JAVA PROGRAMMING BY EXAMPLE

4th Edition

Joy T. de Jesus, Jr. Lesley Abe, M.S. Jaime D.L. Caro, Ph.D.

JAVA™ PROGRAMMING by Example

4th Edition

Joy T. de Jesus, Jr.
Lesley Abe, M.S.
Jaime D.L. Caro, Ph.D.

Philippine Copyright 2012 techFactors Inc

Fourth printing of the fourth edition, 2015
ISBN 978-971-0550-57-9

Published by TechFactors Inc.
Printed in the Philippines

Authors Joy T. de Jesus, Jr. and Lesley Abe, M.S.
Series Editor Jaime D.L. Caro, Ph.D.
Cover Design Jiyas P. Suministrado

Content and Editorial Alvin Ramirez, Frances Ibañez, Joanne Ortiz, and Alexander Lim
Creatives Jiyas Suministrado, Gilbert Lavides, and Regina Zapata
Systems Kim Benebese, Caselyn Dionisio, and Mark Abliter

Exclusively distributed by TechFactors Inc.
101 V. Luna Road Ext., Sikatuna Village
Diliman, Quezon City
1101 Philippines

Telephone number: (632) 929 6924
E-mail address: info@techfactorsinc.com
Website: www.techfactorsinc.com

Exclusively distributed by TechFactors, Inc.
101 V. Luna Road Ext., Sikatuna Village
Quezon City, 1101 Philippines

FOREWORD

We live in an age of marvels.

Nowadays, we can move machines with the touch of a button. We can communicate with people halfway across the world in real-time. Innovations occur everyday because of technology.

For adolescents today, technology is already a part of daily living. They use mobile phones, computers, iPods, and other electronic gadgets that assist and expand their awareness of the world. However, they have so much more to learn about the full potential and limitations of the technology within their reach.

In an increasingly computer-dependent world, it is important to be aware of the changes in computing technology, and to be knowledgeable in the various ways that computers can help us in everyday life. This courseware is not only intended to be an instructional manual discussing individual topics, but also to be a means of exploring the continually improving and expanding world of computers – and our society as well.

Jaime L Caro

Jaime D.L. Caro, Ph.D.
Series Editor

ABOUT THE AUTHORS

Joy T. de Jesus, Jr. specializes in ATM-based and ATM-related systems. He has served as trainor and content developer of Techfactors Inc., co-authoring the following books: Office Productivity, IT Project Management, Java Programming By Example, Animation in a Flash, and C++ Programming Fundamentals. He studied Computer Science major in Software Technology at De La Salle University and pursued further studies at Mapua Information Technology Center.

Lesley Abe, M.S. holds the degree of Master of Science in Computer Science from the De La Salle University. She graduated cum laude with a bachelor's degree in Information Technology from the Polytechnic University of the Philippines. She is an experienced educator in Computer Science and Information Technology. She is currently a full-time professor at the College of Computer Studies in De La Salle University.

ABOUT THE SERIES EDITOR

Dr. Jaime D.L. Caro has more than 20 years of experience in education and research in the areas of Computer Science, Information Technology, and Mathematics. He received the degrees of Bachelor of Science major in Mathematics (cum laude) in 1986, Master of Science in Mathematics in 1994, and Doctor of Philosophy in Mathematics in 1996, all from the University of the Philippines Diliman. He spent a year as a post doctorate research fellow at the University of Oxford from 1997 to 1998. He is presently Assistant Vice President for Development of the University of the Philippines, Program Director of the UP Information Technology Training Center (UP ITTC), and a professor of Computer Science in UP Diliman. He is an honorary member of the Philippine Society of Information Technology Educators (PSITE), President of the Computing Society of the Philippines (CSP), and a member of the Technical Panel on Information Technology Education of the Commission on Higher Education (CHED). Dr. Caro is a recognized expert on Complexity Theory, Combinatorial Network Theory, Online Communities, and e-Learning.

TABLE OF CONTENTS

INTRODUCTION

In this module, the student will find object-oriented programming using Java™ enjoyable. Basic Java concepts and principles are discussed, alongside with the explanation of sample executable programs. The student does not need prior knowledge of any programming language to understand Java. And learning Java does not end after going through this courseware. Additional information, samples, and references can be found in various resources such as the Internet.

LEARNING GOALS

By the end of this courseware, the student is expected to:

1. Appreciate Java as a programming language.
2. Define commonly used terms in object-oriented programming such as classes and objects.
3. Apply the different language elements and constructs of Java to solve simple programming problems.

FORMATTING CONVENTIONS

Italics – used to distinguish filenames, directories, or class names
Boldface – used to indicate important terms in the lesson
`Courier` – used to represent Java syntaxes

DISCLAIMER

Every effort has been made to make this courseware as accurate as possible, but no warranty or fitness is implied. The authors and publishers shall have neither liability nor responsibility to any person or entity with respect to any loss or damage arising from the information contained herein.

LESSON ONE

It is a truth universally acknowledged, that a single man in possession of a large fortune must be in want of a wife.

However little known the feeling or views of such a man may be on his first entering a neighborhood, this truth is so well fixed in the minds of the surrounding families, that he is considered the rightful property of someone or other of their daughters.

– from *Pride and Prejudice*, Jane Austen

Introduction to Java™

Part of the courtship rituals in 19th century England was that women of a certain class were expected to marry men who were within their social and financial means. They were paired with possible husbands who were thought to be best suited to their lifestyle and the lifestyle to which the men's families were accustomed. Qualities were compared and contrasted, and similarities were highlighted while differences were discarded.

The Java programming language initiative started in 1991 and was spearheaded by a group of engineers from Sun Microsystems named the "Green Team," led by James Gossling. The Green Team envisioned that the next big thing was the union of consumer devices and computers. To demonstrate this, they created an interactive home entertainment controller targeted for cable television. Unfortunately, the concept was too advanced for its time, but Java was soon integrated into one of the more popular original Internet browsers, the Netscape Navigator in 1995, ushering Java into the mainstream development. With its "Write Once Run Anywhere" principle, within a few years, Java became one of the programming languages with the fastest growing developer base. With the acquisition of Sun Microsystems by Oracle Corporation in around 2009-2010, Oracle has since described itself as the steward of Java technology.

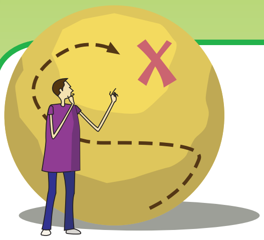
NOTE

Like many programming languages, Java had several names before the developers finally settled for one. Java was once named "Oak" since there was an oak tree outside the engineers' office. Then the language was named "Green," until finally, the team settled for "Java," since they were consuming a lot of Java coffee during the course of their development.

LESSON OBJECTIVES

At the end of this lesson, the student will be able to:

1. Describe the history of Java.
2. Explain the design goals behind Java.
3. Identify various Java versions.



LESSON OUTLINE

1. Design Goals Behind the Java Language
2. Versions of Java

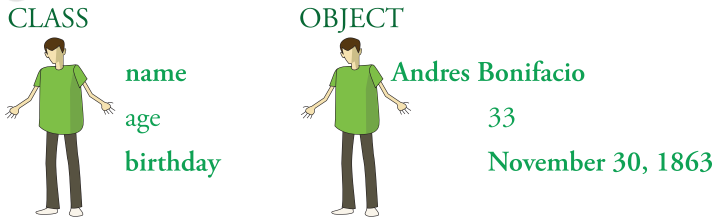
Design Goals Behind the Java Language

Simple, Object-Oriented, and Familiar

One of the goals of the Java developers was to create a simple programming language that can be used even without extensive programmer training. An example that attests to Java's simplicity are concepts such as automatic memory allocation and automatic garbage collection. In some programming languages, the developer is required to program the amount of memory allocated to elements of the program and proper "garbage collection" from the program's functions. Programming these takes a lot of time and effort. With Java, all these are done in the background, thereby freeing the programmer to think of other things related to the program.

The developers of Java also wanted the programming language to be familiar, that is, to look as close to C++ as possible when it comes to language syntax, but removed certain complexities of C++. C++ remains to be one of the more popular programming languages to date, so having Java "look and feel" like C++ encourages more developers to try out the Java programming language and create programs faster.

Object-oriented programming ("OOP" for short) was conceived to model real-world objects. In OOP, a **class** defines the common **variables** (attributes) and **methods** (functions) of a set of objects. An **object** is an instance of a class. For example, all humans have common attributes: name, age, birthday, etc.; and functions: getting old or growing up. Once you understand how components of a program are treated as objects, you will find Java fairly simple. In addition, using other predefined and more complex components will be effortless once they are implemented and utilized in the same way.



Robust and Secure

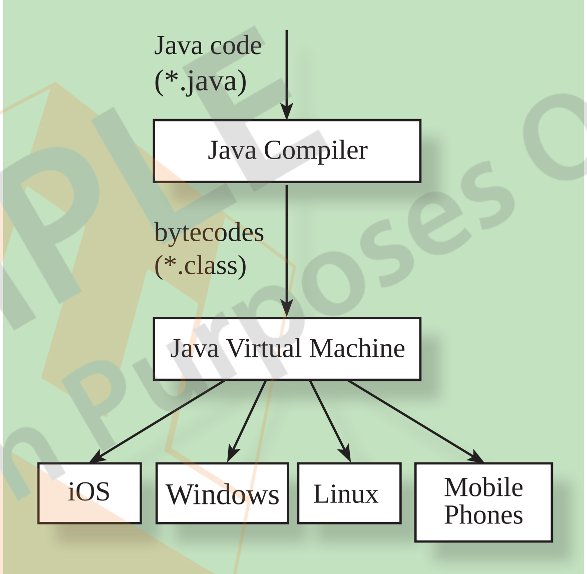
Java has a certain degree of fault-tolerance, which is a property that allows a platform to continue to function in the event of a failure or an error. Java also has good exception handling when running your program. Even after you compile your program, the Java Virtual Machine performs security checks when executing the code.

Java was created to run in various environments, even in those where security is a top concern. To address this, Java lets you construct network applications that can be secured from intrusion by unauthorized code attempting to invade the network.

Architecture Neutral and Portable

Earlier we mentioned that Java adheres to the "Write Once Run Anywhere" principle. This means that Java was designed to create applications that can execute in different operating systems, hardware architectures, and network environments by ensuring that the programming concepts and behavior are the same on every platform without any data incompatibilities. Java is able to accomplish neutrality and portability through the Java Application Programming Interface (Java API) and the Java Virtual Machine.

The Java API is the component that allows you to write your program. The Java API can be used with Integrated Development Environments (IDEs) such as Net Beans, Eclipse, and BlueJ to make programming easier. If you have used other programming languages before, once you compile your program, you should be able to run it right away on your computer as an executable file. With Java, the compiler generates "bytecode" files which is an architecture neutral format used to transport code across various hardware and software platforms, and even various versions of the Java programming language.



Java codes written once can be executed almost anywhere.

Various platforms (i.e. Windows, Linux, iOS, etc) can have different implementations of the Java Virtual Machine but there is only one Virtual Machine specification. Because of this, Java is able to have a uniform programming interface and can run the bytecode file anywhere.

High Performance

High Performance Computing (HPC) are applications that aim to solve complex computational problems. Certain characteristics of Java make it an ideal candidate to program high performance computing applications. Since hardware advancements keep on moving forward, having the latest and faster hardware helps HPC performance. HPC applications must then be portable to adapt to constant hardware upgrades. HPC applications are normally running within a network or via distributed computing and Java has built-in support for distributed systems.

Interpreted, Threaded, and Dynamic

By "interpreted," we refer to how the Java bytecodes are interpreted by the Java Virtual Machine regardless of the hardware and software platforms. Java's interpreted nature has been discussed earlier but one of the benefits of having an interpreted language is faster development cycles.

Java is also a multithreading programming language that allows the programmer to build applications with many concurrent activity threads. Multithreading allows a program to execute many tasks at the same time. This reduces the execution time and improves the performance of the application.

Big projects and applications require several programmers collaborating together. In most cases, a team of programmers can develop one part of the application that will be integrated with another team's work. But what if there are updates to one team's work? Those updates have the ability to mess up the work of other teams who have integrated the old application. Java makes it possible to reuse previously written program to lessen the workload of the programmer. Aside from this, it also allows programmers to make revisions, additions, and alterations to a program without affecting other programs that might already have been using it.

Versions of Java

We can learn more about what kind of applications use Java by knowing the different versions of Java available:

Java Standard Edition (JSE) – This is the default Java version that allows us to develop applications for desktops, servers, and small embedded environments such as SIM cards and SMART cards. It also provides the creation of a rich user interface. These applications serve a small number of users at a time.

Java Enterprise Edition (JEE) – Applications created for this version of Java are oriented towards enterprise software catering to data-intensive server applications meant to have a large number of users. While JSE can also run on servers, JEE is tailor-fit to handle network and web applications.

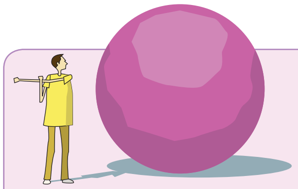
Java Micro Edition (JME) – Applications using JME are normally mobile devices such as mobile phones and other embedded devices such as Blu-ray players, TV boxes, and printers. JME applications can run on many devices but also highlight key features of the device.

SCORE

NAME: ____________________

SECTION: ____________________

DATE: ____________________



SKILLS WARM-UP

Match the Java related concepts with their corresponding meanings.

1. Java
2. Object-Oriented Programming
3. Dynamic
4. Object
5. Bytecodes
6. Multithreading
7. BlueJ
8. High Performance
9. Architecture Neutral
10. Java Virtual Machine

a. feature that allows the making of revisions, additions and alterations to a program without affecting other programs that might already have been using it

b. describes applications that can execute in different operating systems, hardware and network environments

c. the result of compiling Java code with a Java SDK

d. the software that executes compiled Java codes

e. a feature that allows a program to perform several functions at the same time

f. an instance of a class

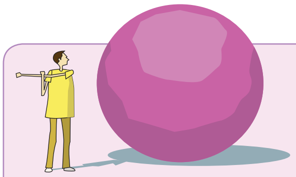
g. describes applications that aim to solve complex computational problems

h. a programming paradigm that was conceived to model real world objects

i. an object-oriented programming language that was created by engineers of Sun Microsystems

j. a type of Java Integrated Development Environment to aid programmers in coding

CUT THIS PAGE



SKILLS WARM-UP

Take a look at the following statements below. Write **True** on the blank provided if the statement is true, otherwise write **False**.

___________ 1. The Java programming language makes object-oriented programming easier.

___________ 2. JME is tailored to create networked applications.

___________ 3. It is possible to reuse Java code previously written for other projects.

___________ 4. It is possible for a Java class compiled in an Apple iOS machine to be executed in a Windows environment provided that the Java Virtual Machine is properly installed.

___________ 5. The Java Virtual Machine can access and modify any file on your computer.

___________ 6. Java bytecodes are executable on their own

___________ 7. Programmers still need to program garbage collection and memory allocation in Java for their individual programs.

___________ 8. The two main components of Java are the Java Application Programming Interface and the Java Virtual Machine.

___________ 9. The Java Virtual machine no longer performs security checks when executing the code.

___________ 10. Java was created to look and feel like C++.

NOTE

One popular misconception is when people assume that the mobile operating system Android runs on Java or JME. In fact, Android has its own virtual machine separate from the Java versions known as "Dalvik." This means that regular Java bytecode cannot be interpreted by the Dalvik Virtual Machine in the same way as JME or JSE. To get your Java program to work with Android, you need to compile it using the Android SDK. The usage of the Dalvik Virtual Machine is currently a point of discussion in the ongoing patent litigation between Oracle and Google, the creators of the Android OS.

SUMMARY

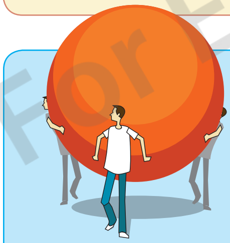
Java is an object-oriented language. In this paradigm, objects are class instances. Like other object-oriented languages, you can reuse codes and create dynamic applications. Moreover, you can use its extensive built-in library to create a wide variety of programs.

WORD BANK

High Performance Computing - applications that aim to solve complex computational problems

Java Virtual Machine - software that executes Java codes

Multithreading - allows a program to execute many tasks at the same time



GROUP WORK

Java as a programming language has found applications in many fields in computer science. Some Web sites use Java applets to provide interactive games or animated Web navigational buttons. Some software projects use Java to aid to model and simulate artificial intelligence or aid in mathematical computations.

Your task is to create a group composed of five students to do some group research work to find three software projects or websites that use Java. If possible, give the URL of the websites that use Java applets or the homepage of the projects that use Java. Try to find out how the benefits of Java, discussed in this lesson helped these applications provide their specific services and requirements. Present the results of your discussion in class for additional insights.