**Lynda.com - Java Essentials Training 2016**

**Chapter 4 - History**

Started on 1991 by Sun Microsystem – Green Project

New Portable Programming Language that can run on multiple OS without recompiling

Original Name: Oak – Oak Tree outside the windows of developer’s office

First release named Java (1995)

Original Tag: Write Once Run Everywhere

1996 – First JDK 1.0 (Java Development Kit)

1997 – Object Oriented Nature, JavaBeans, JDBC, RMI (Remote Distributed Systems), Reflection

1998 – J2SE 1.2 (Java 2 Standard Edition); Swing GUI, Collections Framework, JIT Compiler,

2000 – J2SE 1.3; Hotspot JVM, JNDI Interface, Sound API, Debugging Architecture

2002 – J2SE 1.4; Regular Expressions, IPv6 Network, Logging API, XML and XSLT, Security and Cryptography, Java Web Start

2004 – J2SE 1.5; Generics (Strongly data type), Enumerations, Variable Arguments, Foreach enhancements

2006 – Java SE 6; Performance JDBC 4.0

2010 – Oracle buys Sun

2011 – J2SE 7; Strings in switch, Try-catch enhancements,

2014 – Lambda expressions, method references, collections, new date/time API, Nashorn JavaScript engine

Android: check Java 6 and Java 7

**Chapter 5 – Principles of Java**

* Simplicity – consistent, never deviates from the way the language is architected
* Object-Oriented – encapsulation, inheritance, polymorphism
* Familiar with C and C++
* Robust and Secure – object which consists of methods/functions and properties/fields for ease of debugging
* High-performance – just as fast as C Language
* Interpreted at runtime rather than run directly as machine code
* Multi-threaded – can do more than one thing at a time
* Dynamic – change datatypes at runtime as long as its compatible
* Interpreted Language
* Compiles to bytecode instead of machine language
* Compiled application is portable between platforms without recompiling
* Class-based
* Run in a protected virtual machine (JVM)

Java Editions

* Java Platform, Standard Editions (SE)
  + Core language and runtime environment
* Java Platform, Enterprise Editions (EE)
  + Standard for building industrial-strength web applications
* Java Platform, Micro Edition (ME)
  + Micro-controllers, sensors, and mobile devices (e.g. simcards on smart phones)

Java SE Runtime Environment (JRE)

* JRE is the runtime environment
* Free and downloadable from Oracle
* Runs Java Applications on desktop, borwser, and server
* Must be updated regularly to keep up to date
* Mobile OS have own runtimes

Java SE Development Kit (JDK)

* Compiles and package Java Applications
* Includes tools for compilation and packaging
  + Java: runtime
  + Javac: compiler
  + Javadoc: docs builder
  + Jar: archive builder

**Chapter 6 – Java syntax and compilation**

* All code is defined in classes
* Classes are defined in source code files with .java extensions
* Javac command compiles Java code into bytecode
* Java runs bytecode files

Package Declaration: domain name in reverse order (com.example.java)

Class Declaration: each source code must have at least one public class

Main Method: console application must have a main method (public static void(String[] args))

Executable Code: code inside the main method

Case Sensitive: must be unique within the scope; firstname, firstName, FIRSTNAME

Whitespace does not affect interpretation of code (spaces, tabs, and line feeds). Collapsed during compilation

Semicolon – ends the statement

Keywords cannot be used by class or other identifiers

Identifiers must start with alpha characters or underscore

Class names starts with uppercase characters – code convention

Methods and variables start with lowercase character – code convention

Constants are all uppercase – code convention

* Public static final String FIRSTNAME = “John”;

**Chapter 7 – Memory Management**

* Java manages memory automatically. No need to allocate and deallocate memory
* JVM has feature of memory management

Stack and Heap Memory

Primitive Variables can be stored on both stack and heap

Complex variables are always stored on heap

Variables within method expires along the method. After the method executes, the variable is then deallocated

If the variable is set to null, the variable is explicitly dereferenced

Garbage Collector runs on its own thread