**Programming in Java**

**-Sheet 1-**

1. **Writing your first java program**

To create our Java applications we are going to use the free IDE (Integrated Development Environment) NetBeans. Let’s open the IDE.

1. Using the windows start menu open the NetBeans application
2. Select file > New Project
3. Select Java Application > Click next
4. Give the project the name “HelloWorld” > Click Finish

**1.1 Try this:** Enter the following code into NetBeans. Note that NetBeans creates some code automatically which you can use. Faded grey text can be ignored for now.

|  |
| --- |
| **package helloworld;**  **class** HelloWorld {  **public** **static** **void** main(String[] args) {  System.**out**.println("Hello world!");  }  } |

Compile and run the program. It should output the text "Hello world!". Congratulations you've written your first java program!

Below breaks the program down to three parts. Explaining a little bit about how a basic java program works:

**The program container**

|  |
| --- |
| **class** HelloWorld { } |

After the **class** keyword the program name is declared. Within the curly brackets you place the program code that defines the HelloWorld **class**. Think of the program **class** as a blueprint.

**The main method**

|  |
| --- |
| **public** **static** **void** main(String[] args) |

This code defines the starting point of the majority of java programs. The method "**main**" will contain the actual program instructions within its curly brackets. The method indicates the actions the class will take.

**Public**, **static** and **void** will be explained further later in the course. For now think of these keywords as special modifiers needed for the method **main** to be found and accessible.

**The statement**

|  |
| --- |
| System.***out***.println("Hello world!"); |

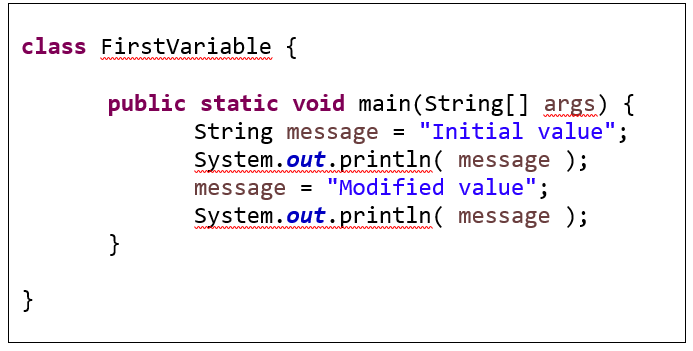
**Statements** must always end with a semi colon. They are program tasks contained within the **method**. A method can contain many statements but this is a single statement which outputs a line of text.

**1.2 Try this:** Change the text "Hello World!" to say the same in French (Use google translate). Re-compile, run and see what happens.

**2. Creating a Variable**

A **variable** in java is a useful container which a value can be stored in and used within the program. The stored value can then be changed (vary) throughout the program as it executes its instructions. A variable is created by writing a variable declaration in the program specifying the type of data that the variable can contain.

**2.1 Try this:** As with task 1, create a new project called FirstVariable. Enter the following code. This code will create a variable and then modify it while outputting both values:



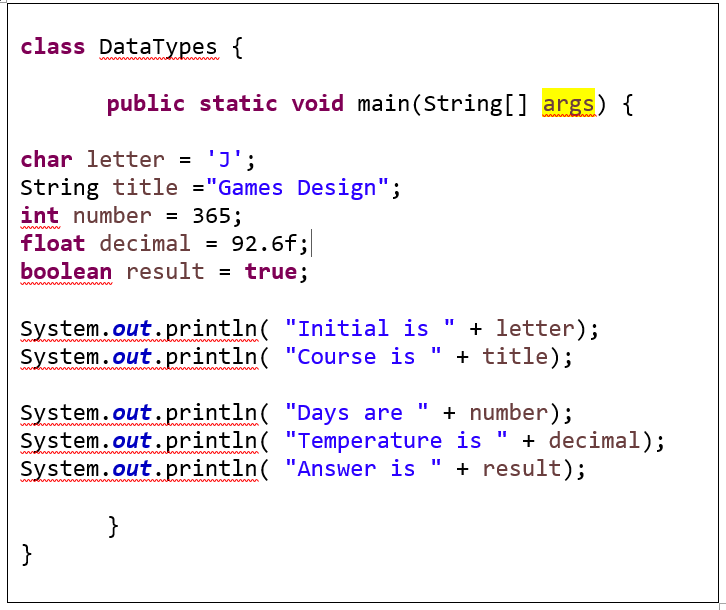
Compile and run the program. You should have two lines of text showing "Initial value" and "Modified value"

**3. Data Types**

The most frequently used data types when declaring a variable are shown below:

|  |  |  |
| --- | --- | --- |
| **Data type:** | **Description:** | **Example:** |
| char | A single Unicode character | 'a' |
| String | Any number of Unicode characters | "This is my String" |
| Int | An integer number, between -2.14 billion and + 2.14 billion | 1000 |
| float | A floating-point number, with a decimal point | 3.14159265f |
| boolean | A logical value of either true or false | true |

**Try this 3.1:** Let’s create a program that creates, initializes and outputs variables of all the common data types. Create the following program:



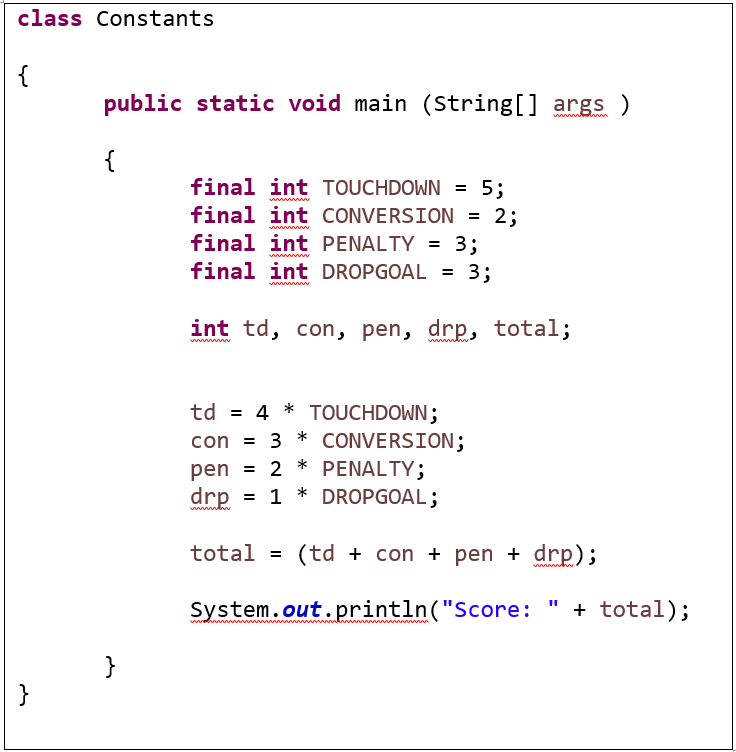
When completed compile and run the program. You should get the output:

Intial is J  
Course is Games Design  
Days are 365  
Temperature is 92.6  
Answer is true

**4. Creating constants**

Constants allow you to declare a variable that cannot be changed. Using the modifier "final" keyword before a data type lets the program know that no further changes are allowed. Convention dictates that we use capital letters when declaring constants.

**4.1 Try this**: Try creating the following program:



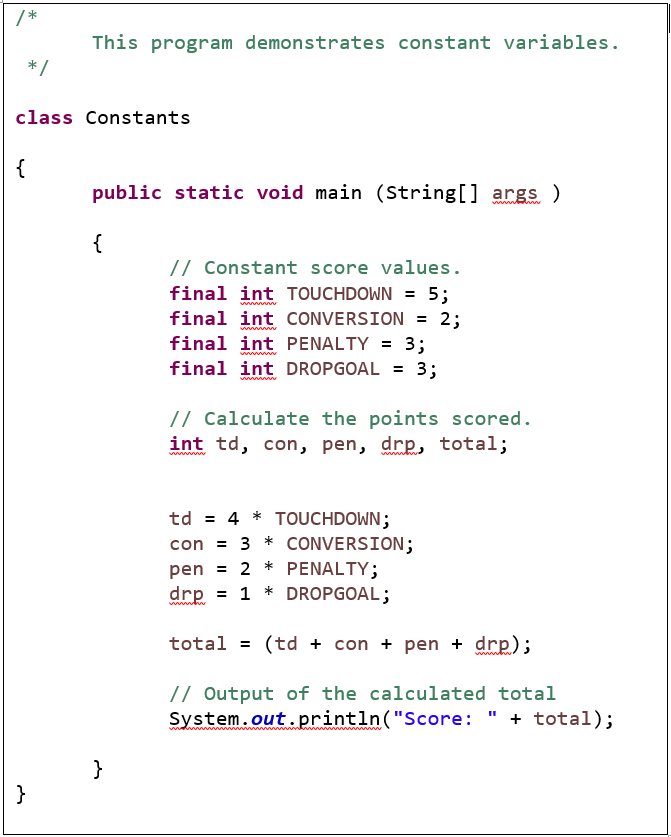
Compile and run the program. You should get the output "**Score: 35**".

**4.2 Try this:** Add a statement that attempts to change the value of a constant, recompile and see what happens.

**5. Comments**

It is good practice when coding to leave comments to explain each section of your code. This helps yourself and other people to understand your code. If you'd like to display code over multiple lines use /\* and \*/ writing your comments in-between. If you'd like to create a single line comment then start with //. Comments are completely ignored by the compiler.

**5.1 Try this:** Add comments to your constants program. It should look something like below.



**6. Exercises**

1. Using Variables

Write a program that stores your name and year you finish your course into variables, and displays their values on the screen.

Make sure that you use two variables, and that the variable that holds your name is the best type for such a variable, and that the variable that holds the year is the best type for *that* variable.

Also make sure that your variable names are good: the name of a variable should always relate to its contents.

*Your program should* ***NOT*** *look like this:*

System.out.println( "My name is Jeff and my course finishes in 2017" );

1. Using Variables 2

Write a program that creates three variables: an int, a float, and a String.

Put the value 413 into the first variable, the value 3.5 into the second, and the value "Programming in Java" into the third. It does not matter what you call the variables... *this time*.

Then, display the values of these three variables on the screen, one per line.

Your program ***SHOULD NOT*** look like this:

System.out.println( "This is room # 113" );

System.out.println( "e is close to 2.71828" );

System.out.println( "I am learning a bit about Computer Science" );

You must use three variables. Your program will probably have nine lines of code inside the curly braces of main().

1. Gallons to litres

Although the preceding sample programs illustrate several important features of the Java language, they are not very useful.

In this project, we will create a program that converts gallons to litres. You will need to declare two **float** variables. One will hold the number of the gallons, and the second will hold the number of litres after the conversion. There are 4.54609 litres in a gallon. So to convert gallons to litres, the gallon value is multiplied (using the \* symbol) by 4.54609. The program should display both the number of gallons and the equivalent number of litres.

Name the **GalToLit.java**

Set the program to convert 10 gallons to litres.