TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2017

TRIMESTER 1

COMP 102
INTRODUCTION TO COMPUTER PROGRAM DESIGN

Time Allowed: TWO HOURS ******** WITH SOLUTIONS **********

CLOSED BOOK

Permitted materials: Silent non-programmable calculators or silent programmable calculators with their memories cleared are permitted in this examination.

Printed foreign language–English dictionaries are permitted.

No other material is permitted.

Instructions: Attempt ALL Questions.
The exam will be marked out of 120 marks.
Brief Java Documentation will be provided with the exam script
Answer in the appropriate boxes if possible — if you write your answer elsewhere, make it clear where your answer can be found.
There are spare pages for your working and your answers in this exam, but you may ask for additional paper if you need it.

October 13, 2017

Questions:

• Understanding Java	[33]
• GUI programs	[8]
• Files	[12]
• ArrayLists of Objects	[26]
• 2D Arrays	[20]
• Defining Interfaces and Classes	[21]

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 Java **[33 marks]**

(a) **[6 marks] Loops and calling methods.**

What will the following printStuff method print out?

```
public void printStuff (){
    UI. println ("printing stuff");
    int num = 12;
    while (num > 0){
        printNum(num, "word");
        num = num - 3;
    }
    UI. println ("done="+ num);
}

public void printNum(int num, String str){
    num = num + 1;
    UI. println (str +"-" + num);
}
```

num: $\boxed{}$

num: $\boxed{}$

str: $\boxed{}$

```
printing stuff
word-13
word-10
word-7
word-4
done=0
```

(Question 1 continued on next page)

(Question 1 continued)

Consider the following Question class, specifying objects with one field and three methods.

```
public class Question {

    // field
    private String text;

    // constructor
    public Question(String s){
        this.text = s;
    }

    // methods

    public void check(){
        if (this.text.contains("?")) {
            UI.println("Yes");
        }
        else {
            UI.println("No");
        }
    }

    public void combine(String word){
        this.text = this.text + word;
        UI.println(this.text);
    }

    public String compare(String word){
        if (this.text.contains(word)) {
            return(word);
        }
        else return(this.text);
    }

}
```

(Question 1 continued on next page)

(b) **[7 marks] Calling methods on objects.**

Given the Question class on the facing page, what will the following runQuestions method print out?

```
public void runQuestions(){
    Question qnA  = new Question("Why program?");
    Question qnB = new Question("How many");

    qnA.check();

    qnA.combine("Java");

    UI. println (qnA.compare("How") );

    qnB.check();

    qnB.combine( qnA.compare("Why") );
}
```

qnA:

qnB:

```
Write your answers here
Yes
Why program?Java
Why program?Java
No
How manyWhy
```

(Question 1 continued on next page)

(c) **[6 marks] Arrays.**

The following arrayTest method creates a pastas array, then modifies it, then prints out each value in the array. What will be printed?

```
public void arrayTest(){
    String[ ] pastas = new String[ ]{"linguine", "elbow", "fusilli", "penne", "vermicelli"};

    int n = 0;
    pastas[n] = pastas[n+3];

    for(int i=0; i < pastas.length; i++){
        if(pastas[i].endsWith("i")){
            pastas[i] = pastas[i/2];
        }
    }

    // print out the array
    for(String p : pastas){
        UI.println(p);
    }
}
```

```
penne
elbow
elbow
penne
elbow
```

(Question 1 continued on next page)

(Question 1 continued)

(d) **[6 marks] Using 2D Arrays.**

The following array2DTest method creates a 4×3 2D array, then modifies it, then prints it out. What will be printed?

```
public void array2DTest(){
    int[ ][ ] numbers = new int[ ][ ]{{15, 2, 8},
                                      {5, 11, 9},
                                      {20, 6, 12},
                                      {4, 25, 10}};

    for (int k = 0; k < 3; k++) {
        numbers[k][k] = numbers[k][k] + numbers[k+1][k];
    }

    // Print out the array, one row per line
    for (int row =0; row < numbers.length; row++){
        for (int col =0; col < numbers[row].length; col++){
            UI.print(numbers[row][col] + " ");
        }
        UI.println();
    }
}
```

```
Numbers:
20 2 8
5 17 9
20 6 22
4 25 10
```

(Question 1 continued on next page)

(Question 1 continued)

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

What will the following arrayListTest method print out? (Write your answer on the facing page.)

```
public void arrayListTest (){
    ArrayList <String> predators = new ArrayList<String>();

    predators.add("cat");
    predators.add("dog");
    predators.add("weasel");
    predators.add("stoat");
    predators.add("possum");

    UI.println ("------A----------");

    UI.println (predators.get(predators.size()−1));
    UI.println (predators.contains("fox"));
    UI.println (predators.indexOf("weasel"));

    UI.println ("------B----------");

    predators.add(3, "rat");
    for(String s : predators){
        UI.println (s);
    }

    UI.println ("------C----------");
    predators.remove(1);
    predators.set (3, "shark");
    for(String s : predators){
        UI.println (s);
    }
}
```

(Question 1 continued on next page)

(Question 1 continued)

```
------A----------
possum
false
2
```

```
------B----------
cat
dog
weasel
rat
stoat
possum
```

```
------C----------
cat
weasel
rat
shark
possum
```

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 2. GUI programs **[8 marks]**

Complete the following WordPlacer program that allows the user to place words on the graphics pane. The program has a textfield where the user can enter a word that will be stored in a field. When the user releases the mouse on the graphics pane, the program will draw the current word at that place.

You need to

- declare a field,
- complete the constructor to add a text field
- complete the setWord method
- complete the doMouse method

```
public class WordPlacer{

    private String word;



    public WordPlacer(){
        UI.setMouseListener(this::doMouse);
        UI.addTextField("Word", this::setWord);



    }

    public void setWord(String v){
        this.word = v;



    }

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





    }
}
```

Question 3. Files **[12 marks]**

This question concerns methods for processing files containing word-count data from books. Each line of a data file contains a word followed by an integer that is the number of times the word occurred in a book.

The following is the first few lines of the "TheCatInTheHat" word-count file:

```
the 96
sun 2
did 10
not 41
shine 1
```

(a) **[6 marks]** Complete the following highFrequencyWords method that prints out every word in a data file that has a count of 10 or greater.

For the section of the example file above, the method would print out "the", "did", and "not".

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

```
public void highFrequencyWords(String filename){
    try{
        Scanner sc = new Scanner(new File(filename));
        while(sc.hasNext()){
            String word = sc.next();
            int count = sc.nextInt();
            if (count >= 10) {
                UI.println(word);
            }
        }
        sc.close();



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

(Question 3 continued on next page)

(Question 3 continued)

(b) **[6 marks]** Complete the following writeWordFile method that will read a data file, and write a new file containing just the words in the file that start with a given letter.

The parameters are the name of the data file and a string containing one letter. The name of the new file should be the name of the data file with the letter appended to it.

For example, writeWordFile("TheCatInTheHat", "t") should write a new file called "TheCatInTheHat-t", which contains all the words starting with "t".

```
public void writeWordFile(String filename, String letter){
    try{
        Scanner sc = new Scanner(new File(filename));
        PrintStream out = new PrintStream(new File(filename+"-"+letter));
        while (sc.hasNext()){
            String word = sc.next();
            sc.nextInt();
            if (word.startsWith(letter)){
                out.println(word);
            }
        }
        sc.close();
        out.close();









    } catch(Exception e){UI.println("file error");}
}
```

Question 4. ArrayLists of Objects **[26 marks]**

This question concerns a program for analysing data about flooding events. The program has two classes: a Flood class and a FloodAnalyser class.

The Flood class below defines Flood objects, which store details about Flood events: date, place, duration of flood, and maximum depth of flooding.

```java
public class Flood {
    private String date;        // year/month/day
    private String place;       // town/city name (no spaces)
    private int duration;       // number of days
    private double maxDepth; // depth of flooding in meters

    /** Constructs a Flood object from a description string.
     *  desc must be of the form: date place duration depth
     *  eg "2017/4/6 Edgecumbe 6 8.33"
     */
    public Flood(String desc) {
        Scanner strSc = new Scanner(desc);
        this.date = strSc.next();
        this.place = strSc.next();
        this.duration = strSc.nextInt();
        this.maxDepth = strSc.nextDouble();
    }

    public String getPlace() {
        return this.place;
    }

    public int getDuration() {
        return this.duration;
    }

    public double getDepth() {
        return this.maxDepth;
    }

    /** Returns true if this Flood happened after the other Flood */
    public boolean isAfter(Flood other) {
        return (this.date.compareTo(other.date) > 0) ;
    }

    /** Return brief description string such as "Edgecumbe on 2017/4/6, 6 days, 8.33m" */
    public String briefDescn(){
        return (this.place +" on "+ this.date + ", " +
                this.duration + " days, " + this.maxDepth + "m");
    }
}
```

(Question 4 continued on next page)

The FloodAnalyser class defines a field to store the list of Flood objects:

```
    private ArrayList<Flood> floods = new ArrayList<Flood>();
```

(a) **[7 marks]** Complete the following loadData method that will load flood data from a file into the floods field. Assume that the file contains one line for each flood, containing the date, place, duration, and depth of the flood in the format required by the Flood constructor.

Here is an example file:

```
2016/3/6 Edgecumbe 5 7.1
2015/3/12 Wellington 1 1.4
2017/2/9 Edgecumbe 3 2.4
2017/4/6 Edgecumbe 6 8.33
2017/1/30 Christchurch 10 4.0
2015/10/20 Christchurch 3 1.2
```

loadData has one parameter, which is the name of the file to read from.

```
public void loadData(String filename){
    try {
        Scanner sc = new Scanner(new File(filename));
        this.floods.clear();     //or   this.floods = new ArrayList<Flood>();
        while (sc.hasNext()){
            this.floods.add(new Flood(sc.nextLine()));
        }
        sc.close();









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

(Question 4 continued on next page)

(b) **[7 marks]** Complete the following selectFloods method that will print out a brief description of each flood with a duration equal to the parameter and a depth of at least 1.0 meters, followed with a count of how many such floods there were.

For example, on the file given in part (a), selectFloods(3) should print out:

```
Edgecumbe on 2017/2/9, 3 days, 2.4m
Christchurch on 2015/10/20, 3 days 1.2m
Number of floods = 2
```

```
public void selectFloods(int dur){
    int count = 0;
    for (Flood fl : this.floods) {
        if ((fl.getDuration() == dur) && (fl.getDepth() >= 1.0)){
            UI.println(fl.briefDescn());
            count++;
        }
    }
    UI.println("Number of floods = " + count);
}
```

(Question 4 continued on next page)

(Question 4 continued)

(c) **[12 marks]** Suppose all floods have different depths, and we are interested in the place where the flood with the greatest depth occured.

Complete the following floodsAtWorstPlace method that will find this place, and then construct and return a new ArrayList containing all the floods that were at the same place.

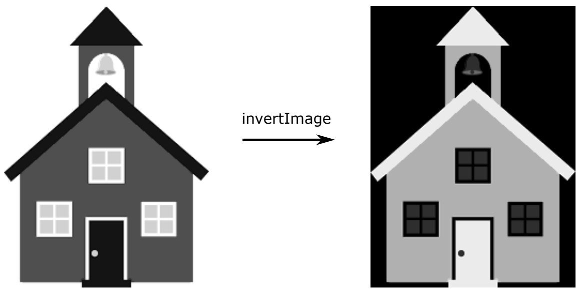
For example, if the worst flood was at Edgecome, floodsAtWorstPlace() should return a list of all the floods at Edgecome.

```
public ArrayList<Flood> floodsAtWorstPlace(){

    // Find worst place
    double max = 0;
    String maxPlace = "";
    for (Flood fl : this.floods) {
        if (fl.getDepth() > max){
            max = fl.getDepth();
            maxPlace = fl.getPlace();
        }
    }









    // create and return list of floods at worst place
    ArrayList<Flood> answer = new ArrayList<Flood>();
    for (Flood fl : this.floods) {
        if (fl.getPlace().equals(maxPlace)){
            answer.add(fl);
        }
    }
    return answer;








}
```

Question 5. 2D Arrays **[20 marks]**

This question involves completing two methods that modify grey-scale images represented by a 2D array of integers between 0 and 255, as in the ImageProcessor program for Assignment 8.

(a) **[8 marks]** Complete the following invertImage method that will invert an image, as in the following images:



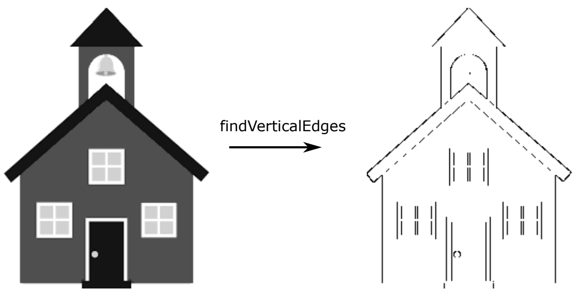
The parameter of invertImage is a 2D array of integers (0...255).
A pixel value v can be inverted by replacing it with $255 - v$.

```
public void invertImage(int[ ][ ] image){
    for(int row = 0; row < image.length; row++){
        for(int col = 0; col < image[row].length; col++){
            image[row][col] = 255 - image[row][col];
        }
    }
}
```

(Question 5 continued on next page)

(Question 5 continued)

(b) **[12 marks]** Complete the following findVerticalEdges method that will identify all the sharp vertical edges in an image. For example:



A pixel is a vertical edge if it is at least 50 units brighter or darker than the pixel to its right.

findVerticalEdges is passed a 2D array of integers, and converts every vertical edge pixel to black (0), and every other pixel to white (255).

```
public void findVerticalEdges (int[ ][ ] image){
    int rows = image.length;
    int cols = image[0].length;
    for(int row = 0; row < rows; row++){
        for(int col = 0; col < cols-1; col++){
            if (Math.abs(image[row][col] - image[row][col+1])>=50){
                image[row][col] = 0;
            }
            else{
                image[row][col] = 255;
            }
        }
        image[row][cols-1] = 255;
    }
}
```

Question 6. Defining Interfaces and Classes **[21 marks]**

For this question, you are to complete an interface class (Culture) and a class (WetSubstrate) that implements the interface.

Suppose you are writing a program for a scientist running a series of biology experiments that involve measuring the growth of bacteria cultures. There are several different kinds of cultures, including high-temperature cultures, wet-substrate cultures, and agar cultures. The program must keep certain information about every culture, including

- an ID (eg "WS-9/22")
- the name of the bacteria,
- a list of the measured values (doubles), and
- the date that the most recent measurement was recorded. There is a Date class to represent this.

The different kinds of cultures have different intervals between making measurements and different instructions for making the measurements.

Part of the ExperimentManager program is shown below, including a field where the culture information is stored, and two methods. Culture is an interface class.

```
public class ExperimentManager {
    ArrayList <Culture> cultures = new ArrayList<Culture>();

    /** Prints instructions for each culture that is still live,
     * including the date that the next measurement should be made.
     */
    public void printInstructions (){
        for (Culture c : this.cultures) {
            if (c.stillLive ()){
                UI.println ("----------");
                UI.println ("Culture "+c.getID()+": Make measurement at " + c.nextMeasureDate());
                UI.println (c.getInstructions ());
            }
        }
    }

    /** Adds a measured value to the list of measured values for the specified
     *  culture, and also records the date of the latest measurement.
     */
    public void recordNewMeasurement(String ID, Date date, double value){
        for (Culture c : this.cultures) {
            if(c.getID().equals(ID)){
                c.addMeasurement(value, date);
                return;
            }
        }
    }
}
```

(Question 6 continued on next page)

(Question 6 continued)

(a) **[6 marks]** Define the Culture interface, so that the code above would compile correctly.

```
public interface Culture{
    public String getID();
    public boolean stillLive ();
    public Date nextMeasureDate();
    public String getInstructions ();
    public void addMeasurement(double value, Date date);









}
```

(Question 6 continued on next page)

(Question 6 continued)

(b) **[15 marks]** Complete the following WetSubstrate class to represent wet-substrate cultures. Measurements on wet-substrate cultures should be made a fixed number of days apart.

This class should implement the Culture interface and work correctly with the code given on the previous pages.

You will need to define the appropriate fields, a constructor and all the necessary methods.

- The constructor should have parameters for the ID, bacteria name, number of days between measurements, the date and value of the first measurement.
- The instructions for a WetSubstrate are very simple:
  `"Measure in 95% humidity atmosphere."`
- Wet-substrate cultures remain alive for ever.

You may assume that Date objects have a method
  public *Date* getLaterDate(*int* days)
that will return a new Date the specified number of days later.

```
public class WetSubstrate  implements Culture {

    private String ID;
    private String bacteria;
    private ArrayList<Double> measurements = new ArrayList<Double>();
    private int daysBetweenMeasurements;
    private Date latest;

    public WetSubstrate(String ID, String bac, int days, Date date, double value){
        this.ID = ID;
        this.bacteria = bac;
        this.daysBetweenMeasurements = days;
        this.latest = date;
        this.measurements.add(value);
    }

    public String getID(){
        return this.ID;
    }








// (WetSubstrate class continued on next page)
```

(Question 6 continued on next page)

(Question 6 continued)

```
// (WetSubstrate class continued)

    public boolean stillLive (){
        return true;
    }
    public Date nextMeasureDate(){
        if (this. latest !=null){
            return this. latest .getLaterDate(this.daysBetweenMeasurements);
        }
        else { return null; }
    }

    public String getInstructions (){
        return "Measure in 95\% humidity atmosphere";
    }

    public void addMeasurement(double value, Date date){
        this.measurements.add(value);
        this. latest =date;
    }
















}
```

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