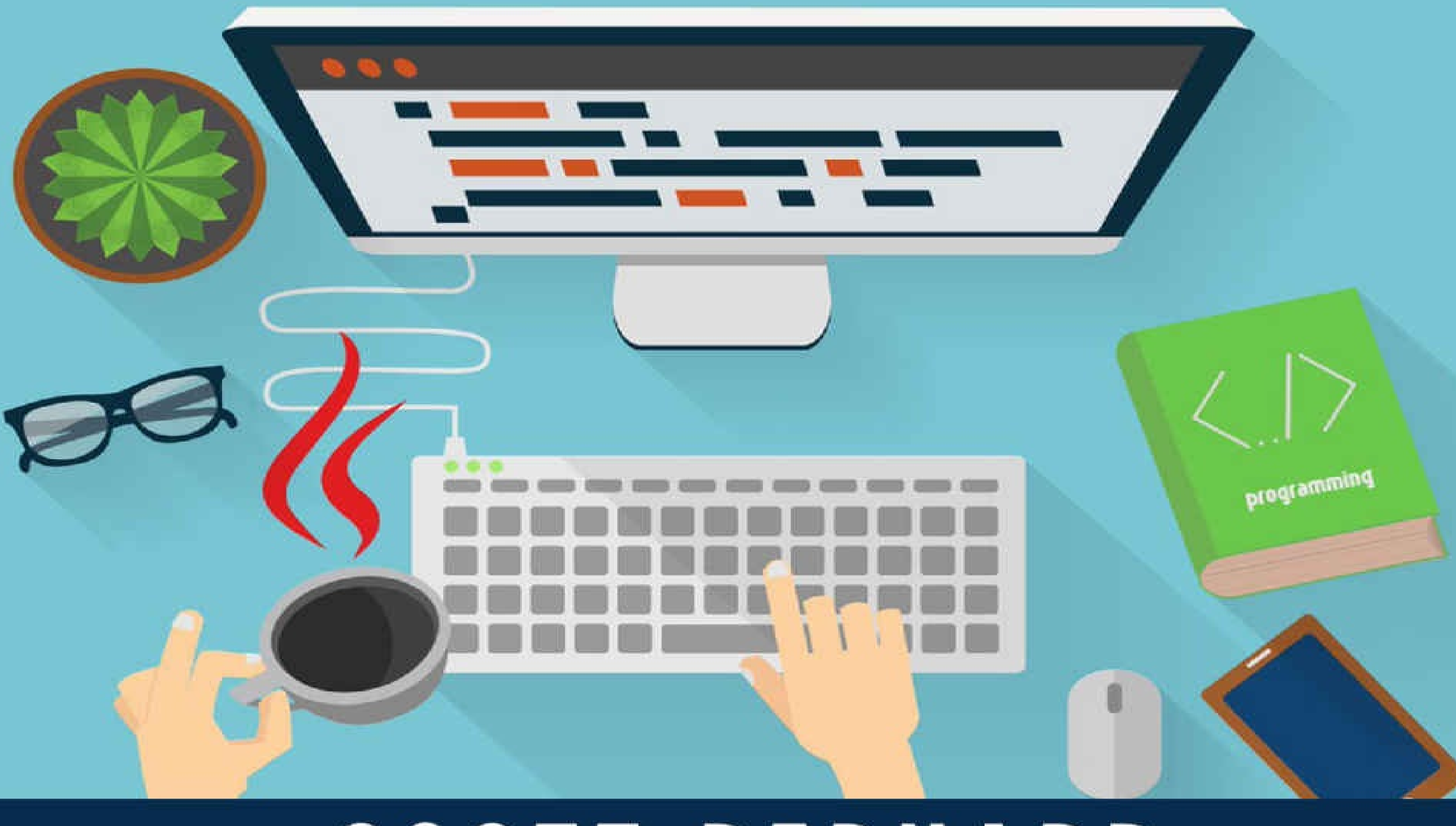
JAVA PROGRAMMING

AN INTRODUCTION, HISTORY, AND THE FUNDAMENTALS FOR CREATING YOUR FIRST PROGRAM



SCOTT BERNARD

JAVA Programming

An Introduction, History, and the Fundamentals for Creating Your First Program

TABLE OF CONTENTS

INTRODUCTION

Congratulations on downloading *JAVA Programming: An Introduction, History, and the Fundamentals for Creating Your First Program*, and thank you for doing so!

The following chapters will discuss the fundamentals of JAVA programming, its uses, and how to start writing JAVA code for yourself.

There are plenty of books on this subject available on the market, so thank you again for choosing *JAVA Programming*! Every effort was made to ensure this book is full of as much useful information as possible. Please enjoy!

CHAPTER 1

An Introduction to JAVA

JAVA is a type of programming language that was designed specifically to be used on the internet. It is easier to use than its counterpart, C++ language, while utilizing a model of programming that is object-oriented. JAVA may be utilized in order to create entire applications that may be distributed among clients and servers in a network or run on a single computer. JAVA can also be used build a small applet, or a small application module, to be used as part of a Web page. Applets enable Web users to interact with that particular Web page. JAVA programs can be found in mobile devices, desktops, Blu-ray Discs, servers, and smart cards.

JAVA was originally known as Oak when it was first created. Oak was created by James Gosling in the year 1991. James Gosling worked for a company at the time known as Sun Microsystems and the original purpose for Gosling's programming language was to be used in small electronics such as televisions, toasters, VCRs, and so forth. The purpose of JAVA since its beginning was to be fast, efficient, small, and portable. Sun Microsystems renamed the programming language and introduced JAVA in 1995. After this introduction, the interactive capabilities of the World Wide Web grew exponentially. A JAVA virtual machine is included in all major Web browsers. Likewise, almost every major operating system developer has added JAVA compilers as a part of what they offer in their products, major operating system developers being companies such as Microsoft and IBM. JAVA is now owned and maintained by a company called Oracle.

So how did the name JAVA come to be? The name JAVA came into existence when James Gosling, along with his team members, were developing the JAVA language and happened to be drinking copiously high amounts of coffee. The JAVA team felt that, because of the large quantity and high quality of the coffee they happened to be drinking, they were then able to better develop a higher quality of a programming language. The coffee that James Gosling and his team were drinking had its own very special role in the development of the JAVA programming language and happened to be distributed to the entire globe from a place with the moniker "Java Island". Essentially, Gosling and his cutting edge team were inspired to christen the brand new programming language with the name of the place where their caffeinated nectar originated. Hence, the symbol for the JAVA programming language is the unmistakable cup and saucer.

The top 10 facts you need to know about Java as you get started reading this book

1. The term Java refers generally to a trifecta combination of these things:
    - The programming language known as JAVA
    - The JVM, or JAVA Virtual Machine, which is the JAVA platform. The JVM is what

runs Java bytecodes that have been compiled, usually calling on a set of standardized libraries specific to JAVA. These libraries include those provided by the Enterprise Edition, referred to as EE or Java Standard Edition, which is referred to as SE. Though the platform for JAVA and the JVM are designed to work hand in hand, the language itself does not necessarily imply the use of JAVA Virtual Machine, and vice versa as well.

- The Java Virtual Machine (JVM) refers to a virtual machine operating at a very high performance level, and executes bytecodes on a very specific computing platform. The JAVA Virtual Machine is abbreviated and is simply referred to by programmers typically as JVM.

2. The majority of the higher layers of the widely used mobile Android operating system, or OS, is built using JAVA, so it is imperative that you learn JAVA if your goal is to eventually build applications for the android operating system. With JAVA programming, you are able to build games, embedded systems, websites, and desktop applications, which may have similar versions of those programs on mobile devices and other devices as well.

3. There are 4 different JAVA platforms available:
   - The JAVA Micro Edition. The JAVA Micro Edition contains the libraries and frameworks needed in order to build the applications that run on smaller devices, known as micro devices, such as tablets and smart phones.
   - The JAVA Enterprise Edition. The JAVA Enterprise Edition is comprised of the libraries and frameworks that are required to build applications which are utilized mostly by enterprises.
   - The JAVA FX Edition. The JAVA FX Edition is comprised of graphic libraries that are used to operate consistently across a wide variety of platforms and also to build strong client applications.
   - The JAVA Standard Edition. The JAVA Standard Edition is comprised of all of the core JAVA functionalities and all of the core JAVA libraries. This is the edition that you will be using to learn JAVA programming from in this book, and also utilizing when or if you continue to pursue your programming education.

4. The JAVA programming language is an object oriented programming language, referred to by programmers as OOP, and operates on a high level.

5. The JAVA programs are run on a special dedicated software known as the JAVA Virtual Machine, referred to as JVM in the programming world.

6. The JAVA virtual machine, or JVM, is supported by a high number of operating systems. Because of the fact that JAVA can be run on so many operating systems, this makes JAVA a cross-platform language. This is where the popular phrase “write once, run

anywhere”, or WORA, originates from. This term is also sometimes known as “write once, run everywhere”, or WORE.

7. You will need to download and then install the JAVA runtime environment, also known as the JRE, in order to install the JAVA virtual machine and run JAVA programs. Note that the JRE cannot be used to develop JAVA programs. The JRE can only be used to read JAVA programs.

8. In order to access the tools necessary to the development of JAVA applications, however, you first need to download and install the JAVA development kit. The JAVA development kit is also referred to by programmers as the JDK. The latest available version of the JDK is version 8. However, you will not need to download the JRE if you download the JDK 8 version, because you are able to freely redistribute the JRE in that version with your application, in accordance to the JRE licensing terms.

9. The acronym IDE refers to a software suite that is known as Integrated Development Environment and which provides the people who program computers the necessary comprehensive tools needed in order to make software writing, testing and overall development possible. Normally, a modern IDE will be made up of a debugger, a source code editor, and an interpreter or compiler. The software developer utilizes a single graphical user interface, referred to by developers as simply a GUI, in order to access those three tools. There are two very popular JAVA IDEs that are also completely free – an IDE known as Netbeans, and an IDE known as Eclipse. In this book, you will be shown how to program utilizing the Eclipse IDE.

10. A great resource of learning all about beginning your JAVA programming experience is right in front of you - this book!

JAVA Features

The major features of JAVA are:

- JAVA’s code is robust. This means that the JAVA objects may contain zero references to other known objects, or data that is external to themselves. In contrast, programs written in C++ and other languages do not have this ability. This process makes sure that the instruction does not contain the address of the stored data in the operating system itself, or that of another application. Either of those occurrences would effectively cause the operating system or the program to crash, or terminate. In order to ensure integrity, the JAVA virtual machine, or JVM, has a number of checks in place.
- JAVA applets have other features that are in place in order to make it work as efficiently as possible, in addition to being executed not at the server, but instead, at the client.

- JAVA is much easier to learn than C++, although the language is certainly not learned overnight.
- Programs created in JAVA are portable in a network. This means that these programs can be used in other operating systems than the one that created it without needing to perform a complicated, time consuming, major rework. The source program is compiled into what JAVA refers to as bytecode. This bytecode is able to be run in any network on a client or server that has a JVM. That JVM then translates the bytecode to a code which will then have the ability to be run on the actual computer hardware itself, which means that distinct differences in platforms, such as the length of instructions for example, are able to be recognized and dealt with locally - as soon as the program is being executed. Basically, JAVA ensures that it is no longer necessary to produce platform- specific versions of your software program.
- JAVA programming is what is known as object oriented. This means that a single object is able be a part of a group of other objects as well as inherit code which is common to that class, among other numerous features. Rather than the traditional procedural "verbs", objects are referred to in terms of "nouns" which someone is more likely to relate to. A method may be looked upon as one of the object's behaviors or capabilities.

More about object oriented programming (OOP)

Object oriented programming, referred to by programmers as OOP, is a model of language that is built around data instead of logic, and also objects instead of actions. After the input data is processed it then gives output data. OOP takes the view of the object of interest to be worked with over the logic that is needed in order to make them work. These objects, for example, can vary from computer widgets to a building and its floors to human beings.

In OOP, the first step is to locate all of the objects the programmer wishes to manipulate, and then to identify how these specific objects are able to relate to each other. This process is usually referred to as data modeling. Once a programmer has selected an object to be identified, the programmer would then generalize it. For example, this one particular book would then stand for all books. This generalization defines the type of information that the object holds as well as the logic sequences which are able to work with it. Every separate sequence is referred to as a method. Objects within OOP communicate with messages, which are interfaces that are clearly defined. The rules, parameters, and concepts utilized in OOP offer these benefits that all programmers need to be aware of and grasp:

- The data class concept makes it possible to define object subclasses which have all or at least some of the key group characteristics. This particular quality of OOP, called inheritance, reduces development time, inheritance ensures more acutely accurate coding, and inheritance also requires a more thorough data analysis.

- A class defines only specific data. When that object or group is running, the code will not mistakenly access data from the program that is not intended. This particular conceptual characteristic known as data hiding avoids unintended data corruption and provides greater system security.
- Since the definition of a certain class is not only able to be reused by a program that it is especially built for but also by other programs that are object oriented, it is then more efficiently distributed for network use. (Refer to the write once, use anywhere concept)
- The data class concept enables a programmer to build any kind of new data that has not already been defined within the parameters of the language itself.

Getting to understand the use of applets

JAVA produces browser run programs which are called applets. Although applets are becoming more and more obsolete as time goes on, they are used to facilitate object intercommunication by Web users and graphical user interface, or GUI. Before there were JAVA applets available, Web pages and sites were normally not interactive and otherwise static. Such competing products such as Microsoft Silverlight and Adobe Flash have made JAVA much less popular, since it once dominated the market. JAVA runs its applets on an internet browser using the JVM. The JVM then translates the specific code into what is known as native processor instructions, also known as native code. This translation allows for indirect platform program or indirect Operating System (OS) execution. JAVA Virtual Machine supplies most of the components that are necessary in order to run bytecode. This bytecode is normally significantly smaller than other programming languages in which executable programs are normally written through. Basically, when used online, JAVA allows for applets to be used and also downloaded through a browser. This process allows the browser to access a feature or perform a function that would normally not be available without the applet. The program or applet must be downloaded or installed by the user before the user is able to fully access and use the JAVA program.

How JAVA differs from JavaScript

While both JAVA and JavaScript are programming languages used to develop features or applications on a Web page, it is highly important that JAVA should not be confused with JavaScript. JavaScript originated at Netscape and is a little easier to learn than JAVA. JavaScript is also interpreted at a higher level than JAVA, meaning that JavaScript uses a programming language with a strong abstraction, or technique for managing the complexity of computer systems, from the details of the computer. JavaScript, however, does lack the speed of JAVA bytecode as well as some of the portability. This means that the running speed of JavaScript is considerably much slower than that of JAVA. JAVA applets are able to run on just about any operating system without necessitating recompilation and have no variations or extensions that are unique to an operating system. This is what makes JAVA generally thought to be the most strategic language in developing Web applications. JavaScript, however, may be useful in extremely small applications that run on either the Web server or the Web client.

By far and wide, the most highly recognizable difference between JAVA and JavaScript is the type of applications in which they are used to create. JAVA programs are utilized for applications which are either initiated through a Web page or run from a computer's desktop. These programs are standalone programs that usually open a completely separate program window. In addition, a computer is absolutely not able to run JAVA applications without JAVA having been installed. JavaScript is always included in all of the up to date browsers, however, and will start the JavaScript on a Web page when the Web page is loading, as long as JavaScript has been enabled.

When a programmer is using JavaScript code, JavaScript's “thinking” and calculations are consistently always performed on the client side. The client side refers to the computer where the page on the Web is initially accessed. On the other hand, JAVA programs do normally perform all of the main “thinking” and the calculations process within a certain JAVA applet that must first be downloaded, or on the server side. Because of the fact that JavaScript runs on the client side, it usually runs faster than JAVA, and is at times even almost instant, depending on the speed of the connection. Since JAVA programs are run on the client side, it takes just a bit more time in order to process, but usually only several seconds or more.

JavaScript uses significantly less memory than JAVA, even very, very little memory in some cases in fact, in order to do its processing and also perform correctly. Because of its lower memory requirements and its capability of offering so many varied Web page features, JavaScript is a very common program language utilized in many various pages on the Web at this present time. JAVA programs, however, may sometimes require that a lot of computer memory be used in order to function properly. This requirement can absolutely cause another program to run at a much slower rate and even cause the computer to slow down altogether. So even while JAVA programs have the innate ability to be developed to do various, often powerful things because it utilizes a higher memory usage, it can also be a speed disadvantage in some respects.

CHAPTER 2

How JAVA is Being Used Today

As of 2016, JAVA is now being utilized by over 9 million software developers around the globe and is also one of the most utilized programming languages available especially for client-server applications. Regardless of a computer's architecture, the applications of JAVA are able to be compiled into code which runs on all JVMs. JAVA used to be identified mainly with slow performance, with having to wait for Sun Microsystems to drop upgrades, with bytecode interpretation, and with the use of Applets. Today however, JAVA has become more associated with reliably high performance, with dynamic compilation of hotspots, with a more and more independent open source community, and with service oriented architecture, or SOAs, with web services, and with web applications. This chapter will go into depth on the uses of the JVM, the JAVA programming language, as well as JAVA platforms.

Using JAVA Programming Language

The JAVA programming language operates on a high level and was inspired by Smalltalk, and by both C and C++ programming. The JAVA programming language has also borrowed a few ideas from other languages. The syntactic design of JAVA programming language was made to be familiar to those already versed in "curly brace", C-descended languages, but has stronger (at times arguably) OO principles than the principles used in C++. Also, garbage collection is fairly automatic, which does away with the need of the developer to have to free up memory used by objects that have become obsolete and are no longer in use.

JAVA is philosophically referred to as a "fail early" language. Since it has syntax restrictions, most programming failures are just not actually possible with Java at all. Because of the fact that JAVA does not have access to pointers directly, errors specific to its arithmetic simply just do not exist. Utilizing a certain object as a type other than what it originally was named as requires a straightforward cast conversion, which allows the code compiler the obvious chance to deny programming that may be illogical and therefor unable to be used.

A varied number of Java enterprise frameworks make it necessary to use the deployment descriptors or configuration files, which are usually written in XML in order to identify a function. This could include the order of the steps that are needed to execute within a rule engine or which class may handle a particular request in HTTP. In other words, they must go beyond the language at hand in order to implement their function properly. JAVA 5.0 adds footnotes to this language, and this allows for classes, fields, and methods to be tagged with values that are then able to be developed and inspected at the time they are run, and most often via reflection. A lot of software programmers prefer these footnotes, usually referred to as annotations, because they are able to simplify the things that may be otherwise called out by certain descriptors or other ways. They are also able to make it difficult to interpret the Java code, though. Whether or not a footnote or annotation is in place may actually affect the execution of the code and also may affect it in ways that may not be completely obvious from the footnote or annotation itself.

Aside from this one singular criticism, JAVA is mainly regarded as the most popular general-purpose computer programming language utilized in the world at the present. The benefits of JAVA programming language are extremely vast. The main benefits include but certainly are not limited to:

- The benefit of a vastly immense knowledge base.
- The benefit of a vastly enormous amount of software developers who are always readily available.
- The benefit of free tools which are widely available on numerous platforms such as Windows, Solaris, Linux and Mac. All of these platforms are able to execute and compile all Java applications.

The JAVA programming language itself hits a unique and valuable point in the quid pro quo between code performance and software developer efficiency. While CPU (central processing unit) cycles continue to steadily reduce in cost factor, software developers, however, usually do not continue to reduce in cost. If anything, the cost of software developers continues to rise for many reasons, among them being rising inflation and an increase of technical education costs. Because of this fact it may inevitably lead to the acceptance of yet an additional piece of abstraction inbetween the CPU opcodes being executed and the software developers themselves – as long as it allows for the software developers to be able to create a higher quality software and at a much more efficient rate.

Explanation of the use of JAVA platforms

JAVA is regarded generally in terms of three different platforms. These three platforms were briefly regarded previously in this book. Note that the JAVA FX Edition is not included here.

1. The Enterprise Edition, also known as JAVA EE
2. The Standard Edition, also known as JAVA SE
1. And
2. The Micro Edition, also known as JAVA ME

Each of these different platforms utilizes a combo of a specific group of libraries, virtual machine, and a language version that is used to execute its code. The EE version contains everything that the SE version does and more, so that means that any EE application can access and utilize the entirety of the SE libraries. Also, the EE's use of the programming language is identical to that of the SE's as well. This means that the SE version is a subset of the EE version.

JAVA Micro Edition, or ME, is significantly different than its counterparts because of the fact that a small device (such as a smart phone) are just simply that much more limited on space and capabilities as well. The Micro Edition is not able to be considered as a subset of the Standard Edition (therefore, cannot be considered a subset of the EE), because of the fact that a number of its libraries only exist in the Micro Edition. In addition, the Micro Edition version completely eliminates some of the programming language features because of the computation limitations of the platform on which it is run. Examples of some such eliminated features include Float class and the float primitive. This means that Micro Edition requires a slightly different set of tools than the Enterprise Edition version and the Standard Edition version. With such deep seated differences in the devices, it makes the complete portability of code much less realistic in the micro space. In fact, a lot of Java developers view ME as a completely different animal altogether, almost a separate entity even.

What to expect when using the JAVA Virtual Machine (JVM)

JAVA source code has to translate into executable platform native code at some point along the way. Typically, doing so requires the use of a two-step process:

1. The software developer needs to compile the source code to JAVA code.
2. The JVM then has to translate that information, for the platform in use, into that platform's native code.

The second step was originally executed by interpretation, or by taking each single JAVA Virtual Machine (JVM) instruction and converting that instruction in an instant to one or more instructions native to the platform. Then after that is accomplished, the just in time compilers, referred to by software developers as JIT compilers, convert the program from bytecode on the JVM into code native to that particular platform as the program is run. There are several ways to achieve this conversion at this present time. Sun Microsystem's HotSpot compiler initializes this process by translating the code and then profiling it at runtime, optimizing and compiling the particular areas that have been concluded to be the first priority in order for that program to continue to work efficiently. IBM's "mixed mode interpreter" of its JVMs operates in a very similar fashion. These procedures circumvent the startup performance lag entailed by utilizing JIT compilers on the whole program, but then this also means that performance is accumulated, arriving consistently over a period of time, when crucial sections of code are detected and made to work more efficiently. Because of this procedure, client applications are far less benefitted by this approach than the long-running server processes.

What to expect when using JAVA without using the JAVA Virtual Machine

It is entirely possible to run JAVA without the use of a JAVA Virtual Machine. Since JAVA source ends up becoming bytecode, then which in turn becomes platform-native code, as it stands, this can actually be accomplished all at one time. The GNU (GNU recursively refers to the term "GNU's Not Unix") Compiler for JAVA, also referred to as GCJ, allows for an up front, one-time, compendium of JAVA code to a command that is able to be executed on a singular platform. Though there is enough information to compile command line and server side applications when using this process, it is not actually able to allow for the support of the

Abstract Windowing Toolkit (AWT), which would obviously render it unsuitable for AWT programming and also Swing GUI programming as well.

This obvious downside to using this process is that cross platform code ends up becoming bound to only one platform and only in one single step. What is more, that static type of compilation does not automatically negate the workings of the HotSpot's dynamic compilation. The author of said compilation did work on a piece of research in which the GCJ performance gain was concluded as being fewer than a mere 5% above the performance gain of the HotSpot version. Even still, GCJ does have the performance gain advantage and has the ability to solve crucial problems, such as the ability to succinctly deploy a particular Java application that is completely able to be ran regardless of which version the JVM is running and whether or not it’s even available.

What to expect when using the JAVA Virtual Machine without JAVA

It is also absolutely possible to get straight to the JVM level immediately and successfully bypass the JAVA language entirely. C-to-JVM bytecode compilers are already in existence; compilers like the Axiomatic Multi-Platform C, which is commercial and supplies a partial version of ANSI C. What is more, the progress of the manipulation of JAVA bytecode allows for the creation of classes at runtime that are completely executable in Java applications. This means that what you will be working with will no longer be JAVA, but instead will be an effective form of programming language used for assembly in the JAVA Virtual Machine.

Explanation of the JAVA Community Process

There is a JAVA community that exists beyond virtual machines, beyond libraries, and beyond programming language. In spite of the overwhelming amounts of software that is considered to be open source and utilized by and written in the JAVA programming language, there is a continuously obvious and open conflict at large between the open source community and the JAVA community.

The JAVA Community Process, sometimes referred to by software developers as simply the JCP, was established in 1998. The JCP is the formalized mechanism by which any interested parties may be allowed to offer input in the development of standard technical specs of the JAVA technology. It is open to the public and anyone may become a member of the JAVA Community Process by simply submitting the appropriate form available at their website.

A hugely varied number of JAVA aspects exist entirely without the JCP’s standardized acceptations. A lot of independent projects have gained in size enough to actually compete with the actual JCP benchmarks in the level of awareness in the minds of JCP consumers. As a whole though, these independent projects have also been able to be quick in adapting to modifications completely without the use of JAVA and also simultaneously utilize the best features of those modifications, as in the feature AJAX, which is a feature that simplifies code. This is a Direct Web Remoting, or DWR project or the Rails-inspired Trails project.

CHAPTER 3

Introduction to JAVA Programming Basics

When it comes to learning JAVA programming language, or any programming language for that matter, there are five basic concepts you must understand before you get started. These five basic concepts include:

1. Variables
2. Data Structures
3. Control Structures
4. Syntax
5. Tools

Each of these five concepts will be thoroughly explained in this book on a beginner's level, to ensure that they are understood.

Variables

Variables are actually the cornerstone of any and all JAVA programs you will run into, and so they are the cornerstone of the JAVA programming language in and of itself. By definition, a variable in computer programming is simply the location of storage as well as the associated name that symbolizes what an unknown or known piece of information or quantity may be. In other words, a particular value. Simply put, a variable is a method of storing some kind of information for use at a later time, and is retrievable by referring to a name or word which will describe the said information.

For example, let's say you choose to navigate to a certain website, and the very first thing this website does is ask you what your name is. This is normally done in order to instate some form of human familiarity – upon your next visit, that website will call you by name. The person who built the website would create a small text box on the screen that would ask you for your name and that small text box would represent a variable. The person who built the page may decide to call that small text box something like "Visitor Name", and that would be the symbolic word or name for the small text box variable.

So then after you type your name into the small text box, your name is then stored as information in a variable called "Visitor Name". This information would be made readily available to the person who had built the page, or the programmer. Then this person would have the ability to revisit the page and ask "What value does the variable "Visitor Name" contain?", and the program would answer the programmer with the value of whatever you may have typed into that small text box that you saw when you first visited the page.

This concept is used constantly over and over again throughout JAVA programming and it is also super powerful in its programming. This particular concept is what makes Twitter and Facebook work; it is what allows you to pay your bills using your online banking or banking app, and it is also what makes it possible to place a bid on sites like eBay. Variables enable the programming world to keep spinning ‘round, as it were.

Okay, now it is time to get a little bit more specific. There are different types of variables when it comes to programming using the JAVA programming language. If a programmer were to choose to store your name in a variable, your name would be stored as a type of variable called a String. Or, perhaps the programmer also wanted to store your age - that would be stored as a type of variable known as an Integer. Finally, maybe the programmer wanted to store your yearly income – your income would then be stored as a type of variable referred to as a Double. Just to recap, there are three different types of variables being covered here:

- String
- Integer
- Double

So, what exactly constitutes a String variable, an Integer variable and a Double variable? When using JAVA, the programming language needs to know what sort of information you will be going to store within a particular variable. The programming language needs to know this because JAVA is what is known as a strongly typed language. Basically, in order for a language to be weakly typed, that means that the types of all of the variables are inferred, or known at the time of compilation. A strongly typed language, however, does not allow for the use of one type as another type, meaning that it utilizes different types of variables. This is where string, integer, and double variables are put to use.

String Variables

Typing in JAVA programming lets the programming language know for sure that the information that is being stored within a variable will be concretely defined. When a string is referred to in JAVA, we are looking at the data as if it were simply a sentence comprised of words in the English language. A String only specifically represents letters placed in a specific order. As in the English language, string variables constitute a series of letters placed in a particular order which gives that series of letters a very specific meaning.

In the way that string variables are made more understandable by comparing it to everyday language, adding two strings together works very much the same way. If you have two sting variables and they stored, for example, the data “Visitor” and “Name”, if they were added together you would get the String: “VisitorName”.

Integer Variables

In JAVA, an integer variable means that you have a number that does not include decimal places. This means a whole number such as 23 or -784. In Java, when it is specified that a variable is an integer, it is simply not allowable to store anything but a whole number.

For example, if you want to add two numbers together, the number 35 and the number 5. Java actually behaves really differently, it depends upon the exact type of variable being used to storing this data. Then adding 35 and 5 together will result in the integer 40. Instead of just compounding the two, as a string would (when 35 and 5 are added together in a string, you would get 355), the integers are added together in simple math.

The Double Variable

The double variable in JAVA will either use very big numbers or very little numbers. The minimum values as well as the maximums are respectively both 17 and then 307 zeros.

A double variable can be used to also contain point values that are known as'floating'. A floating point value is basically any numerical value with a decimal point.

The bottom line is that having a type will help you to start to understand what sort of things you are able to do with the information contained within the variable. Types of variables are really powerful and make sense of what is and what is not allowed within a certain variable group.

How data structures are used in the JAVA programming language

A data structure in computer science is a specific method of organizing and storing information in a computer so that it may be used in the most efficient manner possible. To better explain data structures, let's use the concept of a list of contacts. Usually, a list of contacts contains numerous contacts which could shrink or grow at any time. For this example, let's say you need to keep five contacts in order. If you were to attempt to represent those contacts as variables in a program, you would need to know data structure.

In this instance, you have a list of contacts. When using JAVA programming, there is a data structure called a list. This is what the code looks like:

1. List contacts = new ArrayList();
   Do not worry about the symbols just yet, that will be covered later in chapter 5 of this book. All you need to know at the moment is that there is a way to store a list into a data structure. In a List data structure you can easily both add items to your list and also remove items from your list.

```
contacts.add("John Doe (john.doe@somename.com)");
contacts.add("Jane Doe (jane.doe@somename.com)");
```

When creating a data structure, it is supremely more efficient to create one variable, as shown above, instead of a different variable that you would need to write out for every single item on your list. Because you only created one variable, contacts.add(RandomContact) that means that your code is more dynamic and flexible. In this instance, dynamic refers to the fact that the outcome of the program is able to change depending on which variables are inputted. You ideally want your code to be rendered as dynamic as possible, and you also want your code to have the ability to handle a lot of different situations without needing to have to keep writing more and more code, over and over, as time goes on. Basically, data structure is just a way to avoid having to create more variables than is absolutely necessary.

How control structures are used in the JAVA programming language

A control structure is a block of programming that analyzes variables and chooses a flow based on given parameters. It is the basic process where decisions are made in computing. Flow control regulates how a computer will respond when it is given specific conditions as well as specific parameters. This means that while a particular program is running, the code is read by the computer line by line. This process is known as "code flow". As the code is read from top to bottom, it may reach a point that requires some amount of a decision making process. This decision could result in the code jumping to a different part of that specific program, and it may require that a certain piece of code is rerun, or it may just skip a lot of code in response. This process could be equated to a 'choose your own adventure' book. You reach page 12 of the book, and you have a choice between selection A and selection B. A computer program works much the same way except the program has a strict set of rules to abide by, given by the programmer. The decision that the program makes effects the flow of code, and that is known as a control structure.

How Syntax is Used within the JAVA Programming Language

The syntax of the programming language of JAVA is really just simply the set of rules which define the combinations of symbols considered to be precisely structured programs in that set language. Syntax is basically a particular layout of symbols. In JAVA, an example of this would be curly brackets { } or round brackets ( ). For example, when you look at an email address you are able to identify the fact that it is an email address because of the syntax. You may also identify this as its structure or simply by 'the way it looks'. You are able to recognize an email address and differentiate it from a website address because of the different syntax that lies therein. Syntax in JAVA programming language is similar. There are rules in place, and when followed, those rules allow programming language to comprehend the functioning software and also allows you to create functioning software. You will receive errors if you do not abide by the rules of a programming languages' particular syntax.

When using JAVA, there are four specific steps to the syntax of creating a variable. They are as follows:

1. The first part to the syntax of creating a variable is the use of the word String, which is the variable's type. A String in this particular case allows the storage of regular letters as well as the storage of special characters.

2. The second part to the syntax of creating a variable is using the variable name. A variable name can be made up of numbers and letters, but the only special characters they are allowed are underscores. The first letter in variable names are normally lower case, which is not necessarily a must, but is kind of a rule of

thumb in the JAVA programming world.

3. The third part to the syntax of creating a variable is implementing the value that the particular variable holds. In JAVA strings are clearly defined by placing quotes around regular numbers, letters, and special characters.

4. The fourth and final part to the syntax of creating a variable is implementing the mark of completion. In JAVA, a semi-colon is used. Although there are a few exceptions, nearly all of the lines of a JAVA code will end in a semi-colon; much like a period at the end of a sentence.

Explanation of the use of tools in the JAVA programming language

A tool that is used in programming is a lot like any other tool that we as people use in that it helps you accomplish your goal more efficiently – except that JAVA tools are only useful to JAVA programmers, of course. In JAVA, a tool is a piece of software that enables you to get your program completed more quickly. There is a vast array of tools used in JAVA programming, but for all intents and purposes, this book will focus on the IDE.

An IDE is essentially a piece of software which will basically make coding a lot easier and is the most essential tool of every programmer on the planet. IDEs check the syntax of a code to make sure there are no errors. IDEs also organize your files and give you a specific way to comprehensively view them. IDEs tend to incorporate code completion, which actually fills in code for you, in some common scenarios. IDEs enable easy navigation through code. There are many more advantages of using an IDE, though the most useful are noted here, and you will become more familiar with using an IDE once you set yours up following the simple step by step guide outlined in chapter 5 of this book.

CHAPTER 4

Familiarizing Yourself with JAVA Commands

The vast amount of JAVA commands are far too numbered to include every single one in the list here, but this chapter includes a visual of some of the most common and useful commands available to date. The diagrams in this chapter are supplied as an introduction to the name of the most basic commands and also as a visual reference as you make your way through the next chapter, so you will have numerous ways of checking your work and making more sense of what you have read thus far. These JAVA commands are in no particular order really, with the only exception being the very first command explained in this list, hello world. The Hello World application is by far and wide the most common starting point for every single newbie programmer in the world. There are literally millions of different “Hello World!” application codes out there, written by just as many beginner programmers.

An introduction to the Hello World application code in the JAVA programming language

There are three primary components that make up the application known as “Hello World!". These components consist of the HelloWorldApp main method, the class definition, and the source code comments. Unless you are already familiar with how to program and utilize commands in JAVA, the further implications of “Hello World!” will be able to be understood only after you have completed reading the rest of the “Hello World!” programming material in this book. Below, the keyword “class” (beneath the first arrow in the first line) launches what is known as the class definition which gives meaning to the class “name”. It also launches the code for every individual class which appears in between each set of the opening and closing curly braces in the diagram below. For those of you who may not already be familiar with what curly braces are (“{“and “}”), they are a symbol used to execute code in JAVA. In JAVA, every single application will begin with a class definition, as you will see in the following chapter. Also, when utilizing the JAVA programming language, a “main” method must be used in every application. Lastly, source code comments are only useful to other programmers in order for them to more fully understand the code. The source code comments are completely ignored by the program compiler.

An introduction to Integer variables in the JAVA programming language

When you are writing code with the JAVA programming language, every single variable first has to be declared prior to its ability to be used in that code. When working in JAVA, this is referred to as being statically typed. When a programmer declares a variable that means that the variable’s name and the variable’s type are both stated very clearly in the code. When the variable's name and the variable's type are both clearly stated, this lets your program know that, in that particular code, there is an initial value. It also lets your program know that there is numerical data being held and that there is a particular field named "yournamedobjecthere" that exists within the program and code. The data type of a variable completely determines the particular operations that are able to be performed on it and also determines the values the variable is able to contain.

An introduction to using an Object in the JAVA programming language

As stated previously in this book, JAVA is a programming language that is very heavily object oriented. When a programmer is working in the JAVA programming language, then that programmer utilizes objects in order to complete the code task at hand. The programmer will create those objects, modify those objects, move the objects around within the program, change the objects' variables, call the objects' methods, and then will also combine those objects with other objects in the code as well. When programming with the JAVA programming language, the programmer will develop classes within the code, the programmer will create objects out of those particular classes within that code, and then the programmer will use those classes in the code with other classes and other objects within the code.

When a programmer is writing code in a JAVA program, the programmer will define a set of specific classes within the code. As you will learn further in the next chapter of this book, classes are simply just the templates that are used by programmers in order to create the objects within the programmer's code. The objects that the programmer chooses to create, which are also known as instances, are self-contained aspects of a particular program that also has relative data as well as features. A programmer will, for the most part, use that class simply to create and define instances within a code and then the programmer will work with those particular instances within that one code. Hence, in chapter 5 you will learn how to be able to create any new object from any class that you have created.

In order to create a brand new object, the programmer must first use a new operator along with the name of the particular class, and that class is to be used by the programmer and the IDE as a comprehensive template. Keep in mind as you proceed that parentheses always follow the name of the class, no matter what. This will be shown in the next chapter as well. These parentheses are very important, so make doubly sure you do not leave them out or else your code will not execute properly, or may not execute at all. The parentheses following the name of the class can even be left empty. If you leave the parentheses following the name of the class empty, this means that you will have successfully created the most basic object possible in your code as well as the simplest object possible in your code.

When you have successfully created an object in your code, you will probably then decide to utilize that object in writing your program. When you as the programmer decide on how you wish to use your object, you then may have to call one of the object's methods in order to perform a specific action, you may choose to use a certain value of one of the object's fields within the lines of code, or you may choose to even change one of the object's fields altogether.

An introduction to using a class in the JAVA programming language

A class is really just a simple template or a simple blueprint when a programmer is creating several different objects. This simple template or simple blueprint will act to define a class's behaviors as well as a class's properties. The JAVA programming language class objects show the behaviors as well as the properties as defined by the object's particular class. A class in the JAVA programming language is able to contain methods as well as fields in order to describe how the object is supposed to behave within a line of code and within an entire program as well.

An introduction to using String Data Type in the JAVA programming language

Even though the compiler in the JAVA programming language possesses a particularly special support for the String s, String by itself is not a primitive data type at all since String includes a vast array of characters. In that instance, String is quite simply a reference data type and a class in the JAVA programming language. For the sake of reference, examples of support for the String s include such things as performing what is known as performing String concatenation and also includes the programmer choosing to convert string literals - which is not a primitive type - into String instances, which are a primitive type. When using the JAVA programming language, String actually borrows from the C programming syntax. This means that the JAVA programming language compiler understands String as what is known as char array. Thusly, String is then only an abstract data type in JAVA that is actually made by a primitive data type char array, a data type essentially borrowed from JAVA's programming counterpart – the C programming language.

The anatomy of an if statement in JAVA

In an 'if' statement's simplest and most basic possible form, it executes a statement or block of statements if a boolean expression proves to true. A Boolean value is only one answer of either 1 or 0, either yes or no, or either true or false. Below is the syntax:

The anatomy of an if-else statement in JAVA

When an 'if' statement includes the else clause, it effectively executes a statement if the boolean expression is true or issues a block if the boolean expression is false.

The anatomy of a Loop in the JAVA programming language

In JAVA there are 3 different types of loops. There is the 'for loop', the 'while loop' and the 'do-while loop'. The 'for' loop (aptly) repeatedly executes a block of statements until the condition that the programmer wrote returns as false. The 'while' loop executes a block of statements when the Boolean expression turns out to be true and the 'while' loop is terminated when that same expression turns out to be false. The 'do while' loop is very similar to the ‘while’ loop, with the exception that it processes the boolean condition only after executing the block of the statement.

boolean and Boolean defined

First of all a ‘Boolean’ value is a value that has the option of only one of two choices. These choices are either yes or no, either 1 or 0, or either true or false. In the JAVA programming language, Boolean values have a variable type. For example: boolean user = true. Instead of typing ‘string’ or ‘double’ or ‘int’, the programmer simply types ‘boolean’ with a lower case ‘b’.

CHAPTER 5

How to Start Writing Programs with JAVA Step by Step

Because of its flexibility, JAVA is one of the most widely used programming languages in use by software developers to date. As previously stated in this book, JAVA is the core programming language that is used in the development of the apps for Android, and JAVA is really widely used in the back-end side of web development as well. If you happen to be brand new to the programming world and are looking to or thinking about making a career or hobby in this field, the following step by step beginner's guide will give you a good idea what you may expect when learning to program and write your own code. Since it is easy to use and so widely used by all sorts of programs and programmers, JAVA programming is an excellent option for the interested first time code writer, even if you do not necessarily have your sights set on a full time career in this field. The following step by step guide will supply you with a really solid foundation in the beginning basics of programming, Object Oriented Programming concepts, or OOP, a better understanding of computer science, and even get you on the right path to succeed in software engineering. This information will also successfully put you ahead of the game in the instance that a career in computer science is what you are really interested in.

1. How to install JAVA Development Kit, or JDK.
    - Open your browser and search for JAVA JDK
    - Look for an option from Oracle.com and click
    - Click on the JAVA JDK download
    - Click to accept license
    - Choose your operating system
    - Minimize everything else on your computer
    - Once it is downloaded, be ready to run the executable file
    - Click next and check status
    - JAVA JDK will start installing
    - Click next and wait for JAVA to finish installing
    - Once installed, you can go to your C folder and you should see two files, JDK and JRE

2. How to install and set up Eclipse Integrated Development Environment, or IDE
    - Type www.eclipse.org in your browser

- Opt to download eclipse IDE for JAVA developers
- Choose your operating system
- You will then be directed to choose your PC's nearest location, so click the location that is nearest to you
- You will then be prompted to save the zip file, save zip file
- Open the zip file once it is finished downloading
- Minimize everything else on your computer
- The zip file contains a folder named eclipse, extract to C folder
- Once the extraction is complete, you should be able to see eclipse in your C folder
- Open up Eclipse
- The very first time that you openup Eclipse, a question will appear that will ask you where you want your projects to be saved - choose a location or leave it at C folder default
- Click OK and eclipse should open shortly
- In order to start a brand new project, go to file, then click on new, then click on JAVA project
- Name your project
- Capitalize the first letter of each of the words in your project name
- Click next
- Click finish
- Your "My Project" folder is created

3. How to create your first JAVA code in Eclipse IDE
    - Start a new project within Eclipse
    - You will see that eclipse will also create a subdirectory for the project you create, this will contain your source file
    - Close all projects except for the project you will be working on
    - Right click on the source folder of the project you want to work on and you will be able to add packages to your folder (whenever you add a package it will add a folder to your source folder)
    - Access the package you just created and you will be able to create a new class
    - When naming your class, make sure the first letter of each word of the name you choose is capitalized
    - You will be asked which method stop you would like to create, check public static void mean

- Press finish
- You will want to include a comment section in your project; in order to do this you will first start with a forward slash, followed by an asterisk, then press enter
- This comment section is not executed by your program, it is only info you are documenting for your program
- Another way of creating comments is by first starting with two forward slashes and posting your comment after; this will not be executed either
- Whatever you write in the main class will be executed
- JAVA has built-in classes to help you program; the most important of these is called System – make sure the S is capitalized
- When you type System then a period eclipse will automatically know what to do; it should look like this: System.
- A popup menu will then appear, showing you the available methods of a label in this system class
- Select the method called out.println
- Once that is selected, you will then be able to click on it and start leaving a comment
- When you are ready to run your program, you will then right click on your project and run it as a JAVA application – or – you can click the “run my class” option, which will open the save and launch popup
- Whenever you scroll over a built-in method of class in eclipse editor it will give you a help popup which shows you what that particular code can be used for

4. Programming with variables and Types in JAVA
    - When declaring a variable in JAVA you must first select the data type
    - For example, let’s say you want to store an integer in a variable
    - Declare the data type as “int” (for integer)
    - Name your variable, say you name your variable x
    - Assign value to your variable, such as 10
    - Your line of code should now look like this: int x = 10;
    - You can also directly use your variable in the System.out.println line
    - Instead of typing a message in this line, you can put x as the value; your line should then look like this:
    - System.out.println(x);
    - When this program is run, the value assigned to x, which is 10, will be printed

where the x is in the line

- Basic data types include:
    1. a byte – which is a number and 1 byte
    2. a short – which is a number and 2 bytes
    3. an int – which is a number and 4 bytes
    4. a long – which is a number and 8 bytes
    5. a float – which is a float number and 4 bytes
    6. a double – which is a float number and 8 bytes
    7. a char – which is a character and 2 bytes
    8. a boolean – which is a true or false and 1 byte
- Byte, short, int, and long can all store a number, the only difference is their size – as indicated
- The range of all of the variables increases with byte size
- Short example: short my_variable=10;
- When using the float variable, you need to include the word float into your code: float my_decimal=(float)4.5;
- When using the double variable, it is not necessary to differentiate and should look like this: double my_double=11.52;
- Use char if you want to store any one single character: char my_char='A';
- Boolean can store true or false: boolean is_true=false;
- When printed out, only the values of all of these variables would appear, so in order they would show up as 10, 4.5, 11.52, A, and false

5. What it means to get user input using JAVA
    - There is a class in JAVA called Scanner which enables you receive input and looks like this: Scanner scan=new Scanner(System.in);
    - In order to be able to give output in the same line, it should look like this: Scanner scan=new Scanner(System.out);
    - If you want to gather input from a user, your first line of code should be: System.out.println("Enter some number");
    - You then want to name your variable and continue coding:
    - Int user_input_number = scan.
    - When you reach the part in the second line at scan., eclipse will then prompt you to make a choice, here you want to choose next int, so the completed second line will look like this: int user_input_number = scan.next int();
    - Now if you want to print this value, that line of code should look like this: System.out.println("The entered value is");

- If you want to print your output without breaking the line, first omit the ln after print in the line code, then copy your variable, which in this case is user_input_number, then paste it in the parentheses and run the program, and the very last line will appear this way: System.out.print(user_input_number);
- When you've run this program, you will notice that it asks the user to enter a value. Let's say you enter the value of 1,000
- Once you enter the value, the program will confirm the entered value
- Double works the same way, but you have to make sure you are consistent in changing the entire line of code, for example:
- Scanner scan1 = new Scanner (will appear as "System.in");
- System.out.println (Enter a decimal value here);
- double user_input_double = scan1.nextdouble();
- System.out.println("The entered value is");
- System.out.print(user_input_double);
- If you want to take text input, you need to define the variable as string
- Take the previous code and just change it scan.next line, because you are expecting a line from the user
- The entire program should then look like this:

  First line: "Scanner scan1 = new Scanner (System.in);"
  Second line: "System.out.println("Enter some string");"
  Third line: "string user_input_string = scan1.nextLine();"
  Fourth line: "System.out.println("The entered string is");"
  Fifth line: "System.out.print(user_input_string);"

- When you run this program you will be prompted to enter some string, enter a phrase of your choosing

6. Getting to know the math and arithmetic operators in JAVA
   - First, declare two variables, x and y in this instance
   - Then assign an answer
   - int x, y, answer;
   - Assign value to x and y, 20 and 30 respectively
   - Assign value to the answer, 50
   - If you want to print out your answer, the program should look like this:

     int x, y, answer;

```
x = 20;
y = 30;
answer = x + y;
System.out.println("Answer = " + answer);
```

- When an addition or plus symbol is used in a print function, it is known as a concatenation or concatenation operator
- The correct answer should be printed out in eclipse without you actually having to do the math, as long as the code is correct
- In order to perform a subtraction, all you need to do is replace the addition symbol with the subtraction symbol in the fourth line of code, or the answer line: answer = x – y;
- When performing multiplication, x cannot be used as the multiplication symbol because x is often a variable; instead, an asterisk is used
- All you have to do to perform multiplication is replace the symbol again: answer = x * y;
- Division is a little bit trickier – if you divide a lower number by a higher number and have declared your variable as integer, the answer will always be zero because integers cannot be decimals
- In the same light, when your variable has been declared as integer and you are dividing a higher number that is not divisible by a lower number without using decimals, you will not get an exactly mathematically correct answer either – if you try to divide seventy by thirty, your answer will be two because that is how many times thirty can go into seventy while having a whole number as an answer
- In order to get accurate answers in division, you must declare your variables as double
- A forward slash symbol is used in place of the other symbols when using division
- One more operator in JAVA is called the modulus operator and it gives you the remainder of a division

7. Getting to know the increment operator and the assignment operator
   - An increment operator is used when you would like to increase the value of a variable and somewhat resembles algebra
   - int x = 10;
   - x = x + 1;
   - System.out.println(x);
   - Another way of writing this code to achieve the same goal would be to replace the + 1 with two addition symbols, this is called post increment operation
   - int x = 10;

- x++;
- System.out.println(x);
- The same answer of eleven has been reached both times in print
- If you use x++ in the print line it looks like this:
- int x = 10;
- System.out.println(x++);
- In post increment operation, the value of x will only be changed after this plus operation
- Now you are going to increase the value of x by one which should look like this:
- int x = 10;
- System.out.println(x++);
- System.out.println(x);
- Pre increment operation places the plus symbols in front of the x and increases its value before the operation is performed
- So if we add ++x in place of x++ in the previous code, this is what you would see:
- int x = 10;
- System.out.println(++x);
- System.out.println(x);
- The answer to this is actually:
- 11
- 11
- This is because the pre increment operator, where x had a value of 11 and the post increment operator, where x also had a value of 11 were compiled
- Now let's say you want to add five to this value, you can do it by writing out: x = x + 5
- Another way of writing this is: x+ = 5
- If you would like to multiply the value of 5 and the value of 10 in the equation, you would only need to replace the addition symbol in x+ =5 with an asterisk – that shortcut works for multiplication as well – and it would look like this: x* = 5
- The term 'equal to' in JAVA is referred to as assignment operator
- Thus, x* = 5 and x = x*5 is an assignment operator and this operator works for addition, subtraction, multiplication and division

8. How to incorporate the if else or conditional statements and their relational operators

- A conditional statement is a statement which evaluates whether a condition is true or false, and based upon this condition executes a certain code
- A double equal sign == symbolizes relational operator
- An if statement looks like this:

```
int x = 10;
if (x == 10) {
                                    System.out.println("yes x == 10");
```

This statement is basically saying if x is equal to 10, then I want to run this code. Here, the program answers that yes, x is in fact equal to 10, which is true.

- That means the statement is followed and the condition will be executed
- Now try to see if x is equal to 20 by only changing the statement by placing the number 20 in the if line
- No answer will come up in eclipse because this statement is false and the condition cannot be executed
- An if else statement can execute when an if statement does not and looks like this:
- int x = 10;
- if (x == 20) {
- System.out.println("yes x == 10");
- }
- else {
- System.out.println("no x != 10");
- Take note of the exclamation point that has taken the place of one of the equal signs after the else line – this is called a non-equality operator
- Now when you run the program again eclipse will give you the answer that x is in fact not equal to 10
- In JAVA, comparison operators include:

  1. == is equal to
  2. != is not equal to
     ㄹ is greater than
  3. < is less than
  4. >= is greater than or equal to
  5. <= is less than or equal to

9. How to utilize the logical operators in JAVA

- If you want to evaluate more than one condition using a simple if statement or an if else statement you will need to use a logical operator

- There are two basic kinds of logical operators – the and operator, which is symbolized by double ampersands && and the second type is called an or operator which is symbolized by double pipe symbols ||
- The and operator and the or operator can both be used to define two conditions simultaneously in both if and if else statements
- The and operator checks to see if all conditions are simultaneously true while the or operator checks to see if one or more things is true
- The symbol placed between subjects tells which you are using, as in line four of the following two codes:

```
int subject1 = 40;
int subject2 = 60;
// && ->AND   || ->OR
if ((subject1 >= 35) && (subject2 >= 35)) {
        System.out.println("the condition is true");
```

This condition is true because both 40 and 60 are greater than 35.

- Now try to incorporate an else statement with the if statement, nut first change the value of subject1 to 20

```
if else
int subject1 = 20;
int subject2 = 60;
// && ->AND   || ->OR
if ((subject1 >= 35) && (subject2 >= 35)) {
            System.out.println("the condition is true");
            } else {
            System.out.println("the condition is false");
```

This condition is false.

- Now try running the same program but replace the and operator with the or operator:

```
int subject1 = 40;
int subject2 = 60;
// && ->AND    || ->OR
if ((subject1 >= 35) || (subject2 >= 35)) {
      System.out.println("the condition is true");
} else {
      System.out.println("the condition is false");
This condition is true.
```

10. Working with the switch statement in JAVA
    - Whenever you have to check multiple in JAVA, you will want to utilize

- the conditions switch statement
- You can also switch statements in place of if else statements
- The breaks in a switch statement act to break up the responses you will
- receive from the program when you run it
- Part of a switch statement includes the use of ‘default’ which works like the else in an if else statement
- int = 90;

```
// byte, int, short, or char.
switch (score)
{
case 90 :
      System.out.println(“Very good”);
      break;
case 60 :
      System.out.println(“Good”);
      break;
case 30 :
      System.out.println(“OK”);
      break;
default :
           System.out.println(“Grades are not defined”);
           break;
}
```

With the value of the integer identified as 90, the case 90 statement is true.

- When there is no break between case statements, the program will default to the next lowest in succession when utilizing this type of code

11. Working with the while statements in JAVA, also known as while loops
    - While loops are the most basic loops in JAVA
    - A loop is a piece of code or a statement which executes some block of code again and again until some condition is met
    - If you want to execute some code again and again without having to rewrite it over and over, this is what loop is for:

```
int a = 0;
while (a <=10)
{
    System.out.println(a);
    a++;
}
```

12. Proceeding to the do while statements in JAVA, also known as the do while loops

- The basic dissimilarity from a while loop to a do while loop is that even though loops first evaluate the condition and then execute the code, the do while loops execute the code first and then evaluate the condition

13. How to learn to implement arrays in your JAVA program
    - An array is similar to a variable but it can store more than one value at a time – the only condition being that even though you can store more than one value in an array, it has to be the same type of values
    - For example, you would be able to store 10 integers in an array, but if you wanted to store 5 integers and 5 doubles, you would not be able to do it
    - This is one way of declaring an array: int [] myintarray = {4, 2,1,5,3};
    - Another way is: int myinarray2[] = {4,2,1,5,3} This way is fine but it is not the preferred way of declaring an array
    - There are three more ways of declaring arrays:
      1. int[] myIntArray = new int[3];
      2. int[] myIntArray = {1,2,3};
      3. int[] myIntArray = new int []{1,2,3};

14. How to use the JAVA string in your program
    - A string is a sequence of characters and can also be known as an array of characters
    - In order declare a string in JAVA, you just use the keyword ‘string’
    - You then name your string whatever you would like
    - Then you place an equal sign after that
    - Then in double quotes, place whatever string you want to assign and print it just the same as all the others:

      String mystring = “Hello World”;
      System.out.println(mystring)

15. Introduction to using methods in your JAVA programming
    - The terms method and function can be used somewhat interchangeably, but the preferred term in JAVA is method
    - A method is a piece of code which executes some logic and you can wrap this method with a name and you can recall this method as many times as you would like whenever you want to use it
    - You can name your method anything you wish
    - This is the most basic kind of method:

```
public static void myFirstMethod() {
}
```

- The term ‘public static’ is known as the pacifier
- If you want to print a message using method, this is what the syntax or code would look like:

```
public static void myFirstMethod() {
     System.out.println(“Hello Youtube”);
}
```

CONCLUSION

Thank you again for taking the time out of your schedule to download JAVA Programming: An Introduction, History, and the Fundamentals For Creating Your First Program!

Since you have completed reading this book, you should now have a good understanding of JAVA programming and also possess the skills to complete some basic JAVA code!

If you have enjoyed this book as much as I have enjoyed writing it, I ask that you please take the time to leave me a review on Amazon. I appreciate your positive feedback!