Student ID: .....

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2014

TRIMESTER 1

COMP 102

INTRODUCTION TO COMPUTER PROGRAM DESIGN

Time Allowed: THREE HOURS ******** WITH SOLUTIONS **********

Instructions: Closed Book
Attempt ALL Questions.

The exam will be marked out of 180 marks.

Silent non-programmable calculators or silent programmable calculators with their memories cleared are permitted.

Printed foreign language dictionaries are permitted.

Java Documentation will be provided with the exam script

No other material is permitted.

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.

Questions

	Marks
1. Understanding Java	[27]
2. Writing Java	[25]
3. Event driven input	[18]
4. Defining a Class	[18]
5. ArrayLists of Objects	[23]
6. Files	[12]
7. More ArrayLists	[26]
8. 2D Arrays	[17]
9. Debugging loops	[14]

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 [27 marks]

(a) [5 marks] What will the following printIt method print out?

```
public void printIt (){
    int y = 100;
    while (y > 8){
        UI.println ("y " + y);
        y = y/2;
    }
    UI.println ("finally "+y);
}
```

y:

```
y 100
y 50
y 25
y 12
finally 6
```

(Question 1 continued on next page)

(Question 1 continued)

(b) [6 marks] The testIt method below has one parameter and performs three tests on the parameter, printing out which tests are passed.
Note that there are no **else**'s.

```
public void testIt ( int x){

    if ( x < 8 || x >= 15 ) {
        UI.print ("A ");
    }

    if ( x < 7 && x != 4 ) {
        UI.print ("B ");
    }

    if ( ! (x == 4 || x == 10) ) {
        UI.print ("C ");
    }

    UI.println ();
}
```

What would the following calls to testIt print out?

testIt(4); $\Longrightarrow$ A

testIt(6); $\Longrightarrow$ A B C

testIt(8); $\Longrightarrow$ C

(Question 1 continued on next page)

(Question 1 continued)

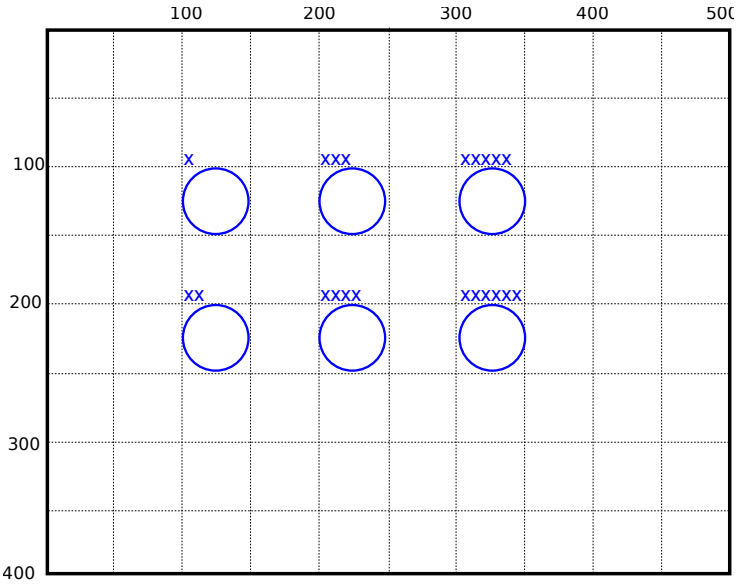
(c) [8 marks] On the grid below, show what will be drawn if the following board method is called with the arguments 3 and 4:

```
board(3, 4);
```

```
public void board (int rows, int cols){
    String title = "x";
    for (int col = 1; col<cols; col++){
        for (int row = 1; row < rows; row++){
            double x = col*100;
            double y = row*100;
            UI.drawOval(x, y, 50, 50);
            UI.drawString( title , x, y);
             title = title +"x";
        }
    }
}
```

title: [ ]

row: [ ] col: [ ]



(Question 1 continued on next page)

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 continued)

(d) [8 marks] Suppose the file `numbers.txt` contains the text:

```
9 80 before
43 after 38 48
6 75 same as 13
equal to 82
60 27
```

What will the following processNumbers method print out?

```
public void processNumbers(){
    try{
        Scanner scan = new Scanner (new File("numbers.txt"));
        String last = "";
        while ( scan.hasNext() ){
            if (scan.hasNextInt()){
                int num = scan.nextInt();
                UI. println ("num " + num);
            }
            else {
                last = scan.nextLine();
            }
        }
        scan.close();
        UI. println ("last " + last);
    }
    catch(IOException e){UI.println("File reading failed");}
}
```

num:

last:

```
num 9
num 80
num 43
num 6
num 75
num 60
num 27
last equal to 82
```

Question 2. Writing Java [25 marks]

(a) [5 marks] Complete the following average method which is passed two doubles and returns their average. You need to complete the header of average, as well as its body.

```
public double average( double x, double y ){
    return (x + y) / 2;

}
```

(b) [6 marks] Complete the following repeated method to construct and return a string consisting of repetitions of a short string. It has two parameters: a string that should be repeated, and an integer specifying the number of repetitions.

For example, repeated("Hi", 4) should return `"HiHiHiHi"`.

It should use a loop to add the string to answer repeatedly.

```
public String repeated (String base, int repetitions ){
    String answer = "";
    for ( int i = 0; i<repetitions; i++){
        answer = answer + base;
    }
    return answer;

}
```

(Question 2 continued on next page)

(Question 2 continued)

(c) [7 marks] Complete the following printArray method which is passed an array of Strings, and should print to the UI text pane all the strings in order, one string per line. If any element of the array is null, printArray should not print it.

```
public void printArray(String [ ] names){
    for (int i=0; i<names.length; i++){
        if (names[i] != null){
            UI.println(names[i]);
        }
    }
or
    for (String name : names){
        if (name != null){
            UI.println(name);
        }
    }
}
```

(d) [7 marks] Complete the following incrementScores method which is passed an array of integers, and increases every value in the array by 1, unless it is already greater than 10.

For example, if passed the array:

8	20	10	15	5
0	1	2	3	4

incrementScores should change the contents of the array to

9	20	11	15	6
0	1	2	3	4

```
public void incrementScores(int [ ] data){
    for(int i=0; i<data.length; i++){
        if( data[i] <= 10 ){
            data[i] = data[i]+1;
        }
    }



}
```

Question 3. Event Driven Input [18 marks]

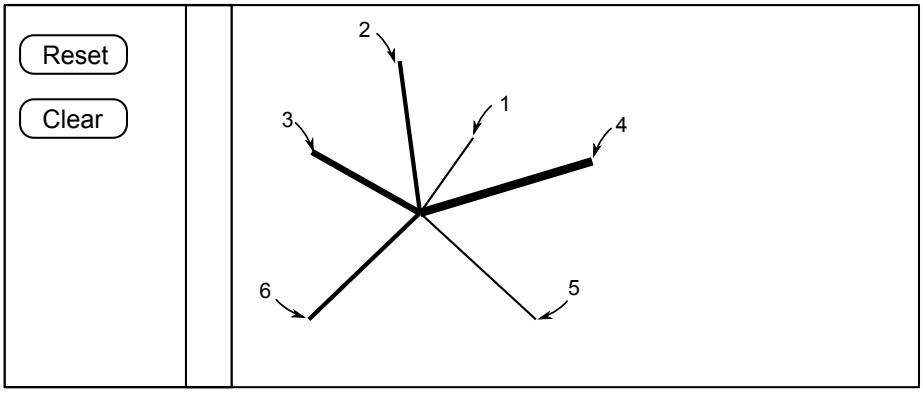
Complete the Star program on the facing page so that it allows the user to draw "stars" made out of lines of increasing line width. The program should have buttons to reset the line width and to clear the graphics pane. It should also listen to the mouse.

When the user releases the mouse in the graphics pane, the program should draw a line from the mouse position to the point (250,250), and then increase the line width by 1 unit.

The line width should initially be 1, and every time the user clicks the Reset button, the program should reset the line width to 1.

The diagram below shows what the program should do if the user

- clicked the mouse at the positions labeled 1, 2, 3, 4
- clicked the Reset button,
- clicked the mouse at the positions labeled 5 and 6



(Question 3 continued)

```
public class StarMaker implements UIButtonListener, UIMouseListener{
    // Fields
    private double width = 1;


    public StarMaker(){
        UI.setMouseListener(this);
        UI.addButton("Reset", this);
        UI.addButton("Clear", this);
        UI.setLineWidth(1);              // not necessary, since this is the default.

    }
    public void buttonPerformed(String button){
        if (button.equals("Reset") ){
            this.width = 1;
            UI.setLineWidth(1);
        }
        else if (button.equals("Clear") ){
            UI.clearGraphics();
        }



    }
    public void mousePerformed(String action, double x, double y) {
        if (action.equals("released")){
            UI.drawLine(250, 250, x, y);
            this.width = this.width + 1;
            UI.setLineWidth(this.width);
        }









    }
    public static void main(String[] arguments){
        new StarMaker();
    }
}
```

Question 4. Defining a Class [18 marks]

Complete the SoftwareRelease class on the facing page which stores information about a release of a software application.

A SoftwareRelease object should contain four fields:

- application, which contains the name of the application (*e.g.* Windows)
- version, which contains the main version number (*e.g.* 8.1)
- minor, which contains the minor release number (*e.g.* 42)
- build, which contains the number of the latest build number of the software (*e.g.* 349). The initial value of build is always 1.

SoftwareRelease should have a constructor that takes a name and version number and stores them in the application and version fields.

SoftwareRelease should have four methods:

- setMinor, which is passed the new minor release number, and sets the minor field to contain this value.
- rebuild, which is called every time the application is rebuilt. It should add one to the number in the build field.
- toString, which returns a string containing the full release identifier. For example, Windows version 8.1, with a release number 42, after 349 rebuilds would have a full release identifier of "Windows 8.1.42:349"
- nextRelease, which returns a new SoftwareRelease object with the same application and version, but the next larger minor number (eg, Windows 8.1.42 would go to 8.1.43).

The headers of the constructor and one of the methods are given.

(Question 4 continued on next page)

(Question 4 continued)

```
public class SoftwareRelease{
    // fields
    private String application;
    private double version;
    private int minor;
    private int build = 1;


    // constructor
    public SoftwareRelease(String app, double vers){
        this.application = app;
        this.version = vers;
    }


    // methods
    public void setMinor(int m){
        this.minor = m;
    }


    public void rebuild(){
        this.build = this.build + 1;
    }

    public String toString(){
        return this.application + "  " + this.version + "." + this.minor + ":" + this.build;
    }

    public SoftwareRelease nextRelease(){
        SoftwareRelease ans = new SoftwareRelease(this.application, this.version);
        ans.setMinor(this.minor+1);
        return ans;
    }









}
```

Question 5. ArrayLists of Objects [23 marks]

This question concerns a program for managing a database of cars owned by members of an automobile club. Two of the classes in the program are a Car class and a CarDB class.

The Car class below defines Car objects, which store information about individual cars.

```
public class Car{
    private String make;
    private String model;
    private int year;

    public Car(String mk, String mdl, int yr){
        this.make =mk;
        this.model = mdl;
        this.year = yr;
    }
    public String description(){
        return this.make +" "+ this.model +"("+ this.year +")";
    }
    public String getModel(){
        return this.model;
    }
    public boolean isVintage(){
        return (year>=1919 && year<=1931);
    }
}
```

(a) [3 marks] What will the following testCar method print out?

```
public static void testCar(){
    Car car1 = new Car("Ford", "Model A", 1928);
    Car car2 = new Car("Bentley", "Speed 6", 1926);
    UI.println(car1.description());
    UI.println(car2.getModel());
    if (car2.isVintage()){
        UI.println(car2.description());
    } else {
        UI.println("Car 2 is not vintage");
    }
}
```

```
Ford Model A (1928)
Speed 6
Bentley Speed 6 (1926)
```

(Question 5 continued on next page)

(Question 5 continued)

The CarDB class defines a field to store the database of Car objects, and several methods, including printAll, addCar, and countVintage.

```
private ArrayList<Car> clubCars = new ArrayList<Car>();
```

Assume that clubCars will never contain any null values.

(b) [6 marks] Complete the following printAll method that prints out (to the UI text pane) the number of cars in the database followed by a description of each car.

```
public void printAll (){
    UI.println ("Number of cars: "+this.clubCars.size());
    for (Car car : this.clubCars){
        UI.println (car.description ());
    }

}
```

(c) [7 marks] Complete the following addCar method that asks the user for the make, model, and year of a new car, makes a new Car object to store the information, and adds it to the database.

```
public void addCar(){
    String make = UI.askString("Make: ");
    String model = UI.askString("Model: ");
    int year = UI.askInt("Year: ");
    this.clubCars.add(new Car(make, model, year));

}
```

(Question 5 continued on next page)

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 continued)

(d) [7 marks] Complete the following countModel method that is passed a model name and returns the number of cars in the database that have a matching model name.

```
public int countModel(String model){
    int count = 0;
    for (Car car : this.clubCars){
        if (model.equals(car.getModel())){
            count++;
        }
    }
    return count;



}
```

Question 6. Files [12 marks]

Suppose a sports club has files containing data about the opponent and scores in each game that the club team has played in each season. Each line of a file contains the name of the opponent, the club team's score, and the opponent's score. For example, the file "scores13.txt" might contain the text:

```
Eagles     10    15
Hawkes      9     8
Spikers    11    15
Hawkes     18     7
Eagles     19    15
Suburbs    14    12
Rongotai    2    11
Suburbs     2     8
Spikers     8    23
Rongotai    7     3
```

Assume the opponent name is always a single word.

(a) [6 marks] Complete the following printWins method, whose parameter is the name of a file. printWins should read the file and print out the opponent for each game that the club team won, all on one line. For example, given the file above, printWins("scores13.txt") should print out

```
Won games against:
  Hawkes  Hawkes  Eagles  Suburbs  Rongotai
```

```
public void printWins(String fileName){
   try{
       UI.println("Won games against:");
       Scanner sc = new Scanner(new File(fileName));
       while (sc.hasNext()){
           String opponent = sc.next();
           int score1 = sc.nextInt();
           int score2 = sc.nextInt();
           if (score1 > score2){
              UI.print(team+" ");
           }
       }
       UI.println()




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

(Question 6 continued)

(b) [6 marks] Complete the following printStats method that prints out the average score of the club team, and the maximum amount they won by in any game. For example, printStats("scores13.txt") would print out:

```
Average score: 10.0
Maximum win by: 11
```

(The club won the second game against Hawkes by 11 points)

```
public void printStats(String fileName){
   try{
        double total = 0;
        int count = 0;
        int maxWin = 0;
        Scanner sc = new Scanner(new File(fileName));
        while (sc.hasNext()){
            String opponent = sc.next();
            int score1 = sc.nextInt();
            int score2 = sc.nextInt();
            total = total + score1;
            count++;
            int margin = score1-score2;
            if (margin > maxWin){
                maxWin = margin;
            }
        }
        sc.close();









        UI.println("Average score: " + (total/count));
        UI.println("Maximum win by: "+ maxWin);

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

Question 7. More ArrayLists of Objects [26 marks]

For this question, you are to complete several methods for a TowerPlanner class that is part of a program for planning the locations of cellphone towers. Each tower covers a certain region. Critical issues include whether a tower covers too large a region and how much the coverage regions overlap.

The TowerPlanner class has a field containing a list of the proposed Towers:

```
private ArrayList<Tower> towers = new ArrayList<Tower>();
```

The program also has a Tower class which has the following methods:

Methods in Tower **class**:

```
public int getID()
    // Returns the ID of the tower.

public double area()
    // Returns the area of the coverage region of the tower

public double overlap(Tower other)
    //  Returns the area of overlap of the regions of this tower and the other tower.
```

.

(Question 7 continued on next page)

(Question 7 continued)

(a) [5 marks] Complete the following largeCoverage method which should construct and return an ArrayList of all the towers with a coverage area of 1000.0 or more.

```
public ArrayList<Tower> largeCoverage(){
    ArrayList<Tower> ans = new ArrayList<Tower>();
    for (Tower tower : towers){
        if (tower.coverage() >= 1000.0) {
        ans.add(tower);
        }
    }
    return ans;
OR
    ArrayList<Tower> ans = new ArrayList<Tower>();
    for (int i=0; i<towers.size(); i++){
        if (towers.get(i).area() >= 1000.0) {
        ans.add(tower);
        }
    }
    return ans;
}
```

A critical issue in planning tower locations is ensuring that the coverage regions of different towers overlap a bit, but not too much.

(b) [8 marks] Complete the following totalOverlap method that computes the total overlap area of a plan by summing the overlap of every pair of towers.

```
public double totalOverlap(){
    double total = 0;
    for (int i=0; i<towers.size(); i++){
        for (int j=i+1; j<towers.size(); j++){
        total += towers.get(i).overlap(towers.get(j));
        }
    }
    return total;
    // should not count pairs twice, nor compute overlap of any tower with itself.
OR //(not so efficient)
    double total = 0;
    for (Tower t1 : this.towers){
        for (Tower t2 : this.towers{
            if (t1!=t2){
                total += t1.overlap(t2);
    }}
}   return total/2;   // we have counted every pair twice!
```

(Question 7 continued on next page)

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.

OR

```
    for (int i=0; i<this.towers.size(); i++){
        Tower tower = this.towers.get(i);
        double area = tower.area();
        for (int j=0; j<this.towers.size(); j++){
            if (i!=j){
                if (tower.overlap(this.towers.get(j))/area > .4){
                    towers.remove(i);
                    i--; // The next tower got moved down; we have to check it now.
                    break;
    }}}}
```

OR

```
    ArrayList<Tower> toRemove = new ArrayList<Tower>();
    for (Tower t1 : towers){
        for (Tower t2 : towers){
            if (t1!=t2 && t1.overlap(t2) > 0.4*t1.area()){
                toRemove.add(t1);
                break;
            }
        }
    }
    for(Tower t : toRemove){
        towers.remove(t);
    }
```

(Question 7 continued)

(c) [10 marks] Complete the following removeBadlyPlaced method that will remove badly placed towers from the list of towers. A tower is badly placed if it has an overlap of more than 40% of its coverage area with any other single tower. The method should step through the list of towers. If it finds a tower that is badly placed, it should remove the tower from the list.

```
public void removeBadlyPlaced(){
    for (int i=this.towers.size()-1; i>0; i--){ // stepping backwards makes it easier
        Tower tower = this.towers.get(i);          // can't use foreach because removing elements
        double area = tower.area();
        for (int j=0; j<this.towers.size(); j++){
            if (i!=j){      //
                double fraction = tower.overlap(this.towers.get(j))/area;
                if (fraction > .4){
                    this.towers.remove(i);
                    break;
                }
            }
        }
    }
    // note that t1.overlap(t2) may be > 40% of t1.area(), but < 40% of t2.area()
    // so we much check each pair both ways round (but not check towers against themselves.)
}
```

(d) [3 marks] Explain why the removeBadlyPlaced method may remove more towers than is really necessary.

If tower C overlapped towers A and B by more than 40%,
but A and B didn't overlap each other very much,
then we only need to remove tower C.
But if the method checked towers A and B before C, then it would remove A and B rather than C.

Question 8. 2D Arrays [17 marks]

You are writing a StudyBooker program for the library to help manage the bookings for its ten study rooms. Students may book a room for a one hour time slot. The StudyBooker program keeps all the bookings in a 2D array of Strings:

```
private String[ ][ ] bookings = new String [24][10];
```

The rows of the array are hours (0 to 23) and the columns of the array are the study rooms, numbered 0 to 9. The values in the array are student names, or null if there is no booking.

(a) [7 marks] Complete the following addBooking method that asks the user for a name and a preferred hour. It then searches the bookings array for a study room that is free at that hour. If it finds such a room, it prints out the number of the room and adds the student name as a booking for that room at that hour. If there is no free room, it simply prints out a message.
You may assume the user enters a valid hour.

```
public void addBooking(){
    String name = UI.askString("Name: ");
    int hour = UI.askInt("Hour: ");
    for (int room=0; room<10; room++){
        if (this.bookings[hour][room] == null){
        this.bookings[hour][room] = name;
        UI.println("Booked in room "+room);
        return;
        }
    }
    UI.println("No room found for that time");
}
```

(Question 8 continued on next page)

(Question 8 continued)

(b) [10 marks] At times, the librarians may need to close a room and move all the bookings to other rooms. Complete the following closeRoom method that asks the user for a room number, and then attempts to move each booking in that room to some other room at the same time. For each booking, it should print a message saying either where the booking was moved to, or that the booking could not be moved.
You may assume the user enters a valid room number.

```
public void closeRoom(){
    int room = UI.askInt("room: ");
    for (int hour=0; hour<24; hour++){
        String booking = this.bookings[hour][room];
        this.bookings[hour][room] = null;
        if (booking!=null){      // don't try to move null bookings.
            for (int r=0; r<10; r++){
                if (r!=room && this.bookings[hour][r] == null){
                    this.bookings[hour][r] = booking;
                    UI.println(booking +" at "+ hour + " moved to room "+r);
                    booking = null;
                    break;
                }
            }
            if (booking != null){
                UI.println(booking +" at "+ hour + " could not be rebooked");
                this.bookings[hour][room] = null;
            }
        }
    }
// checking r!=room is not actually necessary since
// this.bookings[hour][room] is known to be not null









}
```

Question 9. Debugging Loops [14 marks]

The following smooth method is intended to "smooth" an array of numbers by replacing each number by the maximum of the number and the numbers on each side of it. For example, given the array:

nums:	10	20	12	8	15	17	6	7
	0	1	2	3	4	5	6	7

smooth(nums) should change the array to be

nums:	20	20	20	15	17	17	17	7
	0	1	2	3	4	5	6	7

The following version of smooth has multiple errors.

```
/** Replace each value in data by maximum of the value and its immediate neighbours */
    // This version is broken!
public void smooth(int [ ] data){
    for ( int i=1; i<data.length−1; i++){
        int neighbour = Math.max(data[i−1], data[i+1]);
        data[i] = Math.max(data[i], neighbour);
    }
}
```

(a) [4 marks] Show the contents of the nums array after the version of smooth is called on the original value of the nums array above:

nums:	10	20	20	20	20	20	20	7
	0	1	2	3	4	5	6	7

(Question 9 continued on next page)

(Question 9 continued)

(b) [10 marks] Write a correct version of smooth.

```
/** Replace each value in array by maximum of the value and its immediate neighbours */
public void smooth(int [ ] data){
        if (data.length>1){
            int[ ] temp = new int[data.length];
            temp[0] = Math.max(data[0],data[1]);
            for (int i=1; i<data.length-1; i++){
                int neighbour = Math.max(data[i-1], data[i+1]);
                temp[i] = Math.max(data[i], neighbour);
            }
            temp[data.length-1]= Math.max(data[data.length-1],data[data.length-2]);
            for (int i=0; i<data.length; i++){
                data[i] = temp[i];
            }
        }
// Alternative ....
        int prev = -Integer.MIN_VALUE;
        for (int i=0; i<data.length-1; i++){
            int max = Math.max(prev, data[i]);
            if (i<data.length-1){
                max = Math.max(max, data[i+1]);
            }
            prev = data[i];
            data[i] = max;
        }









}
```

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