

JAVA PROGRAMMING – EXERCISE 2

In the first lab you learned how to use simple printing statements and perform calculations. In this lab, we'll be working with data types, operators and other expressions.

COURSE MARKER

A program called Course Marker will be used to provide you with the necessary java files to work with. Later on, we'll be using Course Marker to automatically grade your work.

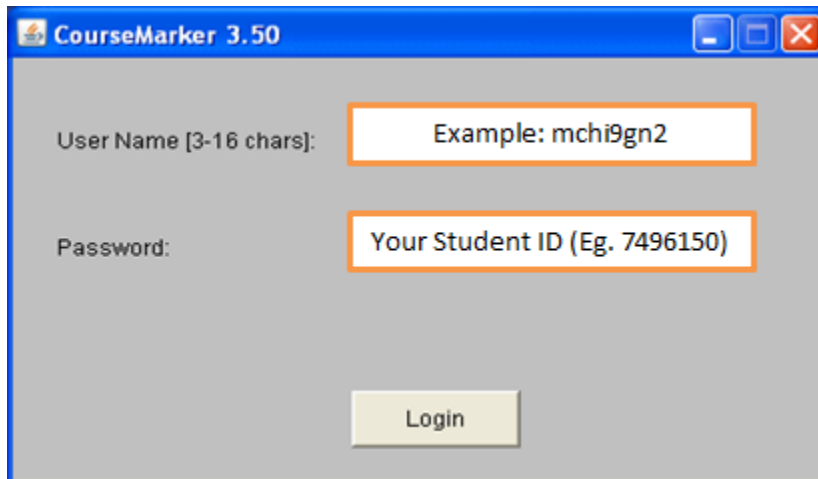
Please read the following instructions carefully.

1) Start Course Marker by selecting

Start>All Programs> CourseMarker >CourseMarker

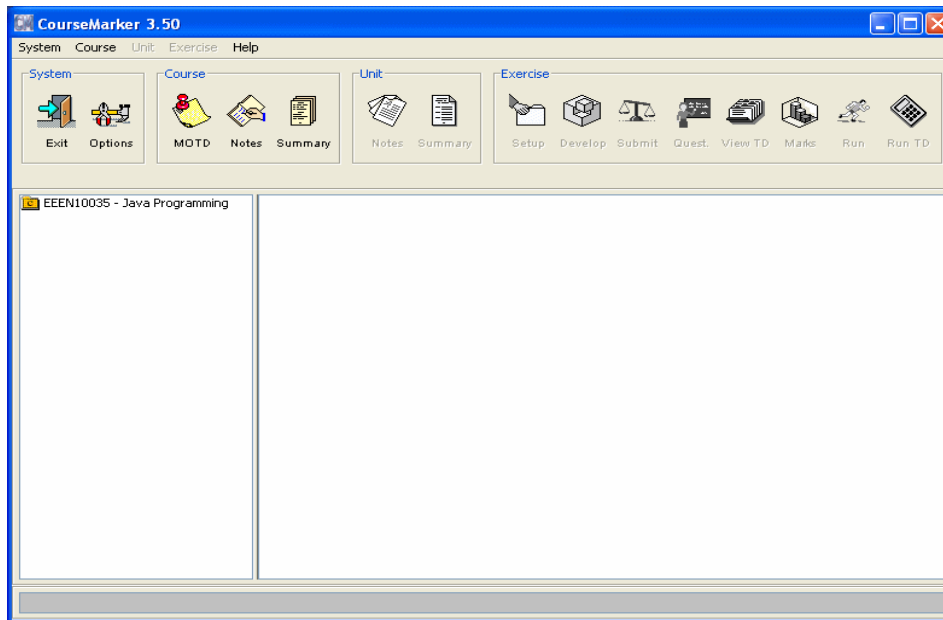
After a few moments you will see the login window –

NOTE: User Name is your usual username (mch...), and the password is your **student ID number**:

The image shows a screenshot of a Windows-style application window titled "CourseMarker 3.50". The window has a blue title bar with standard minimize, maximize, and close buttons. The main area has a light gray background. There are two input fields: the first is labeled "User Name [3-16 chars]:" and contains the text "Example: mchi9gn2"; the second is labeled "Password:" and contains the text "Your Student ID (Eg. 7496150)". Both input fields have orange borders. Below these fields is a yellow "Login" button.

2) Once you're in, access the relevant assignment by selecting:

EEEN10035 – Java Programming>u2 >exercises2 – Data types



- 3) Click the Setup button on top, and then click OK.
Clicking the Setup button copies the files needed for this exercise to the folder P:>mch....>EEEN10035u2exercises2 (where mch.... is replaced with your username).

The files copied are:

Quote.java, MathExercise1.java, MathExercise2.java and MathExercise3.java

These are the java files you'll be working on this time.

- 4) That's all we need from Course Marker, so now you can close it.
- 5) Everything we do from here on out will involve notepad and the command prompt.
- 6) First open command prompt using **start >All Programs** and scroll down until you see the folder **JDK** –click it and then click **cmd.exe**

In the resulting window type

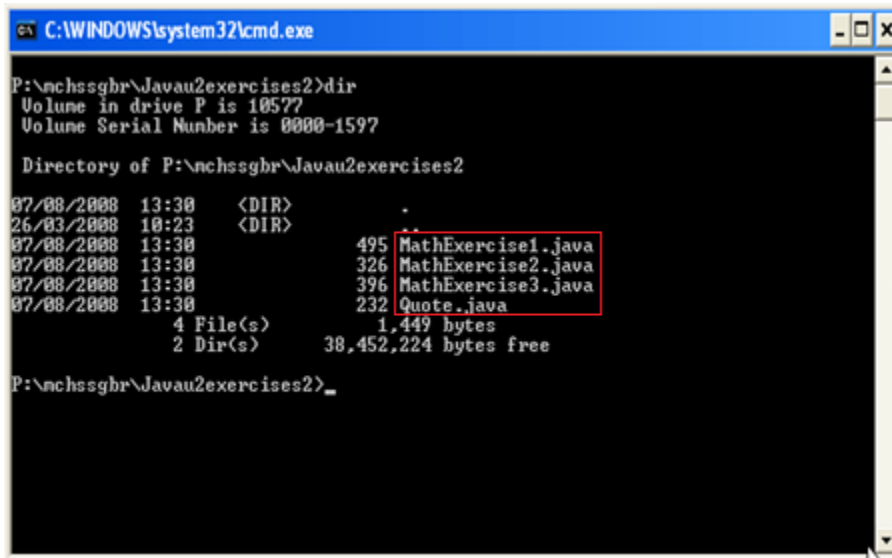
P: <press enter>

and then type

cd mchXXYY\EEEN10035u2exercises2 <press enter>

(We're just navigating to the folder where course marker copied all the necessary java files.)

Now, type the command **dir** and press enter. You should see an output similar to the following: (**dir** helps to list out all the files in a folder)



```
C:\WINDOWS\system32\cmd.exe
P:\nchssgbr\Javau2exercises2>dir
Volume in drive P is 10577
Volume Serial Number is 0000-1597

Directory of P:\nchssgbr\Javau2exercises2

07/08/2008  13:30    <DIR>          .
26/03/2008  10:23    <DIR>          ..
07/08/2008  13:30             495  MathExercise1.java
07/08/2008  13:30             326  MathExercise2.java
07/08/2008  13:30             396  MathExercise3.java
07/08/2008  13:30             232  Quote.java
               4 File(s)      1,449 bytes
               2 Dir(s)      38,452,224 bytes free

P:\nchssgbr\Javau2exercises2>
```

As you can see, we have all the exercises we copied over from Course Marker, in our folder.

7) **Task 1: Video Help?** <http://stream.manchester.ac.uk/Play.aspx?VideoId=9578>
-you may have to type your username and password into the page that loads.

Try to compile Quote.java by typing
javac Quote.java

You will notice a number of errors.

Using Notepad (just type in **notepad Quote.java** to open up the file), correct the code so that it compiles and runs.

8) Before we open the other files in the folder, you need to understand the concept of **data types**.

Video Help? <http://stream.manchester.ac.uk/Play.aspx?VideoId=9585>

There are 8 scalar (or 'primitive') data types in Java:

Data Type	Function	Size
boolean	Stores a value of 'true' or 'false'	1 bit
char	Stores a single character (eg. an alphabet or a single-digit number)	16 bits
byte	Stores an integer number in the range of -128 to 127	8 bits
short	Stores an integer number in the range of -32768 to 32767	16 bits
int	Stores an integer number in the range of -2147483648 to 2147483647	32 bits
long	Stores an integer number in the range of -9223372036854775808 to 9223372036854775807	64 bits
float	Stores any number (which can include decimal values) in a specific range	32 bits
double	Stores any number (which can include decimal values) in a higher range	64 bits

These data types are used to represent numbers, characters, etc. In this lab, we shall concentrate on using **int**, **float** and **double**.

A Java program consists of declarations and statements of data used within the program. Data whose values can change are termed *variables*.

A variable **must be declared and initialised** before it can be used. The declaration must include the item's type, e.g

```
int a;           //declares variable a of type int
```

NOTE: It is possible to declare multiple variables at the same time, e.g.

```
int a,b,c;       //declares variables a, b, c, each of type int
```

int is a data type used to represent integer numbers (in the range of -2^{31} to 2^{31}). It requires 32 bits of memory.

float is a type that can be used to represent floating point numbers (i.e. numbers with fractional parts), in the range 1.4×10^{-45} to 3.4×10^{38} . Float also requires 32 bits of memory:

float e; *//declares variable e of type float*

double is used for greater precision and greater range: 4.9×10^{-324} to 1.79×10^{308} , but requires 64 bits of memory. Example:

double d; *//declares variable d of type double*

A **double** requires twice as much memory compared to a **float**.

A variable is initialised in an assignment statement, e.g.

a = 5; *//assign the value 5 to a*

Often the declaration and assignment are combined:

int a = 5;

The number '5' is called an *integer literal*, and this satisfies the requirement that in all assignment statements the left-hand, and right-hand, sides are of the same type. If this is not the case, there will be a compiler error. There are certain exceptions to this, however, as you will later see.

Example using double:

double speed = 5.0;

The number '5.0' is a *floating point literal*. Floating point literals are always of type **double**, so the above statement satisfies the afore-mentioned requirement that the left-hand and right-hand types are the same. To represent a floating point literal of type **float**, the character 'f' or 'F' is appended, e.g.

float speed = 5.0F;

There is another way to write a floating point literal in Java: in scientific notation. In scientific notation a number consists of a decimal part, followed by an 'e' or 'E', followed by an exponent, e.g.

1.23E10	<i>// represents 1.23×10^{10}</i>
2.34E12	<i>// represents 2.34×10^{12}</i>
-4.75e3	<i>// represents -4.75×10^3</i>
-4.75e-3	<i>// represents -4.75×10^{-3}</i>

Multiple assignments can be made at the same time, e.g.

```
double volume = 23.0, weight = 12.5;  
int length = 30, height = 20;
```

Items that are fixed are termed *constants*; a constant in Java is defined by the word **final**, e.g.

```
final int DENSITY = 4;
```

By convention constants are written in uppercase letters.

An assignment statement can contain an *expression* on the right-hand side, as can be seen in the following example:

```
double area = height * length;
```

The values for height and length are replaced with their previously defined values.

Let's look at a sample program now:

```
public class Anything {  
    public static void main(String args[]) {  
        char a;      // declares a character variable called a.  
        int b;        // declares an integer variable called b.  
        float c;      // declares a float variable called c.  
        double d;     // declares a double variable called d.  
        a='z';        // assigning the letter 'z' to variable a  
        b=56;          // assigning 56 to variable b  
        c=52.7F;       // assigning 52.7 to variable c  
        d=245.5;       // assigning 245.5 to variable d  
        System.out.println(a);  
        System.out.println(b);  
        System.out.println(c);  
        System.out.println(d);  
    }  
}
```

Declaration
(of variables)

Initialisation
(of declared variables)

Notice that when you want
to output variables, you do
not use any quotes.

The output of the above program will be:

z
56
52.7
245.5

Now, considering this sample program, the tables below will help in undertaking the next task.

changing the value of:

(I) char a

Value	Output
a='b'	b
a=b	Program doesn't compile, since a char variable needs to be inside single quotes
a='bb'	Program doesn't compile, since a char variable can only hold ONE character
a='2'	2
a='22'	Program doesn't compile, since a char variable can only hold ONE character
a='*'	*
a='**'	Program doesn't compile, since a char variable can only hold ONE character

changing the value of:

(II) int b

Value	Output
b=27	27
b=27.5	Program doesn't compile, because an int variable cannot store decimal values
b=5.9	Program doesn't compile, because an int variable cannot store

	<p>decimal values</p> <p>Special Note: Whenever you are assigning a value to a variable, make sure the 'data-type' is the same on the right and left hand side.</p> <p>That is, in this case, 'b' is an int variable, but '5.9' is a float or double value.</p>
b=5	<p>5</p> <p>Special Note: If the System.out.println(b) statement in our sample program had been System.out.println(b/2) instead, the output would've been 2 and NOT 2.5. Why?</p> <p>Because int variables cannot display decimal values. It must be noted that the program will compile though.</p>

changing the value of:
(III) float c

Value	Output
c=27F	27.0
c=6.2F	6.2
c=6.2	Program doesn't compile, because the '6.2' needs an F at the end, to tell the compiler that this is a float value.
c=5	5.0

changing the value of:
(IV) double d

Value	Output
d=27D	27.0
d=5.8D	5.8
d=5.8	<p>5.8</p> <p>Special Note: You don't need to write a D at the end for 'double' variables. Yes, it is completely optional, and this only works for 'double' variables.</p>
d=27	27.0

Task 2

Open Notepad on MathExercise1.java and add a multi-line comment that states your name and a version number for the program, before the class declaration.

Then, in the body of the **main** method, add declarations for the following:

- A variable 'count' of type int
- A variable 'volts' of type float
- A variable 'real' of type double initialized to 33×10^{21}
- A constant 'INTEREST_RATE' of type double
- A variable 'current' of type double and initialized to 6.23×10^{-3}

Make sure the code compiles before continuing (no output is expected from the program). Close Notepad.

Special Note:

It is good practice to give important variables a meaningful identifier, e.g. 'speed', 'distance', etc, rather than 'a', 'b', etc. The rules about identifiers are that they must

- start with a letter, underscore (_), or dollar symbol (\$). Subsequent letters must be either alphabetic, numeric, underscores or dollar signs
- they can't be Java reserved words, such as 'class', 'public', etc.
- whitespace cannot be embedded in the name (whitespace are spaces, tabs, etc)

Task 3

Which of the following identifiers are legal?

Prod-b

1abc

a1b2c3

sum.Of

5sum

\$total

To test the above, open Notepad again on MathExercise1.java, and add declarations for each (using say, int as the type) starting immediately after the declaration for 'current'. Save and compile the code each time to test your answers. When you have finished close Notepad.

9) In this next section you will work on the file MathExercise2.java. Open the file for editing using Notepad.

The purpose of the code in **main** is to calculate the speed given values of distance and time, using the expression $\text{speed} = \text{distance} / \text{time}$.

Task 4:

Complete the code as follows:

- a. declare three **int** variables 'speed', 'distance, and 'time'
- b. initialize distance and time using any values you like
- c. calculate and print out the speed

Make sure your code compiles and runs without error before continuing.

Now test your program (i.e. change and save the file, and then re-compile and run, each time) using the following test values for distance and time:

<u>distance</u>	<u>time</u>
3	4
4	4
5	4

Verify that the results are 0, 1 and 1, and not 0.75, 1 and 1.25, for each of the above combinations. These results are because we are doing integer division, and integers cannot hold a fractional part.

The expression can be changed to give a more accurate result if we use **double** for speed, distance and time.

Task 5:

Change the type of speed, distance and time to **double**, re-compile and test again.

10) Type Conversion

In the expression $\text{distance} / \text{time}$, it turns out that we only have to make *one* of the operands (either 'distance' or 'time') as a **double** type, to get an accurate result.

Task 6:

Verify this is true by editing, re-compiling and running MathExercise2.java again.

The reason this works is that for the division operator '/', if one operand is a **double**, the other operand is automatically promoted from an **int** to **double**. The same is true for other basic operators such as add '+', subtract '-' and multiply '*'. Automatic type conversion can occur between a 'narrow' type (**int** is 32 bits) and a 'wider' type (recall that **double** is 64 bits). For example, automatic type conversion would occur if say, we had used **float** for distance and time, but kept speed as **double**; in this case, the conversion would be from **float** (32 bits) to **double** (64 bits).

11) Formatted Output

Video Help? <http://stream.manchester.ac.uk/Play.aspx?VideoId=9596>

Say you calculated the average age of employees in a company to be 27.288756. It would be better to present that answer in 2 or 3 decimal places. How?
By using the 'printf' method, instead of the usual 'println' method!

```
public class Anything {  
  
    public static void main(String args[]) {  
        double avg= 27.288756;  
        System.out.printf("%.2f", avg);  
    }  
}
```

The '%' tells the compiler that this is a special function. The '.2' means the output should be rounded up to 2 decimal places. The 'f' just says that the '.2' is a float. The 'avg' (without quotes) is the average age.

This is a 'printf' statement, not a 'print' or 'println' statement. That's why the syntax is a little different. printf uses a "," symbol instead of a "+" to print variables. However you can still use "+" when its just string concatenation that you want to do. E.g. this will work:

```
System.out.printf("The value of " + "avg = %3.2f", avg);
```

Task 7:

Complete the program MathExercise3.java. This program calculates and prints out the force exerted by gravity on three objects of mass m1=5.0, m2=5.5 and m3=6.0, respectively, using the relationship force = mass * g, where g is the constant 9.81. The output should be as shown below, formatted to 3 decimal places:

Force for mass m1 = xx.xxx

Force for mass m2 = xx.xxx

Force for mass m3 = xx.xxx

Task 8:

Choosing suitable values for the cross-sectional area and length, write a program that calculates and prints out to two decimal places, the resistance of a wire, given the relation

$$\text{resistance} = (\text{resistivity} * \text{length}) / \text{cross-sectional area}$$

where the resistivity is a constant of value $1.68E-3$

Modify this to print out the total resistance of a three such resistances connected in parallel. When you have finished, save your work and logout.