UNIVERSITY COLLEGE LONDON

University of London

EXAMINATION FOR INTERNAL STUDENTS

For the following qualifications:-

B.Sc.

Information Studs. B2: Programming 2

COURSE CODE	:	**INSTB002**
DATE	:	**29-Apr-05**
TIME	:	**14:30**
TIME ALLOWED	:	**2 hours 30 minutes**

2005-C764-3-60

TURN OVER

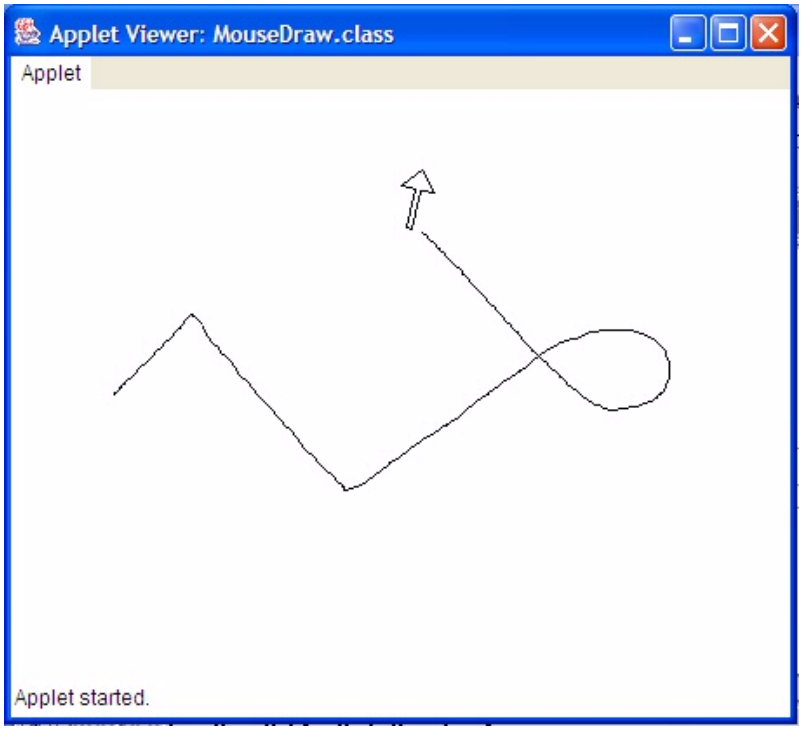
INSTB002: Programming 2

Answer any **THREE** questions, including question 1 or question 2 (or both).

Marks for each part of each question are indicated in square brackets.

Calculators are not permitted.

1. Use Java to write an *applet* in which, when the mouse pointer is dragged, it leaves a line that traces (draws) its progress (see example below).



Your answer need only consider one dragged journey by the mouse pointer during the applet lifetime (as shown above). There should be no limits on the distance travelled or the directions taken during the pointer's dragged journey within the applet panel. You may assume an applet size of 800 x 800 pixels and the availability of the following Java libraries:

```
java.awt.*                  java.awt.event.*
javax.swing.*               java.util.ArrayList
```

The `MouseMotionListener` interface should be employed appropriately and your answer should include a full design and properly commented Java code.

[Total 33 Marks]

2a. Use Java to create a class called `meetings` that is able to store the details of meetings, scheduled to take place within a university department, at the *nodes* of a *linked list*. Each node in the list should be able to store:

- the surname of person booking the meeting (`String`)
- the time of meeting (`int` between 9 & 17)
- the day of meeting (`int` between 1 & 31)
- the month of meeting (`int` between 1 & 12)
- the meeting location (room number) (`String`)

Both the list and node classes should possess an appropriate constructor to initialise the list and the information stored at each node of the list respectively. The `meetings` (list) class should possess an `add` method that adds new nodes to the end of the list.

[18 Marks]

2b. The method `compareToIgnoreCase` is available to objects of the `String` class. When called by one string with a second string as its argument, `compareToIgnoreCase` returns an integer < 0 if the first string is alphabetically lower than the second, > 0 if the first string is alphabetically higher than the second and 0 if the two strings are alphabetically equal. Upper and lower case characters are treated equally. For example,

```
String string1 = "question", string2 = "exam";
int number = string1.compareToIgnoreCase(string2);
```

gives the integer `number` a value > 0, because the word "question" is alphabetically higher than the word "exam".

Adapt the `add` method of your `meetings` class to add each new meeting to the list in a position that maintains the booker surnames in alphabetical order **without** sorting the list contents. Your code should protect against double bookings, i.e. it should not allow meetings to be scheduled at the same time on the same date in the same room.

[15 Marks]
[Total 33 Marks]

3a. Provide a detailed comparison of the *SelectionSort*, *BubbleSort*, and *QuickSort* list sorting algorithms when used to sort a 1-dimensional array of integers. Your answer should be illustrated with example Java code and consider algorithmic *implementation* and *performance*.

[25 Marks]

3b. How many passes through the following array of characters

{I, L, O, V, E, J, A, V, A}

would the *BubbleSort* algorithm require in order to sort the array contents into ascending alphabetical order? Demonstrate how you have arrived at your answer.

[8 Marks]
[Total 33 Marks]

4. Write an essay on the subject of *file I/O* and *exception handling* when creating and reading **text files** with Java. Your answer must include detailed illustrative examples written in Java and cover the following topics:

- I/O *streams* (data and process streams);
- the manner in which *exception handling* is implemented;
- the benefits provided by its employment, and
- its use during file I/O operations.

[Total 33 Marks]

5a. Briefly describe each of the following features of Java and explain how they are used, including illustrative examples of Java code: *try / catch clauses*, *indirect recursion*, *abstract classes*, *inheritance*, and the `JTabbedPane` containment structure.

[25 Marks]

5b. Explain (using example pseudo-code) the use and output of the `printStackTrace()` method of the `Exception` class during exception handling in Java.

[8 Marks]
[Total 33 Marks]

[END OF PAPER]