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) {

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

}
```

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]

File A;

4 2

ABE

[½]

File B:

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

[2 1/2]

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

Remove all the stopping cases – [or any answer valid for all inputs]

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

Iterative solutions do not make use of a stack, while recursive solutions do to keep track of previous recursive calls to the same function. [1] The time required for creating stack frames and pushing them onto the stack impacts on the efficiency of the recursive solution. [1]

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]

5 [2]

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

An unchecked exception. Not specified in the throws clause.

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;

try {
      scan = new Scanner(new FileInputStream("fileB.txt"));
      pw = new PrintWriter(new FileOutputStream("fileA.txt"));
     int level = scan.nextInt();
      pw.println(Stack(level,""));
   }
     catch (PictureException e) {
       System.out.println(e.getMessage());
     }
     catch (FileNotFoundException e) {
        System.out.println("fileB.txt does not exist");
     }
     catch} (InputMismatchException e) {
        System.out.println("fileB.txt should contain a number.");
        }
  finally { pw.close(); }
 }
}
```

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

```
public class PictureException extends Exception
{
public PictureException( )  //[1]
   {
      super("Not a Vowel Exception"); //[1]
   }

   public PictureException(String message) //[1]
   {
      super(message); //[1]
   }
}
```

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

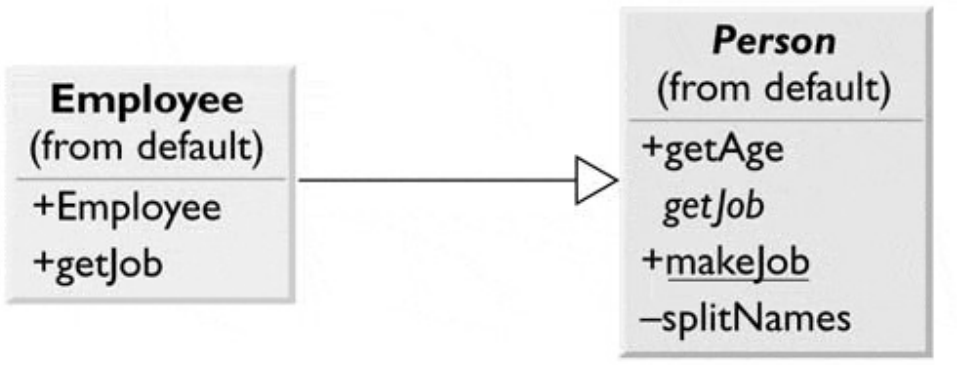
System.out

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

pw = new PrintWriter(new FileOutputStream("fileA.txt")); <- FileOutputStream [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]

It is inheritance/generalization

Person is a super class and Employee is a derived class from Person

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

public: Employee: Employee, getJob; Person: getAge, makeJob

private: Person: splitnames

static: Person: makeJob

abstract: Person: getJob

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

getJob in Person is an abstract method -> Person is an abstract class

whereas getJob in Employee is an inherited and implemented/concrete method of the getJob of Person

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]

This is the inheritance technique. One of the main benefits is Polymorphism.

Other is that we can postpone the definition of the abstract method until later in derived class (a powerful tool when combined with polymorphism)

For example one can call getJob of Person without knowing how they are implemented in the subclasses as this will be decided at run time. Also the way implementing the abstract method getJob of Person in subclasses can be various. This allows code flexibility and re-usage in Java.

Person persons[] = new Person[2];

```
persons[0] = Employee();
persons[1] = Employer();// subclass of Person
for (int i = 0; i < persons.length; i++) {
   persons[i].getJob();
}
```

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]

An interface specifies a set of methods that any class that implements the interface must have. An interface is a contract between a class and the outside world. When a class implements an interface, it promises to provide the behavior published by that interface.

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 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.

```
public class Car implements Comparable
{
 private int capacity = 0;
 private int model = 0;
 public Car(int capacity_, model_) {
  capacity = capacity_;
  model = model_;
 }
 public int compareTo(Car car_) {
  if (this.capacity > car_.capacity)
   return 1;
  else if (this.capacity < car_.capacity)
   return -1;
  else
   if (this.model > car_.model)
    return 1;
   else if (this.model < car_.model)
    return -1;
   else return 0;
 }
}
```

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).

```
public class DriverProgram {
 public static void main(String[] args) {
  Car cars[] = new Car[3];
  cars[0] = new Car(20,10);
  cars[1] = new Car(30,5);
  cars[2] = new Car(20,15);
  GeneralizedSelection.sort(cars, cars.length);
 }
}
```

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

After sorting the array, the order of objects should be:

Car(20,10), Car(20,15), Car(30,5)

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]

```
public boolean isEmpty()
{
 return (head==null);
}
```

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

```
public void clear()
{
 head = null;
}
```

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

The clear method removes any reference to all the nodes. The program can then no longer access these nodes. The garbage collector automatically removes the nodes and recycles the memory they occupied.

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

```
public class LinkedList<T>
{
   private class <T>Node
   {
      private T data;
      private Node<T> next;

      //Node constructors

   }//End of Node inner class

   private Node<T> head;
```

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]

A stack is a LIFO structure where the last item that was added is the first one that can be retrieved first. It can be implemented as a linked list by making the addToStart (or just add) method a push operation and the deleteHead (or just delete) method a pop operation.

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

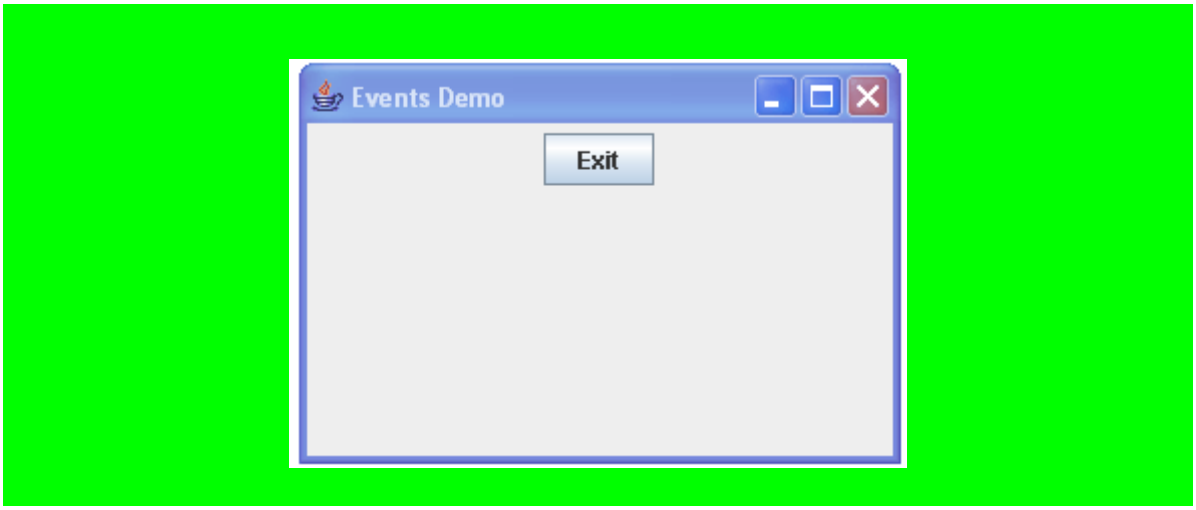
A queue is a FIFO structure where the first item to be added is the first one retrieved. Items are added to the back and removed from the front. The simple linked list must be modified to have a pointer to the back of the list so that items can be added there. The operations for the queue are enqueue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

The program ends.

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

The message "Use the Exit button" appears on the console.

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

It is an abstract class which is Java's own implementation of the WindowListener interface. Using it means that one need only implement the methods required by the program.

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]

Create an inner class which extends the WindowAdapter class. Have this inner class override the windowClosing method. Use an object of this inner class as the argument to the addWindowListener call.

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]

(A) What are the consequences? (teleological)? [1] vs (B) What is my duty? (deontological). [1] So the difference is that one considers outcomes and the other the rules governing action. [1]

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

ubuntu is based on transparency and communal interest in each other. Thus there is a (potential) conflict with notions of privacy

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

Advantages:

a) Qualified people more likely to do a good job.

b) Assured that an appropriate methodology is used.

c) Assured that licensee has appropriate experience (you'll have to be apprenticed before getting

license).

Drawbacks:

a) Competent people, without degrees or accreditation can be excluded from practicing.

b) Small projects - less need for professionals, licensing would be overkill.

Anything else that is relevant.

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

It cannot be done! [1] However a system that incorporates software can be patented.[2] (or any other useful observation)

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

1) Guard can see prisoners but not vica versa [1]

2) Prisoners know that they are being observed [1]

3) This affects their behaviour [1]

As a metaphor: [2 marks here]

1) Observation is considered an infringement of privacy (for a normal person in society who is not a convict).

2) Data gathering from individual transactions & the use of this information by Companies hoping to benefit from it is an invasion of individual privacy.

3) again as a means of controlling behaviour

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

(1) freely copy and distribute copies of source code and software, only with the license,[2]
(2) modify code, but those changes must be clear and made available with same license [2]

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

The term Digital Divide refers to the major disparities in the penetration of the Information Society in the developing world [1]. Digital Divide is the growing gap that exists between those who have access to the Information Society and those who are deprived of such access [1] due to cultural bias in the applications and contents, gaps in their education (for example, illiteracy), personal handicap, poor digital infrastructure, or lack of appropriate computer equipment [2 for some of these]

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)
 {}
}
```