DUBLIN CITY UNIVERSITY

SEMESTER TWO
EXAMINATION 2001-2002

COURSE: B.Sc. in Applied Computational Linguistics
B.Sc. in Computer Applications (Day)
B.Sc. in Computer Applications (Evening)
B.Sc. in Mathematical Sciences
B.Sc. in Financial and Actuarial Mathematics

YEAR: MS1, FM1, CA1, CA2, CAE1, CAE2, CL1 CAE3

MODULE: CA105/CA166

(Title and Code) Algorithms & Data Structures 1
Computer Programming II

EXAMINERS: Prof. J. M. Morris Ext: 8419
Prof. G. Lyons
Prof. D. O' Maidin

TIME ALLOWED: 2 Hours

INSTRUCTIONS: Answer all questions.
There are 8 parts in Question 1, each worth 6 marks.
Questions 2 and 3 are each worth 26 marks.
A Java Reference is provided at the end of the paper.

Requirements for this paper
Please tick (X) as appropriate

- ☐ ***Log Table***
- ☐ ***Graph Paper***
- ☐ ***Attached Answer Sheet***
- ☐ ***Statistical Tables***
- ☐ ***Floppy Disk***

PLEASE DO NOT TURN OVER THIS PAGE UNTIL ASKED TO DO SO

Question 1

(a) Paying particular attention to the return types, which of the following methods should give rise to an error message when compiled? You are not asked to explain your reasoning – just write the names of the erroneous methods.

```
void ff(double d) {
      if (d>0) return d; else return -d;
}

int gg(int j, int k) {
      if (j>k) System.out.print(j);
      else System.out.print(k);
}

boolean hh(int k) {
      if (k>0) {
            return true;
      }
      else if (k=0) {
            return false;
      }
      else {
            System.out.print("Error: negative received.");
      }
}
```

(b) Which of the four assignment statements labelled (i) to (iv) below violate Java's rules on access to public and private components? You are not asked to explain your reasoning.

```
class Tuple {
      int x;
      private int y;

      void f(Tuple p) {
            x=p.x; // (i)
            x=p.y; // (ii)
      }
}

class Test {
      public static void main(String[] args) {
            Tuple r = new Tuple();
            r.x=1; // (iii)
            r.y=2; // (iv)
      }
}
```

(c) Complete the code of method `first` below.

```
class Name {
      String forename; String surname;
      // forename & surname of a person


      boolean first(Name p) {
            // Does my name precede or equal p by the
            // usual ordering of names, i.e.
            // alphabetically by surname, with ties
            // resolved alphabetically by forename?
            ..........
      }

}
```

(d) For p a variable of type `Pair` as defined below, write down the values of `p.x` and `p.y` after each of the following assignments, or state that the assignment is illegal.

(i) `p = new Pair(23.1);`
(ii) `p = new Pair(7, 6.0);`
(iii) `p = new Pair();`

```
class Pair {
      int x = 3; double y;

      Pair(double yy) {y = yy;}

      Pair(int xx, double yy) {x = xx; y = yy;}
}
```

(e) Write a program `Max` which displays the maximum in a collection of integers, the integers being passed *as command line arguments*. For example, the command

```
java Max 6 7 3 5 3
```

should give rise to the output 7. You may assume that at least one integer is supplied.

(f) Suppose you are offered two methods P and Q each of which "squashes" an array (you do not need to know what squashing an array actually means). The methods behave identically but have different execution times. The running time of method P is $A+Bn$ microseconds, and the running time of method Q is $C+Dn+n^2$ microseconds, where n denotes the size of the array, and A, B, C, and D denote constants.

(i) If it takes method P 2000 microseconds to squash an array of length 250 on a particular machine, roughly how long would you expect it to take to squash an array of length 500?
(ii) As for Part (i), but using method Q instead of method P.
(iii) Write down the time complexity of method P in O-notation.
(iv) Write down the time complexity of method Q in O-notation.
(v) Which of P and Q is preferable in general?
(vi) If you wish to squash an array of size 25 in the fastest time, which of P and Q would you choose to use, given the following values for the constants: A=100, B=25, C=10, and D=2.

(g) Complete program `MakeFile` below which creates a *text* file (*not* a binary file) containing the integers from 0 to 99, inclusive, in that order at one integer per line. The name of the file is passed as a command line argument. Note that there is no need for any input to the program from the keyboard. Use class `PrintWriter`.

```
class MakeFile {
    public static void main(String[] args) {
        try {
            .......
        } catch(IOException e) {
            e.printStackTrace();
        }
    }
}
```

(h) Complete the code of the constructor and methods in class `Date` below.

```
class Date {

      private int day, month, year; // a date

      Date(int d, int m, int y) {
      // A Date object with day d, month m, and year y
      // (Assume d, m, and y are valid)
            ........
      }

      Date newYear() {
      // Return a Date object representing next New Year's day,
      // i.e. January 1st in the year following the year
      // represented by this Date object.
            ........
      }

      boolean laterYear(Date dt) {
      // Does dt represent a later year than this year?
            ........
      }

}
```

Question 2

(a) Write a class called `Item` to represent an item in a supplier's stock. Each item has a name (which is a single word), a stock level (an integer), and a price (a double). Include the following methods:

(i) A method `get()` which reads an item from a single line of input consisting of the item's name, the stock level, and the price in that order. A typical line looks like:

```
toaster  7  23.50
```

(ii) A method `put()` which prints the items's name.

(iii) A method `buyIn(n)` which increases the stock level for this item by n units (assume n>0).

(iv) A method `sell(n)` which reduces the stock level for this item by n items (assume n>0). If the stock level is less than n initially, then no action is taken. Return a boolean value to indicate whether or not the stock was reduced.

(v) A method `value()` which returns the total value of the stock for this item (i.e. price times stock level).

The class should have no other methods or constructors.

(b) Write a program which reads a non-empty sequence of stock items, one per line as described in part(a). The program should print the name of the item which has the greatest stock value (i.e. stock level times price). For example, if the input is

```
toaster  7  23.50
kettle  5  38.75
microwave  2  87.00
```

then the output is

```
kettle
```

Your program should consist of a single additional class whose only component is method `main()`, and it should make good use of class `Item` exactly as you have written it in part (a).

Question 3

(a) Complete methods `put` and `isMatureFemale` in the following class definition. Method `put` should write a single record containing a student's details to a (binary) file. Writing occurs at an offset in the file determined by the current value of the file pointer. You may assume that the file pointer does indeed point to a valid record in the file, or is at the end of the file. Method `get`, which reads in the employee's details as written by `put`, is supplied as a guide (note in particular that names are padded to a constant length).

```
class Student {

    private String name; // student's name, max length nameSize
    private int age; // student's age
    private boolean male; // student's sex

    private final static int nameSize = 30;

    Student() {
    }

    Student(String n, int a, boolean s) {
        name = n; age = a; male = s;
    }

    void put(DataOutputStream fout) {
    // write attributes to file
            .........
    }

    boolean get(DataInputStream fin) {
    // read attributes from file, return true iff successful
        try {
            if (fin.available()== 0) return false;
            char[] b = new char[nameSize];
            for (int i=0; i<nameSize; i++)
                b[i] = fin.readChar();
            name = (String.valueOf(b)).trim();
            age = fin.readInt();
            male = fin.readBoolean();
            return true;
        } catch (IOException e) {
            e.printStackTrace();
            return false;
        }
    }

    boolean isMatureFemale() {
    // is student 23 years of age or older and female?
            .........
    }

    void putName() {
    // print name
        System.out.println(name);
    }

}
```

(b) Given a (binary) file of students called `StudentFile` as described in part (a), write a program which prints the names of all mature female students (i.e. female students aged 23 years or older). Your program should consist of a single additional class whose only component is method `main()`, and it should make good use, without modification, of class `Student` as given in part (a).

Not listed under individual classes:
int compareTo(Object)
boolean equals(Object)
String toString()

ArrayList

ArrayList()
ArrayList(Set)
boolean add(Object)
boolean addAll(Collection)
void clear()
boolean contains(Object)
boolean containsAll(Collection)
boolean isEmpty()
boolean remove(Object)
boolean removeAll(Collection)
boolean retainAll(Collection)
int size()
Iterator iterator()
void add(int, Object)
Object get(int)
int indexOf(Object)
Object remove(int)
Object set(int, Object)

Boolean

Boolean(boolean)
boolean booleanValue()

BufferedReader

BufferedReader(FileReader)
String readLine() throws IOException
void close() throws IOException

Character

Character(char)
char charValue()
static boolean isDigit(char)
static boolean isLetter(char)
static boolean isLetterOrDigit(char)
static boolean isLowerCase(char)
static boolean isUpperCase(char)
static boolean isWhitespace(char)
static char toLowerCase(char)
static char toUpperCase(char)

Collections

static void sort(List)

Console

static int readInt()
static boolean readBoolean()
static double readDouble()
static char readChar()
static String readString()
static void skipLine()
static String readToken()
static boolean endOfFile()
static boolean EndOfFile()
static int available()
static boolean hasMoreTokens()
static void skipWhitespace()
static char peekChar()

ConsoleReader

TextInput(String)
int readInt()
boolean readBoolean()
double readDouble()
char readChar()
String readString()
void skipLine()
String readToken()
boolean endOfFile()
boolean EndOfFile()
int available()
boolean hasMoreTokens()
void skipWhitespace()
char peekChar()
void close()

DataInputStream

DataInputStream(FileInputStream)
int readInt() throws IOException
long readLong() throws IOException
boolean readBoolean() throws IOException
char readChar() throws IOException
double readDouble() throws IOException
float readFloat() throws IOException
int available() throws IOException
void close() throws IOException

DataOutputStream

DataOutputStream(FileOutputStream)
void writeBoolean(boolean) throws IOException
void writeInt(int) throws IOException
void writeLong(long) throws IOException
void writeDouble(double) throws IOException
void writeFloat(float) throws IOException
void writeChars(String) throws IOException
void writeChar(int) throws IOException
void close() throws IOException
void flush() throws IOException
int size()

Double

Double(double)
double doubleValue();
static double parseDouble(String)
static String toString(double)

Exception

void printStackTrace()

File

File(String)
boolean exists()
boolean isFile()
boolean canRead()
boolean canWrite()
boolean delete()
boolean renameTo(File)

FileInputStream

FileInputStream(String) throws
FileNotFoundException

FileOutputStream

FileOutputStream(String) throws IOException
FileOutputStream(String, boolean) throws
IOException

FileReader

FileReader(String) throws
FileNotFoundException

FileWriter

FileWriter(String) throws IOException
FileWriter(String, Boolean) throws
IOException

Float

Float(float)
float floatValue();
static float parseFloat(String)
static String toString(float)

GregorianCalendar

GregorianCalendar()
int get(int)
static int Calendar.DAY
static int Calendar.MONTH
static int Calendar.YEAR

HashMap

HashMap()
void clear()
Object put(Object, Object)
Object get(Object)
Object remove(Object)
boolean containsKey(Object)
int size()
boolean isEmpty()
Set keySet()

HashSet

HashSet()
boolean add(Object)
boolean addAll(Set)
void clear()
boolean contains(Object)
boolean containsAll(Set)
boolean isEmpty()
boolean remove(Object)
boolean removeAll(Set)
boolean retainAll(Set)
int size()
Iterator iterator()

Integer

Integer(int)
int intValue()
static int parseInt(String)
static String toString(int)

Iterator

Object next()
boolean hasNext()

Keyboard

Same as Console

LinkedList

LinkedList()
LinkedList(Set)
void addFirst(Object)
Object getFirst()
Object getLast()
Object removeFirst()
other methods as ArrayList

Long

Long(long)
long longValue()
static long parseLong(String)
static String toString(long)

Math

static double random()
static double abs(double)
static int abs(int) etc.
static double ceil(double)
static double floor(double)
static int max(int, int) etc.
static int min(int, int) etc.
static double rint(double)
static long round(double)
static int round(float)
static double sqrt(double)

PrintWriter

PrintWriter(FileWriter)
PrintWriter(FileWriter, boolean)
void print(String)
void println(String)
void close()
void flush()

RandomAccessFile

RandomAccessFile(String, String) throws
IOException
long length() throws IOException
void seek(long) throws IOException
int readInt() throws IOException
long readLong() throws IOException
boolean readBoolean() throws IOException
char readChar() throws IOException
double readDouble() throws IOException

float readFloat() throws IOException
void writeBoolean(boolean) throws
    IOException
void writeInt(int) throws IOException
void writeLong(long) throws IOException
void writeDouble(double) throws IOException
void writeFloat(float) throws IOException
void writeChars(String) throws IOException
void writeChar(int) throws IOException
void close() throws IOException

String

int length()
boolean startsWith(String)
boolean startsWith(String)
boolean endsWith(String)
int indexOf(String)
String substring(int, int)
String substring(int)
char charAt(int)
String toUpperCase()
String toLowerCase()
String trim()
static String valueOf(int)
static String valueOf(long)
static String valueOf(double)
static String valueOf(float)
static String valueOf(boolean)
static String valueOf(char)
static String valueOf(char[], int, int)
static String valueOf(char[])
boolean equalsIgnoreCase(String)
void getChars(int, int, char[], int)

System

static long currentTimeMillis()
static void exit(int)

TextInput

as ConsoleReader

TreeSet

TreeSet()
methods as HashSet

TreeMap

TreeMap()
methods as HashMap()