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

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

COMP 102: Test 5

2022, 21 December ** WITH SOLUTIONS *

Instructions

- Time allowed: **3 hours**
- Attempt **all** the questions. There are 120 marks in total.
- Write your answers in this test paper and hand in all sheets.
- 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 may also boost your mark for any or all of the previous tests.
- 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. Basic Java	[48]	
2. Event-driven Programs	[10]	
3. Using While	[12]	
4. Reading from Files	[20]	
5. ArrayLists	[20]	
6. 2D Arrays	[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. Basic Java **[48 marks]**

(a) **[6 marks] Variables and loops.**

What will the following code fragment print out?

```
int a = 10;
int count = 1;
for (int i = a;  i <= 10000; i=i*10){
    UI.println("i:  " + i);
    count = count+1;
}
UI.println("count:  " + count);
```

```
i:  10
i:  100
i:  1000
i:  10000
count:  5
```

(b) **[7 marks] Counted for loops.**

Complete the following printSquareRoots(int max) method which should print out a table of the square roots of the numbers from 1 to max. The square roots should be printed with 4 decimal places.

The table should have a header line.

For example, printSquareRoots(5) should print out:

```
Number    Square root
   1        1.0000
   2        1.4142
   3        1.7321
   4        2.0000
   5        2.2361
```

Hint: You can compute the square root of a number using Math.sqrt(...)

```
public void printSquareRoots( int max){
    UI.println ("Number   Square root");
    for (int num = 1; num<=max; num++){
        UI.printf ("   %d       %.4f\n", num, Math.sqrt(num));
    }









}
```

(c) **[6 marks] Methods with Parameters.**

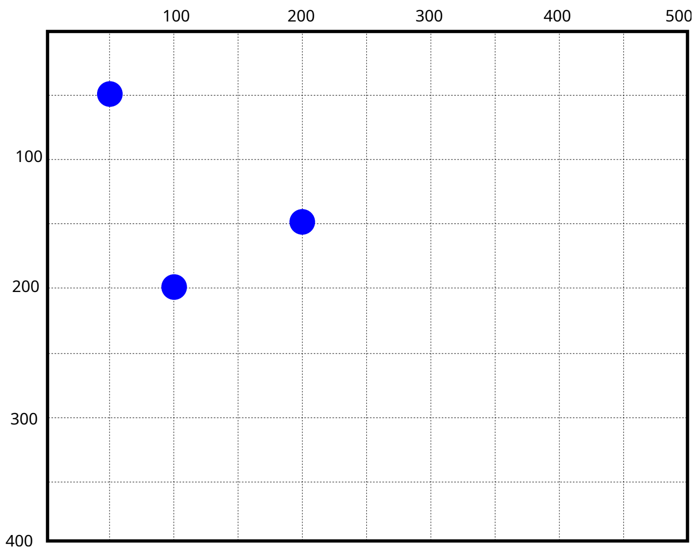
The drawShape(double x, double y) method below draws a 20x20 circle centered at the point (x,y).

The drawStuff() method calls drawShape(...) three times. Show what drawStuff() draws on the graphics window.

Hint: Look carefully at the arguments in the calls to drawShape(...).

```
public void drawStuff(){
    double x = 100;
    double y = 200;
    this.drawShape(x, y);
    x = x + 50;
    this.drawShape(y, x);
    y = y - x;
    this.drawShape(y, y);
}

/** Draw a 20x20 circle centered at (x,y) */
public void drawShape(double x, double y){
    double radius = 10;
    UI.fillOval(x-radius, y-radius, radius*2, radius*2);
}
```



(d) **[8 marks] Reading lines from files.**

Complete the following printLinesWithHash(...) method which will print out every line in a file that starts with a hash: “#”.

The parameter of the method is the name of the file to read from.

Hint: Use the Files.readAllLines(...) method to read the file into a List of String.

```
public void printLinesWithHash(String filename){
    try{
        List<String> lines = Files.readAllLines(Path.of(filename));
        for (String line : lines){
            if (line.startsWith("#")){
                UI.println(line);
            }
        }






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

(e) **[8 marks] Using ArrayLists**

Complete the following isBadWord(...) method which has two parameters: a testWord and an ArrayList of bad words.

isBadWord(...) checks the testWord against every word in the badWords list:

- If the testWord is the same as any words in badWords (ignoring case), then it returns true.
- If the testWord is not the same as any of the words in badWords, then it returns false.

```
public boolean isBadWord(String testWord, ArrayList<String> badWords){
    for(String word : badWords){
        if (testWord.equalsIgnoreCase(word)){
            return true;
        }
    }
    return false;








}
```

(f) **[8 marks] Arrays of Objects.**

Complete the following flipHand(Domino[ ] hand) method. flipHand(...) should flip all the Dominos in the hand by calling the flip() method on each Domino.

The parameter, hand, will contain an array of Dominos which may contain "spaces" – null values.

Hint: do not call the *flip*() method on a null value.

```
public void flipHand(Domino[ ] hand){
    for (int i=0; i<hand.length; i++){
        if (hand[i] != null){
            hand[i].flip();
        }
    }
}
```

(g) **[5 marks] 2D Arrays.**

The arrayPrinter() method operates on a 2D array of integers stored in a field called numbers.

```
public void arrayPrinter (){
    for ( int col=0; col < this.numbers[0].length ; col++){
        double total = 0;
        for ( int row =0; row < this.numbers.length; row++){
            total = total + this.numbers[row][col ];
        }
        UI. println ( total );
    }
}
```

What will arrayPrinter(...) print if the numbers field contains the following array:

```
    private double[ ][ ] numbers = new double[ ][ ]{{4, 2, 8, 7},
                                                    {6, 1, 5, 9},
                                                    {6, 10,2, 3}};
```

```
16.0
13.0
15.0
19.0
```

Question 2. Event-driven Programs **[10 marks]**

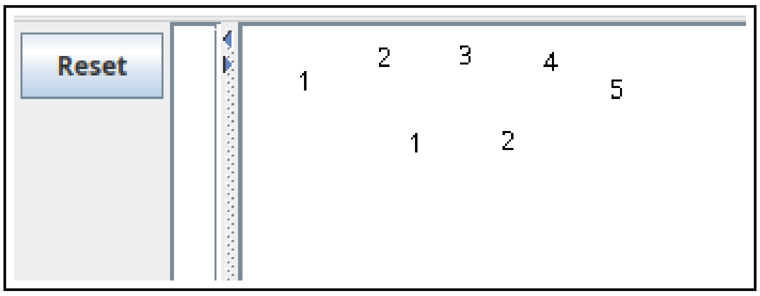
Complete the following program that will allow the user to place numbers on the graphics pane with the mouse.

The program must remember the current value of the number, which should start at 1.

Every time the user releases the mouse, the program should draw the current value of the number at that point, and then increase the number by 1.

The program should have a "Reset" button which will set the number back to 1.

For example, if the user released the mouse in five places, then clicked Reset, then released the mouse two more times, it might produce the following:



Hint: you can draw a number num at (x,y) using UI.drawString(""+num, x, y)

```
public class NumberDrawer {
    private int number = 1;

    public void setupGUI(){
        UI.addButton("Reset", this::reset );
        UI.setMouseListener(this :: doMouse);


    }
    public void reset (){
        this.number = 1;

    }
    public void doMouse (String action, double x, double y) {
        if (action.equals("released")){
            UI.drawString(""+this.number, x, y);
            this.number++;
        }




    }
    public static void main(String[] arguments){
        new NumberDrawer().setupGUI();
    }
}
```

Question 3. Using while. **[12 marks]**

Complete the wordGuess(String target) method below to let the user play a simple guessing game. The method repeatedly asks the user to guess a word. The parameter of the method is the target word.

- If the guess is right, the method prints how many guesses the user took, and finishes the game.
- If the guess not right, but it matches the start of the target, then the program reports that, and tells the user to try again.
- Otherwise, it tells the user to try again.

For example, if wordGuess(...) is called with the argument "cabbage", then the output might be the following (the user's guesses are shown in bold):

```
I've thought of a word.  You have to guess it.
guess:  cab
No, but the answer starts with cab.  Try again.
guess:  cabbie
No, try again
guess:  cabbage
You got it, in 3 guesses
```

```
public void wordGuess(String target){
    UI.println("I've thought of a word. You have to guess it.");
    int numGuesses = 0;
    while (true){
        String guess = UI.askString("guess: ");
        numGuesses++;
        if (guess.equals(target)){
            UI.println("You got it, in "+numGuesses+" guesses");
            break;
        }
        else if (target.startsWith(guess)){
            UI.println("No, but the target starts with "+guess+". Try again");
        }
        else {
            UI.println("No, try again");
        }
    }









}
```

Question 4. Reading from Files **[20 marks]**

A manufacturing company sells its products to retail companies in varying sized orders. It keeps information about the sales for its products in sales files, one file per month.

The products are identified by a SKU such as “N211422” or “WA75F5S6DRA”. The customer companies are identified by a code that is an abbreviation of the company name, such as “BRSC”, “TWH”, “NOLEM”.

Each line of a sales file starts with the SKU and is followed by the unit price then a sequence of company codes and number of units sold to that company that month. For example:

```
N211422  1435 TWH 30 NOLEM 21
HDC80E1  842  HVYNM 3
WW85T984 3985
WA75F5S6DRA  450  TWH 2 HTHCT 15 FRMRS 8 APPLS 5
```

The first line indicates that 30 units of N211422 (which cost $1,435 each) were sold to TWH and 21 units were sold to NOLEM.

The third line indicates that no units of WW85T984 (which cost $3,985) were sold that month.

Complete the following analyseSales(String fileName) method which is passed the name of a sales file. It should then read the file and return an ArrayList containing the SKUs of all products where the revenue (total units sold multiplied by the unit price) was more than $10,000. For the example above, both “N211422” and “WA75F5S6DRA” should be in the list returned, but “HDC80E1” and “WW85T984” should not.

Hints:

- Construct a new ArrayList for the answer.
- Use a new scanner for each line.
- Use a while loop to read and add up the number of sales for a product

(Question 4 continued on next page)

(Question 4 continued)

```
public ArrayList<String> analyseSales(String fileName){
  try{
      ArrayList<String> ans = new ArrayList<String>();
      ArrayList<String> lines = Files.readAllLines(Path.of(fileName));
      for (String line : lines){
          Scanner sc = new Scanner(line);
          String sku = sc.next();
          double price = sc.nextDouble();
          int sales = 0;
          while (sc.hasNext()){
              sc.next(); // customer
              sales = sales + sc.nextInt();
          }
          if (sales * price > 10000) {
              ans.add(sku);
          }
      }
      return ans;









  }catch(IOException e){UI.println("File failed "+ e); return null;}
}
```

Question 5. ArrayLists **[20 marks]**

This question involves lists of cities. A City object contains information about a city, including its name and location. City objects have two methods:

```
class City:
    public String getName()
        // returns the name of a City.

    public double distanceTo(City otherCity)
        // returns the distance (in kilometers) from this City to the otherCity.
```

(a) **[10 marks]** Complete the following removeFurthestCity(...) method which is passed a list of City objects and a new place. It should find the city in the list that is the furthest away from the new place, and should remove that city from the list.

You may assume that cities contains at least one city.

```
public void removeFurthestCity(ArrayList<City> cities, City newPlace){
    City furthestCity = cities.get(0);  // assumes at least one city in cities
    for (City city : cities){
        if (newPlace.distanceTo(city) > newPlace.distanceTo(furthestCity)){
            furthestCity = city;
        }
    }
    cities.remove(furthestCity);
\\OR (more efficient — doesn't recalculate distance to furthest city)
    City furthestCity = null; double maxDist = -1;
    for (City city : cities){
        double dist = newPlace.distanceTo(city);
        if (dist > maxDist){
            furthestCity = city; maxDist = dist;
        }
    }
    cities.remove(furthestCity); // works even if cities is empty
\\OR (most efficient version - doesn't have to search to remove)
    int furthestIndex = -1; double maxDist = -1;
    for (int i=0; i<cities.size(); i++){
        double dist = newPlace.distanceTo(cities.get(i));
        if (dist > maxDist){
            furthestIndex = i;    maxDist = dist;
        }
    }
    if (furthestIndex >=0){cities.remove(furthestIndex);}

}
```

(Question 5 continued on next page)

(Question 5 continued)

(b) **[10 marks]** Complete the following findAllCloseCities(...) method which is passed two lists of City objects, in the sources and targets parameters. It should return a new ArrayList of City objects containing all the cities in targets that are less than 50km away from some city in sources.

Note:

- No city in targets should be included in the answer more than once.
- Assume that all the cities in sources are different.

```
public ArrayList<City> findAllCloseCities (ArrayList<City> sources, ArrayList<City> targets){
    ArrayList<City> answer = new ArrayList<City>();
    for (City target : targets){
        for (City source : sources){
            if (target.distanceTo(source) <= 50){
                answer.add(target);
                break;
            }
        }
    }
    return answer;
//OR (less efficient )
    ArrayList<City> answer = new ArrayList<City>();
    for (City source : sources){
        for (City target : targets){
            if (source.distanceTo(target) <= 50 && ! answer.contains(target)){
                answer.add(target);
            }
        }
    }
    return answer;









}
```

Question 6. 2D Arrays **[10 marks]**

The following zoomInTopLeft(int[ ][ ] image) method is supposed to “zoom in” on the top left quarter of a greyscale image represented as a 2D array of *int*. zoomInTopLeft should move the pixel values around so that the top left quarter of the image is expanded to fill the whole image.

```
public void zoomInTopLeft(int[ ][ ] image){
    for ( int row=0; row<image.length/2; row++){
        for ( int col=0; col<image[row].length/2; col++){
            image[row*2][col] = image[row][col];
            image[row][col*2] = image[row][col];
            image[row*2][col*2] = image[row][col];
        }
    }
}
```

For example, zoomInTopLeft should change the image array on the left to be the image array on the right: (The shading of the squares corresponds to the pixel value in the square.)

20	60	34	48	91
80	30	74	28	71
44	84	94	68	51
18	38	58	48	31
81	61	41	21	11

$\Longrightarrow$

20	20	60	60	34
20	20	60	60	34
80	80	30	30	74
80	80	30	30	74
44	44	84	84	94

(a) **[5 marks]** The version of zoomInTopLeft above has errors.
In the box on the right, show all the values in the image after zoomInTopLeft is called on the image on the left:

20	60	34	48	91
80	30	74	28	71
44	84	94	68	51
18	38	58	48	31
81	61	41	21	11

$\Longrightarrow$

20	60	60	48	91
80	30	30	28	71
80	30	30	68	51
18	38	58	48	31
81	61	41	21	11

(Question 6 continued on next page)

(Question 6 continued)

(b) **[5 marks]** Write a correct version of zoomInTopLeft that would work on any size image.

```
public void zoomInTopLeft(int[ ][ ]  image){
    for (int row=image.length−1; row>=0; row−−){
        for (int col=image[row].length−1; col>=0; col−−){
            image[row][col] = image[row/2][col/2];
        }
    }
//OR
        int rows = image.length;
        int cols = image[0].length;
        int [][] temp = new int[rows+1][cols+1];
        for (int row=0; row<rows/2.0; row++){
            for (int col=0; col <cols/2.0; col++){
                temp[row*2  ][col*2  ] = image[row][col];
                temp[row*2+1][col*2  ] = image[row][col];
                temp[row*2  ][col*2+1] = image[row][col];
                temp[row*2+1][col*2+1] = image[row][col];
            }
        }
        for (int row=0; row<rows; row++){
            for (int col=0; col <cols; col++){
                image[row][col] = temp[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.