Notes for the Student: This is the first of four courseworks which are designed to give you practical experience in composing simple Java programs. The philosophy behind these courseworks is the following: we learn by imitating others, and then by attempting to improve on what we have imitated. The programming assignments get progressively more challenging, while never becoming really difficult. Even though these programs are (relatively) simple, you should proceed to write them carefully, documenting them as you go, and using everything you are being taught about good program design.

Background: You will need to have access to a Java compiler, and know how to enter and run Java programs under your system. You will need access to sources of information on Java programming: the Study Guide, textbooks, web sites, and – ideally – someone who knows Java.

Concepts covered: compilation; importing; classes; objects; methods; data; variables; data types; operators; arguments/parameters; control structures (selection and iteration); input and output; initialization of variables.

A. Programming Theory

1. Basic ideas. Examine the following program, type it in, and run it. Then answer the questions which follow.

```java
/* A simple example program*/

import java.io.*;

class ASimpleExample
{
    public static void main(String[ ] args)
    {
        System.out.println ("Hello, World!");
    }
}
```

1(a) – Why did we put the phrase 'A simple example program' between the delimiters '/*' and '*/'? What is the effect of enclosing anything between these delimiters?
1(b) – What does 'import java.io.*' mean? What are we 'importing'? If we had not imported java.io, what would have happened when we tried to run our program?
1(c) – What is the purpose of the asterisk ('*') after 'io'?
1(d) – What is a 'class' in Java? What is the distinction between a 'class' and an 'object'?
1(e) – What does 'public' mean in the program above?
1(f) – What does 'static' mean in the program above?
1(g) – What does 'void' mean in the program above?
1(h) – What does 'main' mean in the program above?
1(i) – Why do we have a semi-colon after the "Hello, World!" line?
1(j) – There is one 'method' in the program above. What is its name?

2. Data Types.

We have to be aware of the 'type' of data that we deal with in Java. Make up a table listing the various types of data which Java recognizes. The first type has been done for you.

<table>
<thead>
<tr>
<th>Whole (counting) numbers:</th>
<th>Name</th>
<th>Bytes of Storage taken up</th>
<th>Range</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 int</td>
<td>4</td>
<td>-2,147,483,648 .. +2,147,483,647</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2.3

Fractions (decimal numbers)

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes of Storage taken up</th>
<th>Number of significant digits</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.4

Single Characters

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes of Storage taken up</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.5

More than one character

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes of Storage taken up</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.6

'Logical outcomes': true or false

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes of Storage taken up</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. Variables. Data in Java is held in 'variables'.

3.1 What do we mean by a 'variable' in a Java program?
3.2 Each variable in a Java program can only hold data of a certain type. How do we tell the computer what type of data a variable can hold?
3.3 Which of the following names would be allowed for a variable? (If you're not sure, try using each of these names in a Java program, to see if the program will compile.)
   (a) myBankAccount
   (b) my Bank Account
   (c) foreign
   (d) native

4. Operators. Data can be manipulated (changed, or moved around) by the use of operators.

4.1 Assignment -- Give an example of the use of the assignment operator.
4.2 Arithmetic -- Give an example of the use of each of the five arithmetic operators.
4.3 Relational -- List the six relational operators and give an example of the use of each one.
4.4 Logical (boolean) -- List two logical (boolean) operators and give an example of the use of each one.

5. Control Structures.

5.1 Making Choices
5.1.1 Give an example of how the IF { } ELSE { } statement could be used in a Java program.
5.1.2 Give an example of how the SWITCH { } statement could be used in a Java program.

5.2 Doing something repeatedly
5.2.1 Give an example of how the WHILE ( ) { } statement could be used in a Java program.
5.2.2 Give an example of how the DO { } WHILE ( ) could be used in a Java program.
5.2.3 Give an example of how the FOR { } could be used in a Java program.
B. Programming Practice

SPECIFICATION

A military Training Organization needs a simple program for holding the results of its trainees' studies on the various Instructional Modules it offers.

Write a program in Java which can do the following:

INPUT
(1) Read in from the keyboard a trainee’s Trainee Number (an integer between 1000 and 9999).
(2) Read in from the keyboard a trainee's first and family names.
(3) Read in the title of the trainee’s current Instruction Module.
(4) Read in the number of quizzes the trainee wants to enter his or her marks for: this will always be a whole number (an integer) between 1 and 8.
(5) For each quiz, read in the mark, (these may be decimal – ‘real’ numbers).
(6) Read in the Module Test mark.

OUTPUT
The program should print out the trainee’s details, as entered, and then

(a) Calculate and display the trainee’s average (mean) quiz mark.
(b) Display the trainee’s Module Test mark.
(c) Calculate and display the trainee’s overall mark: this is based on taking 20% of the average quiz mark and adding it to 80% of the Module Test mark.
(d) Calculate a letter grade for the trainee, on the basis that 70 or above is an A, 60 up to 70 is a B, 50 up to 60 is a C, 40 up to 50 is a D, and 35 up to 40 is an E.

Displayed marks should be shown with one decimal point of precision.

An example of a typical run of your program might look like this, with user's input in bold:

Trainee Number: 5827
Family Name: Thaksin
First Name: Parvit
Instruction Module: Forward-Scanning Radar Display Repair
How many quizzes shall I calculate?: 3
Module Quiz mark 1: 68
Module Quiz mark 2: 76
Module Quiz mark 3: 87
Final Module Test mark: 77

Results for Parvit Thaksin
Trainee Number 5827
Module: Forward-Scanning Radar Display Repair
Average Module Quiz: 70.3 Final Module Test: 77 Overall Mark: 75.7
Letter Grade: A

Please note: this is NOT how a professional program should do input and output! Real programs (the kind you will be writing later) will use a much more friendly form of input. This is just to get started. You may use any form of input and output that you want.
NOTES ON PROGRAMMING

(1) As with all serious programming, build your program up a step at a time, compiling and testing as you go. With this program, a sensible process would be: get a program that can read in and then print out all the data except the quiz and final test marks. When that is working, add the loop for reading in these and building up the totals. Then add the code for printing out the averages.

(2) For this assignment, you don't have to worry about whether user input is correct or sensible. In other words, don't bother to check user input. (In future, we will worry about this.)

(3) In order to write this program, you will need to know about variables, basic data types, loops, simple arithmetic operations, input and output statements and assignment statements.

(4) Do not use any code generators or database packages to write this program. It should not take up much more than one or two pages.

WHAT TO TURN IN

For Part A: Answers to Questions 1-5.
For Part B:
(1) A print-out of the program code for the program
(2) Screen dump, or printouts of an external text file, of the program's output, shown processing inputs for one trainee.
(3) A one-page description of how you tested the program to make sure that it worked. This should include a table showing various combinations of inputs with expected, and actual, outputs, plus a brief discussion of why you chose the particular values in the table. (HINT: make sure your program works with various combinations of extreme values. For example, it should give a correct output for a trainee with perfect scores, a trainee with medium stores, a trainee with zero for everything, a trainee with zero for the quizzes but one hundred percent for the final Module Test, and so on. You don't have to go over the top on this, but you should test your program with about eight or ten different combinations of values.)
(4) A one-page description of how your program works. Make reference to the different variables and control structures (FOR, etc) you used. You may wish to number the different sections of your program (using the commenting delimiters) and refer to "Section 1" and so on. Be sure you identify the classes and methods that your program uses.

Weighting:
<table>
<thead>
<tr>
<th>Part A: Questions 1-5:</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part B: Program:</td>
<td>30%</td>
</tr>
<tr>
<td>Testing Description:</td>
<td>20%</td>
</tr>
<tr>
<td>Program Description:</td>
<td>20%</td>
</tr>
</tbody>
</table>

END
UNIVERSITY OF LONDON

DIPLOMA AND BSc IN COMPUTING AND RELATED SUBJECTS
FOR INTERNATIONAL PROGRAMMES STUDENTS

COURSEWORK SUBMISSION FORM

IMPORTANT, PLEASE NOTE:

You must complete this form in full and attach it to the coursework that is to be submitted to the University.

Full name: ........................................................................................................................................................................
(as it appears on your Registration Form)

Student number: ................................................................................................................................................................

Unit title: ...........................................................................................................................................................................

Unit number: ....................................................................................................................................................................

Assignment number: ..........................................................................................................................................................

DECLARATION

I declare that:

• I understand what is meant by plagiarism.
• I understand the implications of plagiarism.
• This assignment is all my own work and I have acknowledged any use of the published or unpublished works of other people.

Signature ………………………………………………… Date ……………………

COURSEWORK SUBMISSION CHECKLIST

Before you submit your coursework to the University, have you:

☐ completed the title, unit number and assignment number on the submission form?

☐ signed and dated the Declaration?

☐ put your name and student number on every page?

☐ put the unit number, unit name and assignment number on every page?

☐ used more than one page? If so, have you securely stapled or tied the pages together?

☐ used a disk? If so, have you used a separate disk for each assignment? (Note: you should only use a disk if it is explicitly asked for in the assignment).

☐ labelled clearly and securely attached any disks?

☐ attached all parts of the coursework required (including the submission form)?

If you fail to do these things your coursework may not be accepted.

Please send all coursework to:
The Registration and Learning Resources Office
Room STG13
Stewart House
University of London
32 Russell Square
London WC1B 5DN
United Kingdom
Tel: +44 (0)20 7862 8326
Fax: +44 (0)20 7862 8329
Notes for the Student: This assignment is an extension of Coursework 01. You can use the program you wrote for that coursework as the starting point for this one.

Concepts covered: Storing and accessing multiple instances of the same data type.

SPECIFICATION
This program is an extension of the one you wrote for CW01. You will need to make the following changes to this program:

1. Rather than having the trainee type in the full title of his Training Module, let him choose from a menu: the choices should be: “Forward Scanning Radar Display Repair”, “IR Beacon Maintenance” and “SAM Detection System Replacement”. I have not shown the menu output and user input in the example. Your program should display a menu with these three choices.
2. Allow more than one trainee’s marks to be entered, and then print the results for each trainee at the end of the program, rather than after each trainee’s marks have been entered.
3. Terminate entry by entering a special code.
4. Do not allow marks below 0, or above 100, to be entered.

An example of the program being run. Marks for two trainees are entered. Erroneous entries, or entry for a trainee of coursework marks for a coursework already entered, are not shown here, but should be tested for.

Trainee Number: 5827
Family Name: Thaksin
First Name: Parvit
[Instruction Module chosen from menu: chooses Forward Scanning Radar Display]
How many quizzes shall I calculate?: 3
Module Quiz mark 1: 68
Module Quiz mark 2: 76
Module Quiz mark 3: 87
Final Module Test mark: 77
Another [Y/N]: Y
Trainee Number: 6202
Family Name: Lundberg
First Name: Claes
[Instruction Module chosen from menu: chooses IR Beacon Maintenance]
How many quizzes shall I calculate?: 5
Module Quiz mark 1: 64
Module Quiz mark 2: 72
Module Quiz mark 3: 76
Module Quiz mark 4: 68
Module Quiz mark 5: 84
Final Module Test mark: 62
Another [Y/N]: N

Results for Parvit Thaksin
Trainee Number 5827
Module: Forward-Scanning Radar Display Repair
Average Module Quiz: 70.3 Final Module Test: 77 Overall Mark: 75.7
Letter Grade: A

Results for Claes Lundberg
Trainee Number 6202
Module: IR Beacon Maintenance
Average Module Quiz: 72.8 Final Module Test: 62 Overall Mark: 64.0
Letter Grade: B
NOTES ON PROGRAMMING

(1) The most important thing about this coursework is that we are storing lots of instances of the same thing (module titles and trainee data). Thus we will need to use a Java construct like arrays, or vectors. You may assume that not more than twenty trainees will have their information entered into the program.

(2) Of course, you will not want to begin all over again in writing this coursework. Start with the program you wrote for Coursework 01, and modify it.

(3) And, as usual, go a step at a time! Make the smallest change to the old program that you can, then compile and run it. Then make another small change, and so on. Don't try to make all the changes at once!

(4) At this stage, we are still not very concerned with input and output. You will not be marked on having a good interface. In fact the method of entering data here is awful! In real life you would adapt various Windows-oriented “ready-made” methods for data entry. However, the point of this coursework is for you to learn how to program. Thus, we will use these ugly, but instructional, methods.

WHAT TO TURN IN

(1) A print-out of the program code for the program

(2) Screen dump, or printouts of an external text file, of the program's output, shown processing inputs for four different test cases. You must choose the most useful cases to test. At least one of the test cases should show entry for four different trainees, and all three degree programmes.

(3) A one-page description of how you tested the program to make sure that it worked. This should include a table showing various combinations of inputs with expected, and actual, outputs, plus a brief discussion of why you chose the particular values in the table. (HINT: make sure your program works with various combinations of extreme values.)

(4) A two-page description of how your program works. Make reference to the different variables and control structures (IF ELSE, FOR, etc) you used, and any classes and associated methods you have introduced. You may wish to number the different sections of your program (using the commenting delimiters) and refer to "Section 1" and so on.

Weighting: Program: 50% Testing Description: 30% Program Description: 20%

END
UNIVERSITY OF LONDON

DIPLOMA AND BSc IN COMPUTING AND RELATED SUBJECTS
FOR INTERNATIONAL PROGRAMMES STUDENTS

COURSEWORK SUBMISSION FORM

IMPORTANT, PLEASE NOTE:

You must complete this form in full and attach it to the coursework that is to be submitted to the University.

Full name: ........................................................................................................................................................................
(as it appears on your Registration Form)

Student number:................................................................................................................................................................

Unit title: ..........................................................................................................................................................................

Unit number: ....................................................................................................................................................................

Assignment number:..........................................................................................................................................................

DECLARATION

I declare that:

• I understand what is meant by plagiarism.
• I understand the implications of plagiarism.
• This assignment is all my own work and I have acknowledged any use of the published or unpublished works of other people.

Signature ................................................................. Date .................................
COURSEWORK SUBMISSION CHECKLIST

Before you submit your coursework to the University, have you:

☐ completed the title, unit number and assignment number on the submission form?

☐ signed and dated the Declaration?

☐ put your name and student number on every page?

☐ put the unit number, unit name and assignment number on every page?

☐ used more than one page? If so, have you securely stapled or tied the pages together?

☐ used a disk? If so, have you used a separate disk for each assignment? (Note: you should only use a disk if it is explicitly asked for in the assignment).

☐ labelled clearly and securely attached any disks?

☐ attached all parts of the coursework required (including the submission form)?

If you fail to do these things your coursework may not be accepted.

Please send all coursework to:
The Registration and Learning Resources Office
Room STG13
Stewart House
University of London
32 Russell Square
London WC1B 5DN
United Kingdom
Tel: +44 (0)20 7862 8326
Fax: +44 (0)20 7862 8329
Notes for the Student: This coursework is a modification of the previous one. Don't try to make all the changes at once! Find the smallest thing you can change in the old program, and change that. Make sure it works, then change another feature. Not only is this the best way to program, but it means that if you run out of time in working on this assignment, you will be able to turn in a partly-completed coursework and get some credit for it.

Background: Many, if not most, programs involve both input from, and output to, a user, and input from and output to one or more files, which hold data between runs of the program. In this coursework we extend the previous coursework by allowing the storage of student and course data in an external file. The previous coursework was very unrealistic, since it required all of the course data to be 'hardwired' into the program, and made the user input all of the information about his courses and sets of marks each time the program was run. (If you improved on this, that's fine.) Now we will become more realistic.

Concepts covered: external files.

ASSIGNMENT

Modify the program you wrote for Coursework 02, so that all of the data held in it, or entered into it, is stored on an external file. You will need two external files for this:

1. Training Module Titles File: if this information was 'hardwired' into your previous program, it should now be held instead on an external file, and loaded into your program when it starts.
2. Student Information should be held on an external file that can be updated (or, at a minimum, added to). Thus your program should begin by loading in all previously-entered student information into an array or vector, and allowing your user to add more student information to it, and then saving the updated information back to the file at the end of the program.
3. In all other respects your program should work exactly like CW02’s program.
4. If you wish to improve the input and output you may do so, but this is not required.

WHAT TO TURN IN

1. A print-out of the program code for the program Pay attention to good layout!
2. A screen dump of the final output of your program, showing results for five students. The information (Training Module taken and results) for three of them should have already been held on the file, and you should add two more on-line, saving all five at the end of the program run.
3. A one-to two-page description of how you tested the program to make sure that it worked. This should include a table showing various combinations of inputs with expected, and actual, outputs, plus a brief discussion of why you chose the particular values in the table. (HINT: does your program work correctly the first time, when all external files are empty? How will you handle this situation? You may wish to consider a special program whose only job is to create such files, and possibly give certain of them dummy [all-zero] values.)
4. A two- to three-page description of how your program works. Make reference to the different data structures (including files and arrays), and control structures (IF ELSE, FOR, etc) you used. You may wish to number the different sections of your program (using the commenting delimiters) and refer to "Section 1" and so on.

Weighting: Program: 50% Testing Description: 30% Program Description: 20%
UNIVERSITY OF LONDON

DIPLOMA AND BSc IN COMPUTING AND RELATED SUBJECTS
FOR INTERNATIONAL PROGRAMMES STUDENTS

COURSEWORK SUBMISSION FORM

IMPORTANT, PLEASE NOTE:

You must complete this form in full and attach it to the coursework that is to be submitted to the University.

Full name: ........................................................................................................................................................................
(as it appears on your Registration Form)

Student number: ................................................................................................................................................................

Unit title: ..........................................................................................................................................................................

Unit number: ....................................................................................................................................................................

Assignment number: ..........................................................................................................................................................

DECLARATION

I declare that:

• I understand what is meant by plagiarism.
• I understand the implications of plagiarism.
• This assignment is all my own work and I have acknowledged any use of the published or unpublished works of other people.

Signature ………………………………………………… Date …………………..
Before you submit your coursework to the University, have you:

☐ completed the title, unit number and assignment number on the submission form?

☐ signed and dated the Declaration?

☐ put your name and student number on every page?

☐ put the unit number, unit name and assignment number on every page?

☐ used more than one page? If so, have you securely stapled or tied the pages together?

☐ used a disk? If so, have you used a separate disk for each assignment? (Note: you should only use a disk if it is explicitly asked for in the assignment).

☐ labelled clearly and securely attached any disks?

☐ attached all parts of the coursework required (including the submission form)?

If you fail to do these things your coursework may not be accepted.

Please send all coursework to:
The Registration and Learning Resources Office
Room STG13
Stewart House
University of London
32 Russell Square
London WC1B 5DN
United Kingdom
Tel: +44 (0)20 7862 8326
Fax: +44 (0)20 7862 8329
Notes for the Student: This final coursework is quite different from the previous three. In those courseworks you were learning the basics of Java. Now you need to add 'bells and whistles'.

ASSIGNMENT

You should look for the original source code of an interesting Java program, by searching the web, or looking in a text book. The program does not have to be very long, or elaborate, although it may be. The important thing about this program is that it should use Java concepts which we have not so far covered, such as threads or events. (But it does not have to be either – it could be on-screen graphics, or any of the concepts we have not yet gone into.) Now modify this program, a step at a time, as much as you can, to create a program of your own.

For example, you may find a Java program to play noughts-and-crosses ("Tic-tac-toe"). You could modify this program in many different ways: you could give it better graphics; you could allow users to "take back" a bad move. You could make the program give the user hints.

NOTES ON PROGRAMMING

(1) First, of course, make sure that the original program works.

(2) Then make the smallest change possible to it and make sure that the modified program works. And so on.

(3) It's perfectly all right to use the same original program as someone else. But your changes must be different from his.

WHAT TO TURN IN

(1) A print-out of the original source program which you used to create your own program.

(2) A print-out of your own program, created by modifying the original.

(3) Several (not more than six each) appropriate screen dumps showing the original program's output, and your own.

(4) A description, running not more than five pages, of what you did, organised as follows:
(a) The features of Java, or your own programming goals, which you aimed to learn about in this coursework. For example, you may have wanted to learn about threads, or how to generate moving images on screen.

(b) A description of what the original program does, and how it does it.

(c) A description of what your program does, and how it does it.

(d) A description of the changes you made to the original program to make your own program. Add some numbering to both programs so that your description can refer the reader to the right parts of each program. Refer to the screen dumps, too. A good description will refer to any problems, solved or unsolved, that you ran into while working.

Weighting: Program Description: 100%

Note: high marks in this coursework will come from showing the examiner that you learned something, not from having an astonishing, long, complex program. This means that even if you are a beginner and still don’t know Java very well, and submit relatively simple programs, if you make a good effort in this work and show that you
UNIVERSITY OF LONDON

DIPLOMA AND BSc IN COMPUTING AND RELATED SUBJECTS
FOR INTERNATIONAL PROGRAMMES STUDENTS

COURSEWORK SUBMISSION FORM

IMPORTANT, PLEASE NOTE:

You must complete this form in full and attach it to the coursework that is to be submitted to the University.

Full name: ........................................................................................................................................................................
(as it appears on your Registration Form)

Student number:................................................................................................................................................................

Unit title: ..........................................................................................................................................................................

Unit number: ....................................................................................................................................................................

Assignment number:..........................................................................................................................................................

DECLARATION

I declare that:

• I understand what is meant by plagiarism.
• I understand the implications of plagiarism.
• This assignment is all my own work and I have acknowledged any use of the published or unpublished works of other people.

Signature ................................................................. Date ....................................
UNIVERSITY OF LONDON

DIPLOMA AND BSc IN COMPUTING AND RELATED SUBJECTS
FOR INTERNATIONAL PROGRAMMES STUDENTS

COURSEWORK SUBMISSION CHECKLIST

Before you submit your coursework to the University, have you:

☐ completed the title, unit number and assignment number on the submission form?

☐ signed and dated the Declaration?

☐ put your name and student number on every page?

☐ put the unit number, unit name and assignment number on every page?

☐ used more than one page? If so, have you securely stapled or tied the pages together?

☐ used a disk? If so, have you used a separate disk for each assignment? (Note: you should only use a
disk if it is explicitly asked for in the assignment).

☐ labelled clearly and securely attached any disks?

☐ attached all parts of the coursework required (including the submission form)?

If you fail to do these things your coursework may not be accepted.

Please send all coursework to:
The Registration and Learning Resources Office
Room STG13
Stewart House
University of London
32 Russell Square
London WC1B 5DN
United Kingdom
Tel: +44 (0)20 7862 8326
Fax: +44 (0)20 7862 8329