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

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

Model Solutions
COMP 112: Test 1

15 April, 2014

Instructions

- Time allowed: **50 minutes** .

*shouldprobablyhavebeen*75*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	[6]	
2. Drawing with the UI class	[6]	
3. File Reading and Writing	[6]	
4. Defining Classes	[6]	
5. Using Arrays	[6]	
6. Saving and Loading State	[10]	
7. Algorithms on 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. Understanding and Documenting Code [6 marks]

The following readMe method has unhelpful variable and method names.
(a) Write a documentation comment for readMe that explains what it does.
(b) Give better names for the four variables.

```
public void readMe(int w) {
    int x = 0;
    int y = 0;
    int z = w;
    while (z > 0) {
        int sheep = UI.askInt("How many sheep? ");
        x += sheep;
        if (sheep>y){
            y = sheep;
        }
        z--;
    }
    if (w>0) {x = x/w;}
    UI.println("x sheep "+x);
    UI.println("y sheep "+y);
}
```

Documentation comment:
readMe asks for a specified number of numbers from the user
and prints the average of the numbers and the largest of the numbers

Better name for w: count

Better name for x: total

Better name for y: max

Better name for z: remaining

Alternative:

readMe is passed a number of flocks, asks the user for the number
of sheep in each flock, and prints the average and the largest flock size

Better name for w: flocks

Better name for x: totalSheep

Better name for y: maxFlockSize

Better name for z: remainingFlocks

Question 2. Drawing with the UI class [6 marks]

Complete the following drawCheckersRow method that will draw one row of an $n \times n$ checkers board on the UI graphics pane. The row should have n alternating black and white squares:

The parameters of drawCheckersRow should be the number of squares in the row and the position on the pane of first square in the row. The size of the squares is specified by the constant SIZE.

```
    private static final double SIZE = 30;

    public void drawCheckersRow(    int n, double left , double top){
        double x = left ;
        UI.setColor(Color.black);
        for (int i=0; i<n; i++){
            if (i%2 == 0) {
                UI. fillRect (x, top, SIZE, SIZE);
            }
            else {
                UI.drawRect(x, top, SIZE, SIZE);
            }
            x += SIZE;
        }
OR
    public void drawCheckersRow(int n, double left , double top){
        UI.setColor(Color.black);
        UI.drawRect(left , top, SIZE*n, SIZE);
        for (int i=0; i<n; i=i+2){
            UI. fillRect ( left +i*SIZE, top, SIZE, SIZE);
        }









    }
```

Question 3. File Reading and Writing [6 marks]

Complete the following lineNumber method whose parameter is a string that is the name of a file. It should write a new file, whose name is `"numbered-"` appended to the given file name, which contains the same text as the original file, except that each line should have a line number.

For example if the file `"sheep.txt"` contained the text on the left,

lineNumber(`"sheep.txt"`)

should created the file `"numbered-sheep.txt"` containing the text on the right:

```
Mary had a little lamb
Shrek had lots of wool
Bo Peep lost her sheep
```

```
1 Mary had a little lamb
2 Shrek had lots of wool
3 Bo Peep lost her sheep
```

```
public void lineNumber(String fname) {
    int num = 1;
    try{
        Scanner scan = new Scanner(new File(fname));
        PrintWriter w = new PrintWriter("numbered-"+fname);
        while (scan.hasNext()){
            String line = scan.nextLine();
            w.println(num + " "+ line);
            num++;
        }
        scan.close();
        w.close();
    } catch (IOException e) {UI.println("IOException \n" +e);}

}
```

Question 4. Defining Classes [6 marks]

Complete the following Employee class to define Employee objects. Each Employee should contain a name, an age, and an employee ID. The name and age should be specified by parameters of the constructor; the ID should be an integer greater than 3000 and every Employee should have a different ID. The ID of the first Employee should be 3001.

The Employee class should have a toString method which returns a string containing the name, age, and ID.

```
public class Employee {
    private static int nextID = 3001;
    private String name;
    private int age;
    private int ID;

    public Employee(String nm, int a){
        name = nm;
        age = a;
        ID = nextID++;
    }

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

```
}
```

Question 5. Using Arrays [6 marks]

Complete the following shrinkArray method which is passed an array of numbers, and should return a new array, half the length of the given array. Each value in the new array is the sum of two adjacent numbers in the given array.

For example, given the array `{5, 10, 20, 4, 8, 2}`, shrinkArray should return the array `{15, 24, 10 }`.

shrinkArray may assume that the array has an even number of values.

```
public double[ ] shrink(double[ ] array){
    if (array.length%2 != 0) { return null; }    // ignore arrays of odd length.
    double [ ] ans = new double[array.length/2];
    for (int i=0; i<array.length; i=i+2){
        ans[i/2] = array[i] + array[i+1];
    }
    return ans;









}
```

Question 6. Saving and Loading State [10 marks]

Complete the following saveGame and loadGame methods in the CardGame class that will save the state of the game to a file (asking the user where to save it), and reload a game state from a file. The loadGame method may assume that the file was written by saveGame. It should not cause any errors other than IOExceptions

The CardGame class implements a game involving a sequence of cards laid out on the table, a collection of counters (integers) and a marker token.

At each point in the game, the player places the marker token on one of the cards. The player can also rearrange the cards and buy more cards using the counters.

The CardGame program contains three fields to represent the cards, the counters, and the token:

```
private ArrayList<Card> cards = new ArrayList<Card>();
private ArrayList<Integer> counters = new ArrayList<Integer>();
private int token;
```

The program contains a Card class which has a toString method that will return a String describing a card in some format. All you know about the format is that it does not contain any end-of-line characters. The Card class also has a constructor that takes a single argument — a String of the same form that the toString method generates.

```
public void saveGame(){
    try{
        PrintStream ps = new PrintStream(new File(UIFileChooser.save()));
        ps.println(token);
        if (cards==null){      // must check for null
            ps.println(0);     // (a) to be able to print 0, and
        }                      // (b) to not have error in the loop.
        else {
            ps.println(cards.size());    // must print size, can't use sentinal
                                         // because you don't know card format
            for (Card card : cards){ps.println(card);}
        }
        if (counters!=null){               // must check for null
            for (int counter : counters){ps.println(counter);}
        }
        ps.close();
    }
    catch(IOException e){UI.println("Fail: " + e);}



}
```

(Question 6 continued on next page)

(Question 6 continued)

```
public void loadGame(){
    try{
        Scanner sc = new Scanner(new File(UIFileChooser.open()));
        token = sc.nextInt();
        cards = new ArrayList<Card>();
        int numCards = sc.nextInt();
        sc.nextLine();                    // tricky, but required to clear line
        for (int i=0; i<numCards; i++){
            cards.add(new Card(sc.nextLine()));   // nextLine required, to pass
        }                                         // String to Card constructor
        counters = new ArrayList<Integer>();
        while (sc.hasNext()){
            counters.add(sc.nextInt());
        }
        sc.close();
    }
    catch(IOException e){UI.println("Fail: " + e);}




}
```

There are various different ways of doing this; the way above is just one example. The critical thing is that the data must be saved in a way that the load method knows whether it is reading a card or a counter.

Printing how many items are in the list before the list is a good way.
Printing some kind of sentinel value at the end of the list is another, but that is harder to deal with when loading - you have to check for the sentinel without "using up" the value in case it isn't a sentinel and is part of the next card or counter. Not knowing the format of the Cards made this tricky - we can't tell what sentinel value will be different from a card. Saving and Loading the counters before the cards would make the sentinel approach possible, because we know the counters are saved as integers, so we can make a sentinel that is not an integer.

Another approach is to put "card:" or "counter:" as the first token on each line.

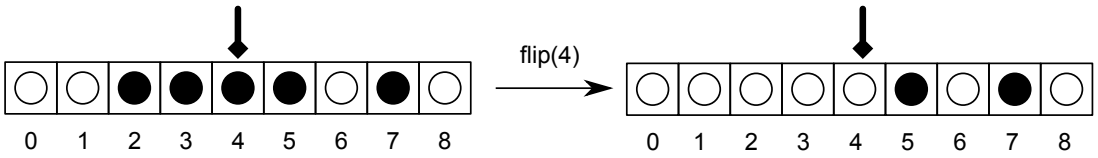
Question 7. Algorithms on arrays [10 marks]

Flip is a game that involves a row of tokens. Each token is black on one side and white on the other.

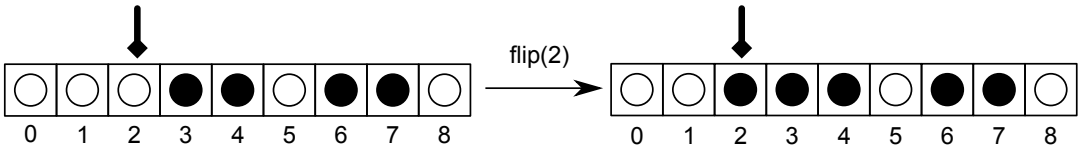
When you "flip" a token t of color c:

- It will be turned over and show the opposite colour.
- If there is at least one token somewhere to the left of t whose colour is the opposite of c, then all the tokens of color c to left of t up until the closest token of the opposite colour will also be turned over.

For example, in the board below, flipping token 4 will also flip tokens 3 and 2.



But flipping token number 2 on this next board will not flip other tokens, because there is no black token to the left of token 2:

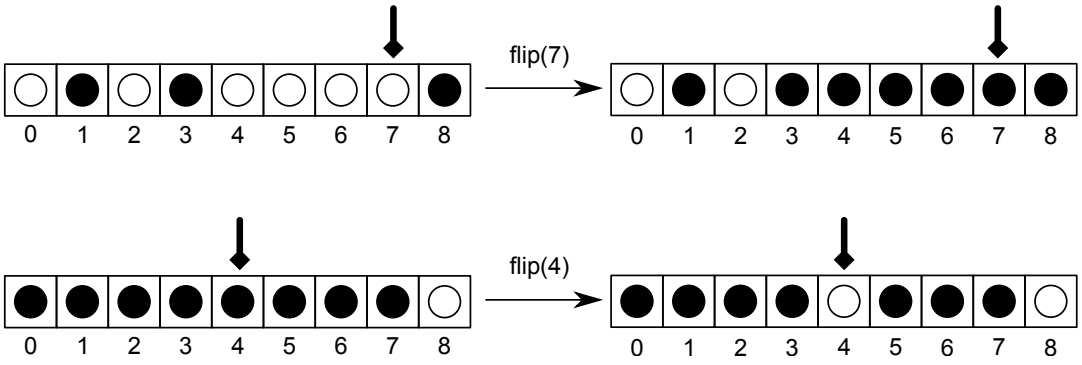


Complete the following flip method which is part of a program for playing Flip. The parameter of flip is the index of the cell to flip over. flip should change the state of the tokens; it does not need to redraw the board.

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

```
private boolean [ ] tokens = new boolean[GAME_SIZE];
```

Two further examples:



(Question 7 continued on next page)

(Question 7 continued)

```
public void flip (int index) {

    int i = index;
    while (i >= 0 &&(tokens[i] == tokens[index]) ) {
        i--;
    }
    // i is now <0 or on a token of the opposite colour
    if (i>=0) {
        i++;
        while(i<index){
            tokens[i] = !tokens[i];
            i++;
        }
    }
    tokens[index] = !tokens[index];
}









}
```

****