

**Notes for the Student:** This is the first of four courseworks which are designed to give you practical experience in composing elementary 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.

**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 Subject Guide, textbooks, websites, 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.

```
/* 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.

**Whole (counting) numbers:**

	<u>Name</u>	<u>Bytes of Storage taken up</u>	<u>Range</u>	<u>Example</u>
2.1	int	4	-2,147,483,648 .. +2,147,483,647	42
2.2				
2.3				
2.4				

### Fractions (decimal numbers)

	<u>Name</u>	<u>Bytes of Storage taken up</u>	<u>Number of significant digits</u>	<u>Example</u>
2.5				
2.6				

### Single Characters

	<u>Name</u>	<u>Bytes of Storage taken up</u>	<u>Example</u>
2.7			

### ‘Logical outcomes’: true or false

	<u>Name</u>	<u>Bytes of Storage taken up</u>	<u>Example</u>
2.8			

## 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) quantity limit
- (b) quantitylimit
- (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

You have been asked to write a program for a company which sells items of hardware over the internet. Your program will hold information about these items of hardware, and when it runs, it will display this information on the screen.

Write a program in Java which can do the following:

#### **“Hardwired” into the program:**

- (1) Information about the company’s products: a product code, description, and unit price.
  - (2) Shipping costs per item.
  - (3) VAT: the percentage of the transaction which is added as tax.
- (This means that values for variables will be assigned in the program, not read in.)

When the program should print out a list of all items in the company catalog. (There will be only four in this example. Their values should be assigned to appropriate variables when the program starts.) Then it should ask the customer to input an order.

#### **INPUT**

- (1) The customer’s name and their address.
- (2) How many of each type of hardware item the customer wants to purchase.

#### **OUTPUT**

The program should print out on the screen an invoice for the customer’s order, consisting of:

- (1) The customer’s name and address.
- (2) One or more order lines, each of which consists of  
Product Code, Product description, quantity ordered, unit price, total price.
- (3) Grand total of the order lines.
- (4) Final price, including 20% VAT (Value-Added Tax) plus \$1.00 per item shipping cost.

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

## HARDWARE ITEMS

CODE	DESCRIPTION	UNIT PRICE
K16	Wood screws, brass, 20mm, pack of 50	\$ 7.75
D24	Wood glue, clear, 1 litre	\$ 5.50
M93	Sandpaper, medium grade, 100 sheets	\$10.25
M94	Sandpaper, fine grade, 100 sheets	\$14.75

PLACE YOUR ORDER!

NAME: **Bill Silic**

ADDRESS-1: **Apt 23**

ADDRESS-2: **Building 8**

ADDRESS-3: **Winchester**

POST CODE: **89763**

ENTER CODE (XX to Stop):

CODE: **D24**

QUANTITY: **2**

CODE: **K16**

QUANTITY: **1**

CODE: **XX**

INVOICE FOR ORDER

Bill Silic

Apt 23

Building 8

Winchester

POST CODE: 89763

D24	Wood glue, clear, 1 litre	2 @ \$ 5.50	\$ 11.00
K16	Wood screws, brass, 20mm, pack of 50	1 @ \$ 7.75	\$ 7.75
		TOTAL:	\$ 18.75
	Shipping: 2 items @ 1.00		2.00
		TOTAL INCL. SHIPPING:	20.75
		VAT at 20%	4.15
		TOTAL TO PAY:	24.90

=====

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, and would display the catalog as clickable webpages. 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, running and testing as you go. With this program, a sensible process would be: get a program that can simply print out a list of hardware items available. When that is working, add the loop for reading in the customer's order, and building up the totals. Then add the code for printing out the final invoice, including the grand total..

(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 assignments, 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.

(5) Although money uses decimal points when it is displayed in dollar form, it is usually a good idea to avoid holding monetary sums as reals. Monetary variables should be held as whole numbers (integers) of pence, but displayed using a special method which prints a decimal point in the right place. In other words, \$15.99 should be held as 1599 pence. This is to avoid round-off errors when performing arithmetic on these values.

(6) Although the postal code in this case is a string of digits, you should hold it as a string, not as an integer. In general, “numbers”, like telephone numbers or URLs, upon which we are not going to do arithmetic, should be held as strings.

## 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: your program will be marked on how well structured it is and how well documented each section is. Note well: many students write correct programs but do not get good marks for them because they omit writing good in-line documentation.

(2) A screen dump, or printouts of an external text file, of the program’s output, shown processing inputs for one customer who 1 each of the first three items in the catalog.

(3) A one-page **description** of how you tested the program to make sure that it worked without error. 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 sensible output for a customer who orders nothing; for a customer who orders just one item; and for a customer who orders 1000 of each item. ) Note that you may assume customer input is always correct – for instance, that a non-existent item is not ordered. We are just testing to see how the program responds to correct input.

(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.

<b>Weighting:</b>	<b>Part A: Questions 1-5:</b>	<b>20%</b>
	<b>Part B: Program:</b>	<b>40%</b>
	<b>Testing Description:</b>	<b>20%</b>
	<b>Program Description:</b>	<b>20%</b>

**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** .....

---

## 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

**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 retrieving multiple instances of the same type of data, loops-within-loops.

## SPECIFICATION

- (1) This program is an extension of the one you wrote for CW01. You will need to make the following changes to this program: add five more kinds of hardware items to the stock of items already offered by the company. You should make their names and information up by yourself.
- (2) Each item which you add should have an associated weight, in grams. Let the shipping cost be based on the total weight of the order, rather than on how many items were ordered. Let your items weigh from between 50 and 2500 grams each. Let shipping costs be \$1.00 per 250 grams or fraction thereof. (Thus, an order weighing 675 grams would attract a shipping cost of \$3.00.)
- (3) Each run of the program should allow processing of an indefinite number of customers. This means you will have to add a dialogue asking whether another customer order is going to be placed. However, the “catalog” of hardware items and their prices should be shown only once.
- (4) Do elementary checking of the customer's inputs: check that a product code is valid, and do not allow orders for more than 100 of any particular item.

## NOTES ON PROGRAMMING

- (1) The most important thing about this coursework is that we are storing lots of instances of the same thing (hardware products, customers). Thus we will need to use a Java construct like arrays, or vectors. I strongly suggest that you use arrays at this point. Note: the previous program “hardwired” information about hardware items into separate variables of the program. This program should assign its information to an array, and search the array to find if a particular item code is valid, and what its description, weight and price are.
- (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 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. Make sure it is well-commented.
- (2) Screen dump, or printouts of an external text file, of the program's output, shown processing inputs for two different test cases. You must choose the most useful cases to test. Both cases should show
- (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: 40%**

**Testing Description: 40%**

**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** .....

---

## 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

**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 courseworks by allowing the storage of product and customer data in external files. The previous coursework was very unrealistic, since it required all of the catalog data to be 'hardwired' into the program. Now we will become more realistic.

**Concepts covered:** external files.

## ASSIGNMENT

- (1) Modify the program you wrote for Coursework 02, so that all of the product data held in it, is stored on an external text file. The VAT rate and shipping cost data should also be stored in an external text file. Thus there will not be any variables in your program which are assigned values like 25 or "woodscrews". Rather, these variables (and constants, if you used any) will be replaced by variables which get their values at the start of the program from a separate external text file. These files should be of type text, so that they can be created and edited by a simple text editor.

NOTE: storing data such as shipping weight parameters and VAT rates in an external file makes your program much more flexible. These data can now be accessed by other programs, and if, for example, the VAT rate changes, we have only to change one file, rather than change every program which makes use of the VAT rate.

- (2) Customer data should still be entered from the keyboard, but customer invoices should be printed to an external text file for permanent storage, rather than printed on the screen.
- (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) Print outs of all of your external text files, including a customer invoice file.
- (3) A screen dump of the final output of your program, showing orders being placed by three different customers..
- (4) 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.
- (5) 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. This description should include any problems you encountered while writing the program, and how you finally solved them.

Weighting: Program and external files: 30% Testing Description: 30% Program Description: 40%

**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** .....

---

## 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

**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 we want to explore other aspects of Java, and do so in the way that most features of programming languages are actually learned: by modifying existing working code.

## ASSIGNMENT

You should look for the original source code of an interesting Java program, by searching the web, or looking in a text book. This program should use Java concepts which we have not so far covered, such as threads or events. (But it does not have to be either of these specifically – 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.

## 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

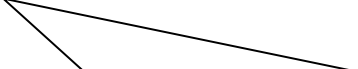
- (1) A print-out of the original source program which you used to create your own program.
- (2) A print-out of your own program, created by modifying the original.
- (3) Several (but not more than six each) appropriate screen dumps showing the original program's output, and your own.
- (4) A description, 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 the 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 have learned several new features of the language, you will get a good mark.

**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** .....

---



## 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