Student ID: .....

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VUW VICTORIA UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2015

TRIMESTER 1

COMP 112

INTRODUCTION TO COMPUTER SCIENCE

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

CLOSED BOOK

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

Printed foreign to English language dictionaries are permitted.

Java Documentation will be provided with the exam script

No other material is permitted.

Instructions: The exam will be marked out of 180 marks. 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. Defining a Class	[15]
2. Files	[30]
3. Event driven input	[30]
4. Shapes	[25]
5. Interface	[25]
6. 2D Arrays and images	[25]
7. Assert, Preconditions, Postconditions and Invariants	[30]

Question 1. Defining a Class [15 marks]

Complete the Part class on the facing page which stores information about parts used by a manufacturer.

Each Part object should contain fields to store:

- description, which contains the part description
- id, which contains a unique integer identifier for the part.
- weight, which contains the weight of the part
- billOfParts, which is a list of subcomponent Part objects

Part should have a constructor that takes a String parameter containing the description and a double containing the weight. These values should be stored in the relevant fields. It should also assign a unique IDCode, using a static field to keep track of the next ID to assign.

Part should have the following methods:

- getID() which returns the part id.
- getWeight() which returns the weight of the part
- setPriority(int p) which sets the priority of the package.
- addPartToBOP(Part p) which adds the input to the billOfparts.
- toString() method that prints the data held in all the fields. Only printing the id of the parts in the billOfParts.

(Question 1 continued on next page)

(Question 1 continued)

```
import java. util . ArrayList ;
public class Part
{    private String description ;
     private static int lastID = 1;
     private final int id ;
     private double weight;
     private ArrayList<Part> bom = new ArrayList<Part>();



     public Part(String d, double w) {

          description = d;
          weight = w;
          id = lastID++;
     }
     public int getID(){ return id ; }
     public double getWeight(){ return weight; }

     public String getDescription(){ return description ;}

     public void addParttoBOM(Part p)
     {          bom.add(p);
     }
     public String toString (){
          String s = "";
          for (Part p:bom) {
               String pid = String.format("%0d4", p.getID() );
               s += (pid +" ");}
          return description+" "+id+" bom = "+s;
     }




}
```

Question 2. Files [30 marks]

Each line of a file stores information about a delivery of parts to a factory. Each line consists of three integers:

The first integer is the quantity of the part delivered,

the second a manufacturer ID of the part and

the third is the ID of the part delivered.

Different lines of the file may be for the same manufacturer and the same part. The lines of the file are sorted by part ID.

Write the printPartIDQty method that takes a String parameter containing the name of the file and prints the total quantity of each part delivered.

From a file containing:

```
10 1 1
20 2 1
30 1 2
40 2 2
50 3 3
20 1 3
10 4 4
50 4 4
```

the printPartIDQty method should output

```
partid = 1 qty = 30
partid = 2 qty = 70
partid = 3 qty = 70
partid = 4 qty = 60
```

(Question 2 continued on next page)

(Question 2 continued)

(a) [15 marks] Complete the following printPartIDQty method,

```
public static void printPartIDQty(String filename)
    {
        int qty,manID,partID;
        int partQty = 0;
        int oldpartID = -1;
        Scanner scan = null;
        boolean first = true;
        try{
            scan = new Scanner(new File(filename));

            while (scan.hasNext()){
                qty = scan.nextInt ();
                manID = scan.nextInt ();
                partID = scan.nextInt ();
                if ( first == true) {
                    oldpartID = partID;
                }
                if (oldpartID == partID){
                    partQty += qty;
                } else {
                    System.out.println("partid = "+oldpartID+" qty = "+partQty);
                    partQty = qty;
                    oldpartID = partID;
                }
                first = false;
            }
            System.out.println("partid = "+oldpartID+" qty = "+partQty);
        } catch(Exception e) {
            System.out.println(e);
        } finally {
            scan.close(); // initialise instance variables
        }
        return;




      }
```

(b) [15 marks] The Inventory of the factory is stored in an Inventory object defined below:

```
public class Inventory {
        private ArrayList<Part> inv = new ArrayList<Part>();
        public Inventory(ArrayList<Part> i){ inv =i;}
        // returns the part for a given id
        public Part getPart( int id){
            for(Part p: inv){
                if (p.getID() == id)
                    return p;
            }
            return null;
        }
    }
```

Write the returnWeight(String filename,int inManID, Inventory inv) method that has three parameters:

1. the file name of the delivery file (see the first part of this question)
2. the id of a manufacturer
3. the inventory of the factory

the method should return the total weight of all the parts delivered from the given manufacturer. You will have to use the getWeight() method for Part objects from Question 1.

(Question 2 continued on next page)

(Question 2 continued)

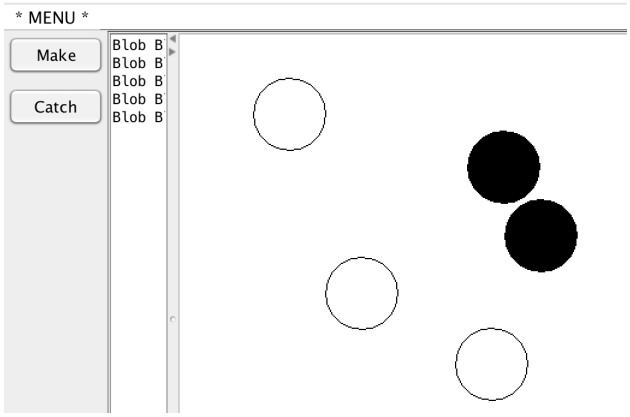
```
public static double returnWeight(String filename, int inManID, Inventory inv)
    {
        Scanner scan = null;
        double totalWeight = 0;
        try{
            scan = new Scanner(new File(filename));
            while (scan.hasNext()){
                int qty = scan.nextInt ();
                int manID = scan.nextInt ();
                int partID = scan.nextInt ();
                if (inManID == manID){
                    totalWeight += (qty * inv.getPart(partID).getWeight());
                }
            }
        } catch(Exception e) {
            System.out.println(e);
        } finally {
            scan.close();
        }
        return totalWeight;
    }
```

Question 3. Event Driven Input [30 marks]

Complete the Q3EDI program on the next **two** pages so that mouse clicking events can have one of two effects. Which of the effects a mouse clicking event performs is controlled by which of the two buttons was last pressed.

Make button last pressed - mouse release makes a Blob and draws it as a hollow circles on the screen.

Catch button last pressed - mouse pressed action defines one corner of a rectangle and the mouse released as the diagonal corner of the rectangle. The representation of all blobs in the rectangle must be changed into solid circles.



The program should use the following Blob class:

```
private class Blob {
    private double bx;        // x coordinate of centre
    private double by;        // y coordinate of centre
    private double bsize;     // radius
    // draw circle with centre=(x,y) radius = size
    public Blob(double x, double y,double size) {
        UI.drawOval(x−size, y−size,2∗size,2∗size);
        bx = x;  by = y;  bsize = size;}

   // is point (x,y) in this circle
    public boolean in(double x, double y){
        if (bsize∗bsize > (bx−x)∗(bx−x) +(by−y)∗(by−y))
            return true;
        else
            return false;
    }
    public double getX(){return bx;}
    public double getY(){return by;}

     // make circle solid ( fill the circle )
    public void fillblob (){ UI. fillOval (bx−bsize, by−bsize,2∗bsize,2∗bsize); }
}
```

.

(Question 3 continued)

(a) [10 marks] Start of Q3EDI program.

```
public class Q3EDI
{
    private ArrayList<Blob> blobs = new ArrayList<Blob>();
    private double halfSize = 30;
    private boolean make = false;
    double ox = 0,oy = 0;


    public Q3EDI()

    {
        UI.setMouseListener(this::mousePerformed);
        UI.addButton("Make", this::buttonM);
        UI.addButton("Catch", this::buttonC);

    }


    public void buttonM()

    {
        make = true;
        return;
    }
    public void buttonC()
    {
        make = false;
        return;
    }

    public static void main(String[] arguments)
      {  UI. initialise ();
        Q3EDI obj = new Q3EDI();

    }
```

(Question 3 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 3 continued)

(b) [20 marks] Q3EDI program continued:

```
/** Respond to the mouse */
  public void mousePerformed(String action, double x, double y) {
    if (action.equals("pressed")) {
        if (make == false) {
            ox = x;
            oy = y;
        }
    } else if (action.equals("released")){
        if (make == true) {
            Blob b = new Blob(x,y,halfSize);
            blobs.add(b);
        } else {
            for(Blob b:blobs){
                double bx = b.getX();
                double by = b.getY();
                UI. println ("Blob "+b);
                if ((ox > bx && bx > x)||
                (ox < bx && bx < x))
                    if ((oy > by && by > y)||
                    (oy < by && by < y))
                        b. fillblob ();
            }
        }

    }
}









  }
```

Question 4. Shapes [25 marks]

In each subquestion write a method for the Q4Shapes object to produce designated designs.

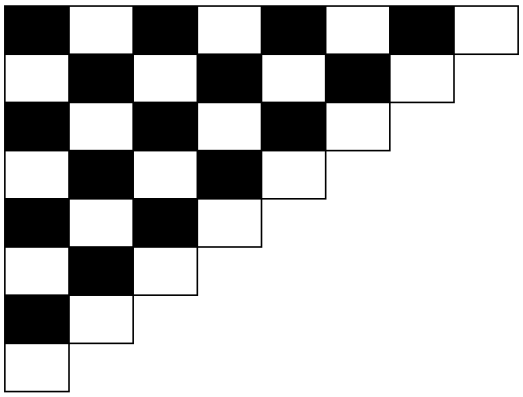
The Q4Shapes object contains the following fields.

```
public class Q4Shapes
{
    private double initx  = 50;        // x co-ordinate of top left corner of design
    private double inity  = 100;       // y co-ordinate of top left corner of design
    private int  SQ = 8;               // max number of squares in a row
    private double WIDTH = 40;         // width of square
    private double HEIGHT = 30;        // height of square
```

(Question 4 continued on next page)

(Question 4 continued)

(a) [12 marks] The buttonTriangle method displays the triangle below. Use the fields in the object and **avoid hard coding** any feature of the design.



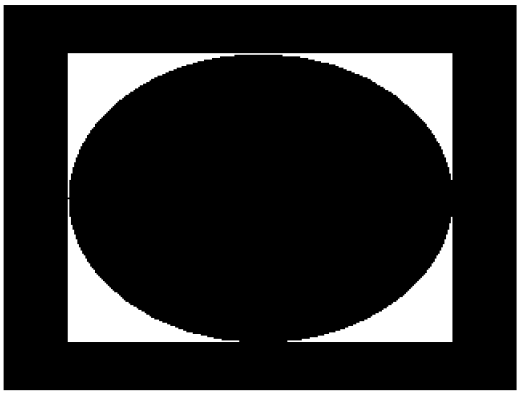
```
public void buttonTriangle() {

        UI.println("Triangle pushed");
        UI.clearGraphics();
        for(int j = 0; j<SQ;j++)
        {
            for(int i = 0; i<(SQ-j);i++){
                if ((i+j)%2 == 0)
                {
                    UI.fillRect ( initx +(i*WIDTH), inity+(j*HEIGHT), WIDTH, HEIGHT);
                }else
                {
                    UI.drawRect(initx+(i*WIDTH), inity+(j*HEIGHT), WIDTH, HEIGHT);
                }
            }
        }
        UI.repaintGraphics();









}
```

(Question 4 continued on next page)

(Question 4 continued)

(b) [13 marks] The buttonFrame method displays the frame below.

Use the fields in the object and **avoid hard coding** any feature of the design.
The edges of the frame can each be built using SQ squares. The size of the oval needed must be computed so that it just touches the four edges of the frame.



```
public void buttonFrame() {

        UI.println("Frame pushed");
        UI.clearGraphics();
        for(int j = 0; j<SQ;j++)
        {
            for(int i = 0; i<SQ;i++){
                if ((i == 0)||(i==SQ-1)||(j==SQ-1)||(j==0))
                {
                    UI.fillRect(initx+(i*WIDTH), inity+(j*HEIGHT), WIDTH, HEIGHT);
                }
            }
        }
        UI.fillOval(initx+WIDTH, inity+HEIGHT, WIDTH*(SQ-2), HEIGHT*(SQ-2));
        UI.repaintGraphics();
    }
      UI.repaintGraphics();









}
```

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. Interface [25 marks]

A game needs to have a collection of Cars. A car must be able to do the following:

1. refuel by spending a sum of money
2. respond to a request to move a number of Kilometres
3. return the total distance it has travelled since it was constructed.

(a) [5 marks] Complete the following Car interface:

```
public interface Car{

        public void refuel( int  dollars );
        public double move(double km);
        public double getDistance();









}
```

Two distinct types of Cars, ElectricCars and PetrolCars have to be implemented in the following subsections.

(b) [10 marks] Complete the PetrolCar method that

1. implements the car interface and
2. has a constructor with three parameters,
   (a) an initial amount of petrol in Litres
   (b) the petrol efficiency in Kilometres per Litre
   (c) the cost of the fuel in litres per dollar.

The PetrolCar only moves if it has sufficient fuel for the distance requested and must return the distance it travelled and update its remaining fuel. Refuelling and reporting the total distance travelled are self explanatory.

(Question 5 continued on next page)

(Question 5 continued)

```
public class PetrolCar implements Car {

        private double litresPerDollar = 0.5;
        private double kmPerlitre = 10;
        private double petrol = 0;
        private double distanceTraveled = 0;

        public PetrolCar(double p, double kpl, double led){
            petrol = p;
            kmPerLiter = kpl;
            litersPerDollar = lpd;
        }
        public void refuel( int dollars ){
            petrol += PetrolCar. litersPerDollar * dollars ;
        }
        public double move(double km) {
            double needed = km / kmPerLiter ;
            if (petrol > needed) {
                petrol -= needed;
                distanceTraveled += km;
                return km;
            }
                return 0;
            }
        public double getDistance(){return distanceTraveled;}
        public String toString (){
            return "PetrolCar Lt/$ = "+litersPerDollar+
                   " liters = "+petrol+
                   " Km/Lt = "+kmPerLiter+
                   " Km = "+distanceTraveled;
        }
    }

}
```

(Question 5 continued on next page)

(Question 5 continued)

(c) [10 marks] Complete the ElectricCar method that

1. implements the car interface and
2. has a constructor with three parameters,
   (a) an initial amount of electricity in Kilowatts
   (b) the efficiency in Kilometres per Kilowatt
   (c) the cost of the fuel in Kilowatts per dollar.

The ElecticCar moves if it has some power, even if it has insufficient for the distance requested and must return the distance it travelled. Refuelling and reporting the total distance travelled are self explanatory.

(Question 5 continued on next page)

(Question 5 continued)

```
public class ElectricCar  implements Car
{
    private double kwPerDollar = 0;
    private double kw = 0;
    private double kmPerKw = 0;
    private double distanceTraveled = 0;
    public ElectricCar(double p, double kpl, double kpd){
        kw = p;
        kmPerKw = kpl;
        kwPerDollar = kpd;
    }
    public void refuel( int dollars ){
        kw = ElectricCar.kwPerDollar * dollars ;
    }
    public double move(double km) {
        double able = kmPerKw *kw ;
        if (able > km) {
            kw -= km/ kmPerKw;
            distanceTraveled += km;
            return km;
        } else {
            kw = 0;
            distanceTraveled += able;
            return able;
        }
    }

    public void sunny(double s) {
        kw += s;
    }
    public double getDistance(){return distanceTraveled;}

    public String toString (){
            return "ElectricCar Kw/$ = "+kwPerDollar+
                   "  Kw = "+kw+
                   "  Km/Kw = "+kmPerKw+
                   "  Km = "+distanceTraveled;
        }
}
}
```

Question 6. 2D arrays and images [25 marks]

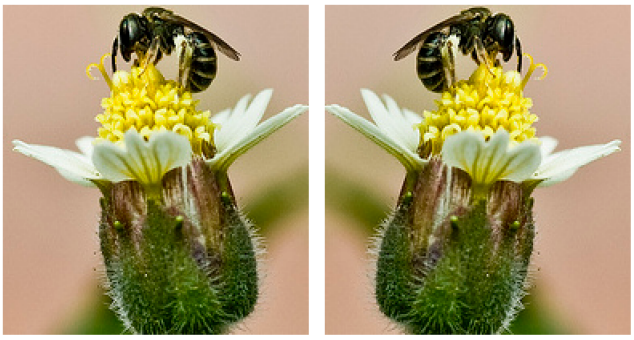
An ImageProcessor program contains an image field consisting of a 2D array of the Color values:

```
private Color [ ][ ] image;
```

(a) [12 marks] Complete the following horizontalFlip method which will horizontally flip the image on the left. For example, horizontalFlip would turn the image on the right into the image on the left.

Note: after calling horizontalFlip, the image in the image field is flipped and should be in the working field.

working **image**



```
public void horizontalFlip (){

        this.checkWorking();
        int maxCol = image[0].length-1;
        for (int row = 0; row < image.length; row++){
            for (int col = 0; col < image[row].length; col++){
                working[row][col]=
                new Color(this.image[row][maxCol-col].getRGB());
            }
        }









}
```

(Question 6 continued on next page)

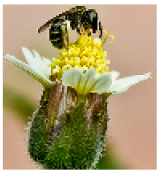
(Question 6 continued)

(b) [13 marks] Complete the following compress method which will compress the image on the left. The compressed image contains every other row and each row consists of every other pixel. Resulting in an image of, approximately, half the height and half the width.

For example, compress would turn the image on the right into the image on the left.

Note: after calling compress, the image in the image field is compressed and should be held in the working field.

working



image

```
public void compress(){


        this.whiteWorking();
        int rw = 0;
        int cw = 0;
        for (int row = 0; row < image.length; row++){
            if (row % 2 == 0) {
                for (int col = 0; col < image[row].length; col++){
                    if (col % 2 == 0)
                    {
                        this.working[rw][cw]=
                        new Color(this.image[row][col].getRGB());
                        cw++;
                    }
                }
                cw = 0; rw++;
            }
        }



}
```

Question 7. Assert, Precondition, PostCondition and Invariants [30 marks]

Run time checks can be inserted at a line of Java code with the following command

assert (condition): error message;

the resulting code can be run with or without these checks.

The StackAssert class implements a stack and has asset statements to check it satisfies the documented specification. It has three public methods:

stackAssert a constructor with a parameter, the size of the stack to be built

push adds the parameter to the top of the stack

pop removes an element from the top of the stack and returns it.

Complete the class filling in the code and the assert statements.

The specification of each of the nine conditions are numbered and appear in comments in the code. The error message of each assert statement should start with the number of the condition it relates to.

For example: an assert statement relating to condition 3 in the constructor should look like:

assert ..... : "3.....";

The class has two invariants and to save repeated coding these checks are defined in the private method invariant that has to be called from the public methods. You need to decide where to call them from.

(Question 7 continued on next page)

(Question 7 continued)

(a) [18 marks] **Constructor and private invariant method**

```
public class StackAssert
{
    private int stackMax ; // Index of last element in stack
    private int stackTop ; // index to next free entry in array
    private int [] myStack =null;   // data in the stack
    /*
     * Class invariant method only used to check the two class invariants .
     * you will have to call this method in the public methods of the class .
     * 1. all elements in stack must be greater than or equal to 10
     * 2. stack can not have more elements than its size
     */
    private void invariant (){




        assert(stackTop<= stackMax +1) : "2 stack exceeds maximum";
        for( int i = 0;i<stackTop; i++){
            assert (myStack[i] >= 10) : "1 some element less that 10";
        }



    }
    /**
     * 3. parameter must be greater than zero
     * 4. stack size must be parameter
     */
    public stackAssert(int ss)
    {
        assert ss > 0 : "3 stack must be ghreater than zero";
        stackMax = ss -1;
        stackTop = 0;
        myStack = new int[ss];
        assert (ss- stackMax == 1) : "4 stack built at wrong size";
         invariant ();



    }
```

(Question 7 continued on next page)

(Question 7 continued)

(b) [12 marks] **Push pop with pre and post conditions and class invariant**

```
/**
 * 5.pop Must only be called if stact contains an element
 * 6.pop Will reduce the stack size by one
 */
public int pop()
{
    assert (stackTop > 0) : "5 poping an empty stack";
    invariant ();
    int old = stackTop;
    stackTop--;
    assert (old -stackTop == 1) : "6 pop stack size failure";
    invariant ();
    return myStack[stackTop];









}
/**
 * 7.push must not be called if stack is full
 * 8.element pushed onto stack must be 10 or greater
 * 9.push will increase the top of stack by one
 */
public void push(int i){

    assert (stackTop <= stackMax) : "7 pushing onto a full stack";
    invariant ();
    assert (i >=10) : "8 pushing element less that 10";
    int old =stackTop;
    myStack[stackTop] = i;
    stackTop++;
    assert (stackTop - old == 1) : "9 pushing stack size failue";
    invariant ();









}
```

Student ID: .....

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