BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 4

DCA2202

JAVA PROGRAMMING

Unit 1

Introduction to Java:

Table of Contents

1. INTRODUCTION

You all must have already experienced the joy of programming the computer language in C or C++. You must have learnt that a program is a sequence of instructions which are called programming statements. These programming statements are executed usually in a sequential manner, i.e., these instructions are executed one after another in the order that they are written. Hence the complete program executes certain results. Programming languages are basically categorized into procedure-oriented language or object-oriented language.

The procedural–oriented languages are based on procedures. Functions are the basic unit of this type of language. Functions in the program and the way data need to be represented are mainly considered in this type of language.

The object-oriented languages are mainly programmed keeping in view the components that the user perceives. Objects are the basic unit in this type of language. All the objects have states and behaviors which represent the data and operations and the way they interact with each other. This unit welcomes you to the world of Java programming and orients you to the concept of object-oriented programming with the help of Java. Java is a powerful computer language that can be learned even if you have just started programming.

1.1 Objectives:

After studying this unit, you should be able to:

- *Write an essay on the history of Java*
- *Explain various features of Java*
- *Explain the concept of Byte Code*
- *Describe Java Development Kit (JDK) and Software Development Kit(SDK)*
- *Discuss the security aspects of Java*

2. HISTORY OF JAVA

The history of Java programming language is usually associated with the World Wide Web (www), i.e., its origin predates the web. During the early period,the first graphical browser called **Mosaic** was released. Non- programmers started accessing the World Wide Web and the Web grew dramatically. People with different types of machines and operating systems started accessing the applications available on the web.

In the early 90s, people could see that the power of network computing can be used to solve various activities of everyday life. In 1991, a small group of Sun engineers called the "Green Team" believed that digital consumer devices and computers can be developed to a higher extent with the use of a new type of programming language. A team was formed under the name Green Project and included **Patrick Naughton** and **Mike Sheridan** led by **James Gosling** to create products for the smart electronics market.

The chief programmer of Sun Microsystems, James Gosling, was given the task of creating the software for controlling consumer electronic devices. This team wanted a fundamentally new way of computing, based on the power of networks, and wanted the same software to run on different kinds of computers, consumer gadgets, and other devices. This new language was originally designed for small, embedded systems in electronic appliances like set-top boxes. At first, it was called "Greentalk" by James Gosling and the file extension was .gt. After that, it was called Oak and was developed as a part of the Green project. Members of the Oak team realized that a new programming language was required that would provide cross-platform independence. It would also be independent of the hardware, the network, and the operating system for its execution.

Since the name of the new programming language had to be patented, the team wanted a new name for it. New names were suggested such as "dynamic", "revolutionary", "Silk", "jolt", "DNA" etc. It had to be named such that it could say that it is revolutionary, dynamic, lively, cool, unique, and easy to spell. In 1995, Oak was renamed "Java" because it was already a trademark by Oak Technologies. Very soon, Java became an integral part of the web.

The main motive for developing Java was to implement a virtual machine and a language that is simple and with the capability of "Write Once, Run Anywhere" with free runtimes on multiple platforms, e.g. Windows, Linux, Sun Solaris, MacOs etc.

According to James Gosling "Java was one of the top choices along with Silk". Since Java was so unique, most of the team members preferred Java. Note that, Java is just a name and not an acronym. In 1995, Sun Microsystems released the version of Java development kit as JDK Alpha and Beta. Within a very short period of time, most of the programmers started shifting to Java. In 1995, Time magazine called Java one of the Ten BestProducts of 1995.

Sun Microsystems, which was acquired by Oracle Corporation in 2010, started releasing Java Development Kit (JDK) with various versions as given below:

- JDK 1.0 got released on 23rd Jan, 1996
- JDK 1.1 got released on 19th Feb, 1997
- J2SE 1.2 got released on 8th Dec, 1998
- J2SE 1.3 got released on 8th May, 2000
- J2SE 1.4 got released on 6th Feb, 2002
- J2SE 5.0 got released on 30th Sep, 2004
- Java SE 6 got released on 11th Dec, 2006
- Java SE 7 got released on 28th July, 2011
- Java SE 8 got released on 18th March, 2014
- Java SE 11 got released in September 2018
- Java SE 17 got released in September 2021

Every new version of Java released includes some new features or new style of coding. Java works just everywhere, from the smallest devices to supercomputer. Java technology components (programs) do not depend on the kind of computer, telephone, television, or operating system they run on. They work on any kind of compatible device that supports the Java platform.

Self-Assessment Questions - 1

1. The earlier name of Java was _____.
2. The members of Green Project were _____, _____and _____.
3. _____is the first graphical browser.

3. FEATURES OF JAVA

Before we proceed further to discuss interesting features of Java, let us get introduced to some important terms and concepts of Java.

- **Class:** The basic unit of Object Oriented Programming (OOP) is a class. A class is a blueprint, template, or prototype for creating different **objects** which define their properties and behaviors. A class is an abstract data type, or user-defined type. Java objects exhibit the properties and behaviors defined by its class. A class can contain fields and methods to describe the behavior of an object.
- **Object:** Java defines data as objects with methods that support the objects. When we talk about an object, we conceptualize software objects.The term "object" usually refers to instances. An instance is a realization of a particular item of a class. We use a class to define a new type of object. Any code that you write in Java is inside a class. Any object consists of state and related behavior. An object stores its state in fields and exposes its behavior through methods. Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication.
- **Java Magic - Byte Code:** A programming language must be portable and also provide security. Java is both interpreted and compiled. Let us briefly discuss what the function of interpreter and compiler is. An interpreter verifies the code and executes it line by line. Only when the execution reaches the statement with error, the error is reported. Compilation is the process of converting the code that you type, into a language that the computer understands i.e. machine language. When you compile a program using a compiler, the compiler checks for syntactic errors in code and lists all the errors on the screen. To achieve the platform independency, the Java designer passes Java code through the compilation stage and interpretation stage.

  Java compiler converts the source code to intermediate binary form called the bytecode. But the bytecode is not a language to be understood by machine or processor. Only Java Virtual Machine (JVM) understands bytecode. Java interpreter converts the bytecode into a machine (processor) readable binary code. The binary

code is then executed by the processor to run the program. In standard form, JVM is an interpreter for bytecode.

Now let us discuss the features of Java programming language. The following definition of Java by Sun Microsystems lists all the features of Java.

'Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multi- threaded and dynamic language.'

The Java buzzwords or features of Java

The fundamental forces that necessitated the invention of Java are portability and security but other factors also played an important role in molding the final form of the language. The key considerations that characterizes Java programming language are the following list of buzzwords:

- Simple
- Secure
- Portable
- Object-oriented
- Robust
- Multithreaded
- Architecture-neutral
- Interpreted
- High performance
- Distributed
- Dynamic
- Platform Independent

Let's examine what each of them implies.

Simple

Java is easy for the professional programmer to learn and use effectively. For an experienced C++ programmer, learning Java will require very little effort because Java inherits the C/C++ syntax and many of the object- oriented features of C++. Expressiveness

is more in Java. Most of the complex or confusing features in C++ are removed in Java like pointers etc. or implemented in a cleaner, more approachable manner.

Secure

Java forces you to handle unexpected errors. This ensures that Java programs are robust (reliable), bug free and do not crash. Due to strong type-checking done by Java on the user's machine, any changes to the program are tagged as error and the program will not execute. Therefore Java is secured. The fact that a Java program is interpreted also helps to make it secure. Because the execution of every Java program is under the control of the JVM, the JVM can contain the program and prevent it from generating side effects outside the system. Java provides data security through encapsulation.

JVM that executes the bytecode contains bytecode verifiers that ensure that the bytecode has not been tampered with.

Besides the basic security features within the language framework, libraries and frameworks are available in Java that provided cryptographic capabilities.

Java also contains tools like keytools, and jarsigners that help in secure communications.

Portable

The key factor that allows Java to solve both the security and the portability problems is that the output of a Java compiler is not executable code. Rather, it is bytecode. The fact that a Java program is executed by the JVM helps to solve the major problems associated with the portability of programs over the Internet. Remember, although the details of the JVM will differ fromplatform to platform, all interpret the same Java bytecode. Due to the bytecode compilation process and interpretation by a browser, Java programs can be executed on a variety of hardware and operating systems. The Java interpreter can execute Java code directly on any machine on which a Java interpreter has been installed. Hence, Java code is portable.

Object-Oriented

Java is also defined by a paradigm that almost "Everything is an Object" and the pragmatists' model – "stay out of my way". All program code and data reside within objects and classes. Java comes with an extensive set of classes, arranged in packages that can be

used in your programs through inheritance. The object model in Java is simple and easy to extend. It supports all the features of object oriented model like: Encapsulation, Inheritance, Polymorphism and Abstraction. Only simple data types such as integers, are kept as high-performance non-objects.

Robust

A program or an application is said to be robust (reliable) when can respond in all the situations. Java help to make the programs robust or reliable due to the features such are: (i) Type checking and

(ii) Exception handling. Java is a strictly typed language and so it checks your code at compile time. It also checks your code at run time. Java helps you to avoid hard-to-track-down bugs that often turn up in hard-to-reproduce run-time situations. So whatever you have written will always behave in a predictable way under diverse conditions.

Java is robust because it can handle memory management mistakes and mishandled exceptional conditions (that is, run-time errors). Java virtually is able to manage memory allocation and de-allocation for you. De-allocation

is completely automatic, because Java provides garbage collection for unused objects. Java handles exceptional conditions (such as division by zero or "File not found" etc.) by providing object-oriented exception handling mechanism. In a well-written Java program, all run-time errors can and should be managed by your program.

Multithreaded

Java supports multithreaded programming, which allows you to write programs to meet the real-world requirement for creating interactive as well as networked programs. Java supports multithreading which is not supported by C and C++. A thread is a light weight process. Multithreading increases CPU efficiency. A program can be divided into several threads and each thread can be executed concurrently or in parallel with the other threads. An example of a real world requirement is - While we are listening to music, we can write in a word document or play a game at the same time. The Java run-time system provides solution for multi-process synchronization and enables you to construct smoothly running interactive systems. Java's easy-to-use approach to multithreading allows you to think

about the specific behavior of your program, not the multitasking subsystem. Multithreading is the ability of an application to perform multiple tasks at the same time. You can create multithreading programs using Java.

Architecture-Neutral

All programmers want their code to satisfy - code longevity and portability. Bytecode helps Java to achieve portability. There is no guarantee with other programming language that if you write a program today, it will run tomorrow – even on the same machine. Operating system upgrades, processor upgrades, and changes in core system resources can all combine to make a program malfunction. One of the goals during for Java was "write once; run anywhere, anytime, forever". Bytecode can be executed on computers having any kind of operating system or any kind of CPU. Since Java applications can run on any kind of CPU, Java is architecture-neutral.

Interpreted and high performance

Although Java bytecode can be interpreted on any system that provides a Java Virtual Machine (JVM), we need to consider the expense of its performance in cross-platform environments. There are other interpreted

systems, such as BASIC, Tcl, and PERL, which suffer from almost insurmountable performance deficits. Java, however, was designed to perform well on very low-power CPUs. In Java 1.0 version there was an interpreter for executing the bytecode. An interpreter is quite slow, when compared to a compiler, so java programs were slow to execute. After Java 1.0 version the interpreter was replaced with JIT (Just-In-Time) compiler. JIT compiler uses Sun Microsystem's Hot Spot technology. JIT compiler converts the byte code into machine code piece by piece and caches them for future use. This enhances the program performance, i.e., programs execute rapidly. Java run-time systems that provide this feature lose none of the benefits of the platform-independent code.

Distributed

Java supports distributed computation, where resources are shared.The server and client(s) can communicate with another and the computations can be divided among several computers which makes the programs to execute rapidly. Java is designed for the

distributed environment of the Internet, because it handles TCP/IP protocols. In fact, accessing a resource using a URL is not much different from accessing a file. This feature brings an unparalleled level of abstraction to client/server programming.

The multiplatform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems.

Dynamic

Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time. The Java Virtual Machine (JVM) maintains a lot of runtime information about the program and the objects in the program. Libraries are dynamically linked during runtime. So, even if you make dynamic changes to pieces of code, the program is not affected. This provides robustness to the applet environment where small fragments of bytecode may be dynamically updated on a running system.

Platform Independent

Java code can be run on multiple platforms e.g. windows, Linux, sun Solaris, Mac OS etc. Java code is compiled by the compiler and converted into byte code. This byte code is a platform-independent code because it can run in

any of the hardware and software environments (on multiple platforms) validating the statement - Write Once and Run Anywhere (WORA).

Self-Assessment Questions - 2

4. _____is the process of converting typed code to machine code.
5. Java code is _____, so that it can easily run on any systems.
6. Java bytecode can be interpreted on any system that provides a _____.
7. _____is the ability of an application to perform multiple tasks at a time.
8. Java is both and _____.
9. The Java Virtual Machine (JVM) maintains a lot of runtime information about the program and the objects in the program. (True/False)

4. JAVA VIRTUAL MACHINE (JVM), JAVA RUNTIME ENVIRONMENT (JRE) AND JAVA DEVELOPMENT KIT (JDK)

Java Virtual Machine (JVM)

JVM (Java Virtual Machine) is an abstract machine, i.e., it is a software implementation of a computer that executes programs like a real machine. It specifies and provides a runtime environment where java bytecode can be executed. Since JVM is a software implementation, it is written specifically for a specific operating system. So JVM is platform-dependent because the configuration of each OS differs. For example, a special implementation is required for Linux and a different type of implementation is required for Windows OS. JVMs are available for many hardware and software platforms.

The main tasks of JVM is to load the code, then verify code and if code is all correct, it will execute code. JVM also provides runtime environment. Java programs are compiled by the Java compiler into bytecode. The Java virtual machine interprets this bytecode and executes the Java program.

Java Runtime Environment (JRE)

The Java runtime environment (JRE) consists of the JVM, Java platform core classes, set of supporting Java platform libraries and contains other files that JVM uses at runtime to start Java programs. The JRE is the runtime portion of Java software, which is all you need to run it in your Web browser. The runtime system includes code necessary to run Java programs, dynamically link native methods, manage memory, and handle exceptions along with JVM implementation.

Java Development Kit (JDK)

A Java software development kit (SDK or "devkit") is a set of development tools that is used in developing applications for certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar platform. If J2SE 1.2 is used to develop applications, then you are using the platform what's known as the Java 2 Platform. The latest Java from Oracle has several enhancements to the Java language.

The Java Development Kit (JDK) is a product of Sun Microsystems used to develop Java software and is an extended subset of a Java software development kit (SDK).

JDK consists of the Application Program Interface (API) classes, a Java compiler, the Java Virtual Machine interpreter, an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development. In short, JDK contains JRE and development tools. JDK is used to compile Java applications and applets. JDK includes tools for developing Java applets and applications.

There are several programming tools and packages in JDK which includes the Java programming language core functionality, the Java Application Programming Interface **(API)** with multiple package sets, and essential tools for developing Java programs.

The packages available in JDK 11 to name the few are: java.applet, java.awt, java.beans, java.io, java.lang, java.math, java.net, java.nio, java.rmi, java.security, java.sql, java.text, java.time, java.util, javax.accessibility, javax.activation, javax.activity, javax.annotation, javax.crypto, javax.imageio, javax.jws, javax.management, javax.naming, javax.net,, javax.rmi, javax.script, javax.security.auth, javax.sound.midi, javax.sql, javax.swing, javax.tools, javax.transaction, javax.xml, org.ietf.jgss, org.omg.CORBA, org.w3c.dom, org.xml.sax etc.

These packages provide everything that needs to start creating powerful Java applications. There are many programming tools available in current version of JDK. These are used to create Java bytecode, view programs, debug code, and provide security features and other utilities. A few of the most common tools available in JDK are given in table 1.

Table 1: Tools in the JDK

Executable Java Tool	Description
appletviewer	Used to view applets without a Web browser (to run appletsoutside of a web browser)
Java	This runs Java bytecode to launch Java command application.

Javac	To compile Java source files (Java programs) to binary classfiles (bytecode)
Javadoc	This command is used to generate API documentation out ofJava
Javah	Creates C-language header and stub files from a Java class,which allows the Java and C code to interact
Javap	Disassembles Java files and prints out a representation of Javabytecode
Jcmd	It is used to send diagnostic command request to the Java JVM. When we run jcmd without arguments it lists the JVM processes that are running at the moment.
Jdb	Helps to find and fix problems in Java class.
Jhat	It is a Java heap analysis tool. It is used to parse and browse a heap dump file. This command parses the heap dump and launches a webserver. Then we can browse the dump in a browser. jhat supports object query language (oql) and also some pre-designed queries.
Jmc	To monitor, analyze and debug Java applications.
Jvisualvm	To monitor, analyze and debug running Java application via aGUI in visual manner.
Jar	To aggregate and compress multiple files into a singe JAR file.
Jarsigner	To digitally sign a jar file and also to verify the signatures and integrity of signed jar files.
Extcheck	To detect version conflict between a target jar file and installed extension jar files.
javafxpackager	To package JavaFx applications for deployment.
java-rmi	To generate stubs, skeletons and other RMI related tasks.
Javaw	To run a GUI based Java application in its own window in MS Windows.
Javaws	To launch a Java application that is distributed through web.
Jconsold	GUI tool to monitor a Java application running in a JVM.
Jdeps	To analyze Java class dependencies.

Jinfo	To print configuration information for a given process or core file or remote debug server.
Jmap	To print shared object memory maps or heap memory details of a Java application process.
Jps	To list instrumented HotSpot Java VMs on a target system.
Jrunscript	To run scipting language files like JavaScript.
Jstak	To print stack trace of threads for a given Java process.
Jstatd	To launch a RMI server to monitor creation and termination of instrumented HotSpot Java virtual machines.
Keytool	To create, manage, store keys and security certificates.
Ktab	To manage the entries in the security key table.
native2ascii	To convert files encoded in any character encoding supported by JRE to files encoded in ASCII.
Orbd	To locate and invoke persistent objects on servers in CORBA environments.
Pack200	To transform a Java JAR file into a compressed pack200 file using the Java gzip compressor.
Policytool	To create and modify the external policy configuration files that define the system Java security policy.
Rmiregistry	To start a remote object registry.
Servertool	To enable programmers to register, unregister, startup and shutdown a persistent server.
Tnameserv	To provide access to Java naming service.
unpack200	To transform back a pack200 file to a Java JAR file.

Self-Assessment Questions - 3

10. JVM is platform dependent because configuration of each OS differs. (True/False)

11. The_____is the runtime portion of Java software, which is all you need to run it in your Web browser.

12. Which of the following is used to develop Java software and is an extended subset of a Java software development kit (SDK).

a) Java Development Kit (JDK)
b) JRE
c) JVM
d) Applet

13. What is the Java command to disassemble Java files and print representation of Java bytecode?

a) javap
b) javac
c) javadoc
d) jar

5. SECURITY IN JAVA

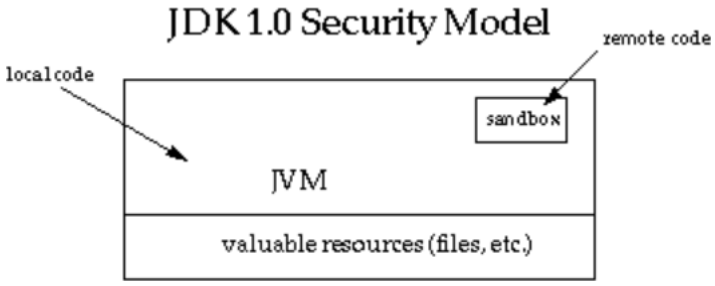
Java was the first programming language that could be used to send interactive programs over the World Wide Web. Since these programs run on the user's system, Java requires highly restrictive security to prevent a malicious programmer from using the language to wreak havoc on the systems of anyone who runs programs from a web page.

Java provides various techniques to make the program secure. Java software programs can run safely, without potential risk to systems or users. Java provides a large set of APIs (Application Programming Interfaces) and tools. It helps in implementing various security algorithms, mechanisms, and protocols. The Java security APIs are available for wide range of areas, including cryptography, public key infrastructure, secure communication, authentication, and access control. Java security technology provides the developer with a comprehensive security framework for writing applications and provides the user or administrator with a set of tools to securely manage applications.

Now let us discuss each security feature in more detail:

- **Platform security:** To provide platform security, Java has built-in language security features enforced by the Java compiler and virtual machine. Java supports strong data typing, automatic memory management, Bytecode verification and secure class loading. Compile- time data type checking and automatic memory management lead to more robust code and reduce memory corruption and vulnerabilities. Bytecode verification ensures code conforms to the JVM specification and prevents hostile code from corrupting the runtime environment. Class loaders ensure that untrusted code cannot interfere with the running of other Java programs.
- **Java Sandbox Model and Java Security Model:**

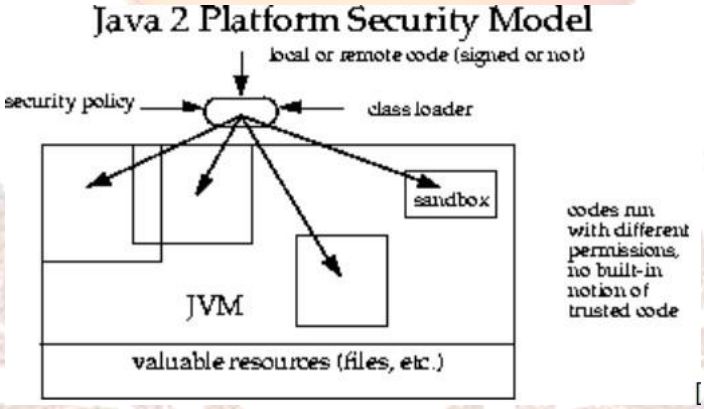
  Since networking was one of the most important use-cases for Java, there was a need to execute remote code. The remote code could be generated by anybody and anywhere. Therefore, there needed to be strict control on the access to the resources for the remote code.



Ref:https://docs.oracle.com/javase/7/docs/technotes/guides/security/spec/securityspec.doc1.html

The local code was assumed to be 'trusted' and the external code was assumed to be 'untrusted'. The external code was executed within a 'sandbox' and had very limited access to the system resources. This was referred to as 'sandbox' model.

This model evolved to 'Java Security Model'.



Ref:https://docs.oracle.com/javase/7/docs/technotes/guides/security/spec/security-spec.doc1.html

This model provided

- Ability to configure the policy instead of having to code them.
- Fine-grained permissions could be granted or revoked
- The local and the remote could need to be signed. A 'digital signature' is an encrypted file or files that accompany a program, indicating exactly from whom the file came. The document that represents this digital signature is called a 'Certificate'. An applet provider must verify its identity using a group called a 'Certificate Authority'. Policies could be set to trust or check the local and remote code.

This security model was mainly used to control the remote code e.g. applets. Since applets are no longer popular, this functionality has been disabled by default in Java.

- **Cryptography:** Java supports cryptography through the wide range of cryptographic services including digital signatures, message digests, ciphers (symmetric, asymmetric, stream & block), message authentication codes, key generators and key factories. Java also provides support for a wide range of standard algorithms (For example, RSA, DSA, AES, Triple DES, SHA etc.). Hence Java is capable of providing an extensible, full featured API for building secure applications. The APIs for the same are provided under the framework – Java Cryptographic Architecture (JCA).
- **Authentication and access Control:** Java security features provide abstract authentication APIs that can incorporate a wide range of login mechanisms through a pluggable architecture. Developer can use a comprehensive policy and permissions API to create and administer applications requiring fine-grained access to security-sensitive resources. This security features (from Java 5 version onward) enables single sign- on of multiple authentication mechanisms and fine-grained access to resources based on the identity of the user or code signer. Features such as timestamped signatures enables long term use of signed code by avoiding the need to re-sign code when the signer's certificate expires. The APIs for authentication and authorization are provided under the framework – Java Authentication and Authorization Service (JAAS).
- **Secure communications**: Java security features allows implementations of standards-based secure communications protocols such as Transport Layer Security (TLS), Secure Sockets Layer (SSL), Kerberos (accessible through GSS-API), and the Simple Authentication and Security Layer (SASL). Features also include full support for HTTPS over SSL/TLS. This protocol authenticates peers over an untrusted network and protects the integrity and privacy of data transmitted between them. These are available under the framework – Java Secure Socket Extension (JSSE).
- **Public Key Infrastructure (PKI):** Java security features also include a tool for managing keys and certificates and comprehensive, abstract APIs. These features support Certificates and Certificate Revocation Lists (CRLs for example- X.509),

Certification Path Validators and Builders (PKIX for example, RFC 3280), On-line Certificate Status Protocol (OCSP), KeyStores (example - PKCS#11, PKCS#12), Certificate Stores (Repositories for example, LDAP, java.util.Collection). These features eases the development and deployment of complex PKI applications. Java supports OCSP for a more scalable and timely method for applications to check certificate revocation status.

There are three tools used commonly for security purpose in the JDK. They are **Keytool, Jar** and **Jarsigner.**

Keytool

Keytool, the Java security key tool, is used to create and manage public keys, private keys and security certificates. It can be used to do the following:

- Manages the own public key / private key pairs.
- Stores the public keys of people and groups which they communicate with.
- Uses the certificates associated with these keys to authenticate the user to others.
- Authenticate the source and integrity of data.

The jar archival tool and jarsigner

The Java archival tool **jar** is used to package into a single archive file a Java program and all resource files that it requires.

The advantage of **jar** tool is to speed up the loading time for an applet on the web, especially when the applet needs a group of other files in order to function. Jar is also necessary to create a digitally signed Java program. All class files which are needed to run a particular Java program need to be packaged into an archive jar file before being signed. When a jar file is included in the program it should be signed using a **jarsigner**. The following two activities should be performed:

- Digitally signing a java archive file
- Verifying the digital signature and contents of a java archive

To digitally sign a java archive file, following have to be specified i.e. a name of the jar archive and the name of the private key to sign it with. The ***jarsigner*** tool has several

command-line options that are identical to those offered by keytool, including ***–keystore***, which is used to specify the folder and the file location of a key store.

The jar tool also can be used to verify a digitally signed archive by adding the ***–verify*** option to the command. If the private key matches the Java archive and the archive's contents did not change after it was signed, it will be verified by the jarsigner.

Self-Assessment Questions – 4

14. _____is used to create and manage public keys, private keys and security certificates.
15. All information which are managed by keytool is stored in a database called _____.
16. Sun Microsystems includes a default key store that uses a new file format called _____.
17. Java archival tool used for packaging all java programs and resource files into a single archive file is _____.
18. _____was the first programming language that could be used to send interactive programs over the World Wide Web.
19. A_____is an encrypted file or files that accompany a program, indicating exactly from whom the file came.
20. The document that represents this digital signature is called a _____.

6. SUMMARY

Let us summarize the important points covered in the unit:

- Java is a programming language developed by Sun Microsystems (nowadays a subsidiary of Oracle Corporation).
- Java is :
  - ✓ Simple – It is easy to learn Java.
  - ✓ Object-oriented – Everything in Java is in form of classes and objects.
  - ✓ Distributed – Java programs can access data across a network.
  - ✓ Compiled and Interpreted – The Java code you write is compiled to bytecode and interpreted when you execute the program.
  - ✓ Robust – Java programs are less prone to error.
  - ✓ Architecture Neutral and Portable – The bytecode can be executed on a variety of computers running on different operating system.
  - ✓ Secure – Java does not allow a programmer to manipulate the memory of the system.
  - ✓ A high performance programming language – Java programs are faster when compared to programs written in other interpreter-based languages.
  - ✓ Multithreaded – It allows multiple parts of a program to run simultaneously.
  - ✓ Dynamic – Maintaining different versions of an application is very easy in Java.
- JDK is a product of Sun Microsystems (nowadays a subsidiary of Oracle Corporation) aimed for developing Java software. Java SDK is the set of development tools that is used in developing software applications.
- Java requires security for the system that runs program from the web. And there are security tools available that can prevent the malicious program which can wreak havoc in a system.

7. TERMINAL QUESTIONS

1. Explain any five features of Java.
2. What is Bytecode? Explain in brief.
3. Write short notes on: JDK and SDK.
4. List any ten Java command tools with a short description of each.
5. Explain jar archival tool and jarsigner in brief.

8. ANSWERS

Self Assessment Questions

1. Oak
2. Patrick Naughton, Mike Sheridan, James Gosling
3. Mosaic
4. Compilation
5. Portable
6. Java Virtual Machine (JVM)
7. Multithreading
8. Compiled and Interpreted
9. True
10. True
11. Java Runtime Environment (JRE)
12. (a) Java Development Kit (JDK)
13. (a) javap
14. Keytool
15. Keystore
16. JKS (Java Key Store)
17. Jar
18. Java
19. Digital signature
20. Certificate
21. Certificate Authority

Terminal Questions

1. Features of Java: It is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multi- threaded and dynamic language. (Refer section 3 for more details)

2. Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). (Refer section 3)

3. JDK: JDK is an extended subset of SDK. JDK contains JRE and development tools. SDK: SDK is the set of development tools that is used in developing software applications. (Refer section 4)

4. Commonly used ten Java command tools are: appletviewer, java, javac, javadoc, javah, javap, jcmd, jdb, jhat, jmc, jvisualvm, jar (Refer to section 4 for more details)

5. jar archive tool: The Java archival tool jar is used to package into a single archive file a Java program and all resource files that it requires. jarsigner: All class files which are needed to run a particular Java program need to be packaged into an archive jar file before being signed. When a jar file is included in the program it should be signed using a jarsigner. (Refer to section 5 for more details)