PAGE :
DATE : / /

Chapter=1
Elements of Java Programming

1) Define java. List the advantages of java programming language.

Ans: Java is a high-level programming language developed by Sun Microsystems and released in 1995. It is strictly an object-oriented programming language. Java programs are interpreted by the Java Virtual Machine (JVM) which runs on multiple platforms. This means all Java programs are multiplatform and can run on different platforms, including Macintosh, Windows, and Unix computers.

The advantages of java programming language are explained below:-

i) Java is easy to learn
= Java was designed to be easy to use. Therefore it is easy to write, compile, debug, and learn than other programming languages.

ii) Java is Object-Oriented
= In Java, everything is an object. It allows you to create standard programs and reusable code.

PAGE :
DATE : / /

iii) Java is platform-independent

= One of the most significant advantage of Java is its ability to move easily from one computer system to another. Java code runs on any computer that doesn't need any special software to be installed, but the JVM needs to be present on the computer.

iv) Java is secure

= Java considers security as part of its design. The Java language, compiler, interpreter, and runtime environment were each developed with security in mind. It has no explicit pointer. Apart from this, it has a security manager that defines the access of classes.

v) Java is robust

= Robust means reliability. Java makes an effort to eliminate error prone situation by emphasizing mainly on compile time error checking and runtime checking.

vi) Java is multithreaded

= With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

PAGE :
DATE : / /

12) What are the most powerful features of java? Why java is popular than C/C++.

Ans: The most powerful features of java are explained below:-

(i) Object Oriented

= In Java, everything is an object. It allows you to create standard programs and reusable code.

(ii) Platform Independent

= Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rater into platform independent byte code. This byte code is distributed over the web and interpreted by the Java Virtual Machine (JVM) on whichever platform it is being run on.

(iii) Secure

= With Java's secure feature it enables to develop virus-free, tam per-free systems. Authentication techniques are based on public-key encryption.

(iv) Architecture-neutral

= Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

PAGE :
DATE : / /

vi) Portable
= Being architecture-neutral and having no implementation dependent aspects of the specification makes java portable. Compiler in Java is written in ANSI C with a clean portability boundary which is a POSIX subset.

vii) High-Performance
= With the use of Just-In-Time compilers, Java enables high performance.

viii) Distributed
= Java is designed for the distributed environment of the internet. Writing network program in Java is like sending and receiving data to and from a file.

Java is popular than C or C++ because unlike them, Java programs are compiled independently of the platform in bytecode language which allows the same program to run on any machine that has a JVM installed. Java doesn't use pointers that make it a type-safe programming language. The JVM helps in an efficient code optimization so the performance of execution of the program is better than as that of C or C++.

PAGE :
DATE : / /

3) Define java as Platform Independent Programming Language.

OR

Describe how java is a platform neutral language by a diagram.

Ans: Software that is designed without regard to the target platform is called architectural neutral or platform independent programming language. Unlike many other programming language including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent bytecode. This bytecode is interpreted by Java Virtual Machine (JVM) on which ever platform it is being run. A Java Virtual Machine (JVM) is a process virtual machine that can execute Java bytecode. Each Java source file is compiled into a bytecode file, which is executed by the JVM. Java was designed to allow application programs to be built that could be run on any platform, without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible, because it is aware of the specific instruction lengths and other particularities of the underlying hardware platform.

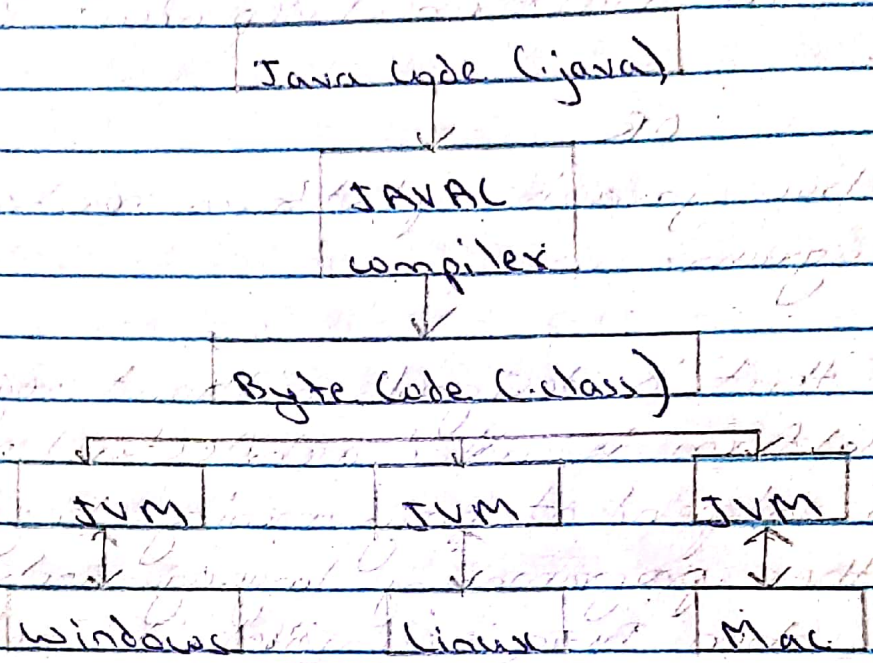


fig:- Java program execution

What are the wrapper classes. Explain.

Ans: Wrapper class in java provides the mechanism to convert primitive into object and object into primitive. A wrapper class is a class whose object wraps or contains a primitive data types. When we create an object to a wrapper class it contains a field and this field, we can store a primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

Wrapper classes helps in following ways:

i) They convert primitive data types into objects. Objects are needed if we wish to modify the arguments passed into a method (because primitive types are passed by value

PAGE :
DATE : 1. 1

iii) The classes in java.util package handles only objects and hence wrapper classes help in this case also.

iii) Data Structures in the collection framework, such as ArrayList and Vector, store only objects (reference types) and not primitive types.

iv) An object is needed to support synchronization in multithreading.

Primitive Datatypes and their corresponding Wrapper class

Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
long	Integer
float	Float
double	Double
boolean	Boolean

4) Is JVM machine independent? Explain the use of JVM. How JVM makes Java machine independent.

Ans: No, JVM is not machine independent. JVM is platform dependent while java language is platform independent. JVM is one kind of interface or middleware between OS and Java language. JVM provides the environment to execute the java file. So, at the end it depends on the kernel and kernel is different from OS to OS. So, JVM is platform dependent.

JVM is a virtual machine that enables a computer to run a Java Program as well as program written in other language. It is responsible to call the main method in Java. A JVM is able to execute the class file which may be generated in any platform like Windows, Linux, MacOS, etc. It acts as a run-time engine to run Java applications.

When a Java is compiled, it is ~~most~~ not compiled into platform specific machine, rater into platform independent bytecode. This bytecode is interpreted by JVM on whichever platform it is being run. JVM can execute bytecode. Java was designed to allow application programs to be built that could be run on any platform without having to be rewritten or recompiled by the programmer for each separate platform. A JVM makes this possible, because it is aware of the specific instruction lengths and other particularities of the underlying

hardware platform.

5) What do you mean by Java System Overview? Explain why Java program are known as portable and architecture neutral.

Ans. Java is an object oriented programming language. It is a simple, general-purpose, interpreted, robust, secure, architecture-neutral, portable, high-performance, multi-threaded computer language.

Java program are known as portable for it's compability with different machines, devices and network. Once a java program is written to run on one operating system, works on another operating system without changing anything. Java is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform doesn't need to be recompiled to run on another.

Since, it is portable, it is ~~also~~ architecture neutral. When a java program is compiled, it is compiled into platform independent bytecode. This bytecode is interpreted by Virtual Machine (JVM) on whichever platform it is being run. Application programs built on Java would be run on any platform, without having to be rewritten or recompiled by the programmer for each separate platform. Hence, Java program are known as portable and architecture neutral.

DATE: / /

6) Justify the statement, "Java is designed for distributed application." Explain the different types of Java variables.

Ans. Java is designed for the distributed environment of the internet. Java has networking facilities, so java can create application on network. Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.

In Java, we can split a program into many parts and store these parts on different computers. A Java programmer sitting on a machine can access another program running on the other machine. This feature in Java gives the advantage of distributed programming, which is very helpful when we develop large projects. Java helps us to achieve this by providing the concept of RMI (Remote Method Invocation) and EJB (Enterprise JavaBeans).

Java comes with an extensive library of classes for interacting using TCP/IP protocols such as HTTP and FTP, which makes creating network connections much easier than in C/C++.

PAGE :
DATE : / /

variable is name of reserved area allocated in memory. In other words, it is a name of memory location.

eg:-

```
int data = 50; // Here data is variable.
```

There are three types of variables in Java. They are listed below:-

i) Local variable

= A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists. A local variable cannot be defined with "static" keyword.

ii) Instance Variable

= A variable declared inside the class but outside the body of the method is called instance variable. It is not declared as static. It is called instance variable because its value is instance specific and is not sharing among instances.

iii) Static variable

= A variable which is declared as static is called static variable. It cannot be local. We can create a single copy of static variable and share among all the instances of the class. Memory allocation for static

DATE : / /

variable happens only once when the class is loaded in the memory.

7) How java is more secure than other programming language? Explain.

Ans: Security is an important aspect and Java's security model is one of the key architectural features that make it most trustful choice when it comes to developing enterprise-level applications. Security becomes critical when software is downloaded across network and executed locally, and Java easily mitigates the security vulnerabilities associated with the projects or applications.

Java is considered ~~bet~~ safe because of:

i) Java programs run inside a virtual machine (JVM). Though the java program can even then have access to your files, it is pretty much safer. They have to belong to a trusted resource with a valid signature. Even if an untrusted application has gained this level of access, we can force quit the JVM, the application dies. It works pretty much like a sandbox but it is exploitable to an extent. On the other hand, most C++ or C applications can easily continue running in the background, and even create new services to be safe from you. Obtaining the same results from a java application is tough.

PAGE:
DATE: / /

(bytecode are tested by JVM at the time of program execution for viruses and other malicious files)

ii) Everything the programmer writes in java is compiled to byte code. This byte code is not easy to exploit/modify by third-parties. However, there are decompilers present which decompile the byte-code to the java code. However, this will be again time-consuming for the third party. (~~bytecode are tested by~~

iii) Java code is verified before execution. This protects a java application from running a method which is flawed, hence usually saving the application from force exits. The variables are also null-checked to save errors in runtime.

iv) No use of pointers for memory management. Pointers can often cause data leaks to unauthorized applications. This is rare in the case of java. Automatic garbage collection also plays a role here.

v) The concept of exception handling enables Java to capture a series of errors that helps developers to get rid of risk of crashing the system.

vi) Java allows developers to declare classes or methods as FINAL. Any class or method declared as final can't be overridden, which helps developers to protect code from security attacks like creating a subclass and replacing it with the original class and override methods.

PAGE :
DATE : / /

vii) Java's access code functionality on variables and methods within the objects provide secure program by preventing access to the critical objects from the untrusted code.

Simple

(i) Java's security model is intended to help and protect users from hostile programs downloaded from some untrusted resource within a network through "sandbox". It allows all the Java programs to run inside the sandbox only and prevents many activities from untrusted resources including reading or writing to the local disk, creating any new process or even loading any new dynamic library while calling a native method.

PAGE:
DATE: / /

8) Explain with example: Static block, static variable & static method. Why main method is always static & public in Java?

Ans: They are explained below:-

i) Static block:

= The static block is a block of statement inside a Java class that will be executed when a class is first loaded. It is used for initializing the static variables. Static blocks are executed only once. A class can have multiple static blocks which will execute in the same sequence in which they have been written into the program.

```
//Java program to demonstrate the use of static block
import java.util.*;
public class BlockExample {
    //static variable
    static int j=10;
    static int n;
    //static block
    static {
        System.out.println("Static block initialized.");
        n=j*8;
    }
    public static void main(String[] args)
    {
        System.out.println("Inside main method");
        System.out.println("Value of j:"+j);
        System.out.println("Value of n:"+n);
    }
}
```

Output:

```
Static block initialized
Inside main method
Value of j: 10
Value of n: 80
```

PAGE :
DATE : / /

ii) Static variable

= When we declare a variable as static, then a single copy of the variable is created and divided among all objects at the class level. Static variables are essentially, global variables. Basically, all the instances of the class share the same static variable. Static variables can be created at class-level only.

Example:

```
import java.util.*;
public class Variable example {
    static int j=n();
static {
    System.out.println("Inside the staticblock");
    }
static int n() {
    System.out.println("from n");
    return 20;
    }
public static void main(String[] args)
    {
    System.out.println("value of j: "+j);
    System.out.println("Inside main method");
    }
    }
```

Output:

```
from n
Inside the static block
Value of j: 20
Inside main method
```

PAGE :
DATE : / /

iii) Static method

: When a method is declared with the static keyword, it is known as static method. It belongs to the class and not the object. The most common example of a static method is the main() method. Static method can access static data only. They can directly call other static methods only.

Example.

```
public class StaticMethodExample{
    static int j=100;
    int n=200;
    static void a(){
    a=200;
    System.out.println("Print from a");
    n=100;
    a2();
    System.out.println(super.j);
    }
    void a2()
    {
    System.out.println("Inside a2");
    }
    public static void main(String[] args)
    {
        // main method
    }
    }
```

→ A static method cannot refer to "this" or "super" keyword.

DATE : / /

The main method is static in Java, so that compiler can call it without the creation of an object or before the creation of an object of the class. Since the main method is static JVM can call it without creating any instance of a class which contains the method. If main method were not declared static than JVM has to create instance of main class.

Java specifies several access modifiers e.g. private, protected and public. Any method or variable which is declared public in Java can be accessible from outside of that class. Since the main method is public in Java, JVM can easily access and execute it.

9) WAP to print "Hello Nepal" in console.

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

Output:

Hello Nepal

PAGE :
DATE : / /

11) WAP to show the usages of various data types in java.

```
// Java program to demonstrate primitive data types
// in java

public class Example {
    public static void main(String args[])
    {
        //declaring character
        char a = 'G';
        //declaring integer
        int i = 89;
        //declaring byte
        byte b = 4;
        //declaring short
        short s = 56;
        //declaring double
        double d = 4.35543;
        //declaring float
        float f = 4.7334;

        System.out.println("char:" +a);
        System.out.println("integer:" +i);
        System.out.println("byte:" +b);
        System.out.println("short:" +s);
        System.out.println("float:" +f);
        System.out.println("double:" +d);
    }
}
```

PAGE :
DATE : / /

Output
char: G
integer: 89
byte: 4
short: 56
float: 4.7837
double: 4.35543

Short Questions

1) History of Java
: Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at that time. The history of Java starts with the Green Team. Java team members (Green Team), initiated this project to develop a language for digital devices such as set-top boxes, television etc. However, it was suited for internet programming. Later, Java technology was incorporated by Netscape.

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic." Java was developed by James Gosling, who is known as father of Java, in 1995. James Gosling and his team members started the project in early 90s.

DATE : / /

Currently, Java ~~use~~ is used in internet programming, mobile devices, games, e-business solutions, etc.

ii) JVM (Java Virtual Machine)

= JVM is an abstract machine. It is called virtual because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode. It is platform dependent.

JVM is the one that actually calls the 'main' method present in a java code. JVM is a part of JRE (Java Runtime Environment). Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java enabled system without any adjustment. This is all possible because of JVM.

When we compile a .java file, .class files (contains byte-code) with the same class names present in .java are generated by the Java compiler.

The JVM performs the following main tasks:

i) Loads code
ii) Verifies code
iii) Executes code
iv) Provides runtime environment

PAGE :
DATE : / /

iii) C++ Vs Java

Comparision Index	C++	Java
1) Platform dependency	C++ is platform dependent.	Java is platform independent
2) Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web based, enterprise and mobile applications
3) Multiple Inheritance	C++ supports multiple inheritance	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in Java.
4) Operator overloading	C++ supports operator overloading.	Java doesn't supports operator overloading
5) Structure & Union	C++ supports structure & union.	Java doesn't support structure & union.
6) Documentation comment	C++ doesn't support documentation comment.	Java supports documentation comment i.e (/** ... */)
7) Pointers	C++ supports pointer. We can write pointers program in C++.	Java supports pointer implicitly. We can't write pointer program in java.

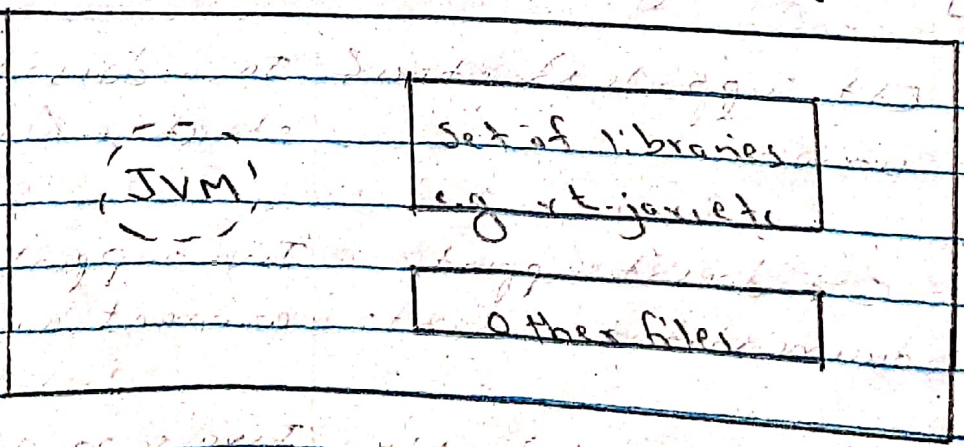
PAGE :
DATE : / /

iii) JRE

= JRE stands for Java Runtime Environment.
The Java Runtime Environment is a set of software tools which are used for developing Java applications.
It is used to provide the runtime environment.
It contains a set of ~~libraries~~ supporting libraries in a combination with core classes and various other files that are used by JVM at runtime.
JRE is a part of JDK but can be downloaded separately. JRE physically exists.

JRE is required to execute Java programs. It consists of:

1) Java Virtual Machine (JVM)
2) Java libraries
   i) Static - functions that are required at compile time.
   ii) Dynamic - functions that are required at runtime and not a compile time.



JRE

PAGE :
DATE : / /

v) Data Types

= Data Types specify the different sizes and values that can be stored in the variable. There are two types of data Types in Java.

i) Primitive data types: The primitive data types include boolean, char, byte, short, int, long, float, and double.

ii) Non-primitive data types: The non-primitive data types includes strings, arrays, classes & interfaces.

There are 8 types of primitive data types.

i) Boolean data type (Size - virtual machine dependent)

= Boolean data type represents only one bit of information, either ~~true or false~~. It is used to store only two possible values: true or false.

Example: Boolean one = false (default value false)

ii) Byte data type (size = 1 byte (8 bits))

= The byte data type is an 8-bit signed two's complement integer. The byte data type is useful for saving memory in large arrays. (Range: -128 to 127)

Example: byte a = 10; (default value 0)

iii) Short data type (Size = 2 byte (16 bits))

= The short data type is a 16-bit signed two's complement integer. Similar to byte, use a short to save memory in large arrays, in situation where the memory

PAGE :
DATE : / /

Eg:- short s=1000, Range (-32,768 to 32,767)
savings actually matters. (default value 0)

iv) Int Data type
= the int data type is a 32-bit signed two's complement integer. It's value range lies between -2^{31} to 2^{31-1}. It's default value is 0.
Eg:- int a = 10,000. (4 byte - 32 bits)

v) Long Data Type (16 byte - 64 bits)
= The long data type is a 64-bit two's complement integer. It's default value is 0. The long data type is used when you need a range of values more than those provided by int.
Eg:- long a = -200000L

vi) float Data Type
= The value range of float is unlimited. The size of float is 4 byte (32 bits). It's default value is 0.
Example:- float f1 = 234.5

vii) Double data Type
= The value range of double data type is unlimited for decimal values, this data type is generally the default choice. It's size is 8 byte (64 bits). It's default value is 0.
Eg:- double d = 12.3

PAGE :
DATE : / /

iii) Char Data Type

= The char data type is a single 16-bit unicode character. The char data type is used to store characters. Its size is 2 bytes (16 bits)

Eg:-

char letterA = 'B'.

6) JIT Compile

= Just-In-Time (JIT) compiler is an essential part of the JRE i.e Java RunTime Environment, that is responsible for performance optimization ~~based on~~ of java based applications at runtime. Compiler is one of the key aspects in deciding performance of an application for both parties i.e the end user and the application developer.

In order to improve performance, JIT compilers interact with the Java Virtual machine (JVM) at runtime and compile suitable bytecode sequences into native machine code. While using a JIT compiler, the hardware is able to execute the native code, as compared to having the JVM interpret the same sequence of bytecode repeatedly and incurring an overhead for the translation process. This subsequently leads to the performance gains in the execution speed, unless the compiled methods are executed less frequently.

The optimizations performed by JIT compilers are data-analysis, reduction of memory accesses by register allocation, translation from stack operations to register operations, elimination of

common subexpressions, etc. The greater is the degree of optimization done, the more time a JIT compiler spends in the execution stage.

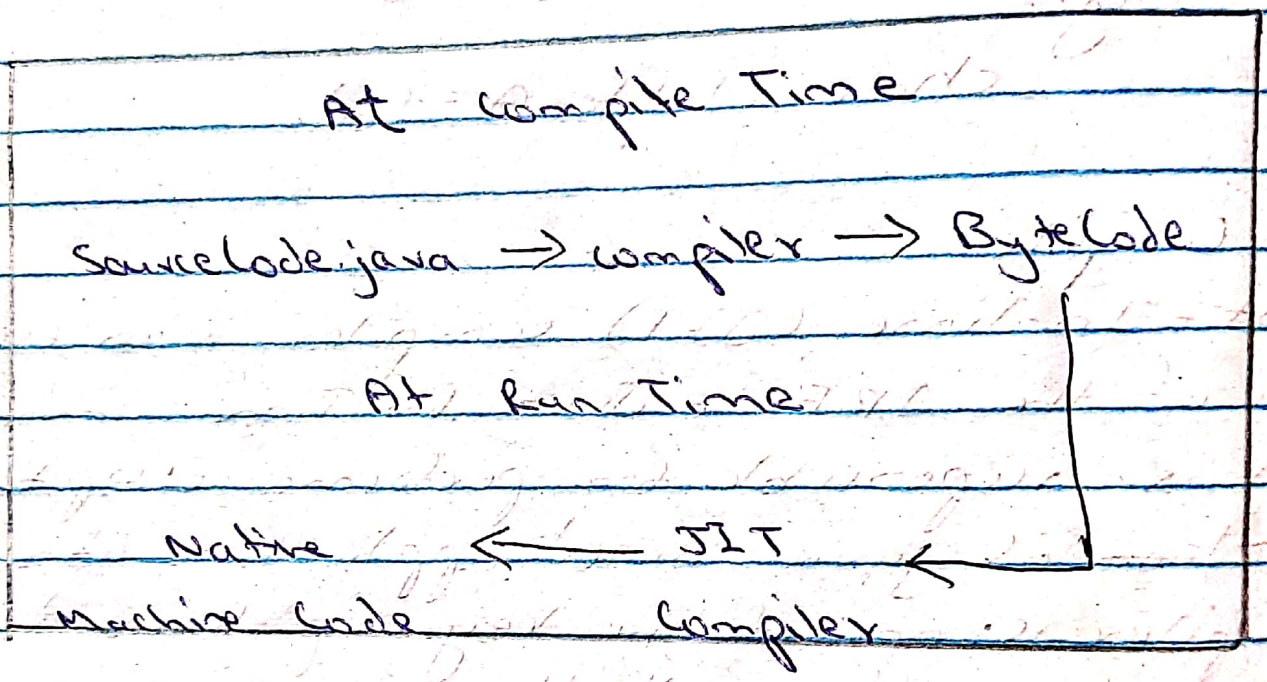


Fig:- Working of JIT compiler