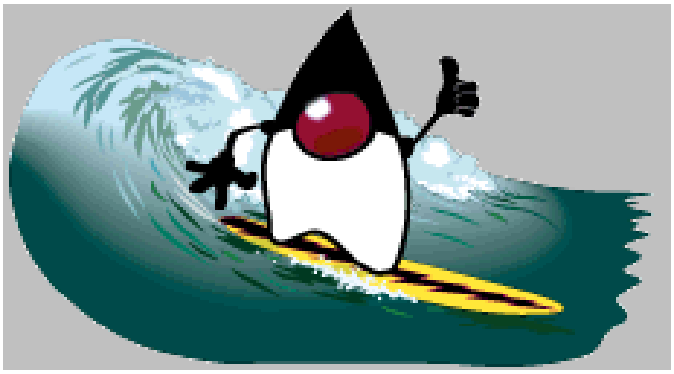


CONTENTS

You have to eat before you can cook. You have to wear before you can sew. You have to ride before you can drive. And you have to run computer programs before you can write computer programs

Desktop

The objective of this booklet is to give awareness to the Java users and all students who are interested to know other information about Java. This booklet is certainly not a Java tutorial book. The information compiled in this booklet focuses, Java releases, Sun certification notifications, Java feature highlights, different J2EE servers, Java database drivers and vendors, Java IDE's, Sun downloads centre, etc.

Lakshya is one of the leading IT training centers situated in the heart of Bhubaneswar since 1995. It aims in creating an atmosphere of advance software development technology awareness and expertise in Orissa. Currently there are more than 5000 students getting trained in different IT courses and projects every year. Lakshya has its own software development wing and a business partner for development of commercial software with Genesys Labs and iPod.

The training pattern adopted here at Lakshya is entirely different from the conventional IT training imparted at different places. Classes extensive demonstrate online workshop session.

What is Java?

The name Java is a trademark of Sun Microsystems and refers to the programming language developed by Sun and released in public alpha and beta versions in 1995. Java is used to create executable content that can be distributed through networks. Used generically, the name Java refers to a set of software tools for creating and implementing executable content using the Java programming language

Java is a general-purpose concurrent class-based object-oriented programming language, specifically designed to have as few implementation dependencies as possible. Java allows application developers to write a program once and then be able to run it everywhere on the Internet.

Java was originally developed by Sun Microsystems and released in 1995. Java source program is compiled into a universal format - instructions for a *virtual machine*. Java applications are typically compiled to bytecode, although compilation to native machine code is also possible. At runtime, bytecode is usually either interpreted or compiled to native code for execution, although direct hardware execution of bytecode by a Java processor is also possible.

Java was originally called Oak, and designed for use in embedded consumer-electronic applications by James Gosling. After several years of experience with the language, and significant contributions by Ed Frank, Patrick Naughton, Jonathan Payne, and Chris Warth it was retargeted to the Internet, renamed Java, and substantially revised to be the language specified here. The final form of the language was defined by James Gosling, Bill Joy, Guy Steele, Richard Tuck, Frank Yellin, and Arthur van Hoff, with help from Graham Hamilton, Tim Lindholm and many other friends and colleagues.

Java focuses on creating objects (data structures or behaviors) that can be accessed and manipulated by the program.

Why Is Java Interesting?

In one of their early papers about the language, Sun Microsystems described Java as follows:

> Java: A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language.

Sun acknowledges that this is quite a string of buzzwords, but the fact is that, for the most part, they aptly describe the language. In order to understand why Java is so interesting, let's take a look at the language features behind the buzzwords.

Simple
Java is a simple language. The Java designers were trying to create a language that a programmer could learn quickly, so the number of language constructs has been

kept relatively small. Another design goal was to make the language look familiar to a majority of programmers, for ease of migration.

Object-oriented

Java is an object-oriented programming language. As a programmer, this means that you focus on the data in your application and methods that manipulate that data, rather than thinking strictly in terms of procedures. If you're accustomed to procedure-based programming in C, you may find that you need to change how you design your programs when you use Java.

Unlike C++, Java was designed to be object-oriented from the ground up. Most things in Java are objects; the primitive numeric, character, and boolean types are the only exceptions. Strings are represented by objects in Java.

Distributed

Java is also called a distributed language. This means, that it provides a lot of high-level support for networking. Java also provides traditional lower-level networking support, including datagrams and stream-based connections through sockets. When programmers say “distributed,” they’re describing geographically dispersed computers running programs that talk to each other - in many cases, via the Internet.

The Java RMI (Remote Method Invocation) makes use of the distributed resources across the network. One application in Java may be distributed over several machines.

Interpreted

Java is an interpreted language: the Java compiler generates universal byte- codes for the Java Virtual Machine (JVM), rather than native machine code. To actually run a Java program, you use the Java interpreter to execute the compiled byte-codes. Because Java byte-codes are platform-independent, Java programs can run on any platform that the JVM (the interpreter and run-time system) has been ported to.

Robust

Java has been designed for writing highly reliable or *robust* software. Java certainly doesn't eliminate the need for software quality assurance; it's still quite possible to write buggy software in Java. However, Java does eliminate certain types of programming errors, which makes it considerably easier to write reliable software.

Secure

One of the most highly concerned aspects of Java is that it's a *secure* language. This is especially important because of the distributed nature of Java. Without an assurance of security, you certainly wouldn't want to download code from a random site on the Internet and let it run on your computer. Yet this is exactly what people do with Java applets every day. Java was designed with security in mind, and provides several layers of security controls that protect against malicious code, and allow users to comfortably run untrusted programs such as applets.

Architecture Neutral

Because Java programs are compiled to an *architecture neutral* byte-code format, a Java application can run on any system, as long as that system implements the Java

Virtual Machine. This is a particularly important for applications distributed over the Internet or other heterogeneous networks.

Portable

The fact that Java is interpreted and defines a standard, architecture neutral, byte-code format is one big part of being *portable*. But Java goes even further, by making sure that there are no "implementation-dependent" aspects of the language specification. For example, Java explicitly specifies the size of each of the primitive data types, as well as its arithmetic behavior. This differs from C, for example, in which an **int** type can be 16, 32, or 64 bits long depending on the platform.

High-Performance

Java is an interpreted language, so it is never going to be as fast as a compiled language like C & C++. Java 1.0 was said to be about 20 times slower than C. Java 1.1 is twice as fast as 1.0. The best was Java 5.0 (1.5) that is estimated eighty percent compared to C++. The future release of Java

Multithreaded

Java programs allow running several methods concurrently. The different methods share CPU time in a queue. A user could be listening to an audio clip while she is scrolling a page, and in the background the browser is downloading an image. Java is a *multithreaded* language; it provides support for multiple threads of execution (called lightweight processes) that can handle different tasks. An important benefit of multithreading is that it improves the interactive performance of graphical applications for the user.

Dynamic

Java is a dynamic language. Any Java class can be loaded into a running Java interpreter at any time. These dynamically loaded classes can then be dynamically instantiated. Native code libraries can also be dynamically loaded.

Different versions of Java

Like most products, Java gets periodic upgrades and enhancements. Since its initial release in 1995, the Java project has seen many release versions.

Java Version	Alias	Date of release
Java 1.0	Oak	1995
Java 1.1		1997
JDK 1.1.4	Sparkler	September 12, 1997
JDK 1.1.5	Pumpkin	December 3, 1997
JDK 1.1.6	Abigail	April 24, 1998
JDK 1.1.7	Brutus	September 28, 1998
JDK 1.1.8	Chelsea	April 8, 1999
J2SE 1.2	Playground	December 4, 1998
J2SE 1.2.1		March 30, 1999
J2SE 1.2.2	Cricket	July 8, 1999
J2SE 1.3	Kestrel	May 8, 2000
J2SE 1.3.1	Ladybird	May 17, 2001
J2SE 1.4.0	Merlin	February 13, 2002
J2SE 1.4.1	Hopper	September 16, 2002
J2SE 1.4.2	Mantis	June 26, 2003
J2SE 5.0 (1.5.0)	Tiger	September 29, 2004
Java SE 6 (1.6.0)	Mustang	December 11, 2006
Java SE 7 (1.7.0)	Dolphin	Anticipated for 2008

- **Java 1.0:** The original release of Java in 1995. Most of the language itself is still pretty much the same as it was in version 1.0, but the API has changed a lot since this release.

- **Java 1.1:** This version was the first upgrade to Java, released in 1997. This release is important because most Internet browsers include built-in support for applets based on Java 1.1. To run applets based on later versions of Java, you must, in most cases, download and install a current JRE.

- **Java 1.2:** This version, released in late 1998, was a huge improvement over the previous version. So much so, in fact, that Sun called it “Java 2.” It included an entirely new API called Swing for creating graphical user interfaces, as well as other major features.

- **Java 1.3:** This version, released in 2000, was mostly about improving performance by changing the way the runtime system works. Interestingly, Java 1.3 is actually called Java 2 version 1.3. Go figure.

- **Java 1.4:** Released in 2001, this version offered a slew of improvements. As you might guess, it is called Java 2 version 1.4. Keep figuring. . . .

- **Java 1.5:** Released in 2004, this version of Java is the latest and greatest. To add to Sun's apparent unpredictability with its version numbering, this version officially has two version numbers. Sun's official Java Web site explains it like this:

"Both version numbers "1.5.0" and "5.0" are used to identify this release of the Java 2 Platform Standard Edition. Version "5.0" is the *product version*, while "1.5.0" is the *developer version*."

Why should I learn Java?

Java is an innovative programming language that has become the language of choice for programs that need to run on a variety of different computer systems. First of all, Java enables you to write small programs called **applets**. These are programs that you can embed in web pages to provide some intelligence. Being able to embed executable code in a web page introduces a vast range of exciting possibilities. Instead of being a passive presentation of text and graphics, a web age can be interactive in any way that you want. You can include animations, games, interactive transaction processing - the possibilities are almost unlimited.

Java supports Server Side Programming (SSP). The SSP's dynamically generate HTML script depending on the runtime conditions, in response to the request received from the web browser.

The Java platform consists of the Java application programming interfaces (APIs) and the Java virtual machine (JVM). Java APIs are libraries of compiled code that you can use in your programs. They let you add ready-made and customizable functionality to save you programming time.

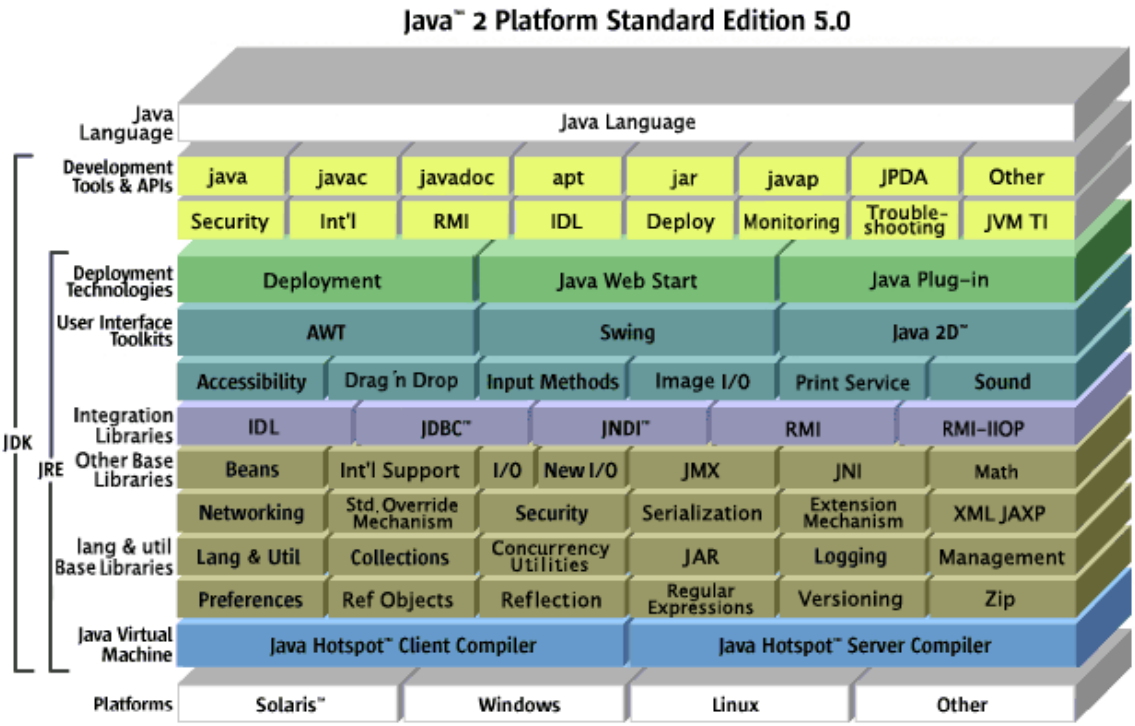
Features of Java

Java is a feature rich language. The only language that is capable of creating three different categories of computer program. The Application, Applet and the Servlet.

Java looks over your program as it runs and automatically reclaims used memory that is no longer required. This is known as automatic garbage collection. This means you don't have to keep track of memory pointers or manually deallocate memory. This feature means a program is less likely to crash and that memory can't be intentionally misused.

Threads are a unique feature of Java. The virtual machine that runs behind every java program is capable of running several programs concurrently.

The official documentation of Java highlights all its features in an illustration given below:



Java Applications

Applets and Web page
Applets are small intelligent programs that run from within the web page. All web pages are written in HTML language. HTML is a passive language. Java applets give dynamic capabilities to a web page.

E-commerce Solution
Businesses selling on the Web often use different applications to manage inventory, sales transactions and key customer information, resulting in redundant data entry and separate, inconsistent islands of data.

Internet Security
Java programs are secured. Internet security is the process of protecting data and privacy of devices connected to Internet from information robbery, hacking, malware infection and unwanted software.

Graphics and Animations
Java provides two sets of API in 2 different packages, the AWT and JFC (Swing) respectively for Graphics and Animations. The 2D graphics supports coordinate systems, modeling, constructive area geometry, color models, affine transformations, compositing, splines, clipping, fonts, raster images, animation and image processing. Java 3D is currently available from Sun for Solaris and Win32 platforms. Additional

download like JMF (Java Media Framework) is an extremely powerful API for managing and manipulating images.

Games and Planning
Although the number of commercial Java games is still small compared to those written in C or C++, the market is expanding rapidly. Recent updates to Java make it faster and easier to create powerful gaming applications-particularly Java 3D-is fueling an explosive growth in Java games. Java games like Puzzle Pirates, Chrome, Star Wars Galaxies, Runescape, Alien Flux, Kingdom of Wars, Law and Order II, Roboforge, Tom Clancy's Politika, and scores of others have earned awards and become bestsellers.

Web Store Solutions
Java is most suited for web applications that are database driven and need complex server side programming. E-commerce applications that need database generated catalogues of products, shopping carts and integration to payment gateways.

Customer Relationship Management (CRM)
Hipergate, OFBiz, Ohioedge, Compiere, CentricCRM, CentraView, Daffodil CRM, openCRX, SourceTap, Cream, Queplix, etc. are some of the most popular web-based open source CRM products developed using Java language.

Sprit in Mars (Robot)
Java gave NASA a low-cost and easy-to-use option for running Spirit, the robotic rover that rolled onto the MARS, the planet's surface on Thursday, January 15th, 2004 in search of signs of water and life.

Embedded Systems & Electronic Equipments
Using Java for Embedded program enables you to develop highly functional, reliable, portable and secure applications for today's more powerful embedded systems. Java offer a full range of products, services, and support that makes it easy for you to develop with Java for your embedded projects.

Embedded systems are computers or microchip-based control systems that are dedicated to perform specific tasks or groups of tasks. Embedded systems have been applied to a variety of industrial and consumer products, including disk drivers in computers, airbag controls and antilock brakes in automobiles, automated teller machines at banks, and space-borne imaging systems on the space shuttles.

Java Mobile Phone Applications
Java provides a separate set of API named J2ME (Java 2 Micro Edition) for mobile phone programming.

Java Packages

A Java package is a mechanism for organizing Java classes into namespaces. Java packages can be stored in compressed files called JAR files, allowing classes to download faster as a group rather than one at a time. Programmers also typically use packages to organize classes belonging to the same category or providing similar functionality.

There are as many as 166 and 202 documented packages in Java SE 5.0 and Java SE 6.0 respectively.

The following table illustrates the complete list of documented packages that are wrapped in the J2SE 5.0 toolkit.

Application	Package Names
Applet	java.applet
Abstract Window Toolkit	java.awt
	java.awt.color
	java.awt.datatransfer
	java.awt.dnd
	java.awt.event
	java.awt.font
	java.awt.geom
	java.awt.im
	java.awt.im.spi
	java.awt.image
	java.awt.image.renderable
	java.awt.print
Java Beans	java.beans
	java.beans.beancontext
Java IO	java.io
Java Language	java.lang
	java.lang.annotation
	java.lang.instrument
	java.lang.management
	java.lang.ref
	java.lang.reflect
Math	java.math
Networking	java.net
New IO	java.nio
	java.nio.channels
	java.nio.channels.spi
	java.nio.charset
	java.nio.charset.spi
Remote Method Invocation (RMI)	java.rmi
	java.rmi.activation
	java.rmi.dgc
	java.rmi.registry

	java.rmi.server
Java Security	java.security
	java.security.acl
	java.security.cert
	java.security.interfaces
	java.security.spec
Database	java.sql
Text	java.text
Java Utility	java.util
	java.util.concurrent
	java.util.concurrent.atomic
	java.util.concurrent.locks
	java.util.jar
	java.util.logging
	java.util.prefs
	java.util.regex
	java.util.zip
Accessibility	javax.accessibility
Activity	javax.activity
Cryptography	javax.crypto
	javax.crypto.interfaces
	javax.crypto.spec
Image IO	javax.imageio
	javax.imageio.event
	javax.imageio.metadata
	javax.imageio.plugins.bmp
	javax.imageio.plugins.jpeg
	javax.imageio.spi
	javax.imageio.stream
Management	javax.management
	javax.management.loading
	javax.management.modelmbean
	javax.management.monitor
	javax.management.openmbean
	javax.management.relation
	javax.management.remote
	javax.management.remote.rmi
	javax.management.timer
Java Naming And Directory Interface (JNDI)	javax.naming
	javax.naming.directory
	javax.naming.event
	javax.naming.ldap
	javax.naming.spi
Advance Networking	javax.net
	javax.net.ssl
Java Print	javax.print
	javax.print.attribute
	javax.print.attribute.standard

	javax.print.event
Advance RMI	javax.rmi
	javax.rmi.CORBA
	javax.rmi.ssl
Advance Security	javax.security.auth
	javax.security.auth.callback
	javax.security.auth.kerberos
	javax.security.auth.login
	javax.security.auth.spi
	javax.security.auth.x500
	javax.security.cert
	javax.security.sasl
Advance Sound	javax.sound.midi
	javax.sound.midi.spi
	javax.sound.sampled
	javax.sound.sampled.spi
Advance Database (Rowset)	javax.sql
	javax.sql.rowset
	javax.sql.rowset.serial
	javax.sql.rowset.spi
Java Foundation Class (JFC-Swing)	javax.swing
	javax.swing.border
	javax.swing.colorchooser
	javax.swing.event
	javax.swing.filechooser
	javax.swing.plaf
	javax.swing.plaf.basic
	javax.swing.plaf.metal
	javax.swing.plaf.multi
	javax.swing.plaf.synth
	javax.swing.table
	javax.swing.text
	javax.swing.text.html
	javax.swing.text.html.parser
	javax.swing.text.rtf
	javax.swing.tree
	javax.swing.undo
Advance Transaction	javax.transaction
	javax.transaction.xa
Java XML	javax.xml
	javax.xml.datatype
	javax.xml.namespace
	javax.xml.parsers
	javax.xml.transform
	javax.xml.transform.dom
	javax.xml.transform.sax
	javax.xml.transform.stream
	javax.xml.validation

	javax.xml.xpath
Security	org.ietf.jgss
Common Object Request Broker Architecture (CORBA)	org.omg.CORBA
	org.omg.CORBA_2_3
	org.omg.CORBA_2_3.portable
	org.omg.CORBA.DynAnyPackage
	org.omg.CORBA.ORBPackage
	org.omg.CORBA.portable
	org.omg.CORBA.TypeCodePackage
	org.omg.CosNaming
	org.omg.CosNaming.NamingContextExtPackage
	org.omg.CosNaming.NamingContextPackage
	org.omg.Dynamic
	org.omg.DynamicAny
	org.omg.DynamicAny.DynAnyFactoryPackage
	org.omg.DynamicAny.DynAnyPackage
	org.omg.IOP
	org.omg.IOP.CodecFactoryPackage
	org.omg.IOP.CodecPackage
	org.omg.Messaging
	org.omg.PortableInterceptor
	org.omg.PortableInterceptor.ORBInitInfoPackage
	org.omg.PortableServer
	org.omg.PortableServer.CurrentPackage
	org.omg.PortableServer.POAManagerPackage
	org.omg.PortableServer.POAPackage
	org.omg.PortableServer.portable
	org.omg.PortableServer.ServantLocatorPackage
	org.omg.SendingContext
	org.omg.stub.java.rmi
Document Object Model (XML-DOM)	org.w3c.dom
	org.w3c.dom.bootstrap
	org.w3c.dom.events
	org.w3c.dom.ls
Simple API for XML (SAX)	org.xml.sax
	org.xml.sax.ext
	org.xml.sax.helpers

The Java Certifications

The Sun Microsystems conduct several global certification programs:

Sun Certified Associate (SCJA)
This certification provides an ideal entry into an application development or a software project management career using Java technologies. This worldwide credential validates basic knowledge of Object-Oriented Concepts, UML representation of OO concepts, the Java programming language, and general knowledge of Java Platforms and Technologies.

Sun Certified Programmer (SCJP)
This certification is for programmers interested in demonstrating proficiency in the fundamentals of the Java programming language using the Java 2 Platform, Standard Edition (J2SE) technology.

Sun Certified Developer (SCJD)
This certification is for Sun Certified Programmers who are already familiar with the basic structure and syntax of the Java programming language, and who have a need to demonstrate advanced proficiency in developing complex, production level applications using Java 2 Platform, Standard Edition (J2SE) technology.

Sun Certified Web Component Developer (SCWCD)
This certification is for Sun Certified Programmers specializing in the application of JavaServer Pages and Servlet technologies used to present web services and dynamic web content using Java 2 Platform, Enterprise Edition (J2EE) platform.

Sun Certified Business Component Developer (SCBCD)
This certification is for Sun Certified Programmers specializing in leveraging the Java 2 Platform, Enterprise Edition (J2EE) platform technologies used to develop server-side components that encapsulate the business logic of an application.

Sun Certified Developer for Java Web Services (SCDJWS)
This certification is for Sun Certified Programmers who have been creating Web services applications using Java technology components such as those supported by the Java Web Services Developer Pack and the Java 2, Enterprise Edition (J2EE) Platform.

Sun Certified Enterprise Architect (SCEA)
This certification is for enterprise architects responsible for architecting and designing Java 2 Platform, Enterprise Edition (J2EE) technology compliant applications, which are scalable, flexible and highly secure.

Sun Certified Mobile Application Developer (SCMAD)
This certification is for Sun Certified Programmers who create mobile applications using the Java 2 Platform, Mobile Edition (J2ME) platform for cell phones or "smart" devices.

Java Download Center

From enterprise software to developer tools, Sun offers a comprehensive suite of products that help to create solutions and increase productivity. The releases that are available for download during the time, this booklet is printed are as follows:

Development Tools

- Sun Studio Compilers & Tools
- Java Studio Creator
- Java Studio Enterprise
- NetBeans and Add-on Packs

Java SE

- Java SE (JDK) 6
- Java Accessibility
- Java Access Bridge
- JavaBeans Architecture
- Java Plug-in Software for Windows XP
- Java Web Start Software
- Java Database Connectivity Technology
- Java Advanced Imaging API
- Java Authentication And Authorization Service
- Java Communications API
- Java Cryptography Extension
- Java Help System
- Java Management Extensions
- Java Media Framework API
- Java Secure Socket Extension

Java EE

- Java EE 5 SDK
- Java Application Platform SDK
- J2EE 1.4 SDK and previous versions
- ECperf Software
- Enterprise JavaBeans Technology
- J2EE Application Deployment API
- J2EE Client Provisioning Software
- J2EE Connector Architecture
- J2EE Management Specification
- JavaBeans Activation Framework (JAF)
- JavaMail
- JavaServer Faces Technology
- JavaServer Pages Standard Tag Library
- JavaServer Pages Technology
- Java Application Verification Kit (Java AVK)
- Java Authorization Contract for Containers (Java ACC)
- Java BluePrints
- Java Database Connectivity Technology
- Java Data Objects (JDO)
- Java Message Service API
- Java Persistence API
- Java Servlet API

- Java Transaction API (JTA)
- Java Transaction Service (JTS)
- SOAP with Attachment API for Java

Java ME

- Connected Device Configuration
- Connected Limited Device Configuration
- Foundation Profile
- Java API for Bluetooth
- Java Card Technology
- Java ME Content Handler API
- Java ME RMI Optional Package
- Java ME Security and Trust Services API
- Java ME Web Services
- JavaPhone API
- Java Technology for Wireless Industry
- Java TV API
- Location API for Java ME
- Mobile 3D Graphics API for J2ME
- Mobile Information Device Profile
- Personal Basis Profile
- Personal Java Technology
- Personal Profile
- SIP API for J2ME
- Sun Java Toolkit for CDC
- Sun Java Wireless Toolkit Software
- Wireless Messaging API

Database Technologies

- Java DB

Infrastructure Software

- Identity Management
- Web Server
- Directory Server
- Portal Server
- Web Proxy Server
- Sun Java System Active Server Pages
- Application Server

Networking Technologies

- Java Dynamic Management Kit
- Java Metadata Interface Software
- Jini Technology Starter Kit
- Project JXTA
- OSS through Java Initiative

Solaris

- Solaris Express Developer Edition
- Solaris OS Binaries
- Solaris Security for Developers Code Example
- Driver Development Downloads
- Tru to Solaris Migration Tool for C/C++ Source Code

- Solaris Patches
- GNOME 2.0 Desktop
- Sun Management Center
- Solaris Admin Pack
- BigAdmin

XML Technologies

- Java Architecture for XML Binding (JAXB) Software
- Java API for XML-Based RPC API
- Java API for XML Messaging (JAXM)
- Java API for XML Processing Software
- Java API for XML Registries Software
- Java Web Services Developer Pack

Java Curriculum

J2SE (Java 2 Standard Edition) Syllabus

1. **The JVM (Java Virtual Machine)**
   - Introduction to Virtual Machine
   - Virtual Machine vs. Original Machine (OS)
   - Advantages of virtual machine in a language
2. **Introduction to Computer Programming**
   - General Introduction
   - Open Source Software: GNU (GPL-General Public Lisence)
   - Programming Approaches: Passive, Active, Dynamic
   - Programming Models: Procedural, Object Oriented, Component Based
   - History and Features of Java and Sun Microsystems
3. **Getting Started with J2SE**
   - What is Java?
   - How to Get Java
   - A First Java Program
   - Compiling and Interpreting Applications
   - The JDK Directory Structure
4. **Language Fundamentals**
   - Java Keywords
   - Operators and Precedence
   - if Statements
   - switch Statements
   - Loop Statements
   - Syntax Details
   - Primitive Datatypes
   - Variables
   - Expressions in Java
   - Strings
   - Arrays
   - Enhanced for Loop
5. **Objects and Classes**
   - Defining a Class
   - Creating an Object
   - Instance Data and Class Data
   - Methods
   - Constructors
   - Access Modifiers
   - Encapsulation
6. **Using Java Objects**
   - Printing to the Console
   - printf Format Strings
   - StringBuilder and StringBuffer
   - Methods and Messages
   - toString

- Parameter Passing
- Comparing and Identifying Objects
- Destroying Objects
- Using the Primitive-Type Wrapper Classes
- Autoboxing

7. Inheritance in Java

- Inheritance
- Inheritance in Java
- Casting
- Method Overriding
- Polymorphism
- super
- The Object Class

8. Advanced Inheritance and Language Constructs

- Enumerated Types - Pre-Java 5.0
- Enumerated Types Today
- More Enumerated Types
- Abstract Classes
- Interfaces
- Using Interfaces
- Comparable
- Collections
- Generics

9. Packages

- What is Package?
- The `import` Statement
- Static Imports
- CLASSPATH and Import
- Defining Packages
- Creating jar files
- Package Scope

10. Exception Handling

- Exceptions Overview
- Catching Exceptions
- The finally Block
- Exception Methods
- Declaring Exceptions
- Defining and Throwing Exceptions
- Errors and RuntimeExceptions
- Assertions and User defined Exceptions

11. Input/Output Streams

- Overview of Streams
- Bytes vs. Characters
- Converting Byte Streams to Character Streams
- File Object
- Binary Input and Output
- PrintWriter Class
- Reading and Writing Objects
- Basic and Filtered Streams

12. Core Collection Classes

- The Collections Framework
- The Set Interface
- Set Implementation Classes
- The List Interface
- List Implementation Classes
- The Queue Interface
- Queue Implementation Classes
- The Map Interface
- Map Implementation Classes

13. Collection Sorting and Tuning

- Using Java 5.0 Features with Collections
- Sorting with Comparable
- Sorting with Comparator
- Sorting Lists and Arrays
- Collections Utility Methods
- Tuning ArrayList
- Tuning HashMap and HashSet

14. Inner Classes

- Inner Classes
- Member Classes
- Local Classes
- Anonymous Classes
- Instance Initializers
- Static Nested Classes

15. Introduction to Swing

- AWT (Abstract Window Toolkit)
- Swing and MVC Technology
- Displaying a Window
- GUI Programming in Java
- Handling Events
- Arranging Components
- A Scrollable Component
- Configuring Components
- Menus
- Using the JfileChooser

16. Swing Events and Layout Managers

- The Java Event Delegation Model
- Action Events
- List Selection Events
- Mouse Events
- Layout Managers
- BorderLayout, FlowLayout, GridLayout, CardLayout & GridbagLayout
- BoxLayout
- JtabbedPane, JTree, JTable

17 Introduction to JDBC

- The JDBC Connectivity Model
- Database Programming

- Connecting to the Database
- Creating a SQL Query
- Getting the Results
- Updating Database Data
- Finishing Up

18. JDBC SQL Programming

- Error Checking and the SQLException Class
- The SQLWarning Class
- JDBC Types
- Executing SQL Queries
- ResultSetMetaData
- Executing SQL Updates
- Using a PreparedStatement
- Parameterized Statements
- Stored Procedures
- Transaction Management

19. Introduction to Threads

- Non-Threaded Applications
- Threaded Applications
- Creating Threads
- Thread States
- Runnable Threads
- Coordinating Threads
- Interrupting Threads
- Runnable Interface
- ThreadGroups

20. Thread Synchronization and Concurrency

- Race Conditions
- Synchronized Methods
- Deadlocks
- Synchronized Blocks
- Thread Communication - wait()
- Thread Communication - notify()
- Java 5.0 Concurrency Improvements
- Thread-Aware Collections
- Executor
- Callable

21. Java Performance Tuning

- Is Java Slow?
- Don't Optimize Until You Profile
- HotSpot Virtual Machine
- Garbage Collection Concepts
- Garbage Collection Generations
- Garbage Collection in Java 5.0
- Object Creation
- String, StringBuffer, and StringBuilder
- Synchronized
- Inline methods
- Tuning Collections

22. **Appendix A - Regular Expressions (JDK 1.5)**

• Pattern Matching and Regular Expressions
• Regular Expressions in Java
• Regular Expression Syntax
• Special Characters
• Quantifiers
• Assertions
• The Pattern Class
• The Matcher Class
• Capturing Groups

23. **Appendix B - J2EE Overview**

• Introduction to J2EE
• J2SE Building Blocks
• Servlets, JSPs, and Web Applications
• Web Services
• Enterprise JavaBeans
• Additional J2EE APIs
• J2EE Clients
• The J2EE Platform

Advance J2SE Syllabus

1. **Advanced I/O - Object Serialization**

• What is Serialization?
• Serializable Objects
• Writing and Reading an Object
• Handling Exceptions
• Customizing and Controlling Serialization
• Versioning

2. **Advanced I/O - New I/O**

• The java.nio package
• Buffers and Channels
• Buffer Implementations and Methods
• FileChannel
• File Locking
• MappedByteBuffer
• Transferring Data Between Channels
• Character Sets

3. **Reflection**

• Introduction to Reflection
• The Class class
• The reflect Package
• Constructors, Fields, and Methods
• Exception Handling and Reflection
• JavaBeans
• Dynamic Programming

4. **Advanced JDBC**

• JDBC SQL Escape Syntax
• The execute() Method

- Batch Updates
- Scrollable/Updatable Result Sets
- Large Objects
- Working with Savepoints
- Introduction to RowSets
- RowSet Implementations
- DataSources

5. Networking with Sockets

- Clients and Servers
- Ports, Addresses and Protocols
- The Socket Class
- Communication Using I/O
- Servers
- The ServerSocket Class
- Concurrent Servers
- The URL Class
- The URLConnection Class

6. Advanced RMI - Dynamic Classloading and Activation

- Callbacks
- Introduction to Activation
- Automatic Class Distribution
- Using Activation
- The Distributed Garbage Collector
- Introduction to JNDI
- Using JNDI to access the RMIRegistry
- RMI-IIOP

7. Managing Security Policies

- Untrusted code - RMI and Applets
- The Java Security Model
- Policy Entries and Files
- Using the Policy Tool
- Security Managers
- Securing Applets and Applications

8. Keys, Signatures, and Certificates

- Jar Files
- Data Security Concerns
- Message Digests
- Digital Signatures
- Using keytool
- Using jarsigner
- Certificates
- Managing Keys and Certificates
- Security Policies for Signed Code
- Java Cryptography Architecture
- Java Cryptography Extension

9. Encryption with the javax.crypto Package

- Cryptography Concepts
- Algorithms
- Padding Schemes and Modes

- The Cipher Class
- Encrypting and Decrypting Data
- Encrypting and Decrypting Streams
- Exchanging Encrypted Keys
- Sealed Objects
- Unlimited Strength Encryption

10. Java Authentication and Authorization Service (JAAS)

- Authentication and Authorization
- JAAS Overview
- LoginContext
- Subjects and Principals
- Authentication with the NT Login Module
- Defining Permissions in Policy Files
- Callbacks
- NameCallback and PasswordCallback
- Authentication with the JNDI Login Module
- The Policy Class

11. Java Naming and Directory Interface (JNDI)

- Naming and Directory Services
- Namespaces and Contexts
- Naming Operations
- Bindings
- Attributes
- Directory Operations
- DNS Lookups with JNDI
- JNDI in J2EE

12. Parsing XML with Java – JAXP

- The Java API for XML Processing
- Introduction to SAX
- SAXParser
- SAX Event Methods
- Introduction to DOM
- DocumentBuilder
- The DOM API
- Validation
- Introduction to XSLT
- Transformer

13. Java Native Methods

- Overview of Java Native Methods and JNI
- How to Create and Use Native Methods
- Native Method Declaration
- Using javah
- Creating the Implementation Code
- Compilation
- Distribution
- Using the Native Methods
- JNI (Java Native Interface)
- Passing Arguments
- Calling Java Methods in Native Code

14. **Java Design Patterns - Creational Patterns**

- What are Design Patterns?
- What are Creational Patterns?
- Singleton – Introduction
- Singleton – Implementation
- Singleton - When to use?
- Factory Method – Introduction
- Factory Method – Implementation
- Factory Method - When to use?
- Builder – Introduction
- Builder – Implementation
- Builder - When to use?

15. **Java Design Patterns - Structural Patterns**

- What are Structural Patterns?
- Facade – Introduction
- Facade – Implementation
- Facade - When to use?
- Adapter – Introduction
- Adapter – Implementation
- Adapter - When to use?
- Composite – Introduction
- Composite – Implementation
- Composite - When to use?

16. **What are Behavioral Patterns?**

- What are Behavioral Patterns?
- Template – Introduction
- Template – Implementation
- Template - When to use?
- State – Introduction
- State – Implementation
- State - When to use?
- Observer – Introduction
- Observer – Implementation

17. **Appendix A - JDBC SQL Programming**

- Hands on Model DBMS/RDBMS/OODBMS
  - ✓ Flat files/CSV files
  - ✓ dBASE/FoxPro files
  - ✓ MS-Excel/MS-Access
  - ✓ MySQL
  - ✓ PostgreSQL
  - ✓ IBM Cloudscape
  - ✓ MS-SQL Server
  - ✓ Oracle 10g
  - ✓ IBM DB2

18. **Appendix B - (Java IDE)**

- Eclipse
- NetBeans
- JBuilder
- JDeveloper

19. **Appendix C - SCJP & SCJD Certification Guide**

- Extensive drilling of sample questions
- Excercising past test questions
- Mock test
- Lakshya certification test

J2EE (Java 2 Enterprise Edition) Syllabus

1. **J2EE**
   - J2EE Overview
   - Client Tier
   - Middle Tier
   - Application Server Tier
   - The J2EE Platform
   - J2EE Skills
2. **Getting Started with JSP**
   - Dynamic Web Content
   - The JSP Solution
   - JSP Syntax
   - JSP Deployment
   - Variables and Expressions
   - Implicit Objects
   - page and taglib Directives
   - Include and Forward
   - Exception Handling
3. **Forms and JavaBeans**
   - HTML Forms
   - JavaBeans
   - JavaBeans and JSP
   - Bean Properties
   - Property Types
   - Properties and Forms
   - Bean Scopes
4. **Introduction to JSTL**
   - JSP Expression Language
   - Expression Language Implicit Objects
   - What is JSTL?
   - Core Tags – Conditionals
   - Core Tags - Iteration and Import
   - XML Manipulation Tags
   - Internationalization Tags
   - SQL Tags
5. **Servlet Basics**
   - Browsers, Servers and Servlets
   - The Basic Servlet
   - The Servlet Life Cycle
   - The HttpServlet Approach
   - More do Methods
   - Threading in Servlets
   - Debugging

6. **Request and Response**

• Request and Response Basics
• The HttpServletRequest Object
• Request Headers
• Status Codes
• Response Headers
• Ensuring Valid Characters

7. **Session Tracking**

• Understanding Cookies
• The Cookie Class
• Cookies in JSP
• Cookie Properties
• Session Tracking
• The HttpSession Class
• Sessions in JSP
• Encoding URLs
• Terminating Sessions

8. **Web Applications**

• Web Application Components
• ServletContext
• Forward and Include
• Supporting Files
• Deployment Descriptor
• Deployment Descriptor Elements
• Security – Authentication
• Security in the J2EE Application Server
• Security - Authorization

9. **Introduction to JNDI**

• Naming and Directory Services
• Namespaces and Contexts
• Naming Operations
• Bindings
• Attributes
• Directory Operations
• DNS Lookups with JNDI
• JNDI in J2EE

10. **DataSources**

• DataSources
• Connection Pools in the J2EE Application Server
• Data Sources in the J2EE Application Server
• Connecting to a DataSource

11. **Introduction to JavaMail**

• Mail Systems and JavaMail
• The javax.mail Packages
• Establishing a Session
• The Message Interface
• Sending a Message
• Message Stores
• Mail Folders

- Multipart Messages

12. JMS (Java Messaging Sevice)

- Introduction to JMS Concepts
- What is JMS?
- Parent Interfaces and GMD
- JMS Definitions
- Message Object
- Multi-Threading and JMS Exception
- PTP Domain and Interfaces
- Pub/Sub Domain and Interfaces
- J2EE Application Server Administered Objects
- Creating the Client
- Handling the Message
- Producing the Message

13. EJB and the J2EE Architecture

- Evolution of Distributed Computing on the Web
- The J2EE Solution
- The Enterprise JavaBean
- Roles in Enterprise JavaBeans Development
- EJB Container and Application Server
- Web Services and J2EE

14. Getting Started with EJB

- Defining the Bean Class
- Remote Interface
- Writing Business Methods
- Home Interface
- Deployment Descriptors and Deployment
- The Client
- Locating the Bean
- Create an Enterprise Bean Instance
- Invoking the Bean's Methods
- Compiling and Running the Client

15. Three Types of EJB

- A Session Bean
- A Message-Driven Bean
- An Entity Bean
- What About State?
- Stateless Session Beans
- Stateful Session Beans
- MDB Code
- Entity Bean Persistence Models
- Entity Bean Code
- Deployment Descriptor

16. Case Study

- Account Local and Local Home Interfaces
- Account EJB
- ejb-jar.xml
- sun-ejb-jar.xml
- Teller Remote and Home Interfaces

- TellerEJB
- ejb-jar.xml - Take 2
- Teller.html
- Teller Servlet
- TransferBean
- Results.jsp
- web.xml
- application.xml

17. Appendix A - Deploying a JSP with the deploytool

- WAR Wizard
- Changing the WAR
- ANT Tool
- JUnit Java testing tool

18. Appendix B - JSP Framework

- Hands on Struts
- Hands on Springs
- Hands on Hibernate

19. Appendix C - Hands on Servers

- Blazix Server from Desiderata Software
- Apache Tomcat 5.5.17
- Macromedia JRun 4
- JBoss Web Server from RedHat Inc.
- BEA WebLogic Server 7.0 & WebLogic Server 8.1
- IBM WebSphere Application Server 5.0,5.1
- Sun ONE Application Server J2EE 1.3 SDK Sun ONE Studio 5
- Borland Enterprise Server, AppServer Edition JBuilder 6.0

Do you know JAVA is?

- ✓ Java is Object Oriented Programming (OOP's) language. This gives the capabilities of creating flexible, modular and reusable program code.
- ✓ Java adds 2.5 percent of the GDP to our country.
- ✓ There are 20,000 Java professionals required every year in our country alone.
- ✓ The Java language concepts, keywords, operators and syntax are derived from C/C++ language.
- ✓ Java is easy to use, good for beginners and experts alike.
- ✓ Java is open source. It is completely free to procure, use and distribute with the source code.
- ✓ Java is the most powerful and preferred application development platform in the computer world.
- ✓ Java benefited from the fact that it ran in a place where no programs had run before - inside the Internet browser window.
- ✓ Java is platform independent, portable across platforms and operating systems. Java source is compiled into a universal format - instructions for a *virtual machine*.
- ✓ Java is secured. No virus can remain undetected due to an infected java files. Java provides several layers of protection from dangerously flawed code, as well as more mischievous things like viruses and Trojan horses.
- ✓ There is a large number of Open source IDEs (Integrated Development Environment) that makes Java development easier and faster.

LOGOS OF SOME GIANT PRODUCTS, COMPANIES AND INSTITUTIONS

Sun Java		Sun Microsystems	
Apache Software Foundation		IBM WebSphere	
BEA Weblogic		Microsoft Corporation	
MySQL AB		Net Beans	
Oracle Corporation		PHP	
PostgreSQL		Apache Struts	
Linux		International Business Machine	
Howlett Packard		Hibernate	

	BlueJ		Borland Software Corporation
	American Megatrend		GNU General Public Licence
	Apple Machintos		Firefox
	Seagate Corporation		Symantec
	Python		Red Hat Linux
	Eclipse IDE		Netscape Navigator
	BaaN International		SAP AG
	Mozilla Foundation		Novell, Inc.
	Infosys		Digital Electronics
	Santa Cruz Operation		Microsoft .NET
	JBoss.org		Springs Framework
	Delphi		Sybase

DB2

Sun Solaris

Informix

Firebird

SAMPLE JAVA PROGRAMS

SOLVE-1

This program displays all the files and directories of a specified path. The path may be a drive name or a directory name. The output of the program may be stored into the optionally specified text file.

Compile the program using the following command from command prompt:

```
javac Tree.java
```

Run the program using the following command from the command prompt:

```
java Tree <[path:\]dirname> [<outputFile>]
```

```
/*
    Program developed by:
    Milan Das@Lakshya Solutions Ltd.
    Tel:    9238585618
    email:  milan.das@lakshyasolutions.com
            milan.das@lakshyatraining.org
*/
import java.io.File;
import java.io.FileOutputStream;

public class Eleventh_4 {
     int tab;
     String buffer;
     FileOutputStream output;

     //   Constructor receiving the path name and
     //   the output file name
     public Eleventh_4(String dirName, String outputFile) {
          tab = 0;
          try {
               output = new FileOutputStream(outputFile);
               output.write((dirName+"\r\n").getBytes());
          } catch(FileNotFoundException e) {
```

```
      } catch(NullPointerException e) {
            System.out.println(dirName);
      } catch(Exception e) {
      }

      File f = new File(dirName);
      this.tree(f);

      try {
            output.close();
      } catch(NullPointerException e) {
      } catch(IOException e) {
      } catch(Exception e) {
      }
}

private void tree(File f) {
      File children[] = f.listFiles();
      for(int i=0; i<children.length; i++) {
      if(children[i].isDirectory()) {
            try {
                  buffer = this.replicate("|   ",tab)+
                        "+-- "+children[i].getName();
                        output.write(
                        (buffer+"\r\n").getBytes());
            } catch(IOException e) {
            } catch(NullPointerException e) {
                  System.out.println(buffer);
            } catch(Exception e) {
            }
            tab++;
            this.tree(children[i]);
            tab--;
      } else {
            try {
                  buffer = this.replicate("|   ",
                  tab) + "|-- " + children[i].getName();
                  output.write(
                        (buffer+"\r\n").getBytes());
            } catch(IOException e) {
            } catch(NullPointerException e) {
                  System.out.println(buffer);
            } catch(Exception e) {
         }
      }
   }
}

private String replicate(String data, int times) {
      String rValue = new String();
      for(int i=0; i<times; i++) rValue += data;
      return rValue;
}

public static void main(String arg[]) {
      try {
```

```
                new Eleventh_4(arg[0], arg[1]);
            } catch(ArrayIndexOutOfBoundsException e) {
                try {
                    new Eleventh_4(arg[0], null);
                } catch(ArrayIndexOutOfBoundsException e1) {
                    System.out.println("Parameter
                    missing!\r\nFile Tree View\r\nUsage:
                    java Eleventh_4 <dirName> [<outputFile>]");
                } catch(Exception e1) {
                }
            } catch(Exception e) {
            }
        }
}
```

SOLVE-2

Write a simple JSP script to display the data retrieved from a given SQL string. The data must be displayed in a tabular format. The SQL string is given into the input text field of a HTML page which will call the JSP script on clicking on the SUBMIT button.

Environment:

The program is intended to run on a single machine where Apache Tomcat Application Server is installed and active. In the same machine, MySQL database is also installed. Various tables in different database under MySQL is also readily available.

Deploy the two files given below in the following directory:

```
C:\Program Files\Apache Software Foundation\Tomcat
5.5\webapps\example
```

To invoke the program type the following line in the address bar of the internet browser:

```
http://localhost:8080/example/DynamicQuery.html
```

HTML Script (File name: **`DynamicQuery.html`**)

```
<!--
    Program developed by:
    Milan Das@Lakshya Solutions Ltd.
    Tel:    9238585618
    email:  milan.das@lakshyasolutions.com
            milan.das@lakshyatraining.org
-->
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
      <head>
            <title>Dynamic Query</title>
      </head>

      <body>
            <center>
```

```
            <font face="Arial" color="green">
                  <h1>Dynamic Query</h1>
                  <br><hr><br>
                  <form action="http://localhost:8080/dynamicquery/
                                    DynamicQuery" method="get">
                        SQL Query:
                        <input type="text" name="sql"
                                size="100">
                        <br><br>
                        <input type="submit">
                             
                        <input type="reset">
                        <br><br>
                  </form>
            </font>
            </center>
      </body>
</html>
```

JSP script (File name: **`DynamicQuery.jsp`**)

```
<!--
    Program developed by:
    Milan Das@Lakshya Solutions Ltd.
    Tel:    9238585618
    email:  milan.das@lakshyatraining.org
-->
<%@ page import="java.sql.*"%>
<html>
   <head>
      <title>JSP Program</title>
      <%
         String sql = request.getParameter("sql");
         Class.forName("org.gjt.mm.mysql.Driver");
         Connection connect = DriverManager.getConnection(
              "jdbc:mysql://localhost/group8");
         Statement statement = connect.createStatement();
         ResultSet result = statement.executeQuery(sql);
         ResultSetMetaData resultMD = result.getMetaData();
         int colCount = resultMD.getColumnCount();
      %>
   </head>
   <body>
      <center>
         <h1>Dynamic Query</h1><hr>
         <table>
            <tr>
               <% for(int i=1; i<=colCount; i++) { %>
                  <th bgcolor="gray">
                  <%=resultMD.getColumnLabel(i).toUpperCase()%>
                  </th>
               <% } %>
            </tr>
            <% while(result.next()) { %>
```

```
            <tr>
            <% for(int i=1; i<=colCount; i++) { %>
               <td><%=result.getString(i)%></td>
            <% } %>
            </tr>
         <% } %>
      </table>
    </center>
  </body>
  <%
      result.close();
      statement.close();
      connect.close();
  %>
</html>
```

PRACTICE QUESTIONS

J2SE (Java 2 Standard Edition)

SIno.	Question with options

1.	Given the following program,
	`1. public class Test {`
	`2. public static void main(String [] args) {`
	`3. signed int x = 10;`
	`4. for (int y=0; y<5; y++, x--)`
	`5. System.out.print(" " + x);`
	`6. }`
	`7. }`
	
	What is the result? (Choose one.)
	A. 10 9 8 7 6
	B. 9 8 7 6 5
	C. Compilation fails
	D. An exception is thrown at runtime

2.	Which is a reserved word in the Java programming language? (Choose one.) 1. `method` 2. `native` 3. `subclasses` 4. `reference` 5. `array`
3.	Which one of these lists contains only Java programming language keywords? (Choose one.) A. `class, if, void, long, Int, continue` B. `goto, instanceof, native, finally, default, throws` C. `try, virtual, throw, final, volatile, transient` D. `strictfp, constant, super, implements, do` E. `byte, break, assert, switch, include`

4.	Which two are keywords? (Choose two.) A. `interface` B. `unsigned` C. `Float` D. `this` E. `String`
5.	Which three are legal array declarations? (Choose three.) A. `int [] myScores [];` B. `char[] myChars;` C. `int[6] myScores;` D. `Dog myDogs[];` E. `Dog myDogs[7];`
6.	Which of the following are valid Java comments? A. `\\ This is a comment.` B. `/* This is a comment. */` C. `/** This is a comment. */` D. `\* This is a comment *\`
7.	Which of the following are valid Java identifiers? A. `%id` B. `$id` C. `_id` D. `#id`
8.	To create a `public` class `MyClass` and successfully compile, which of the following are true? A. `MyClass` must have a correctly formed `main()` method. B. `MyClass` must be defined in the file `MyClass.java`. C. `MyClass` must be defined in the `MyClass` package. D. `MyClass` must be imported.
9.	The Java source code file, containing the `public class Test`, To successfully compile, which of the following must be true? A. It must import java.lang. B. It must declare a public class named Test. C. It must be named Test.java. D. It must have a package statement.
10.	In order for the `MyProgram` program to be compiled and run, which of the following must be true? A. The `MyProgram` class must be defined in `MyProgram.java`. B. `MyProgram` must be declared `public`. C. `MyProgram` must have a correctly formed `main()` method. D. `MyProgram` must `import java.lang`.
11	Which of the following are true? A. If a `package` statement is included in a source code file, it must appear as the first non-blank line. B. If an `import` statement is included in a source code file, it must appear as the first non-blank line. C. If a `main()` method is included in a source code file, it must appear as the first non-blank line.

	D. If a `public interface` is declared in a source code file, it must have the same name as the source code file.
12.	Which one of the following is a valid declaration of an applet? A. `public class MyApplet extends java.applet.Applet {` B. `public Applet MyApplet {` C. `public class MyApplet extends applet implements Runnable {` D. `abstract class MyApplet extends Applet {` E. `class MyApplet implements Applet {`
13.	Referring to the line below, what datatype could be returned by method `check4Biz()`? `if(check4Biz(storeNum) != null) { }` A. `Boolean` B. `int` C. `String` D. `char` E. `byte`
14.	Which of the following are valid main() methods? A. `public static void main() { }` B. `public static void main(String[] argc) { }` C. `void main(String[] args) { }` D. `public static void main(String []args) { }`
15.	What is the output of the following program when it is invoked using the command line `java Test this is a test?` 1. `class Test {` 2. `public static void main(String[] args) {` 3. `System.out.println(args[1]);` 4. `}` 5. `}` Choose the right output: A. `this` B. `is` C. `a` D. `test?`

Adv. J2SE (Advance Java 2 Standard Edition)

1.	Which code segment could execute the stored procedure "countRecs()" located in a database server?
	A. `Statement stmt = connect.createStatement();`
	`stmt.execute("COUNTRECS()");`
	B. `CallableStatement cs = con.prepareCall("{call COUNTRECS}");`
	`cs.executeQuery();`
	C. `StoreProcedureStatement spstmt = connect.`

	createStoreProcedure("countRecs()");
	spstmt.executeQuery();
	C. PrepareStatement pstmt =
	connection.prepareStatement("countRecs()");
	pstmt.execute();
	D. Statement stmt =
	connection.createStatement();
	stmt.executeStoredProcedure("countRecs()");

J2EE (Java 2 Enterprise Edition)

1.	

Java Web Application Servers

ATG Dynamo Application Server	BEA WebLogic Server 7.0 & WebLogic Server 8.1	Borland Enterprise Server, AppServer Edition JBuilder 6.0	Fujitsu INTERSTAGE Application Server
Hitachi Cosminexus	WebSphere Application Server 5.0 WebSphere Application Server 5.1 WebSphere Application Server for z/OS 5.1	IONA Orbix E2A Application Server	KingDee KingDee Apusic Application Server V4.0
Macromedia JRun 4	NEC WebOTX 5	Novell exteNd	Oracle 9i Application

		Application Server 5	Server
Pramati Server 3.0 & Studio 3.0	SAP Web Applications Server	SAS AppDev Studio 2.0.2 Preview Release	SeeBeyond ICAN
SpiritSoft	Sun ONE Application Server J2EE 1.3 SDK Sun ONE Studio 5	Sybase EAServer 4.1, 5.0, 5.2, 5.3	Tmax Soft JEUS 4.0
TongTech Co., Ltd. TongWeb App Server v4.1	Trifork Application Server 3.1	Advanced JAVA Application / Web Server	Tomcat Apache Software Foundation

Gnu (General Public License)
Apache Software Foundation
BSD (Berkeley)