A PRIMER ON JAVA

Second Edition

by Rahul Batra

A Primer on Java

Second Edition

Rahul Batra

This book is for sale at http://leanpub.com/aprimeronjava

This version was published on 2014-05-26

To Pria

Contents

1 An introduction to Java

1.1 What is Java?

To perform useful tasks on a computer, we either use a prebuilt software application (like a text editor) or build one ourselves. This process of making software using instructions that a computer can understand is called *programming*. The set of instructions given is called a *program*.

These instructions or programs must be given in a precise, formal language which is referred to as a *programming language*. **Java** is one such programming language.

Before a program can be run, we require another program to translate the program from the language we wrote in (say Java or Pascal) to a language the underlying hardware understands, i.e. a *machine language*. While this is not a one step process, for simplification we consider it as such. This program is called a *compiler*. Another class of programs with similar goals is an *interpreter*, which translates program statements directly into actions without the need for compilation to a machine language. This is also referred to as *program execution*.

Java uses both a compiler and an interpreter to execute its programs. In the first stage, the Java compiler translates the program listing to a intermediate *bytecode*. This bytecode is in the form of coded instructions a virtual hardware machine called the **Java Virtual Machine (JVM)** understands. The Java interpreter then runs this bytecode according to the specification of this virtual computer of sorts (the JVM) and we have our program execution.

1.2 Advantages of Java

Since Java is compiled to the JVM bytecode specification, it means that a Java program will run on any architecture or operating system where a Java interpreter exists. This is different from the typical execution flow of compiled languages like Pascal or C, because their source code has to be recompiled to target whichever underlying machine exists. The reason for this particular design choice was the fact that one of Java's original goals was to be the programming language for varied consumer devices, not just personal computers. Since these devices would not be the ideal computational machines for doing programming and compilation, the concept of a virtual machine to run intermediate bytecode became necessary. You can develop and compile your Java programs on the machine of your choice, and execute the resulting bytecode on any other machine having a JVM.

Java also uses a familiar C/C++ style syntax which is widely used. Java itself has become one of the most popular languages since its inception in the mid-90s. The good side effect is that there are a huge number of jobs, books and reusable code available for the Java programmer.

History of Java

Around 1991, James Gosling and his team at Sun Microsystems were trying to create a new programming language for use in appliances like televisions and toasters. The name of this project was *Oak*. While the concept of programmable devices did not take off back then, the language was kept and subsequently used in programming on the web. This language is now called *Java*.

1.3 Getting Java

When you go to download Java, you are faced with two variants of it: the **Java Development Kit (JDK)** and the **Java Runtime Environment (JRE)**. The former is used by people wanting to write programs in Java whereas the latter by those who just wish to run Java programs. Thus for the purpose of this book, we will be needing a JDK.

The JDK itself comes in 3 *editions*.

1. ***Java SE*** - Java Standard Edition
2. ***Java EE*** - Java Enterprise Edition
3. ***Java ME*** - Java Mobile Edition

We will be focussing on Java SE since the intended target machines for this edition is personal computers. As of the date of this text, the current version of Java is Java 8 but any version upwards of Java 6 should serve you fine. While the step-by-step installation of Java SE for all the major operating systems is beyond the scope of this book, it is a fairly simple procedure much like the install process of any other package on your choice of OS. You can download your copy of Java SE from the Oracle Technology Network[1] website. Installation instructions are also available from this URL.

Another thing you would need is a good text editor for editing Java source code. Choose any editor that you are comfortable with, preferably one with syntax highlighting (colors different words in source code with special meaning). If you are yet to decide on one, refer to the Appendix: Code Editors and Integrated Development Environments at the end of the book to get a list of suggested editors.

1.4 Writing your first Java program

The first program we are going to try out will simply print out the words “Hello World!” on the screen. This is a common first program to teach and is a good way to learn about basic program structure. Fire up your editor and enter the text given below saving the file as ***HelloWorld.java***.

[1]http://www.oracle.com/technetwork/java/javase/downloads/index.html

Listing: the source code of our Hello World Program

```
class HelloWorld {
        public static void main(String[] args) {
                System.out.println("Hello World!");
        }
}
```

Note the case (capitalization) of certain words in the program. Also note how we have *indented* certain lines to give the appearance of a nested structure to our listing. While such indentation is not necessary in Java (to a compiler), it is done for human readability and is not considered optional by other programmers.

To run this program, we go through a two step process. Firstly we need to compile the program using the Java compiler *javac* and then run it through the interpreter which is called simply *java*. Look at the commands entered below alongwith the output of our program shown on the last line. Run the same on a console (command prompt) using your choice of directories and remember to save your program in the same directory.

Figure: output of our Hello World program

```
d:\code\experiments\java>**javac HelloWorld.java**
d:\code\experiments\java>**java HelloWorld**
Hello World!
```

If you enounter an error like *'javac is not recognized as an internal or external command'*, you have probably not added your JDK *bin* directory to your **PATH** variable. Check your OS manual or check online documentation on how to achieve this. If the first step ran successfully, you would have a file name **HelloWorld.class** in your directory. Be careful not to include the *.class* extension when running the program through the Java interpreter in step 2. If all goes well, you should see the text printed on the command line. Congratulations!

1.5 Analyzing your first program

A full explaination of all the components of this program listing goes beyond the scope of this chapter. For now it is important to study it as a taste of what Java programs look like. The first line declares the title of the program as ***HelloWorld*** and this must be done in the same casing (uppercase and lowercase) as the name of your *.java* file.

The second line declares the ***main***, which is where the program starts running. Notice how this is within the program title's curly braces. It also has its own set of curly braces, and within it we encounter the line which actually produces our desired output. The ***println*** is a way to tell Java to output something on the screen. We terminated this line with a semicolon and finally wrapped up our code listing with the closing curly braces which we had opened before.

Syntactically Java resembles the C/C++ family of programming languages. It is from them that it gets the concept of terminating a statement with a semicolon. Nesting of curly braces define heirarchy and a set of curly braces are used to combine multiple statements into a block. A program terminates when the last statement of the *main* completes its execution.