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

November Exam

Question	Max	Internal	External	Question	Max	Internal	External
1	25			7	25		
2	12						
3	13						
4	10						
5	5						
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 Exam2008 {
   public static void main(String[] args)
    throws PictureException, FileNotFoundException {
      Scanner scan = null;
      PrintWriter pw = null;
      scan = new Scanner(new FileInputStream("fileB.txt"));
      pw = new PrintWriter(new FileOutputStream("fileA.txt"));
      int level = scan.nextInt();
      pw.println(level);
      pw.println(Stack(level,""));
      pw.close();
   }

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

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

   public static String Stack(int n,String offset)
    throws PictureException      {
      if (n<0)
         throw new PictureException("Can't draw a picture of
negative size!");
      else if (n==0)
         return "";
      else
         return Tri(n,offset)+ Stack(n-1,offset+Line(n,' '));

   }
}
```

a) Write a **recursive** definition for the method ***Tri*** in the program listed above. Note that no marks will be awarded for iterative solutions. [4]

```
public static String Tri(int n,String shift) {
```

_____

_____

_____

_____

_____

_____

_____

_____

```
}
```

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

fileA.txt:

```
4
3 2
ABE
```

fileB.txt:

```
3 2
5
```

Now write down the **exact** contents of each of these files **after** the program is run. [4]

fileA.txt:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

fileB.txt:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

c) What changes could you make to the method ***Stack*** to make it infinitely recursive for all input values? [2]

d) Explain clearly and briefly why an iterative binary sort algorithm tends to execute faster than a recursive binary sort algorithm. [2]

e) If you are using recursive binary search to search an array with 15 elements for a key, what will be the maximum possible depth of recursive methods calls (including the original call to the search method)? [2]

f) The program above can throw a java.util.InputMismatchException. Is this a checked or unchecked exception? [1]

g) Rewrite the main method in the program above so that the exceptions PictureException, FileNotFoundException and InputMismatchException are handled separately so the program will not crash if these occur. Sensible, relevant messages must be printed for each of the exception situtations. [4]

```
public static void main(String[] args)  {

        Scanner scan = null;
        PrintWriter pw = null;
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

```
}
```

h) Write a suitable definiton for the class PictureException including all necessary constructors. [4]

```
public class PictureException extends Exception
{
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

```
}
```

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

_____

_____

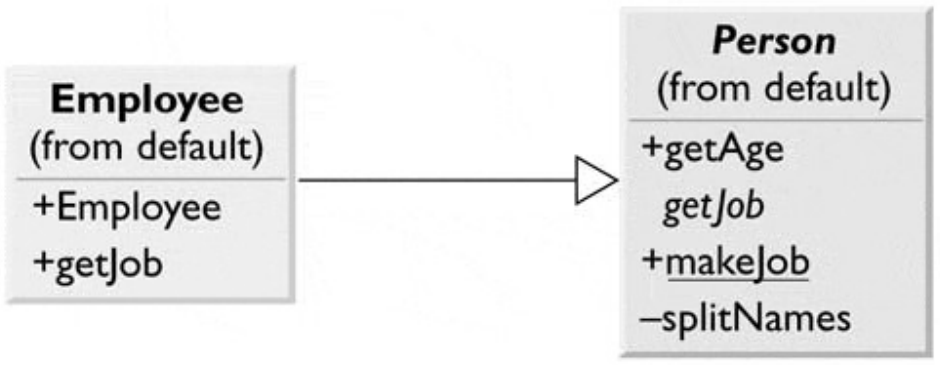
j) Give an example of the use of an anonymous object in the program above. [1]

_____

_____

Question 2: UML, Abstract classes, Inheritance and Polymorphism [12]

Use the following UML diagram to answer the questions that follow.



a) What kind of relationship is there between the classes? [2]

b) State the accessibilities of all members of the classes. [2]

c) Explain the difference between getJob in Employee and getJob in Person. [2]

d) Why do both the classes have the getJob method? What are the benefits of this implementation technique and why (at least 2 reasons)? Write a code example to illustrate at least one of the benefits. [6]

Question 3: Interfaces and Sorting [13]

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] < 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) What is an interface and what is it used for? [2]

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

c) Write a class called Car that implements the Comparable interface. Order is based on car capacity and then model. [4]

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

d) Write a driver program that uses GeneralizedSelectionSort to sort an array of Car objects. [3]

To answer this question, create an array of 3 Car objects, initialize them, and sort the array based on the given GeneralizedSelectionSort. The three objects are: Car(20,10), Car(30,5) and Car(10,15).

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

Question 4: 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;

        //Node constructors

    }//End of Node inner class

    private Node head;

    // LinkedList constructors and methods
    public boolean isEmpty()
    {
    }

    public void clear()
    {
    }
}
```

a) Fill in the method named **isEmpty**, which returns true if the list is empty and false otherwise. [1]

b) Fill in the method named **clear**, which empties the list. [1]

c) What is the role of the Java garbage collector in relation to the **clear** method? [3]

d) Rewrite the partial definition above so that the list can be used to store data of any type, not just Strings. [5]

Question 5: Stacks and Queues [5]

a) Explain what operations are required in a stack and how to implement a stack as a linked list. [2]

b) Explain what a queue is and how to implement it as a linked list. [3]

Question 6: GUIs [10]

Study the following program and answer the questions that follow.

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

public class EventsDemo extends JFrame implements ActionListener,
WindowListener
{
 public static void main(String[] args)
 {
  EventsDemo gui = new EventsDemo();
  gui.setVisible(true);
 }

 public EventsDemo()
 {
  setTitle("Events Demo");
  setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
  setSize(300, 200);
  setLayout(new FlowLayout());
  addWindowListener(this);

  JButtonexitButton = new JButton("Exit");
  exitButton.addActionListener(this);
  add(exitButton);
 }

 public void actionPerformed(ActionEvent e)
 {
  System.exit(0);
 }

 public void windowOpened(WindowEvent e)
 {}
 public void windowClosed(WindowEvent e)
 {}
 public void windowClosing(WindowEvent e)
 {
  System.out.println("Use the Exit button");
 }

 public void windowIconified(WindowEvent e)
 {}
 public void windowDeiconified(WindowEvent e)
 {}

 public void windowActivated(WindowEvent e)
 {}
 public void windowDeactivated(WindowEvent e)
 {}
}
```

a) Draw the GUI that results from running the program. [3]

b) What happens when the button labelled Exit is clicked? [1]

_____

_____

_____

_____

c) What happens when the X button at the top right hand corner is clicked? [1]

_____

_____

_____

_____

d) What is the WindowAdapter class and what advantage does it have over the WindowListener interface? [2]

_____

_____

_____

_____

_____

e) List all the changes that you would make to the program in order to use the WindowAdapter class rather than the WindowListener interface. [3]

_____

_____

_____

_____

_____

Question 7: Ethics, Cyberlaw and Development [25]

a) What are the two basic philosophical approaches to the ethical question "How should I act?" Give a very brief explanation of the difference between them. [3]

b) What would be one possible drawback or problem with using the ethical philosophy of "ubuntu" as a basis for computer ethics? [2]

c) Should software engineers be licensed? Discuss the advantages and drawbacks of turning software engineers into licensed professionals. [5]

d) What is the current position with patenting software in South Africa? [2]

e) What is the Panopticon? How is it used as a metaphor for a possible future? [5]

f) What rights and obligations does the GNU General Public License (GPL) confer? [4]

g) What is the Digital Divide? Explain fully. [4]

Appendix: Question 1 Program

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

public class Exam2008 {
   public static void main(String[] args)
    throws PictureException, FileNotFoundException {
      Scanner scan = null;
      PrintWriter pw = null;
      scan = new Scanner(new FileInputStream("fileB.txt"));
      pw = new PrintWriter(new FileOutputStream("fileA.txt"));
      int level = scan.nextInt();
      pw.println(level);
      pw.println(Stack(level,""));
      pw.close();
   }

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

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

   public static String Stack(int n,String offset)
    throws PictureException      {
      if (n<0)
         throw new PictureException("Can't draw a picture of
negative size!");
      else if (n==0)
         return "";
      else
         return Tri(n,offset)+ Stack(n-1,offset+Line(n,' '));

   }
}
```

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] < 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.*;
import java.awt.event.*;

public class EventsDemo extends JFrame implements ActionListener,
WindowListener
{
 public static void main(String[] args)
 {
  EventsDemo gui = new EventsDemo();
  gui.setVisible(true);
 }

 public EventsDemo()
 {
  setTitle("Events Demo");
  setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
  setSize(300, 200);
  setLayout(new FlowLayout());
  addWindowListener(this);

  JButtonexitButton = new JButton("Exit");
  exitButton.addActionListener(this);
  add(exitButton);
 }

 public void actionPerformed(ActionEvent e)
 {
  System.exit(0);
 }

 public void windowOpened(WindowEvent e)
 {}
 public void windowClosed(WindowEvent e)
 {}
 public void windowClosing(WindowEvent e)
 {
  System.out.println("Use the Exit button");
 }

 public void windowIconified(WindowEvent e)
 {}
 public void windowDeiconified(WindowEvent e)
 {}

 public void windowActivated(WindowEvent e)
 {}
 public void windowDeactivated(WindowEvent e)
 {}
}
```