

Contents – DAY 1

* Introduction
* About JAVA
* OOP – Object Oriented Programming
* Part 1 – Getting Started
* The Java Development Kit – JDK
* My first Java program
* Using an IDE
* Variables and Data Types
* Variables
* Test your skills

Introduction

About JAVA

“Java refers to a number of computer software products and specifications from Sun Microsystems

(The Java™ technology) that together provide a system for developing and deploying cross-platform applications. Java is used in a wide variety of computing platforms spanning from embedded devices and mobile phones on the low end to enterprise servers and super computers on the high end. Java is fairly ubiquitous in mobile phones, Web servers and enterprise applications, and somewhat less common in desktop applications, though users may have come across Java applets when browsing the Web.

Writing in the Java programming language is the primary way to produce code that will be deployed as Java bytecode, though there are compilers available for other languages such as JavaScript, Python and Ruby, and a native Java scripting language called Groovy. Java syntax borrows heavily from C and C++ but it eliminates certain low-level constructs such as pointers and has a very simple memory model where every object is allocated on the heap and all variables of object types are references. Memory management is handled through integrated automatic garbage collection performed by the Java Virtual Machine (JVM).”

OOP – Object Oriented Programming

OOP is a particular style of programming which involves a particular way of designing solutions to particular problems. Most modern programming languages, including Java, support this paradigm.

When speaking about OOP one has to mention:

1. Inheritance
2. Modularity
3. Polymorphism
4. Encapsulation (binding code and its data)

However at this point it is too early to try to fully understand these concepts.

This guide is divided into two major sections; the first section is an introduction to the language and illustrates various examples of code while the second part goes into more detail.

Part 1 - Getting Started

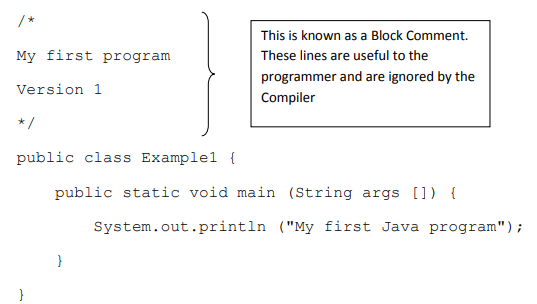
The Java Development Kit – JDK

In order to get started in Java programming, one needs to get a recent copy of the Java JDK. This can be obtained for free by downloading it from the Sun Microsystems website, http://java.sun.com/

Once you download and install this JDK you are ready to get started. You need a text editor as well and Microsoft’s Notepad (standard with all Windows versions) suits fine.

My first Java program

Open your text editor and type the following lines of code:



Save the file as Example1.java

The name of the program has to be similar to the filename.

Programs are called classes. Please note that Java is case-sensitive. You cannot name a file

“Example.java” and then in the program you write “public class example”. It is good practice to insert comments at the start of a program to help you as a programmer understand quickly what the particular program is all about. This is done by typing “/\*” at the start of the comment and “\*/” when you finish. The predicted output of this program is:



In order to get the above output we have to first compile the program and then execute the compiled class. The applications required for this job are available as part of the JDK:

1. javac.exe – compiles the program
2. java.exe – the interpreter used to execute the compiled program

In order to compile and execute the program we need to switch to the command prompt. On windows systems this can be done by clicking Start>Run>cmd

At this point one needs some basic DOS commands in order to get to the directory (folder), where the java class resides:

1. cd\ (change directory)
2. cd\[folder name] to get to the required folder/directory

When you get to the required destination you need to type the following:

c:\[folder name]\javac Example1.java

The above command will compile the java file and prompt the user with any errors. If the compilation is successful a new file containing the bytecode is generated: Example1.class

To execute the program, we invoke the interpreter by typing:

c:\[folder name]\java Example1

The result will be displayed in the DOS window.

Using an IDE

Some of you might already be frustrated by this point. However there is still hope as one can forget about the command prompt and use an IDE (integrated development environment) to work with Java programming. There are a number of IDE’s present, all of them are fine but perhaps some are easier to work with than others. It depends on the user’s level of programming and tastes! The following is a list of some of the IDE’s available:

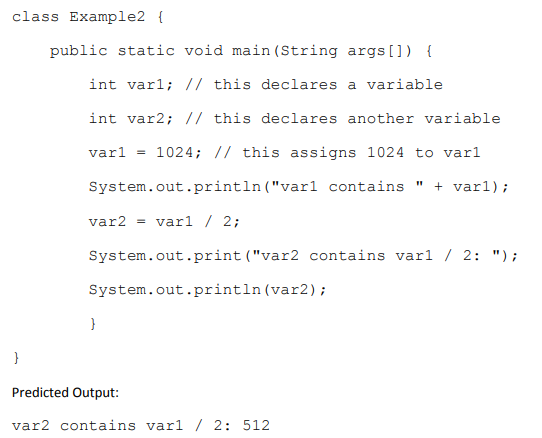
1. **BlueJ** – www.bluej.org (freeware)
2. **NetBeans** – www.netbeans.org (freeware/open-source)
3. **JCreator** – www.jcreator.com (freeware version available, pro version purchase required)
4. **Eclipse** – www.eclipse.org (freeware/open-source)
5. **IntelliJ IDEA** – www.jetbrains.com (trial/purchase required)
6. **JBuilder** – www.borland.com (trial/purchase required)

Beginners might enjoy BlueJ and then move onto other IDE’s like JCreator, NetBeans, etc. Again it’s just a matter of the user’s tastes and software development area.

Variables and Data Types

Variables

A variable is a place where the program stores data temporarily. As the name implies the value stored in such a location can be changed while a program is executing (compare with constant).



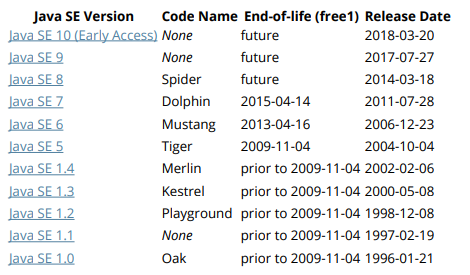
Test your skills

As we saw above, we used the ‘/’ to work out the quotient of var1 by 2. Given that ‘+’ would perform addition, ‘-‘ subtraction and ‘\*’ multiplication, write out a program which performs all the named operations by using two integer values which are hard coded into the program.

Hints:

1. You need only two variables of type integer
2. Make one variable larger and divisible by the other
3. You can perform the required calculations directly in the print statements, remember to enclose the operation within brackets, e.g. (var1-var2)

Extra Points to deliver to students:



**Note**: For Java to recognize this as a public class (and not throw a compile time error), the filename must be the name as the class name (Example1 in this example) with a .java extension. There should also be a public access modifier before it.

Naming conventions recommend that Java classes begin with an uppercase character, and be in camel case format (in which the first letter of each word is capitalized). The conventions recommend against underscores (\_) and dollar signs ($).

To compile, open a terminal window and navigate to the directory of Example1.java:

cd /path/to/containing/folder/

Note: cd is the terminal command to change directory.

Enter javac followed by the file name and extension as follows:

$ javac Example1.java

It's fairly common to get the error 'javac' is not recognized as an internal or external command, operable program or batch file.

When you have installed the JDK and are able to run the program from IDE ex. eclipse etc. Since the path is not added to the environment by default.

A closer look at the Hello World program

The "Hello World" program contains a single file, which consists of a HelloWorld class definition, a main method, and a statement inside the main method.

***public class Example1 {***

The class keyword begins the class definition for a class named HelloWorld. Every Java application contains at least one class definition (Further information about classes).

***public static void main(String[] args) {***

This is an entry point method (defined by its name and signature of public static void main(String[])) from which the JVM can run your program. Every Java program should have one. It is:

**public**: meaning that the method can be called from anywhere mean from outside the program as well.

**static**: meaning it exists and can be run by itself (at the class level without creating an object).

**void**: meaning it returns no value. Note: This is unlike C and C++ where a return code such as int is expected

(Java's way is System.exit()).

This main method accepts:

An array (typically called args) of Strings passed as arguments to main function (e.g. from command line arguments).

Almost all of this is required for a Java entry point method.

Non-required parts:

The name args is a variable name, so it can be called anything you want, although it is typically called args.

Whether its parameter type is an array (String[] args) or Varargs (String... args) does not matterbecause arrays can be passed into varargs.

Note: A single application may have multiple classes containing an entry point (main) method. The entry point of the application is determined by the class name passed as an argument to the java command.

Note: Each statement in Java must end with a semicolon (;).