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

ID Number: . . . . . **Signature** . . . . . .

Model Solutions
COMP 112: Test 1

22 April, 2015

Instructions

- Time allowed: **50 minutes** .
- Answer **all** the questions. There are 50 marks in total.
- Write your answers in the boxes in this test paper and hand in all sheets.
- If you think some question is unclear, ask for clarification.
- Brief Java documentation is provided with the test
- This test contributes 15% of your final grade
  (But your mark will be boosted up to your exam mark if that is higher.)
- You may use paper translation dictionaries, and calculators without a full set of alphabet keys.
- You may write notes and working on this paper, but make sure your answers are clear.

Questions	**Marks**	
1. Understanding and Documenting Code	[10]	
2. File Reading and Writing	[10]	
3. Defining Classes	[10]	
4. Using Arrays	[8]	
5. Algorithms on Arrays	[12]	
	TOTAL:	

Question 1. Understanding and Documenting Code [10 marks]

The following eatMe method has unhelpful variable and method names.
(a) Write a documentation comment for eatMe that explains what it does.
(b) Give better names for the four variables, x,y,z and w.

```
public void eatMe(int w) {
        int x = 0;
        int y = Integer.MAX_VALUE;
        int z = 0;
        int v = 0;
        while (v < w) {
            v++;
            int drinkMe = UI.askInt(
                    String.format("How many students in course COMP%03d ? ",v));
            x += drinkMe;
            if (drinkMe<y){y = drinkMe;}
            if (drinkMe>z){z = drinkMe;}
        }
        if (v>0) {x = x/v;}
        UI.println("z students "+z);
        UI.println("y students "+y);
        UI.println("x students "+x);
    }
```

Documentation comment:
eatMe asks for the number of students in each of a specified number of courses and prints the maximum number or students in any courses, then the smallest number of students in any course and finally the average number of students in all the courses.

Better name for w: noOfCourses

Better name for x: studentAverage

Better name for y: courseMin

Better name for z: courseMax

Alternative:

eatMe is passed a number of courses as a parameter, then for each course asks the user for the number of students. Finally prints the max course size,min course size and average course size

Better name for w: courses

Better name for x: average

Better name for y: min

Better name for z: max

Question 2. File Reading and Writing [10 marks]

Complete the following fileSum method with a string parameter, **fname**, the name of an existing file. The method should write a new file named `"sum-"` appended to **fname**. The new file contains the sum of the doubles in the existing file.

For example fileSum(`"num.txt"`) when applied to file `"num.txt"` shown below should created the file `"sum-num.txt"` shown on the right.

`"num.txt"`

```
1.1 2.2 3.3 4.4 5.5
10 20 30 40 50

1000.001
```

`"sum-num.txt"`

```
1166.501
```

```
public void fileSum(String fname) {

    double total = 0;
    try{
        Scanner scan = new Scanner(new File(fname));
        while (scan.hasNextDouble()){
            total += scan.nextDouble();
        }
        scan.close();
        PrintWriter w = new PrintWriter("sum-"+fname);
        w.println(total);
        w.close();
    } catch (IOException e) {UI.println("IOException %n" +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.

Question 3. Defining Classes [10 marks]

A manufacturer needs to track the component parts they use. Complete the following Part class to define Part objects. Each Part should contain a name, a weight, and a unique ID that all start with "Pt". The name and weight should be specified by parameters of the constructor; the ID should be "Pt" concatenated with an integer greater than 0000 and every Part should have a different ID. The ID of the first three Parts should be "Pt0001","Pt0002" and "Pt0003".

The Part class should have both a constructor and a **toString()** method which returns a string containing the name, weight, and ID.

```
public class Part {
    private static int nextID = 1;
    private String name;
    private double weight;
    private String ID;

    public Part( String nm, double w){
        name = nm;
        weight = w;
        ID = String.format("Pt%04d", nextID++) ;
    }

    public String toString(){
        return name+" weight: "+weight + "("+ID+")";
    }









}
```

Question 4. Using Arrays [8 marks]

Complete the following sumPair method which accepts a parameter, an array of numbers, and returns a new array. The new array is built from parameter by calculating the sum of each pair of adjacent numbers in order.

For example, given the array `{5, 10, 20, 4, 8}`, sumPair should return the array `{15,30,24,12}`.

```
public static double[] sumPair(double[] array) {


    double[] out = new double[arr.length -1];
    for( int i = 1;i<arr.length; i++){
      out[i-1] = arr[i-1] + arr[i];
    }
    return out;









}
```

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.

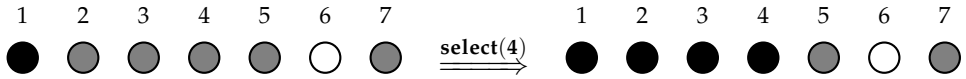
Question 5. Algorithms on arrays [12 marks]

Delta is a game that involves a row of tokens. Each token is either black, gray or white.

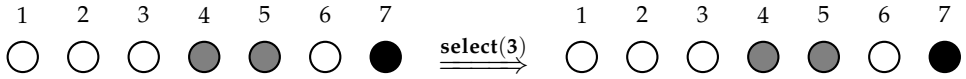
When you select a token **t** of color **c** it will change its color to a new color **d** if and only if not all the tokens to the left of **t** are of color **c**.

- If **t** changes color, to **d**, then **d** will be the color of the token **tl** where:
  1. **c** $\neq$ **d**
  2. **tl** is to the left of *t* and
  3. **tl** is the closest token not of color *c* and to the left of *t*.
- token **t** and all the tokens to the left of **t** up until **tl** will be changed to color **d**

For example, in the row of tokens below, selecting token 4: **t**= 4, **tl**= 1 will change tokens 4, 3 and 2.



But selecting token number 3 on this next board will not change any tokens, because all tokens to the left of token 3 are the same color as token 3:

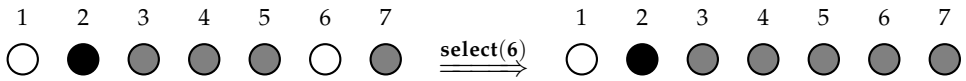


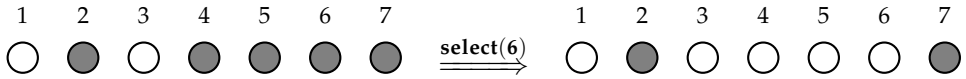
Complete the following select method which is part of a program for playing Delta. The integer parameter of select can be assumed to be a valid index into the array of tokens and is the selected cell. The method select should change the state of the tokens; it does not need to redraw the board.

The row of tiles is represented by the following tokens field:

```
private enum Token {black, gray, white}
private Token [ ] tokens = new Token[GAME_SIZE];
```

Two further examples:





(Question 5 continued on next page)

(Question 5 continued)

```
public static void select(int index) {
        int i = index;
        while (i >= 0 &&(tokens[i] == tokens[index]) ) {
            i--;
        }
        // i is now <0 or on a token of different colour
        if (i>=0) {
            Tkn t = tokens[i];
            i++;
            while(i<=index){
                tokens[i] = t;
                i++;
            }
        }
*****************
**   Alternative  **
*****************
    public static void recselect(int index,Tkn[] tokens) {
        UI.println("Starting recselect i ="+ index);
        Tkn fst = tokens[index];
        Tkn ret = rselect(index,tokens,fst);
    }
    private static Tkn rselect(int index,Tkn[] tokens,Tkn fst) {
        Tkn ret;
        if (index < 0) {return null;}
        else
        if (tokens[index].equals(fst)) {
            ret = rselect(index-1,tokens,fst);
        } else { return tokens[index];}
        if (ret != null) {tokens[index] = ret;}
        return ret;
    }



    }
```

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.

****