

JAVA: Improvised Approach to Java

Jorawar Singh & Jobin Abraham

Department of Computer Science and Engineering, Dronacharya College Of Engineering, Gurgaon, India

Abstract-

Java is, as of 2014, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licences. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).

Index Terms-

Cache memory; cache access mechanism; NUCA cache; pre-fetching; victim cache; chip multiprocessors

1. Introduction

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995

as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

As of December 2008, the latest release of the Java Standard Edition is 6 (J2SE). With the

advancement of Java and its widespread popularity, multiple configurations were built to suite various types of platforms. Ex: J2EE for Enterprise Applications, J2ME for Mobile Applications.

Sun Microsystems has renamed the new J2 versions as Java SE, Java EE and Java ME respectively. Java is guaranteed to be **Write**



Once, Run Anywhere. Sun Microsystems released the first public implementation as Java 1.0 in 1995. It promised "Write Once, Run Anywhere" (WORA), providing no-cost run-times on popular platforms. Fairly secure and featuring configurable security, it allowed network- and file-access restrictions. Major web browsers soon incorporated the ability to run *Java applets* within web pages, and Java quickly became popular. The Java 1.0 compiler was re-written in Java by Arthur van Hoff to comply strictly with the Java 1.0 language specification.¹ With the advent of *Java 2* (released initially as J2SE 1.2 in December 1998 – 1999), new versions had multiple configurations built for different types of platforms. *J2EE* included technologies and APIs for enterprise applications typically run in server

environments, while *J2ME* featured APIs optimized for mobile applications. The desktop version was renamed *J2SE*. In 2006, for marketing purposes, Sun renamed new *J2* versions as *Java EE*, *Java ME*, and *Java SE*, respectively.

History of Java:

James Gosling initiated the Java language project in June 1991 for use in one of his many set-top box projects. The language, initially called Oak after an oak tree that stood outside Gosling's office, also went by the name Green and ended up later being renamed as Java, from a list of random words.

2. Tools you will need:

For performing the examples discussed in this tutorial, you will need a Pentium 200-MHz computer with a minimum of 64 MB of RAM (128 MB of RAM recommended).

You also will need the following softwares:

- Linux 7.1 or Windows 95/98/2000/XP operating system.
- Java JDK 5
- Microsoft Notepad or any other text editor

This tutorial will provide the necessary skills to create GUI, networking, and Web applications using Java.

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to architecture-specific machine code. Java bytecode instructions are analogous to machine code, but they are intended to be executed by a virtual machine (VM) written specifically for the host hardware. End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java applets. Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

Sun released the first public implementation as Java 1.0 in 1995. It promised **Write Once, Run Anywhere** (WORA), providing no-cost run-times on popular platforms.

On 13 November 2006, Sun released much of Java as free and open source software under the terms of the GNU General Public License (GPL).

On 8 May 2007, Sun finished the process, making all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.

A major benefit of using bytecode is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would. Just-in-Time (JIT) compilers were introduced from an early stage that compile bytecodes to machine code during runtime.

Implementations

Oracle Corporation is the current owner of the official implementation of the Java SE platform, following their acquisition of Sun Microsystems on January 27, 2010. This implementation is based on the original implementation of Java by Sun. The Oracle implementation is available for Windows, Mac OS X, Linux and Solaris. Because Java lacks any formal standardization recognized by Ecma International, ISO/IEC, ANSI, or other third-party standards organization, the Oracle implementation is the de facto standard.

The Oracle implementation is packaged into two different distributions: The Java Runtime Environment (JRE) which contains the parts of the Java SE platform required to run Java programs and is intended for end-users, and the Java Development Kit (JDK), which is intended for software developers and includes development tools such as the Java compiler, Javadoc, Jar, and a debugger.

Open JDK is another notable Java SE implementation that is licensed under the GNU GPL. The implementation started when Sun began releasing the Java source code under the GPL. As of Java SE 7, OpenJDK is the official Java

reference implementation.

The goal of Java is to make all implementations of Java compatible. Historically, Sun's trademark license for usage of the Java brand insists that all implementations be "compatible". This resulted in a legal dispute with Microsoft after Sun claimed that the Microsoft implementation did not support RMI or JNI and had added platform-specific features of their own. Sun sued in 1997, and in 2001 won a settlement of US\$20 million, as well as a court order enforcing the terms of the license from Sun. As a result, Microsoft no longer ships Windows with Java.

Platform-independent Java is essential to Java EE, and an even more rigorous validation is required to certify an implementation. This environment enables portable server-side applications.

Performance

Programs written in Java have a reputation for being slower and requiring more memory than those written in C++. However, Java programs' execution speed improved significantly with the introduction of Just-in-time compilation in 1997/1998 for Java the addition of language features supporting better code analysis (such as inner classes, the `StringBuilder` class, optional assertions, etc.), and optimizations in the Java virtual machine itself, such as HotSpot becoming the default for Sun's JVM in 2000.

Some platforms offer direct hardware support for Java; there are microcontrollers that can run Java in hardware instead of a software Java virtual machine, and ARM based processors can have hardware support for executing Java bytecode through their Jazelle option.

3. Local Environment Setup

If you are still willing to set up your environment for Java programming language, then this section guides you on how to download and set up Java on your machine. Please follow the following steps to set up the environment.

Java SE is freely available from the link Download Java. So you download a version based on your operating system.

Follow the instructions to download java and run

the `.exe` to install Java on your machine. Once you installed Java on your machine, you would need to set environment variables to point to correct installation directories:

Setting up the path for windows 2000/XP:

Assuming you have installed Java in `c:\Program Files\java\jdk` directory:

- Right-click on 'My Computer' and select 'Properties'.
- Click on the 'Environment variables' button under the 'Advanced' tab.
- Now, alter the 'Path' variable so that it also contains the path to the Java executable. Example, if the path is currently set to 'C:\WINDOWS\SYSTEM32', then change your path to read 'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin'.

Setting up the path for windows 95/98/ME:

Assuming you have installed Java in `c:\Program Files\java\jdk` directory:

- Edit the 'C:\autoexec.bat' file and add the following line at the end:
'SET PATH=%PATH%;C:\Program Files\java\jdk\bin'

Setting up the path for Linux, UNIX, Solaris, FreeBSD:

Environment variable PATH should be set to point to where the Java binaries have been installed. Refer to your shell documentation if you have trouble doing this.

Example, if you use `bash` as your shell, then you would add the following line to the end of your '.bashrc: export PATH=/path/to/java:\$PATH'

Popular Java Editors:

To write your Java programs, you will need a text editor. There are even more sophisticated IDEs available in the market. But for now, you can consider one of the following:

- **Notepad:** On Windows machine you can use any simple text editor like Notepad (Recommended for this tutorial), TextPad.
- **Netbeans:** is a Java IDE that is open-source and free which can be downloaded from <http://www.netbeans.org/index.html>.
- **Eclipse:** is also a Java IDE developed by the eclipse open-source community and can be downloaded from

<http://www.eclipse.org/>.

Example of a program in java

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

When we consider a Java program it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what do class, object, methods and instance variables mean.

- **Object** - Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors - wagging, barking, eating. An object is an instance of a class.
- **Class** - A class can be defined as a template/ blue print that describes the behaviors/states that object of its type support.
- **Methods** - A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instance Variables** - Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

4. First Java Program:

Let us look at a simple code that would print the words *Hello World*.

```
public class MyFirstJavaProgram {
    /* This is my first java program.
    * This will print 'Hello World' as the output
    */

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

Let's look at how to save the file, compile and run the program. Please follow the steps given below:

- Open notepad and add the code as above.
- Save the file as: MyFirstJavaProgram.java.
- Open a command prompt window and go to the directory where you saved the class. Assume it's C:\.
- Type ' javac MyFirstJavaProgram.java ' and press enter to compile your code. If there are no errors in your code, the command prompt will take you to the next line (Assumption : The path variable is set).
- Now, type ' java MyFirstJavaProgram ' to run your program.
- You will be able to see ' Hello World ' printed on the window.

```
C : > javac MyFirstJavaProgram.java
C : > java MyFirstJavaProgram
```

Hello World

Basic Syntax:

About Java programs, it is very important to keep in mind the following points.

- **Case Sensitivity** - Java is case sensitive, which means identifier **Hello** and **hello** would have different meaning in Java.
- **Class Names** - For all class names the first letter should be in Upper Case.

If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.

Example *class MyFirstJavaClass*

- **Method Names** - All method names should start with a Lower Case letter.

If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.

Example *public void myMethodName()*

- **Program File Name** - Name of the program file should exactly match the class name.

When saving the file, you should save it using the class name (Remember Java is

case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match your program will not compile).

Example : Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as 'MyFirstJavaProgram.java'

- **public static void main(String args[])** - Java program processing starts from the main() method which is a mandatory part of every Java program..

6. Java Identifiers:

All Java components require names. Names used for classes, variables and methods are called identifiers.

In Java, there are several points to remember about identifiers. They are as follows:

- All identifiers should begin with a letter (A to Z or a to z), currency character (\$) or an underscore (_).
- After the first character identifiers can have any combination of characters.
- A key word cannot be used as an identifier.
- Most importantly identifiers are case sensitive.
- Examples of legal identifiers: age, \$salary, _value, __1_value
- Examples of illegal identifiers: 123abc, -salary

7. Java Modifiers:

Like other languages, it is possible to modify classes, methods, etc., by using modifiers. There are two categories of modifiers:

- **Access Modifiers:** default, public, protected, private
- **Non-access Modifiers:** final, abstract, strictfp

We will be looking into more details about modifiers in the next section.

Java Variables:

We would see following type of variables in Java:

- Local Variables
- Class Variables (Static Variables)
- Instance Variables (Non-static variables)

Java Arrays:

Arrays are objects that store multiple variables of the same type. However, an array itself is an object on the heap. We will look into how to declare, construct and initialize in the upcoming chapters.

Java Enums:

Enums were introduced in java 5.0. Enums restrict a variable to have one of only a few predefined values. The values in this enumerated list are called enums.

With the use of enums it is possible to reduce the number of bugs in your code.

For example, if we consider an application for a fresh juice shop, it would be possible to restrict the glass size to small, medium and large. This would make sure that it would not allow anyone to order any size other than the small, medium or large.

Example:

```
class FreshJuice {
    enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
    FreshJuiceSize size;
}

public class FreshJuiceTest {
    public static void main(String args[]){
        FreshJuice juice = new FreshJuice();
        juice.size = FreshJuice.FreshJuiceSize.MEDIUM ;
        System.out.println("Size: " + juice.size);
    }
}
```

Above example will produce the following result:
Size: MEDIUM

Note: enums can be declared as their own or inside a class. Methods, variables, constructors can be defined inside enums as well.

8. Java Keywords:

The following list shows the reserved words in Java. These reserved words may not be used as constant or variable or any other identifier names.

abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	void
volatile	While		

Comments in Java

Java supports single-line and multi-line comments very similar to c and c++. All characters available inside any comment are ignored by Java compiler.

```
public class MyFirstJavaProgram{
```

```

    /* This is my first java program.
    * This will print 'Hello World' as the output
    * This is an example of multi-line comments.
    */

    public static void main(String []args){
        // This is an example of single line comment
        /* This is also an example of single line
comment. */
        System.out.println("Hello World");
    }
}

```

Using Blank Lines:

A line containing only whitespace, possibly with a comment, is known as a blank line, and Java totally ignores it.

9. Inheritance:

In Java, classes can be derived from classes. Basically if you need to create a new class and

here is already a class that has some of the code you require, then it is possible to derive your new class from the already existing code.

This concept allows you to reuse the fields and methods of the existing class without having to rewrite the code in a new class. In this scenario the existing class is called the superclass and the derived class is called the subclass.

Interfaces:

In Java language, an interface can be defined as a contract between objects on how to communicate with each other. Interfaces play a vital role when it comes to the concept of inheritance.

An interface defines the methods, a deriving class(subclass) should use. But the implementation of the methods is totally up to the subclass.

10. Conclusion

One reason for Java's success is its agility. Java has rapidly adapted to changes in the programming environment and to changes in the way that programmers program. Most importantly, it has not just followed the trends, *it has helped create them*. Unlike some other languages that have a revision cycle of approximately 10 years, Java's release cycle averages about 1.5 years! Java's ability to accommodate the fast rate of change in the computing world is a crucial part of why it has stayed at the forefront of computer language design. With the release of Java SE 6, Java's leadership remains unchallenged. If you are programming for the Internet, you have chosen the right language. Java has been and continues to be the preeminent language of the Internet.

11. References

1. "The History of Java Technology". Retrieved October 6, 2012.
2. Java 5.0 added several new language features (the enhanced for loop, autoboxing, varargs and annotations), after they were introduced in the similar (and competing) C# language
3. Gosling, James; and McGilton, Henry (May 1996). "The Java Language Environment".
4. Gosling, James; Joy, Bill; Steele, Guy; and

- Bracha, Gilad. "The Java Language Specification, 2nd Edition".
5. "The A-Z of Programming Languages: Modula-3". Computerworld.com.au. Retrieved 2010-06-09.
 6. Niklaus Wirth stated on a number of public occasions, e.g. in a lecture at the Polytechnic Museum, Moscow in September, 2005 (several independent first-hand accounts in Russian exist, e.g. one with an audio recording: Filippova, Elena (September 22, 2005). "Niklaus Wirth's lecture at the Polytechnic Museum in Moscow".), that the Sun Java design team licenced the Oberon compiler sources a number of years prior to the release of Java and examined it: a (relative) compactness, type safety, garbage collection, no multiple inheritance for classes -- all these key overall design features are shared by Java and Oberon.
 7. Patrick Naughton cites Objective-C as a strong influence on the design of the Java programming language, stating that notable direct derivatives include Java interfaces (derived from Objective-C's protocol) and primitive wrapper classes.
 8. TechMetrix Research (1999). "History of Java". *Java Application Servers Report*. "The project went ahead under the name "green" and the language was based on an old model of UCSD Pascal, which makes it possible to generate interpretive code"
 9. "A Conversation with James Gosling – ACM Queue". Queue.acm.org. 2004-08-31. Retrieved 2010-06-09.
 10. "Programming Language Popularity" 2014. Retrieved 2014-02-07.
 11. "TIOBE Programming Community Index". 2009. Retrieved 2009-05-06.
 12. Byous, Jon (c. 1998). "Java technology: The early years". *Sun Developer Network*. Sun Microsystems. Archived from the original on April 20, 2005. Retrieved 2005-04-22.