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

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

Model Solutions
COMP 112: Test 1

13 April, 2016

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.
- 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	[10]	
5. Algorithms on Arrays	[10]	
	TOTAL:	

Question 1. Understanding and Documenting Code [10 marks]

In a student monitoring system the following method has as a first parameter an array of student marks. The method is unhelpfully named `walrus` and has unhelpful variable and parameter names.
(a) Give better names for the method, the two parameters and the four variables.
(b) Write documentation that explains what the renamed `walrus` method does. The documentation can be in the form of English text or an equation or a mixture of both text and equations.

```
public boolean walrus(int [] carpenter, int cabbages) {
    int shoes = 0;
    int ships = 9999;
    int sealingWax;
    for(sealingWax = 0; sealingWax<carpenter.length; sealingWax++){
        if (ships > carpenter[sealingWax]) {
            ships = carpenter[sealingWax];
        }
        shoes += carpenter[sealingWax];
    }
    int kings = shoes/sealingWax;
    if (cabbages < kings - ships ) {
        return true;
    } else {
        return false;
    }
}
```

Better name for walrus: isAnyOneFallingBehind

Better name for carpenter: results

Better name for cabbages: allowedGap

Better name for sealingWax: i

Better name for shoes: sum

Better name for ships: min

Better name for kings: average

Documentation comment:
Students that scores less than (the average - allowed gap) are falling behind.

isAnyOneFallingBehind has parameters an array of test results and a single int input gap. it computes the average of the values in the array and reports if the gap between the average and the minimum test result is greater that the input gap.

Question 2. File Reading and Writing [10 marks]

Complete the following `fileHappy(String fname)` method. The string parameter **fname** can be assumed to be the name of an existing file. On each line of the file there is a persons name and an integer representing the reported happiness of the named person.

The method should write a new file named `"Average-"` appended by **fname**. The new file contains one line containing the average happiness of the people in the original file.

Example: `fileHappy("happy.txt")` below left should create `"Average-happy.txt"` below right.

`"happy.txt"`

```
david 2
fred 4
sue 3
John 1
ann 5
```

`"Average-happy.txt"`

```
3.0
```

```
public static void fileHappy(String fname) {


    double total = 0;
    int i = 0;
    try{
        Scanner scan = new Scanner( new File (fname));
        while (scan.hasNext()){
            i++;
            scan.next();
            total += scan.nextDouble();
        }
       PrintWriter w = new PrintWriter("Average-"+fname);
       w.println(total/i);
       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 has a warehouse full of wheels of different types and conditions. Each Wheel object should contain a type, condition, and a unique ID all three are to be held as Strings. The type and condition should be specified by parameters of the constructor and the ID should be "Wh" concatenated with a four digit integer greater than 0000. Thus the ID of the first three Wheels should be "Wh0001","Wh0002" and "Wh0003".

The Wheel class should have three methods (1) a constructor, (2) a **toString()** method which returns a string containing the ID, type and condition and (3) **equals(Object o)** method where two wheels are equal if and only if they are of the same condition and type.

```
public class Wheel {
    private static int  ids = 1;
    private String ID;
    private String type;
    private String condition;
    public Wheel(String t, String c)
    {   type = t;
        condition = c;
        ID = String.format("Wh%04d", ids++) ;
    }
    public String getType(){return type;}
    public String getCond(){return condition;}

    public boolean equals(Object object) {
        if(object instanceof Wheel &&
        ((Wheel)object).getType() == this.type &&
        ((Wheel)object).getCond() == this.condition) {
            return true;
        } else {
            return false;
        }
    }
    public String toString()
    {   return ID + " Type = "+type+
        "  condition = "+condition;
    }




}
```

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 4. Using Arrays [10 marks]

Complete the following `boostBehind(String name)` method which accepts a parameter, a name of a `RaceCar` and an array of `RaceCars`.

```
public class RaceCar
{  private String name;
    private int location = 0;  // the loction of the car on the race track
    public String getName() { return name;}
    public int getLocation() {return location ;}
    public RaceCar(String t) {  name = t;}
    public void move(int in){  location += in; }
}
```

Find the location on of the named `RaceCar`. Then move forward (increase the location by 2) each car that is behind the named `RaceCar`. Where "being behind a named car" means having location less than, the named car.

```
public void boostBehind(String name, RaceCar[] race){
        //find the location of the race car called name
        int location = -1;
        for(int i = 0; i<race.length ;  i++){
            if (race[i].getName().equals(name)) {
                location = race[i].getLocation();
            }
        }
        // move forward by two
        // all cars behind the race car called name
        for(int i = 0; i<race.length ;  i++){
            if (race[i].getLocation() < location) {
                race[i].move(2);
            }
        }
.
    }
```

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 [10 marks]

Flip is a game that involves a row of tokens. Each token is either black or white.

If the number tokens of same colour and adjacent and to the left of the selected token is equal to the number tokens of same colour and adjacent and to the right of the selected token then change the colour of all token adjacent to and with the same colour as the selected token.

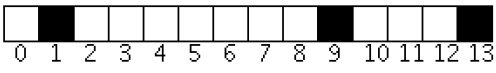
Examples: Starting with:

:

Select token 5. Number of tokens to the left of token 5 and with the same color = 0. Number of tokens to the right of token 5 and with the same color = 2. Hence do nothing.

:

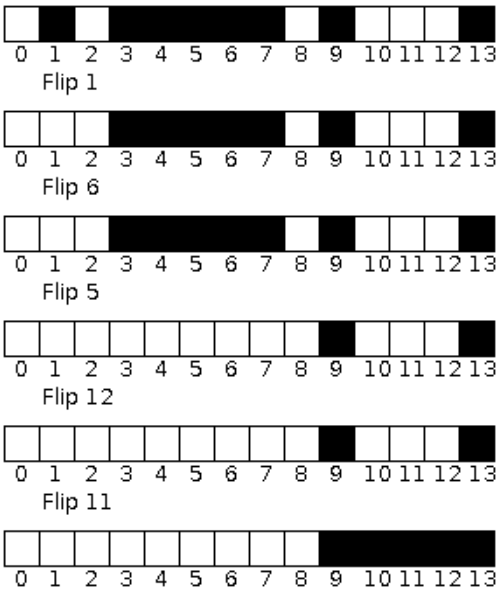
Select token 6. Number of tokens to the left of token 5 and with the same color = 1. Number of tokens to the right of token 5 and with the same color = 1. Hence flip tokens 5,6 and 7.



:

TO DO Complete the `Flip(boolean[] tokens, int index)` method, the first parameter is the array of tokens (true = black and false = white) and the second parameter is the index of the selected token.

Further example executions: Starting from the top row of tiles and applying **Flip** with selected token = 1 results in the second row of tiles, and so on down the rows.



:

(Question 5 continued on next page)

(Question 5 continued)

```
public void flip (boolean[] tokens, int index) {
        boolean hold = tokens[index];
        int i = index-1;
        int left = 0, right = 0;
        while (i >= 0 &&(tokens[i] == hold) ) {
            i--; left++;
        }
        i = index +1;
        while (i < tokens.length &&(tokens[i] == hold) ) {
            i++; right++;
        }
        // i is now <0 or on a token of the opposite colour
        if ( left == right) {
            i = index -left;
            while(i<=index+ right){
                tokens[i] = !tokens[i];
                i++;
            }
        }
.
}
```

****