Java
Unit-1

1	describe JVM
2	state importance of bytecode
3	Explain the features of Java Programming Language.
4	Discuss the various components of Java Environment Setup.
5	Explain the Java Program Structure with example.
6	Explain the difference between Procedure-Oriented Programming (POP) and Object-Oriented Programming (OOP)
7	Write any two difference between Java and C.
8	Write a Java program to reverse the digits of a given number.
9	Write a Java program to find factorial of a given number.
10	Write syntax to compile and execute Java program
11	Explain main method of Java program
12	Describe Applets.
13	Write only compile command and run command of test.java file.
14	Define byte code and source code.
15	Write a Java program that demonstrates the use of static method.
16	How Java is strongly associated with the Internet?
17	Define 'static' keyword in Java.
18	How Java is secure & portable than other language.
19	What is the task of main method in Java program?
20	Write a program in Java to generate first 'n' prime numbers.
21	List four different OOP concepts.
22	Write any four differences between Java and C++.
23	List types of array and Explain variable type (Non Rectangular type) array with example.
24	Write a Java program to print first five terms of Fibonacci series.
25	Explain the following keyword of Java: public static void main (String args[]).
26	_____ keyword is used to make a class
27	_____ operator is used to allocate memory for an object
28	
29	Explain following
30	(i) Byte Code (ii) JVM (iii) Platform Independent
31	
32	Explain following
33	(i) Portable (ii) JRE (iii) Applet
34	Justify Java is strongly typed language
35	Explain following JDK tools: Compiler and interpreter
36	Difference between JRE and JVM

Unit 7

Q1 Describe JVM

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.
JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).
What is JVM
It is:

1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Oracle and other companies.
2. **An implementation** Its implementation is known as JRE (Java Runtime Environment).
3. **Runtime Instance** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

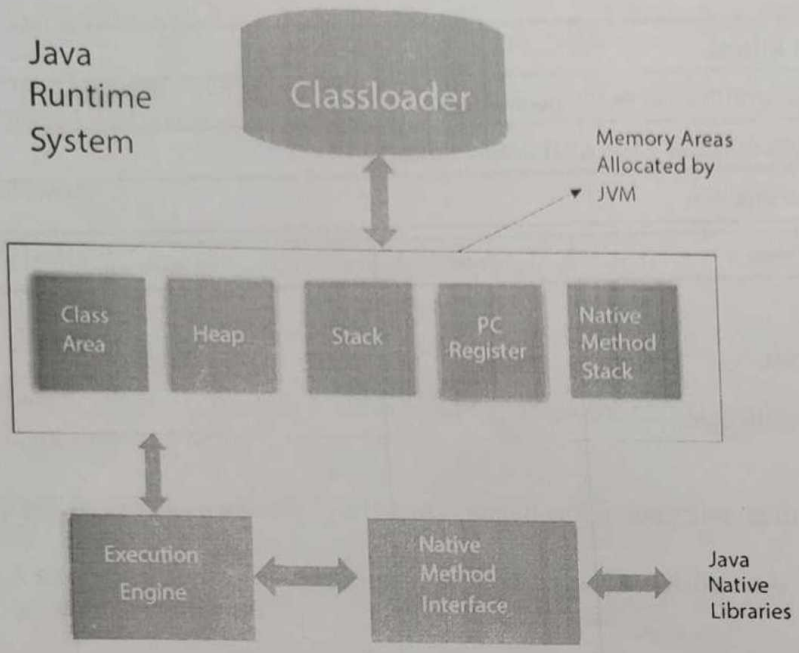
What it does?

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap
- Fatal error reporting etc.



1) Classloader

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

1. **Bootstrap ClassLoader**: This is the first classloader which is the super class of Extension classloader. It loads the *rt.jar* file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes etc.
2. **Extension ClassLoader**: This is the child classloader of Bootstrap and parent classloader of System classloader. It loades the jar files located inside *$JAVA_HOME/jre/lib/ext* directory.
3. **System/Application ClassLoader**: This is the child classloader of Extension classloader. It loads the classfiles from classpath. By default, classpath is set to current directory. You can change the classpath using "-cp" or "-classpath" switch. It is also known as Application classloader.

2) Class(Method) Area

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap

It is the runtime data area in which objects are allocated.

4) Stack

Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register

PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack

It contains all the native methods used in the application.

7) Execution Engine

It contains:

1. **A virtual processor**
2. **Interpreter:** Read bytecode stream then execute the instructions.
3. **Just-In-Time(JIT) compiler:** It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here, the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

8) Java Native Interface

Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc. Java uses JNI framework to send output to the Console or interact with OS libraries.

Q2) Explain Java is platform independent

Java is called platform independent because of Java Virtual Machine. As different computers with the different operating system have their JVM, when we submit a *.class* file to any operating system, JVM interprets the bytecode into machine level language.

Q3) Explain difference between JDK,JRE and JVM

JDK

JDK is abbreviated as Java Development Kit which has a physical existence. It can be considered as a kit inside which resides the JRE along with developing tools within it. The programmers and developers mostly use it.

JVM

JVM is abbreviated as Java Virtual Machine, is basically a dummy machine or you can say an abstract machine which gives Java programmers a runtime environment for executing the Bytecode. For each execution of your program, the JDK and JRE come into use, and they go within the JVM to run the Java source code.

JRE

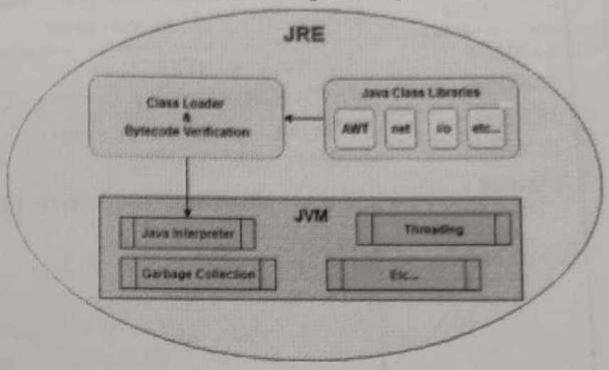
JRE is abbreviated as Java Runtime Environment, as the name suggests used as a package that gives an environment to run the Java program on your machine.

Q4) Explain JRE

JRE stands for Java Runtime Environment which is used to provide an environment at runtime. It is the cause of implementation of JVM (as discussed earlier). It contains a set of supporting libraries in combination with core classes and various other files that are used by JVM at runtime. JRE is a part of JDK (Java Development Toolkit) but can be downloaded separately.

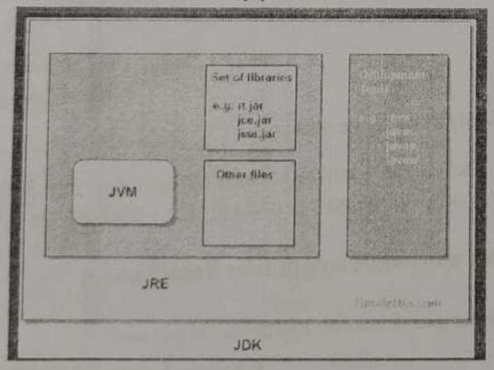
You need JRE to execute your program, which includes two things:

- JVM
- Java Library
  - Static - Functions that are required at compile time.
  - Dynamic - Functions that are required at runtime and not at compile time.



Q5) Explain JDK

JDK (Java SE Development Kit) includes a complete JRE (Java Runtime Environment) plus tools for developing, debugging, and monitoring Java applications. JDK is required to build and run Java applications and applets.



Q6) State importance of byte code

Byte code is a highly optimized set of instructions that is executed by the Java Virtual Machine. Byte code helps Java achieve both portability and security.

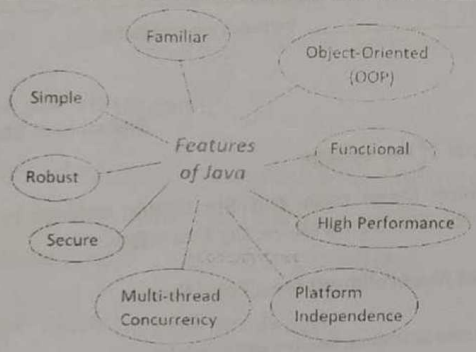
Disadvantages:

- requires both compiler and interpreter
- slower program execution

Advantages:

- portability
  - » very important
  - » same program can run on computers of different types (useful with the Internet)
  - » Java compiler for new types of computers can be made quickly

Q7) Explain the features of Java Programming Language.



1. Java is Simple:

The Java programming language is easy to learn. Java code is easy to read and write.

2. Java is Familiar:

Java is similar to C/C++ but it removes the drawbacks and complexities of C/C++ like pointers and multiple inheritances. So if you have background in C/C++, you will find Java familiar and easy to learn.

3. Java is an Object-Oriented programming language:

Unlike C++ which is semi object-oriented, Java is a fully object-oriented programming language. It has all OOP features such as abstraction, encapsulation, inheritance and polymorphism.

4. Java supports Functional programming:

Since Java SE version 8 (JDK 8), Java is updated with functional programming feature like functional interfaces and Lambda Expressions. This increases the flexibility of Java.

5. Java is Robust:

With automatic garbage collection and simple memory management model (no pointers like C/C++), plus language features like generics, try-with-resources,... Java guides programmer toward reliable programming habits for creating highly reliable applications.

6. Java is Secure:

The Java platform is designed with security features built into the language and runtime system such as static type-checking at compile time and runtime checking (security manager), which let you creating applications that can't be changed from outside. You never hear about viruses attacking Java applications.

7. Java is High Performance:

Java code is compiled into bytecode which is highly optimized by the Java compiler, so that the Java virtual machine (JVM) can execute Java applications at full speed. In addition, compute-intensive code can be re-written in native code and interfaced with Java platform via *Java Native Interface* (JNI) thus improve the performance.

8. Java is Multithreaded:

The Java platform is designed with multithreading capabilities built into the language. That means you can build applications with many concurrent threads of activity, resulting in highly interactive and responsive applications.

9. Java is Platform Independence:

Java code is compiled into intermediate format (bytecode), which can be executed on any systems for which Java virtual machine is ported. That means you can write a Java program once and run it on Windows, Mac, Linux or Solaris without re-compiling. Thus the slogan "*Write once, run anywhere*" of Java.

10 Dynamic: Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

Java supports dynamic compilation and automatic memory management (garbage collection).

Q8 Difference between C++ and Java

Platform-independent	C++ is platform-dependent.	Java is platform-independ ent.
Multiple inheritance	C++ supports multiple inheritance	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
Compiler and Interpreter	C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent.	Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent.
Call by Value and Call by reference	C++ supports structures and unions.	Java doesn't support structures and unions.
Object-oriented	C++ is an object-oriented language. However, in C language, single root hierarchy is not possible.	Java is also an object-oriented language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object.

Program structure	`#include <iostream>`
	`using namespace std;`
	
	`int main() {`
	`cout << "Hello C++ Programming";`
	`return 0;`
	`}`
		`class Simple{`
		`public static void main(String args[]){`
		`System.out.println( "Hello Java");`
		`}`
		`}`

Q9) Explain simple jva program

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

- **class** keyword is used to declare a class in java.
- **public** keyword is an access modifier which represents visibility. It means it is visible to all.
- **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create an object to invoke the main method. So it saves memory.
- **void** is the return type of the method. It means it doesn't return any value.
- **main** represents the starting point of the program.
- **String[] args** is used for command line argument. We will learn it later.
- **System.out.println()** is used to print statement. Here, System is a class, out is the object of PrintStream class, println() is the method of PrintStream class. We will learn about the internal working of System.out.println statement later.

Q10) Explain the difference between Procedure-Oriented Programming (POP) and Object-Oriented Programming (OOP)

	Procedure Oriented Programming	Object Oriented Programming
Divided Into	In POP, program is divided into small parts called **functions**.	In OOP, program is divided into parts called **objects**.
Importance	In POP, Importance is not given to **data** but to functions as well as **sequence** of actions to be done.	In OOP, Importance is given to the data rather than procedures or functions because it works as a **real world**.
Approach	POP follows **Top Down approach**.	OOP follows **Bottom Up approach**.
Access Specifiers	POP does not have any access specifier.	OOP has access specifiers named Public, Private, Protected, etc.
Data Moving	In POP, Data can move freely from function to function in the system.	In OOP, objects can move and communicate with each other through member functions.
Expansion	To add new data and function in POP is not so easy.	OOP provides an easy way to add new data and function.
Data Access	In POP, Most function uses Global data for sharing that can be accessed freely from function to function in the system.	In OOP, data can not move easily from function to function,it can be kept public or private so we can control the access of data.
Data Hiding	POP does not have any proper way for hiding data so it is **less secure**.	OOP provides Data Hiding so provides **more security**.
Overloading	In POP, Overloading is not possible.	In OOP, overloading is possible in the form of Function Overloading and Operator Overloading.
Examples	Example of POP are : C, VB, FORTRAN, Pascal.	Example of OOP are : C++, JAVA, VB.NET, C#.NET.

Q11) Write a Java program to reverse the digits of a given number.

```
public class ReverseNumber {
public static void main(String[] args) {
int num = 1234, reversed = 0;
while(num != 0) {
int digit = num % 10;
reversed = reversed * 10 + digit;
num /= 10;
}
```

Q12) Write a Java program to find factorial of a given number.

```
class FactorialExample{
 public static void main(String args[]){
  int i,fact=1;
  int number=5;//It is the number to calculate factorial
  for(i=1;i<=number;i++){
     fact=fact*i;
  }
  System.out.println("Factorial of "+number+" is: "+fact);
 }
}
```

Q13) Write syntax to compile and execute Java program

Compile: Javac Name of program.java

Run: java name of program

Q14) Write only compile command and run command of test.java file.

Compile: Javac test.java

Run: java test

Q15) Explain main method of Java program

Public static void main(String args[])

{

}

- **public** keyword is an access modifier which represents visibility. It means it is visible to all.
- **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create an object to invoke the main method. So it saves memory.
- **void** is the return type of the method. It means it doesn't return any value.
- **main** represents the starting point of the program.
- **String[] args** is used for command line argument. We will learn it later.
- **System.out.println()** is used to print statement. Here, System is a class, out is the object of PrintStream class, println () is the method of PrintStream class. We will learn about the internal working of System.out.println statement later.

Q16) Explain following JDK tools: Compiler and interpreter

1) **Java Compiler:** Java compiler is **javac** tool located in */bin* folder of the JDK installation directory. The **javac** tool (accessed using javac command) reads class and interface definitions, written in the Java programming language, and compiles them into bytecode class files. It can also process annotations in Java source files and classes.

```
Syntax: D:\JavaProgram>javac hello.java
```

2) **Java Interpreter:** Java interpreter is used to interpret the .class Java files that have been compiled by Java compiler (javac). Java interpreter is accessed using java command. The java command starts a Java application. It does this by starting a Java runtime environment, loading a specified class, and calling that class's main method.The method must be declared public and static, it must not return any value, and it must accept a String array as a parameter. The method declaration has the following form:

```
public static void main(String[] args)
```

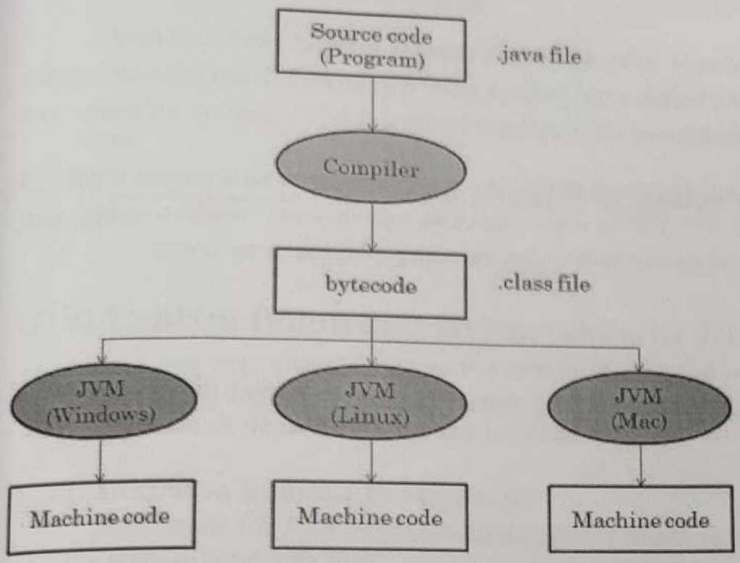
Q17) Explain following: (i) Byte Code (ii) JVM (iii) Platform Independent

1) **Bytecode:**

Java bytecode is the instruction set for the Java Virtual Machine. As soon as a java program is compiled, java bytecode is generated. java bytecode is the machine code in the form of a .class file. With the help of java bytecode we achieve platform independence in java.

How does it works?

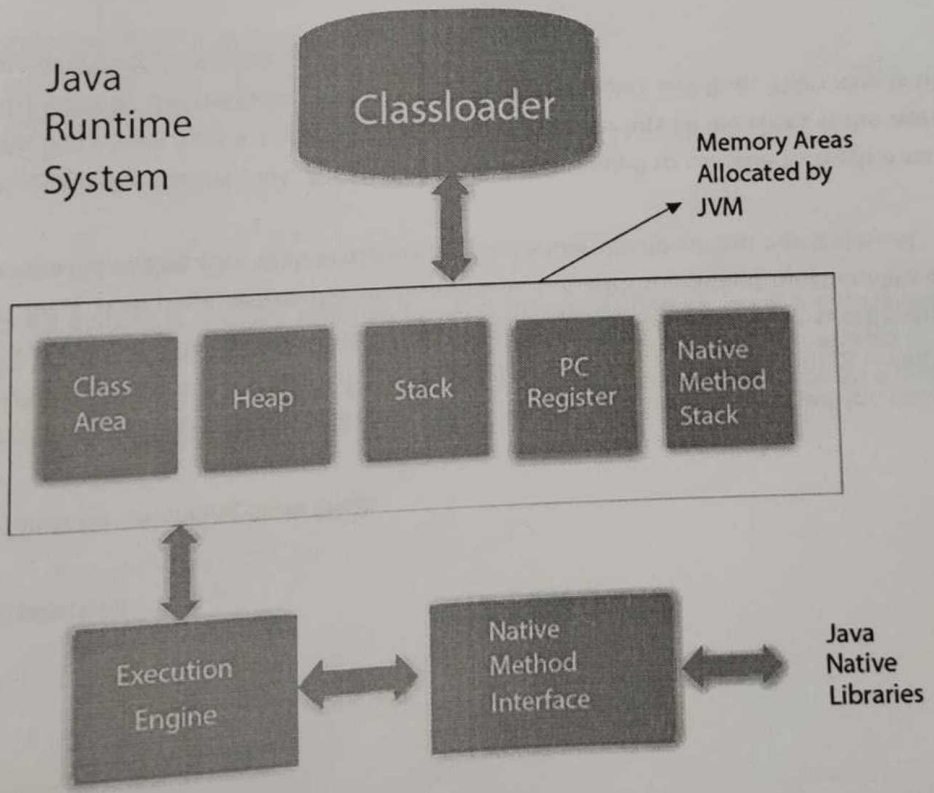
When we write a program in Java, firstly, the compiler compiles that program and a bytecode is generated for that piece of code. When we wish to run this .class file on any other platform, we can do so. After the first compilation, the bytecode generated is now run by the Java Virtual Machine and not the processor in consideration. This essentially means that we only need to have basic java installation on any platforms that we want to run our code on. Resources required to run the bytecode are made available by theJava Virtual Machine, which calls the processor to allocate the required resources. JVM's are stack-based so they stack implementation to read the codes.



2) JVM

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).



3) **Platform independent**

Java is called platform independent because of Java Virtual Machine. As different computers with the different operating system have their JVM, when we submit a *.class* file to any operating system, JVM interprets the bytecode into machine level language.

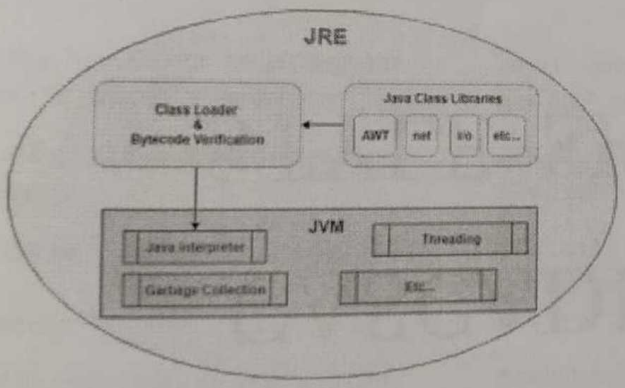
Q18) _____ keyword is used to make a class

Ans: class

Q19) Explain following: (i) Portable (ii) JRE (iii) Applet

1) **Portable:** Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

2) **JRE: Java Runtime Environment**
JRE stands for Java Runtime Environment which is used to provide an environment at runtime. It is the cause of implementation of JVM (as discussed earlier). It contains a set of supporting libraries in combination with core classes and various other files that are used by JVM at runtime. JRE is a part of JDK (Java Development Toolkit) but can be downloaded separately.



3) **Applet:** An **applet** is a small Internet-based program written in **Java**, a programming language for the Web, which can be downloaded by any computer. The **applet** is also able to run in HTML. The **applet** is usually embedded in an HTML page on a Web site and can be executed from within a browser.

Q20) Difference between JRE and JVM

JVM	JRE
JVM (Java Virtual Machine) is an abstract machine.	JRE is an acronym for Java Runt Environment. It is used to provide runt environment.
It is a virtual specification of runtime environment.	It is the implementation of JVM
It is virtual that is why it exists only inside the computer memory.	It physically exists. It contains set of lib + other files that JVM uses at runt

Q21) Difference between JRE and JDK

JRE and JDK

JRE (Java Runtime environment)	JDK (Java Development Toolkit)
It is an implementation of the Java Virtual Machine* which actually executes Java programs.	It is a bundle of software that you can use to develop Java based applications.
Java Run Time Environment is a plug-in needed for running java programs.	Java Development Kit is needed for developing java applications.
JRE is smaller than JDK so it needs less Disk space.	JDK needs more Disk space as it contains JRE along with various development tools.
JRE can be downloaded/supported freely from java.com	JDK can be downloaded/supported freely from java.sun.com
It includes JVM , Core libraries and other additional components to run applications and applets written in Java.	It includes JRE, set of API classes, Java compiler, Webstart and additional files needed to write Java applets and applications.

Q22) Justify Java is strongly typed language

A **strongly-typed** programming **language** is one in which each type of data (such as integer, character, etc.) is predefined as part of the programming **language** and all constants or variables defined for a given program must be described with one of the data types.

In Java, when a variable is declared, it must be informed to the compiler what data type the variable stores like integer, float, double or string etc.

For Example, consider the simple code snippet :

```
int age = 20;
String name = "Tridib";
boolean ans = true;
```

In this snippet, each variable is declared with the data type. Similarly, an array object must be instantiated with the size of the array.

Q23) Explain the following keyword of Java: public static void main (String args[]).

- **public** keyword is an access modifier which represents visibility. It means it is visible to all.
- **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create an object to invoke the main method. So it saves memory.
- **void** is the return type of the method. It means it doesn't return any value.
- **main** represents the starting point of the program.
- **String[] args** is used for command line argument. We will learn it later.
- **System.out.println()** is used to print statement. Here, System is a class, out is the object of PrintStream class, println () is the method of PrintStream class. We will learn about the internal working of System.out.println statement later.

Q24) Write a Java program to print first five terms of Fibonacci series.

```
1.  public class Fibonacci {
2.
3.    public static void main(String[] args) {
4.
5.      int n = 5, t1 = 0, t2 = 1;
6.      System.out.print("First " + n + " terms: ");
7.
8.      for (int i = 1; i <= n; ++i)
9.      {
10.       System.out.print(t1 + " + ");
11.
12.       int sum = t1 + t2;
13.       t1 = t2;
14.       t2 = sum;
15.     }
16.   }
17. }
```

Q25) List four different OOP concepts.

- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Inheritance

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Polymorphism

If *one task is performed in different ways*, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

Abstraction

Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

In Java, we use abstract class and interface to achieve abstraction.

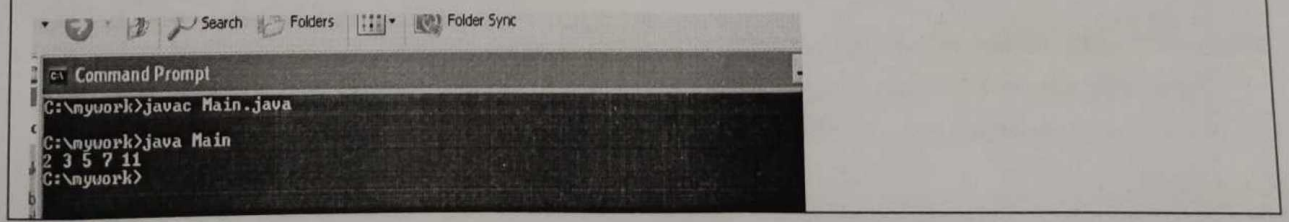
Encapsulation

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Q26) Write a program in Java to generate first 'n' prime numbers.

```
public class Main {
public static void main(String[] args)
 {
int count = 0;
int number = 2;
while(count < 5)
{
 int c=0;
for(int i = 2; i < number ; i++)
{
if (number%i == 0)
{
c=1;
break;
}
}
if(c==0)
{
System.out.print(number + " ");
count++;
}
number++;
}//end of while
} //end of main
} //end of class
```



Q27) What is the task of main method in Java program?

The **main** method is the entry point of your Java App. Whenever you execute a program, the main is the first function to be executed. You can call other functions from main to execute them. In a standard app, there is one main function which uses other classes' instances to function.

The standard declarations possible are:

```
1.  public static void main(String... args){}
2.  public static void main(String[] args){}
3.  public static void main(String args[]){}
```

Q28) How Java is secure & portable than other language.

The Java platform is designed with security features built into the language and runtime system such as static type-checking at compile time and runtime checking (security manager), which let you creating applications that can't be changed from outside. You never hear about viruses attacking Java applications.

Following features of Java makes it more secure than other languages

- Bytecode
- Encapsulation,
- Exception handeling
- No use of pointer
- Garbage collection

Q29) Write a Java program that demonstrates the use of static method.

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

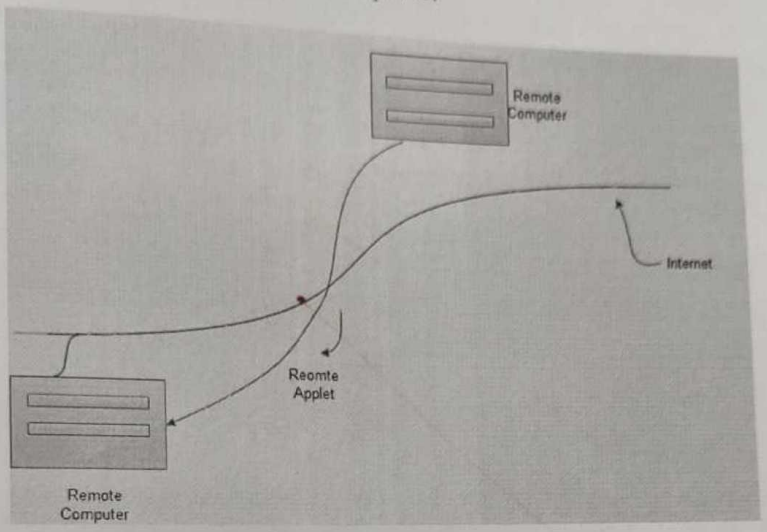
Output

Q30) Define 'static' keyword in Java.

static is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create an object to invoke the main method. So it saves memory

Q31) How Java is strongly associated with the Internet?

Java is strongly associated with the Internet because the first application program written in Java was HotJava,a browser to run the applet on Internet .So the Internet users use the Java to create the applet programs and run them locally using a 'Java-enabled browsers' like HotJava..The users can use the 'Java-enabled browsers to download the applet located on the computer system anywhere in the internet ans run it on their computer.



So the Internet users start setting up the their web sites containing Java applets that could be used by the other remote internet users .So this ability make the Java programming language popular among the Internet programmers ,so the Java is popularly known as 'Internet Language'.

Q32) Define byte code and source code.

Source code:

Source code is the list of human-readable instructions that a programmer writes—often in a word processing program—when he is developing a program. The source code is run through a compiler to turn it into machine code, also called object code, that a computer can understand and execute. Object code consists primarily of 1s and 0s, so it isn't human-readable.

Byte code:

Java bytecode is the instruction set for the Java Virtual Machine. As soon as a java program is compiled, java bytecode is generated. java bytecode is the machine code in the form of a .class file. With the help of java bytecode we achieve platform independence in java.