create a Scanner around the whole file and read the numbers one at a time from the Scanner.

You may assume the file contains at least three numbers.

(Question 4 continued on next page)

**(Question 4 continued)**

```
public void findSpikes(String filename){
    try{
        Scanner sc = new Scanner(Path.of(filename));
        double previous = sc.nextDouble();
        double number = sc.nextDouble();
        while (sc.hasNextDouble()){
            double nextNum = sc.nextDouble();
            if ( (number - previous) >10 && (number - nextNum) > 10){
                UI.printf("Spike: %.0f - %.0f - %.0f\n", previous, number, nextNum);
            }
            previous = number;
            number = nextNum;
        }
        sc.close();

// Note, could have used int and sc.nextInt() instead of double and sc.nextDouble()
// Also, previous and number could have been initialised with Double.MAX_VALUE
//  instead of with the first two numbers in the file
// There are other correct ways of doing it.









    }
    catch(IOException e){UI.println("Fail: " + e);}
}
```

* * * * * * * * * * * * * * *

**SPARE PAGE FOR EXTRA ANSWERS**

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.