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

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

COMP 102 : Test 1

2024, April 15 ** WITH SOLUTIONS **

Instructions

- Time allowed: **60 minutes**
- Attempt **all** the questions. There are 50 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 30% of your final grade.
  (But your mark will be increased to your exam mark if that is higher.)
- 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.	User input	[10]	
2.	Writing programs with <u>if</u>	[10]	
3.	Defining methods with parameters	[10]	
4.	Writing methods that use objects	[10]	
5.	Writing programs with <u>while</u>	[10]	
		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. User Input **[10 marks]**

Complete the printMinus() method below so that it:

- Asks the user for two doubles.
- Prints out the result of the calculation as shown below in the example.

Example:

```
First number:  10
Second number:  7
10.0 - 7.0 = 3.0
```

```
public void printMinus () {

    double n1 = UI.askDouble("First number:");
    double n2 = UI.askDouble("Second number:");

    UI.println(n1 + " - " + n2 + " = " + (n1 - n2));
//OR
    UI.printf("%.1f - %.1f = %.1f\n", n1, n2, n1-n2);

}
```

Question 2. Writing programs with if **[10 marks]**

Complete the calculateTax method on the facing page to calculate income tax. It must **return** the calculated tax as a (double). Calculate the tax based on the following tax rates:

Salary	Tax rate
$10000.00 or below	0%
From $10000.01 and up to $50000.00	30%
From $50000.01	50%

Examples:

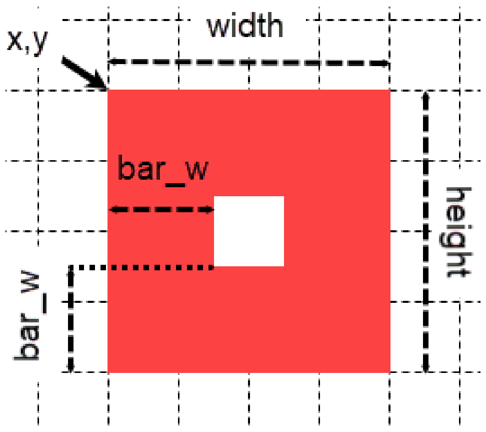
- calculateTax(-10000) should return 0
- calculateTax(7000) should return 0
- calculateTax(15000) should return 1500
  - because (15000-10000) * 0.3 = 1500
- calculateTax(50000) should return 12000
  - because (50000-10000) * 0.3 + (50000-50000) * 0.5 = 12000
- calculateTax(150000) should return 62000
  - because (50000-10000) * 0.3 + (150000-50000) * 0.5 = 62000

```
/** Returns the income tax associated with the given salary */
public double calculateTax(double salary){



        double tax;

        if(salary < 10000) {                              //(salary <= 10000)
            tax = 0;
        }
        else if(salary < 50000) {                         //(salary <= 50000)
            tax = (salary - 10000) * 0.3;
        }
        else {
            tax = 40000 * 0.3 + (salary - 50000) * 0.5;
        }

        return tax;
















  }
```

Question 3. Defining methods with parameters **[10 marks]**

The program on the facing page has two methods. The testDrawBoxes() method calls the drawBox(...) method two times to draw two different boxes.

(a) **[8 marks]** Complete the drawBox(...) method to draw boxes, similar to the one below. A box is made up of two rectangles, a colored rectangle with an inner box which is drawn using Color.white. The inner box is placed in the middle of the colored box.



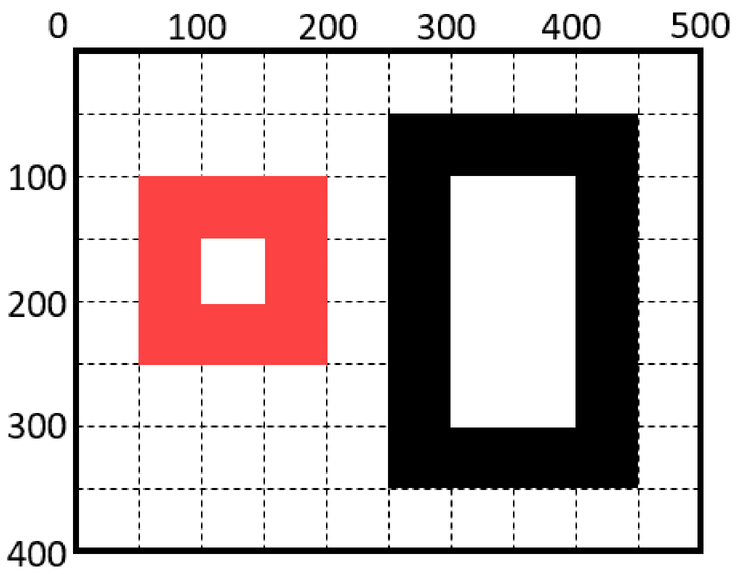
You can assume bar_w is always smaller than width/2 and height/2

The dotted lines should not be drawn. They are in the picture to indicate the positions and sizes of the boxes.

You will need to define six parameters for drawBox(...):

- the x, y position of the box (i.e. top-left of the box as indicated above).
- the width of the box.
- the height of the box.
- the bar_w which specify the width of the colored area around the inner-rectangle.
- the color of the colored rectangle

(b) **[2 marks]** Complete the testDrawBoxes() method to draw two boxes like the ones in the example picture below. The box on the left is using Color.red, the other uses Color.black. The boxes align perfectly with the dotted lines in the diagram.



```
public class Drawer {
    // Draws one box at given position, width, height, bar_w, and color.
    public void drawBox(   double x, double y, double width, double height,
                           double bar_w, Color c
                                                                          ) {
        double white_x  = x + bar_w;
        double white_y  = y + bar_w;
        double white_wd = width - 2*bar_w;
        double white_ht  = height - 2*bar_w;

        UI.setColor(c);
        UI.fillRect(x, y, width, height);

        UI.setColor(Color.white);
        UI.fillRect(white_x, white_y, white_wd, white_ht);










    }

    // Draws two boxes.
    public void testDrawBoxes() {
        UI.clearGraphics();
        this.drawBox(
                     50, 100, 150, 150, 50, Color.red



                                                                          );
        this.drawBox(
                     250, 50, 200, 300, 50, Color.black



                                                                          );
}
```

Question 4. Writing methods that use objects **[10 marks]**

A dog simulator draws various types of dogs on the screen at specified position. A dog can:

1. Run a specified distance either North, South, East, or West
2. Eat food
3. Check if it collides with another Dog

The Dog class defines four constants, a constructor, and three methods. Here is the documentation:

```
class Dog {
//Constants
    public static final String NORTH = "NORTH";
    public static final String SOUTH = "SOUTH";
    public static final String WEST = "WEST";
    public static final String EAST = "EAST";

// Constructor to make a new Dog object
    public Dog(String type, double x, double y)
    /**The type parameter can only be "Beagle", "Bulldog", or "Poodle"
       Creates a new Dog object of a specified type at the (x,y) position in the screen */

// Methods that can be called on a Dog object

    public void run(String dir, double distance)
    /** dir should only be passing the constants (NORTH, SOUTH, EAST, or WEST)
        distance > 0, the object moves the distance (in pixels) towards dir
        distance =<0, the object stays in the current position */

    public boolean eat()
    /** Eats any food at the current location.
        return true, if there was food at the location to eat
        return false, if there was no food at the location */

    public boolean isColliding (Dog other)
    /** Checks if the current Dog collides with the other Dog object */
}
```

Complete the following animate() method in the DogGame class so that it

1. Creates three Dog objects. The screen is 600 on the x direction and 400 on the y direction. The first should be a "Beagle" at position (300,500), the second should be a "Poodle" at position (300,200), and the third should be a "Bulldog" at a random position within the screen.
2. The first dog runs 50 North. (Using the appropriate constant)
3. The second dog attempts to eat where it is. If there were food to eat, it should run 300 South. (Using the appropriate constant)
4. Then, prints "Dog fight", if any dog collides with another dog.

You **should** use methods in the Dog class to do all the steps above, i.e. there is no need to use loops to achieve the actions.

```
public class DogGame{
    public static final String NORTH = "NORTH";
    public static final String SOUTH = "SOUTH";
    public static final String WEST = "WEST";
    public static final String EAST = "EAST";

    public void animate(){
        Dog dog1 = new Dog("Beagle", 300, 500);

        Dog dog2 = new Dog("Poodle", 300, 200);

        double x = Math.random() * 600;
        double y = Math.random() * 400;
        Dog dog3 = new Dog("Bulldog", x, y );

        dog1.run(NORTH, 50);

        if (dog2.eat()) {
            dog2.run(SOUTH,300);
        }

        if (dog1. isColliding (dog2) || dog1. isColliding (dog3) ||
            dog2. isColliding (dog3))) {
            UI. println ("Dog fight");
        }












    }
}
```

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.

Student ID: ....

Question 5. Writing methods with <u>while</u> **[10 marks]**

The program for this question has two sub-questions.

(a) **[5 marks]** Complete the `count(int start, int end)` method, using a while loop, to return each number in a String from start to end. If end is smaller than start, return "ERROR".

For example

- count(-1, 1) returns "-1 0 1 "
- count(1, 1) returns "1"
- count(1, -1) returns "ERROR"

```
public String count(int start , int end) {

    if(end < start) {
        return "ERROR";
    }

    String result = ""

    while( start <=end) {
        result = result + start + " ";
        start ++;
    }

    return result ;

}
```

(Question 5 continued on next page)

(Question 5 continued)

(b) **[5 marks]** Complete the drawCircles(double x, double y, double width, int numx, int numy, Color c) method, so that it can draw a rectangle filled with circles.

The parameters of the method specifies the following:

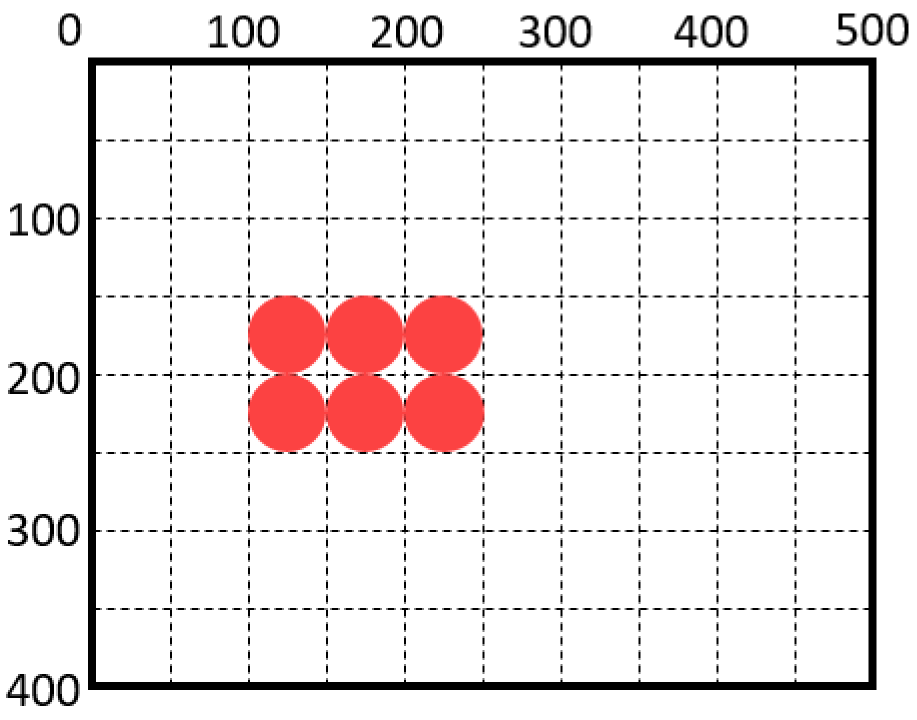
- x and y the top-left corner of the rectangle.
- width of the rectangle.
- numx the number of circles in the x-direction.
- numy the number of circles in the y-direction.
- c the color of the circles.

Note, the height is not provided by the call. The diameter of the circles is calculated using the width and numx.

The following call, drawCircles(100, 150, 150, 3, 2, Color.red) draws the below square of circles (excluding any lines and numbers in the figure.)

You must only use while loops to perform looping in your code.

For experienced students : Recursion is not allowed.



(Question 5 continued on next page)

(Question 5 continued)

```
public void drawCircles(double x, double y, double width,
                        int numx, int numy, Color c) {

    //Only use while loops to perform the looping in the code

    UI.setColor(c);

    double dia = width / numx;

    int col = 0;

    while(col < numx) {
        int row = 0;
        while(row<numy) {
            double cx = x + col * dia;
            double cy = y + row * dia;
            UI.fillOval (cx, cy, dia, dia);
            row++;
        }
        col++;
    }

}
```

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

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.