**Introduction to JAVA**

# Java – A001 — Hello World

**Assignment:**

Let's study the entire program below:

// The "A001" class. import java.awt.\*; import hsa.Console; public class A001

{

static Console c; // The output console

public static void main (String[] args)

{

c = new Console ();

c.println("Hello World!");

} // main method

} // A001 class

Any portion of the program that starts with "//" is a comment. It is completely ignored by the program, and it only there as an aid to the programmer. (Use lots of comments!)

The lines that start with "import" are needed to include specific libraries (called classes in java). The two specific classes imported here are the Abstract Window Toolkit (awt) and the Console class.

The section that starts with "public class A001" is the actual A001 class. The class is always surrounded by braces ("{ }") and is always named starting with a capital letter. See if you can spot the start of the A001 class and the end of it.

The line "static Console c;" is a *declaration*, which means that we are declaring a variable named "c" and "c" is a console object. Notice that the word Console starts with a capital. This is important, because Console is a "class" file, and all classes must be capitalized. This particular variable is declared as "static", which means that is a class variable (as opposed to other variables we will be learning about later on).

Within the A001 class is the line "c = new Console ();". This is an *assignment*, where we are assigning the variable "c" to a new console object. (This is different than the declaration. First we must declare that we are going to use a variable called "c" and that it is going to be a console, then we do the actual assignment where the console object is created.)

We then get to the section starting with "public static void main". Believe it or not, this is the (finally) the start of our program. This is where we can concentrate on making the program do what we want it to do. Don't worry too much about what the specific words on this line mean. It will all make sense to you when you start learning more about java. For now, trust that the program must be written this way and trust that you will understand it in due time. Notice that this section is also surrounded with braces.

And now, we finally get to the part of the program that does something for us — "c.println("Hello World!");" This is the line that prints "Hello World!" to the screen. The "c."

part of the line means we want to run a "method" in the console class. We know it's from the console class because "c" was declared as type "Console" in a previous part of the program. The part after the "c." is the specific method in the Console class we want to run. In this case, it is the "println" method.

Undo any changes you may have made to the program to test for errors, then make sure the program prints "Hello World!" to the console. Save as "A001.java".

# Java -Introj1— Printing Lots of Lines

**Assignment:**

Create a new Console class program named Introj1. Add two "c.println()" statements, one printing your first name, the second printing your last name.

Run the program to see what happens. How many lines are printed? Save as "Introj1.java".

# Java – Introj2 — Printing Lots of Lines (sort of)

**Assignment:**

Load Introj1. Replace "Introj1" with "Introj2" throughout your program. (Use the Search, Replace menu item to do this.)

Save as "Introj2.java".

Change the program by replacing the "println" methods with "print". Run the program to see what happens. What can you say is the difference between println and print? Does the output look okay? See if you can fix the output so it looks better.

Re-save your program.

# Java – Introj3 — Concatenation

You have now learned that you can use two print() statements to print two things on the same line. But there is a different way that is sometimes more efficient.

# Assignment:

Start a new Console class program named Introj3. Put the following code snippets into your program to observe any differences, if any.

// print on the same line

c.print ("abc" + "def");

// print separator line c.println ();

// two print statements c.print ("abc");

c.print ("def");

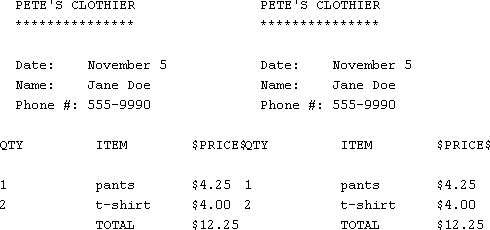
Notice that the two sets of output are the same? The first method, using the "+" operator to join two "strings" together is called concatenation. It is a very common way to join two or more strings together.

Save as "Introj3.java".

# Java – Introj4 — Concatenation Assignment

**Assignment:**

Using the knowledge you learned in Introj3 about concatenation, write a program that produces an output that looks like the following screenshot:



It would be easy to print each line using a single println() statement, but for this assignment you must print each word or each series of spaces within its own set of quotes. For example, the first line would be produced with the following statement:

c.println(" " + "PETE'S" + " " + "CLOTHIER");

Is this the most efficient? No, but it is useful to teach you how concatenation works. Be sure all lines line up the way they do in the screenshot.

Save as "Introj4.java".

# Java – Introj5 — Easy Math

Java math follows the same basic set of precedence rules as regular math — BEDMAS. There are some other precedence rules to follow, but they will come a little later on.

The following symbols are used to represent the four primary mathematical operators:

|  |  |
| --- | --- |
| + | addition |
| - | subtraction |
| \* | multiplication |
| / | division |

Exponents are handled quite differently in java, so we will discuss those in a lesson on their own. Also, division in java sometimes does some funny things (try the statement c.println(5/2); to see what I mean), so that too will get special treatment in its own lesson.

# Assignment:

For this assignment, you are to load Introj4, modify it so it can be saved as Introj5, then modify the subtotals and the grand total so they are performing the relevant math.

Save as "Introj5.java".

# Java –Introj6 — Field Widths

**Assignment:**

Load program Introj5, then modify to save as Introj6.

Wouldn't it be nice if we didn't have to print all the sections of blank spaces? Well we don't! If we put a comma after a string and then specify a number, that is how much space the string will take up. If the string is not wide enough, it will be padded with spaces.

For example, create a new test Console class application and add the following lines to see what I mean:

c.print ("PETE'S", 10);

c.println ("CLOTHIERS");

If you count the number of spaces taken up by the "PETE'S" string, you will see that it equals 10, the exact number specified by the print() statement.

Modify the new program, getting rid of all the sections of code where you were printing blank spaces. Replace these with suitable field widths. You will notice that when you use field widths, you can no longer use concatenation; you'll have to use separate c.print() statements. (You win some, you lose some!)

Save as "Introj6.java".

*Hint: You can print blank spaces at the beginning of a line using the same method! Just use c.println(" ", n), where n is the number of spaces you want.*

# Java – Introj7— Escape Characters

The string "\t" represents a tab, while "\n" represents a new line. These escape characters can be inserted anywhere in a string; they do not have to be used on their own.

For example, the statement c.println("A\tB\nC\t\tD"); produces the following output:

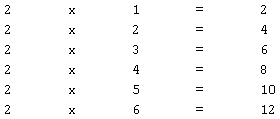


# Assignment:

Write a program that produces an output similar to the screenshot in Introj4. The output does not have to look exactly the same; just try to line up the columns using tabs. Do not use "println()" methods in your program, only "print()".

# Java – Introj8 — Times Tables

**Assignment:**

Write a program that uses escape characters and the suitable mathematical operators to produce a neatly formatted times table from 2 x 1 to 2 x 6 similar to the following diagram:

.

Save as "Introj8.java".