Please fill in your Student Number and Name.

Student Number : _____

Name:

_____

_____

Student Number:

_____

University of Cape Town ~ Department of Computer Science

Computer Science 1011H/1016S ~ 2008

January Exam

Question	Max	Internal	External	Question	Max	Internal	External
1	25			7	25		
2	10						
3	15						
4	5						
5	10						
6	10						
				TOTAL	**100**		

Marks : 100

Time : 180 minutes

Instructions:

a) Answer all questions.

b) Write your answers in the space provided.

c) Show all calculations where applicable.

Question 1: Recursion, Exceptions and File handling [25]

Study the program below carefully and answer the questions that follow.

```
import java.io.*;
import java.util.*;

public class Exam2008Supp {
   public static void main(String[] args)
     throws FileNotFoundException {

      Scanner scan = null;
      PrintWriter pw = null;

      try {
         scan = new Scanner(new FileInputStream("fileA.txt"));
         pw = new PrintWriter(new FileOutputStream("fileB.txt"));
         pw.println("Stack:");
         int level = scan.nextInt();
         pw.println(Stack(level,""));
      }
      catch (FileNotFoundException e) {
         pw.println(e.getMessage());
      }
      catch (InputMismatchException e) {
         pw.println("Input exception occurred");
      }
      finally {
         pw.println("Completed."); pw.close();
      }
   }

   public static String Line(int n, char C) {
      String tmp="";
      for(int i=0;i<n;i++)
         tmp+=C;
      return tmp;
   }

   public static String Tri(int n,String shift) {
      if (n>0) {
         String tmp = shift+Line(n*2-1,'*')+'\n';
         return tmp + Tri(n-1,shift+" ");
      }
      else
         return "";
   }

   public static String Stack(int n,String offset)      {
      //code hidden
   }
}
```

a) Write a recursive definition for the method Line in the program listed above. Note that no marks will be awarded for iterative solutions. [3]

```
public static String Line(int n, char C) {
if (n>0) //[1]
  return C+Line(n-1,C); //[2]
 return"";
}
```

b) For different values of the input parameters, the method Stack produces output as follows:

Stack(1,"") **returns the string:**

```
*
```

Stack(2,"") **returns the string:**

```
***
 *
  *
```

Stack(4,"") **returns the string:**

```
*******
 *****
  ***
   *
    *****
     ***
      *
       ***
        *
         *
```

Stack(3,"") **returns the string:**

```
*****
 ***
  *
   ***
    *
     *
```

etc.

Write a **recursive** definition for the method ***Stack***. Note that no marks will be awarded for iterative solutions. [4]

```
public static String Stack(int n,String offset) {
if (n<=0) [1]
        return "";
      else
        return Tri(n,offset)+ Stack(n-1,offset+Line(n,' ')); [3]
}
```

c) Explain clearly what a ***StackOverflowError*** is and why this type of error can occur in recursive functions. [2]

Stack overflow error means that the stack grows beyond it's maximum range. This happens as a result of infinite recursion.

d) Explain clearly what checked exceptions are in Java and give an example of one from the program above. [3]

Conform to the "catch-or-declare rule" – must be caught or declared to be thrown. An example is FileNotFoundExcpetion but not InputMismatchExcpetion

e) List three conditions under which a FileNotFoundException would be thrown by the program listed above. [3]

If fileA does not exist

You can't write to fileB [permissions]

You can't read from fileA [permission]

Also if you can't wrtie to current directory

f) Assume that, before the program is run, the files contain the following text:

`fileA.txt`:

```
ABE 4 5
```

`fileB.txt`:

```
3 2
5
```

Now write down the **exact** contents of each of these files **after** the program is run. [3]

File A;

ABE 4 5 [1]

File B:

Stack:

Input exception occurred

Completed. [2]

g) Define the concept of a stream in the Java programmng language. [1]

A stream is a flow of data

h) Give an example of a standard stream used in the program above. [1]

System.out

i) Explain clearly why it is a good idea to include the line **`pw.close();`** in the finally block in the program above. [2]

calling close() explicitly flushes the buffer. If this is not done, writes to the files could not actually be written to the file in an error situation. It's good to have this in a finally block, as the close will happen whether or not an error situation occurs – finally is always executed.

j) Explain why Java binary files are portable, whereas binary files from other programming languages (like C++) are not. [3]

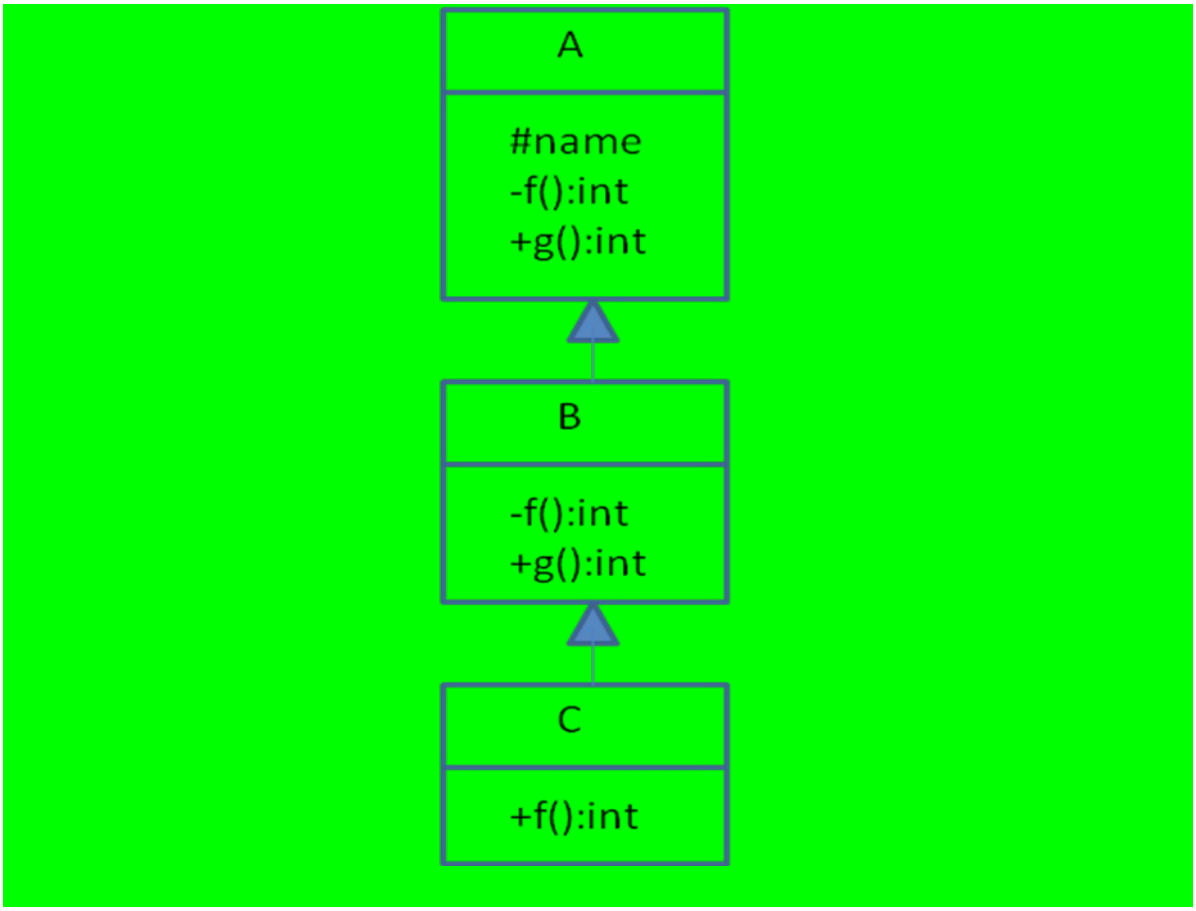
Java binary files are in Byte code, which is the same for all Java virtual machines. Binary files in C++ are specific to the architecture that they are compiled on and therefore are not portable.

Question 2: UML, Abstract classes, Inheritance and Polymorphism [10]

Use the following code to answer the questions that follow.

```
public class Polymorphism {

    public static void main(String[] args) {
        A ref1 = new C();
        B ref2 = (B) ref1;
        System.out.println(ref2.g());
    }
}

public class A {
    protected String name = "CSC1016S";
    private int f() { return 0; }
    public int g() { return 3; }
}

public class B extends A {

    private int f() { return 1; }
    public int g() { return f(); }
}

public class C extends B {
    public int f() { return 2; }
}
```

a) Draw a UML diagram to illustrate the relationships among classes A, B and C. In each class diagram, all the class members should be described with their corresponding accessibility. [4]



b) What will be the result of attempting to compile and run the program Polymorphism? Select the one correct answer below. [2]

i. The program will fail to compile.
ii. The program will compile without error and print `0` when run.
iii. The program will compile without error and print `1` when run.
iv. The program will compile without error and print `2` when run.
v. The program will compile without error and print `3` when run.

iii

c) Explain the keyword protected and compare it with the private and public keywords. [2]

protected means public to all decedent classes in the hierarchy but means private to all outside classes. The keyword protected prevents access from outsider

d) Which statements below are valid? Select all that apply. [2]

i. Variable name can be accessed inside class A
ii. Variable name can be accessed inside class B
iii. Variable name can be accessed inside class C
iv. Variable name can be accessed inside class Polymorphism

i, ii and iii

Question 3: Interfaces and Sorting [15]

Use the following program to answer the questions that follow.

```
public class GeneralizedSelectionSort
{
  /**
   Precondition: numberUsed <= a.length;
   The first numberUsed indexed variables have values.
   Action: Sorts a so that a[0],a[1],...,a[numberUsed - 1] are in
   increasing order by the compareTo method.
  */
  public static void sort(Comparable[] a, int numberUsed)
  {
    int index, indexOfNextSmallest;
    for (index = 0; index < numberUsed - 1; index++)
    { //Place the correct value in a[index]:
      indexOfNextSmallest = indexOfSmallest(index, a, numberUsed);
      interchange(index,indexOfNextSmallest, a);
      //a[0],a[1],..., a[index] are correctly ordered and these are
      //the smallest of the original array elements. The remaining
      //positions contain the rest of the original array elements.
    }
  }

  /**
   Returns the index of the smallest value among
   a[startIndex], a[startIndex+1], ... a[numberUsed - 1]
  */
  private static int indexOfSmallest(int startIndex,
    Comparable[] a, int numberUsed)
  {
    Comparable min = a[startIndex];
    int indexOfMin = startIndex;
    int index;
    for (index = startIndex + 1; index < numberUsed; index++)
      if (a[index].compareTo(min)<0)//if a[index] is less than min
      {
        min = a[index];
        indexOfMin = index;
        //min is smallest of a[startIndex] through a[index]
      }
    return indexOfMin;
  }

  /**
   Precondition: i and j are legal indices for the array a.
   Postcondition: Values of a[i] and a[j] have been interchanged.
  */
  private static void interchange(int i, int j, Comparable[] a)
  {
    Comparable temp;
    temp = a[i];
    a[i] = a[j];
    a[j] = temp; //original value of a[i]
  }
}
```

a) Explain what a `Comparable` interface is and what it is used for. [2]

The Comparable interface is in the java.lang package and so is automatically available to your program. Defining a class to implement the Comparable interface means you are agreeing to redefine the compareTo method [1mark] so that instances of the class can be ordered semantically [1mark]

b) List and explain the main differences between interfaces and abstract classes (at least 2). [2]

Abstract Classes ;Interfaces

Abstract classes are used only when there is a "is-a" type of relationship between the classes. ;Interfaces can be implemented by classes that are not related to one another.

You cannot extend more than one abstract class. ;You can implement more than one interface.

Abstract class can implemented some methods also. ;Interfaces can not implement methods.

With abstract classes, you are grabbing away each class's individuality.;With Interfaces, you are merely extending each class's functionality.

c) Write a class called Student that implements the Comparable interface. Order is based on student mark and then name. [4]

To answer this question, declare the class, constructor, private `mark` and `name` variables, then implement the `compareTo` method that should cover all the comparison cases.

```
public class Student implements Comparable
{
 private int mark = 0;
 private String name = "";
 public Student(int mark_, name_) {
  mark = mark_;
  name = name_;
 }

 public int compareTo(Student student_) {
  if (this.mark > student_.mark)
   return 1;
  else if (this.mark < student_.mark)
   return -1;
  else
   return this.name.compareTo(student_.name)
 }
}
}
```

d) Write a driver program, called test that uses GeneralizedSelectionSort to sort an array of Student objects. [3]

To answer this question, create an array of 3 Student objects, initialize them, and sort the array based on the given GeneralizedSelectionSort. The three objects are: Student(9, "Abba"), Student(7,"Anna") and Student (9,"Bertus").

```
public class test {
 public static void main(String[] args) {
  Student students[] = new Student[3];
  students[0] = new Student(9, "Abba");
  students[1] = new Student(7,"Anna");
  students[2] = new Car(9,"Bertus"); GeneralizedSelection.sort(students, students.length);
 }
}
```

e) What is the order of the objects after running the sort? [2]

Student(7,"Anna"), Student(9, "Abba"), Student(9,"Bertus")

f) If there is no compareTo() method in the Student class, can we compile the program test successfully? Explain why or why not. [2]

No [1mark] because the commitment to redefine the compareTo method in MyClass's heading (by implementing Comparable interface) and MyClass is not an abstract class [1mark]

Question 4: Data Structures [5]

The following is a partial definition of a binary tree.

```
public class BinaryTree
{
    private class Node
    {
        private int data;
        private Node left;
        private Node right;

        public Node(int newData, Node newLeft, Node newRight)
        {
            data = newData;
            left = newLeft;
            right = newRight;
        }
    }

    private Node root;

    private void showElements ... // incomplete
}
```

Complete the method showElements which performs inorder traversal of the tree and prints each item in the tree. [5]

```
private void showElements(Node r)
{
  if (r != null)
  {
    showElements(r.left);
    System.out.println(r.data);
    showElements(r.right);
  }
}
```

Question 5: Data Structures [10]

The following is a partial definition of a simple linked list.

```
public class LinkedList
{
    private class Node
    {
        private String data;
        private Node next;

        public Node(String newItem, Node nextValue)
        {
            item = newItem;
            next = nextValue;
        }

    }//End of Node inner class

    private Node head;

    public void add(String newData, Node nextValue)
    {
    }

    public boolean delete()
    {
    }

}
```

a) Fill in the method named **add**, which stores a new item at the start of the list. [3]

```
public void add(String newData)
{
 head = new Node(newData, head);
}
```

b) Fill in the method named **delete**, which removes the item at the start of the list and returns **true**. If the list is empty, it returns **false**. [4]

```
public boolean delete()
{
    if (head != null)
    {
        head = head.next;
        return true;
    }
```

```
    else
        return false;
}
```

c) Rewrite the partial definition above (excluding the methods **add** and **delete**) so that the list becomes a doubly linked list. [3]

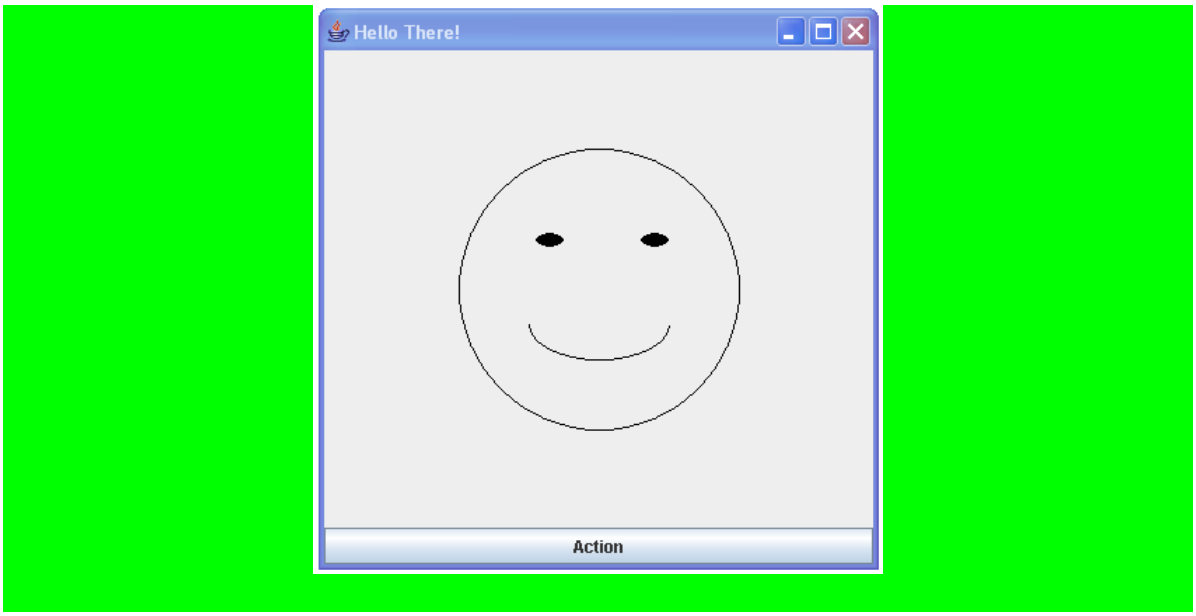
A queue is a FIFO structure where the first item to be added is the first one retrieved. Items are added to the back and removed from the front. The simple linked list must be modified to have a pointer to the back of the list so that items can be added there. The operations for the queue are enqueue (add to the back) and dequeue (delete from the front).

Question 6: GUIs [10]

Study the following program and answer the questions that follow.

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class ActionFace extends JFrame implements ActionListener
{
    private boolean x;

    public void actionPerformed(ActionEvent e)
    {
        x = true;
        repaint( );
    }

    public static void main(String[] args)
    {
        ActionFace drawing = new ActionFace( );
        drawing.setVisible(true);
    }

    public ActionFace( )
    {
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Hello There!");
        setLayout(new BorderLayout( ));

        JButton actionButton = new JButton("Action");
        actionButton.addActionListener(this);
        add(actionButton, BorderLayout.SOUTH);
        x = false;
    }

    public void paint(Graphics g)
    {
        super.paint(g);
        if (x)
        {
           g.setColor(Color.RED);
           g.drawArc(150, 200, 100, 50, 0, 180);
        }
        else
            g.drawArc(150, 200, 100, 50, 180, 180);
        g.drawOval(100, 100, 200, 200);
        g.fillOval(155, 160, 20, 10);
        g.fillOval(230, 160, 20, 10);
    }
}
```

a) Draw the GUI produced by the program. [7]



b) What does the program do when the Action button is clicked? [3]

The mouth changes from a smile to a frown and the whole drawing is red.

Question 7: Ethics, Cyberlaw and Development [25]

a) Give a short definition of Utilitarianism. [2]

An action is right if it maximises benefits over costs for all involved, everyone counting equal. OR You do something because it is useful. What makes behaviour right or wrong depends wholly on the consequences

b) Discuss the ways in which legality (the law) may or may not coincide with ethicality (ethics). Try to give examples (which need not be related to computing) of the various possibilities. [8]

Essentially have to fill in the matrix and give an example at each box

	Legal	*Not Legal*
Ethical	*An act that is ethical and legal*	*An act that is ethical but not legal*
Not Ethical	*An act that is not ethical but is legal*	*An act that is not ethical and not legal*

c) Critically discuss the question: "Is Computing a Profession?" Consider amongst other things the extent to which the features of a profession are already present. [9]

Direct answer can be: Yes, No, Evolving. Has to be motivated. No or Evolving are the better answers. [1] Need to say to what extent the features of a profession are present.

The features they were told about are: 1) Mastery of an Esoteric Body of Knowledge, 2) Autonomy, 3) Formal Organisation, 4) Code of Ethics, 5) Social Function. [5]

Point 5 usually does not exist, or exist only up to some point: (A) Society not yet convinced that only people with special skills should do (specialised) computing jobs (B) Difficulty in defining the exact material that needs to mastered to be a Computing professional. [3 for reasonable discussion]

d) Discuss two important ways in which the Digital Divide applies (or shows itself) in South Africa. [6]

The South African Digital Divide grows out of a history of division and historical backlogs for large groups of people: a particular South African version of colonial history. The Digital Divide also arises from global circumstances which apply to all developing countries. Thus there are two aspects to the Digital Divide: [2]

a) Global Digital Divide: The global disparity between those countries at the forefront of the Information Economy and the developing countries. [2]

b) Local Digital Divide: This refers to the disparities between groups in a particular country. [2]

Appendix: Question 1 Program

```
import java.io.*;
import java.util.*;

public class Exam2008Supp {
   public static void main(String[] args)
     throws FileNotFoundException {

      Scanner scan = null;
      PrintWriter pw = null;

      try {
         scan = new Scanner(new FileInputStream("fileA.txt"));
         pw = new PrintWriter(new FileOutputStream("fileB.txt"));
         pw.println("Stack:");
         int level = scan.nextInt();
         pw.println(Stack(level,""));
      }
      catch (FileNotFoundException e) {
         pw.println(e.getMessage());
      }
      catch (InputMismatchException e) {
         pw.println("Input exception occurred");
      }
      finally {
         pw.println("Completed."); pw.close();
      }
   }

   public static String Line(int n, char C) {
      String tmp="";
      for(int i=0;i<n;i++)
         tmp+=C;
      return tmp;
   }

   public static String Tri(int n,String shift) {
      if (n>0) {
         String tmp = shift+Line(n*2-1,'*')+'\n';
         return tmp + Tri(n-1,shift+" ");
      }
      else
         return "";
   }

   public static String Stack(int n,String offset)      {
      //code hidden
   }
}
```

Appendix: Question 2 Program

```
public class Polymorphism {

    public static void main(String[] args) {
        A ref1 = new C();
        B ref2 = (B) ref1;
        System.out.println(ref2.g());
    }
}

public class A {
    protected String name = "CSC1016S";
    private int f() { return 0; }
    public int g() { return 3; }
}

public class B extends A {

    private int f() { return 1; }
    public int g() { return f(); }
}

public class C extends B {
    public int f() { return 2; }
}
```

Appendix: Question 3 Program

```
public class GeneralizedSelectionSort
{
  /**
   Precondition: numberUsed <= a.length;
   The first numberUsed indexed variables have values.
   Action: Sorts a so that a[0],a[1],...,a[numberUsed - 1] are in
   increasing order by the compareTo method.
  */
  public static void sort(Comparable[] a, int numberUsed)
  {
    int index, indexOfNextSmallest;
    for (index = 0; index < numberUsed - 1; index++)
    { //Place the correct value in a[index]:
      indexOfNextSmallest = indexOfSmallest(index, a, numberUsed);
      interchange(index,indexOfNextSmallest, a);
      //a[0],a[1],..., a[index] are correctly ordered and these are
      //the smallest of the original array elements. The remaining
      //positions contain the rest of the original array elements.
    }
  }

  /**
   Returns the index of the smallest value among
   a[startIndex], a[startIndex+1], ... a[numberUsed - 1]
  */
  private static int indexOfSmallest(int startIndex,
    Comparable[] a, int numberUsed)
  {
    Comparable min = a[startIndex];
    int indexOfMin = startIndex;
    int index;
    for (index = startIndex + 1; index < numberUsed; index++)
      if (a[index].compareTo(min)<0)//if a[index] is less than min
      {
        min = a[index];
        indexOfMin = index;
        //min is smallest of a[startIndex] through a[index]
      }
    return indexOfMin;
  }

  /**
   Precondition: i and j are legal indices for the array a.
   Postcondition: Values of a[i] and a[j] have been interchanged.
  */
  private static void interchange(int i, int j, Comparable[] a)
  {
    Comparable temp;
    temp = a[i];
    a[i] = a[j];
    a[j] = temp; //original value of a[i]
  }
}
```

Appendix: Question 6 Program

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class ActionFace extends JFrame implements ActionListener
{
    private boolean x;

    public void actionPerformed(ActionEvent e)
    {
        x = true;
        repaint( );
    }

    public static void main(String[] args)
    {
        ActionFace drawing = new ActionFace( );
        drawing.setVisible(true);
    }

    public ActionFace( )
    {
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Hello There!");
        setLayout(new BorderLayout( ));

        JButton actionButton = new JButton("Action");
        actionButton.addActionListener(this);
        add(actionButton, BorderLayout.SOUTH);
        x = false;
    }

    public void paint(Graphics g)
    {
        super.paint(g);
        if (x)
        {
           g.setColor(Color.RED);
           g.drawArc(150, 200, 100, 50, 0, 180);
        }
        else
            g.drawArc(150, 200, 100, 50, 180, 180);
        g.drawOval(100, 100, 200, 200);
        g.fillOval(155, 160, 20, 10);
        g.fillOval(230, 160, 20, 10);
    }
}
```