# Basic Java

## Unit 1 - Introduction to Java

Pratian Technologies (India) Pvt. Ltd.
www.pratian.com

PRATIAN
TECHNOLOGIES

PRATIAN TECHNOLOGIES

# Topics

- History of Java
- The 3 editions of Java
- Installation of Java
- Simple Java application
- JRE and JDK
- How Traditional Programs Run
- How Java Programs Run
- Features of Java
- Java Applications Vs Java Applets
- Security in Java

PRATIAN TECHNOLOGIES

# History of Java

- Developed at Sun Microsystems between 1991 and 1995
  - Started as a small, secret project called *The Green Project* in 1991; headed by James Gosling and Bill Joy

James Gosling

Bill Joy

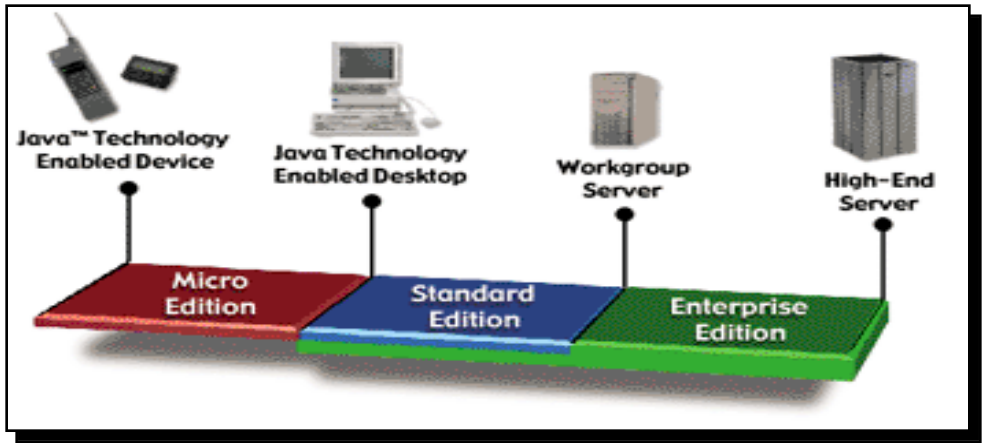
- The original objective of Java was:
  - to provide a platform-independent programming language and operating system for consumer electronics.

- Java became popular as a language of the Web due to:
  - Bytecode
  - Applet
  - Platform-independent nature

PRATIAN TECHNOLOGIES

# The 3 editions of Java

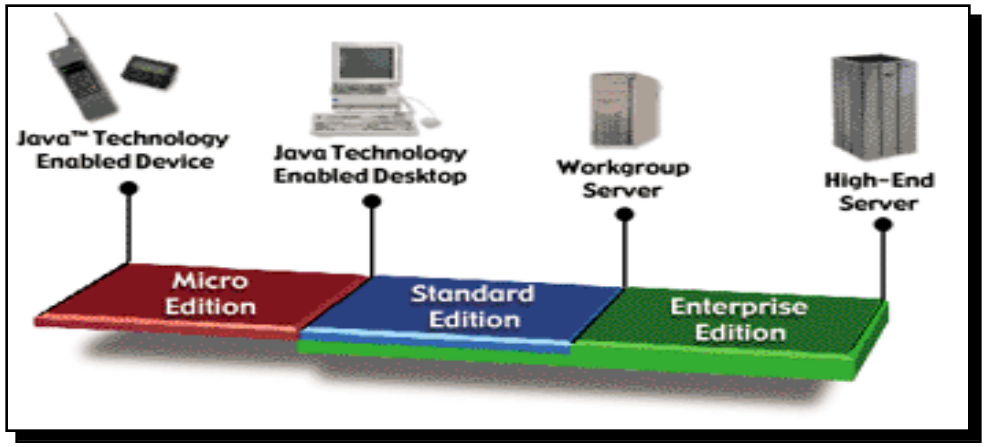
- Java Micro Edition (JME)
- Java Standard Edition (JSE)
- Java Enterprise Edition (JEE)



PRATIAN TECHNOLOGIES

# The 3 editions of Java

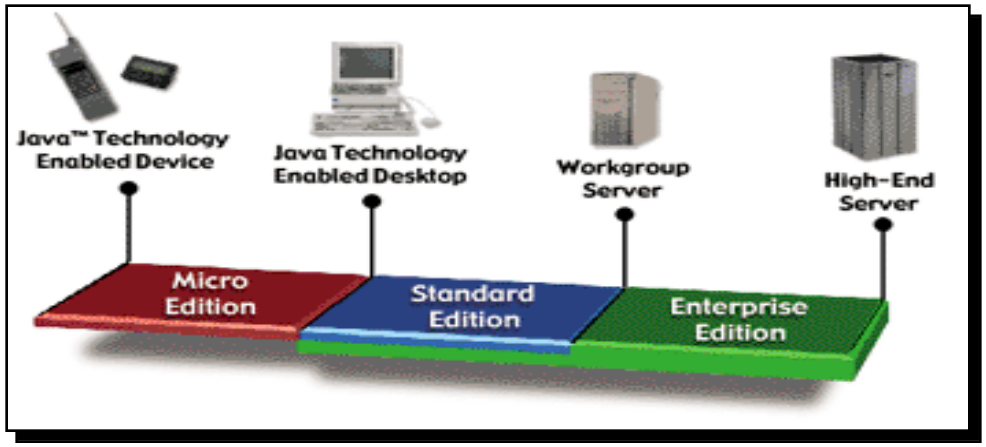
- Java Micro Edition (JME):
  An application platform for Java applications to run on mobile phones, PDAs, etc.
- Java Standard Edition (JSE)
- Java Enterprise Edition (JEE)



PRATIAN TECHNOLOGIES

# The 3 editions of Java

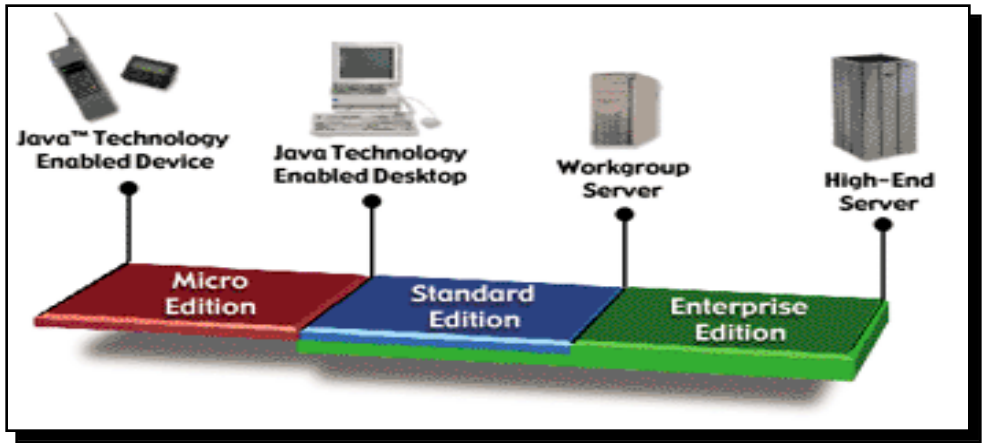
- Java Micro Edition (JME)
- Java Standard Edition (JSE):
  An application platform for Java applications to run on desktops
- Java Enterprise Edition (JEE)



PRATIAN TECHNOLOGIES

# The 3 editions of Java

- Java Micro Edition (JME)
- Java Standard Edition (JSE)
- Java Enterprise Edition (JEE):
  Is the industry standard for developing portable, robust, scalable, and secure server-side distributed applications



PRATIAN TECHNOLOGIES

# Developing our first Java application

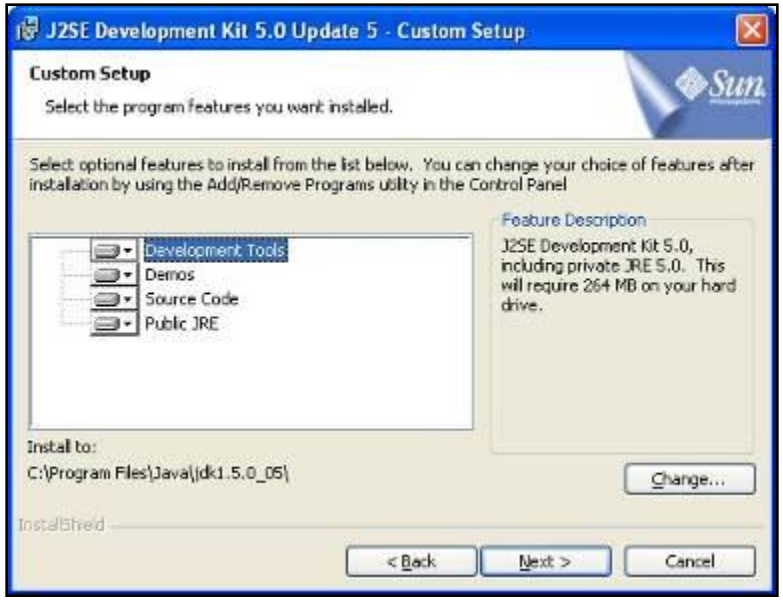
- Pre requisites for developing a java application
  - Installation
  - Setting the path
- Steps involved
  - Write the java program
  - Compile and debug compile-time errors, if any
  - Execute

PRATIAN TECHNOLOGIES

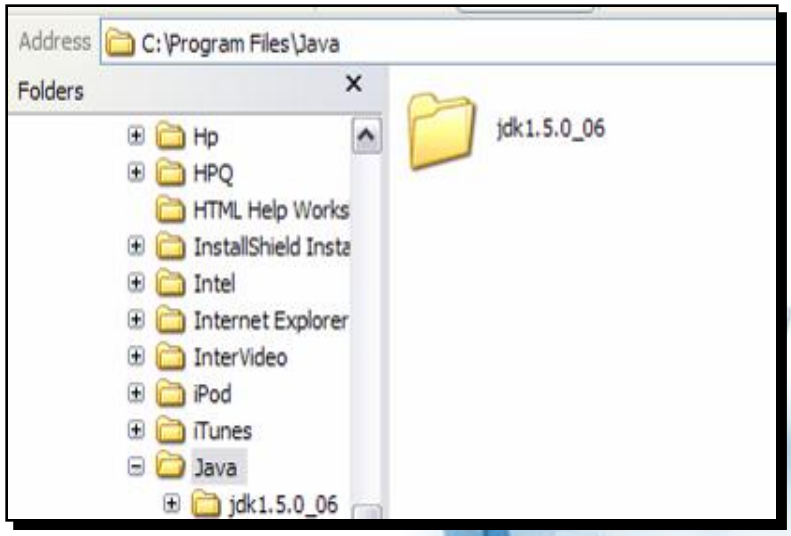
# Installation of Java

1. JDK can be downloaded from Sun Microsystems Web site.

2. The JDK comes as an executable file; on execution of the file, the JDK would be installed on the computer.



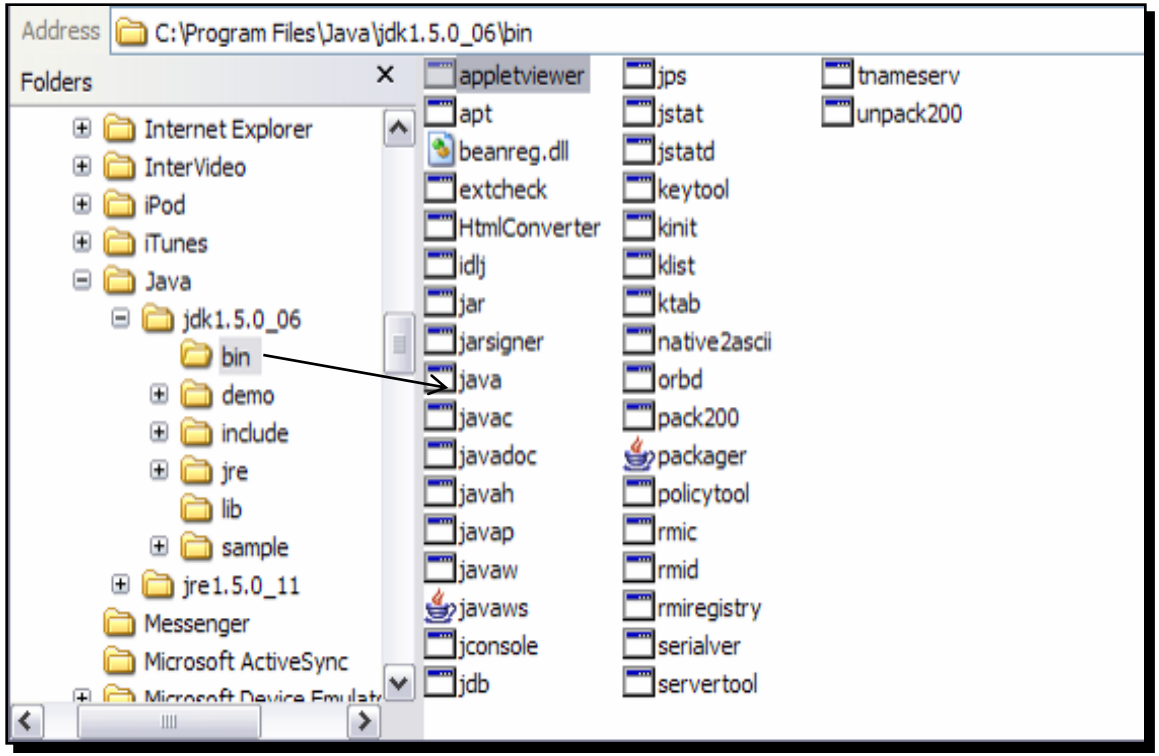

3. Upon installation, you will be able to view the jdk1.6 folder in the following file path: …**/Program Files/Java**.



PRATIAN TECHNOLOGIES

# jdk/bin Folder

- The jdk1.6/bin folder contains all the executable applications, such as:
  - javac
  - java
  - javadoc
  - jar




Add the *bin* directory to an appropriate file path:

- For example, if Java is installed in the file path, C:\Program Files, add C:\Program Files\Java\jdk1.6.0_14\bin to the environment variable path.

PRATIAN TECHNOLOGIES

# Creating a Simple Java Application

1. Open a text editor and create the source file by using the below code

HelloWorld.java

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World");

    }

}
```

2. The file should be saved with a .java extension:

   In this case, name the file as HelloWorld.java

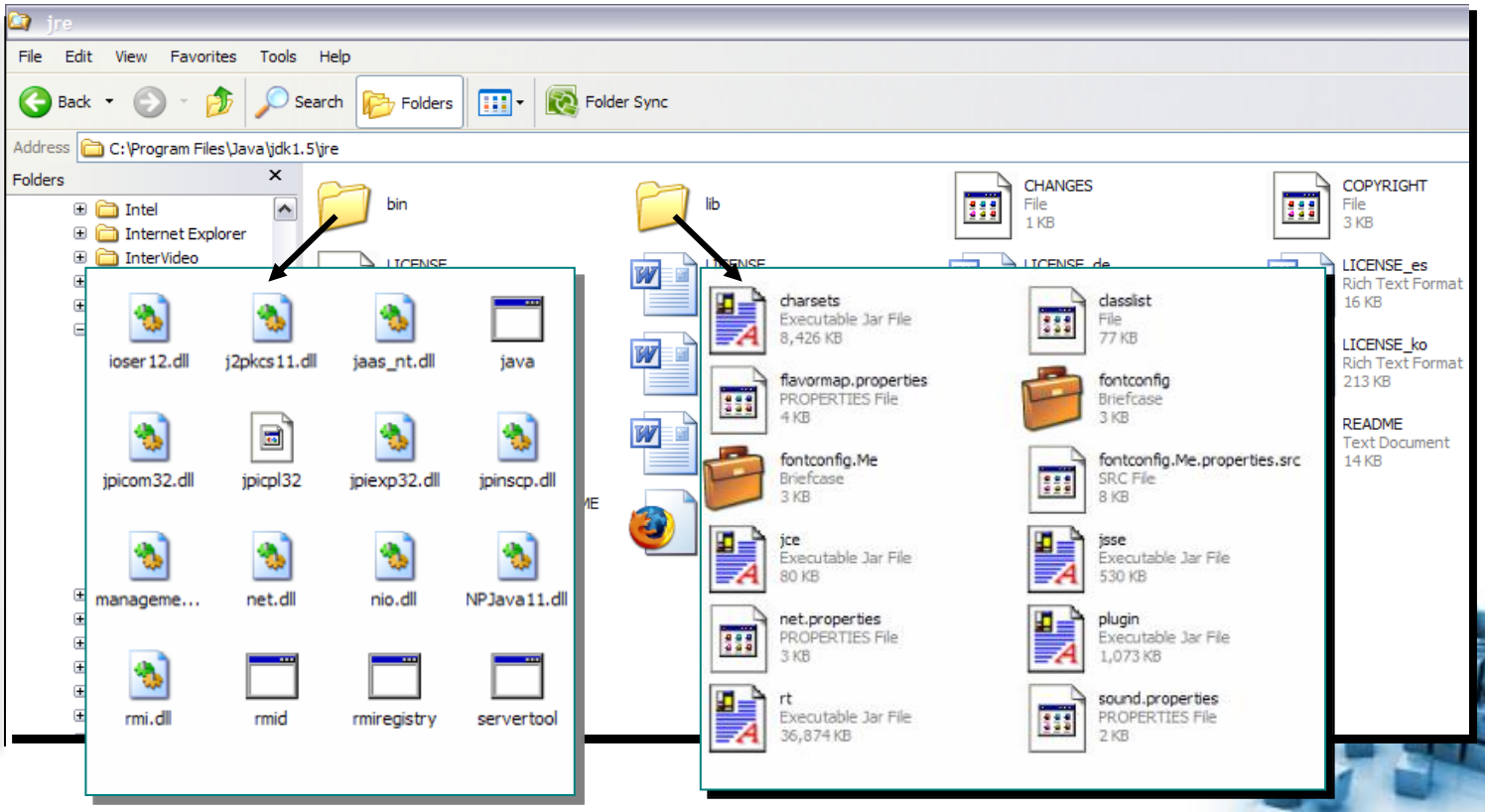
3. Compile the file in the console by using **javac** HelloWorld.java

4. Finally, execute the file by using **java** HelloWorld

PRATIAN TECHNOLOGIES

# JRE (Java Runtime Environment)

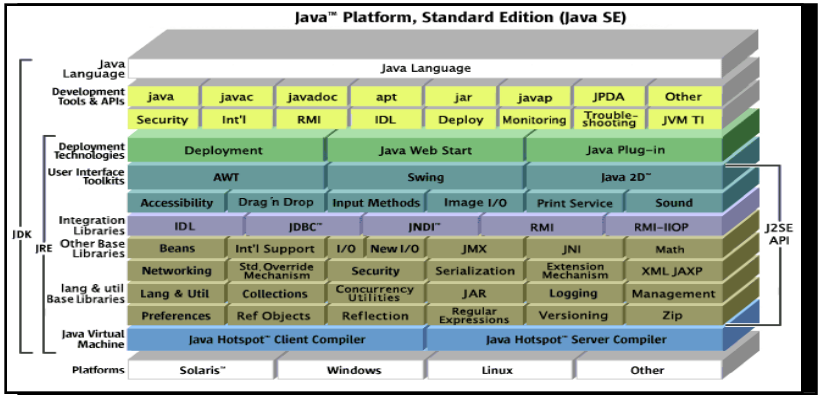
- JRE is found in the jdk1.6 folder:
  - executables for Java Runtime are found in the jdk1.6/jre/bin folder, and
  - classes from different APIs and other plug-ins are in the jdk1.6/jre/lib folder.



PRATIAN TECHNOLOGIES

# JRE and JDK

- Java Development Kit (JDK) is a software required to develop Java applications.
- It contains tools such as the compiler, debugger, libraries, etc.




- When you install the JDK, the JRE is also installed along with it.
- To run a Java application, it is enough to have JRE installed in your machines

PRATIAN TECHNOLOGIES

# Test your skills…

- Write a java program to swap values of two numbers.
- Compile your code and debug compile time errors, if any
- Execute the code

# Traditional programs v/s Java programs

- We will now see the difference in the execution of
  - Traditional programs
  - Java programs

PRATIAN TECHNOLOGIES

# Features of Java

- Simple
- Object-Oriented
- Distributed
- Portable
- Interpreted
- High performance
- Robust
- Secure


Buzzwords

PRATIAN TECHNOLOGIES

# Features of Java

- **Simple**
- Object-Oriented
- Distributed
- Portable
- Interpreted
- High performance
- Robust
- Secure

- Simple programming style
- Easy to use APIs
- Programmers can focus on problem-solving rather than memory-handling
- Ambiguous syntax and programming constructs of C++ that made application difficult to maintain are eliminated.

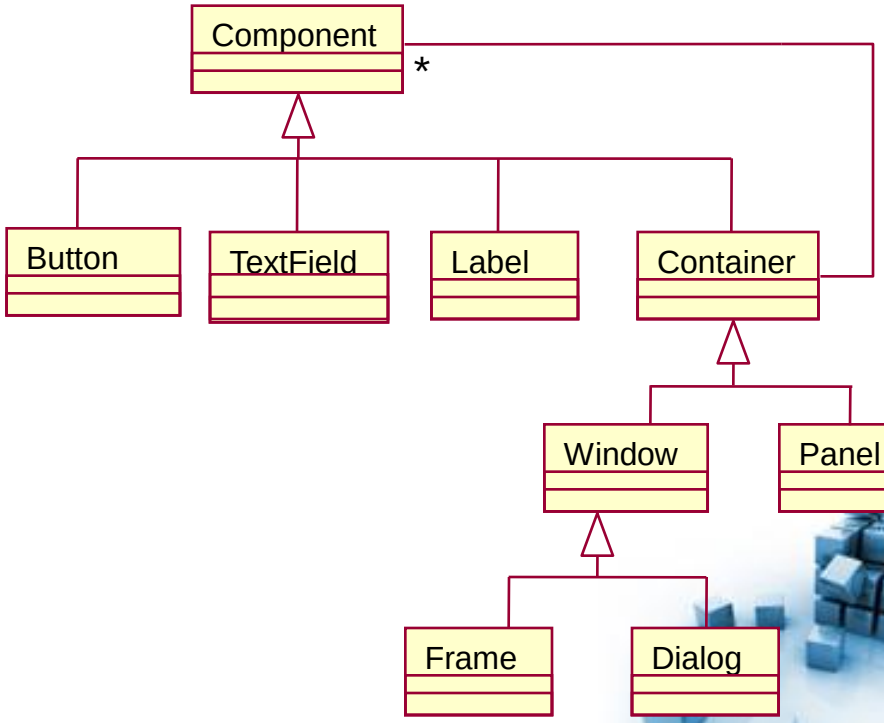
```
public class AuthenticationMgr {
    public Employee loginUser(String empid, String password)throws EmployeeNotFoun
        Employee employee = null;
        try {
            employee = EmployeeDAO.getInstance().getEmployee(empid);
            if (employee == null) {
                throw new EmployeeNotFoundException("Invalid user Id");
            } else {
                if (employee.getPassword().equals(password))
                    return employee;
                else
                    throw new InvalidPasswordException("Password not matching");
            }
        } catch (DAOException e) {
            e.getMessage();
            throw new EmployeeNotFoundException("Invalid user Id");
        }
```

PRATIAN TECHNOLOGIES

# Features of Java

- Simple
- **Object-Oriented**
- Distributed
- Portable
- Interpreted
- High performance
- Robust
- Secure

- Default programming paradigm supported by Java is OOP.
- Java retains the best Object Oriented Programming principles from C++ and Smalltalk.
- All of Java's built-in libraries are fully object-oriented.

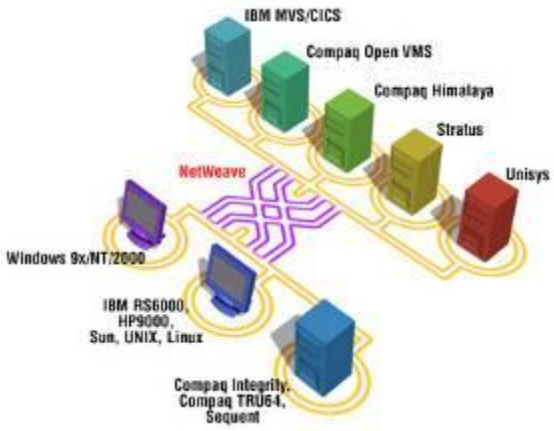


PRATIAN TECHNOLOGIES

# Features of Java

- Simple
- Object-Oriented
- **Distributed**
- Portable
- Interpreted
- High performance
- Robust
- Secure

- Supports a rich set of built-in APIs for building distributed applications
- Contains **extensive library of routines** for TCP/IP protocols
- Its libraries make creating network connections easier than C or C++.
- Java applications can open and access objects across the Net in a manner similar to accessing a local file system.

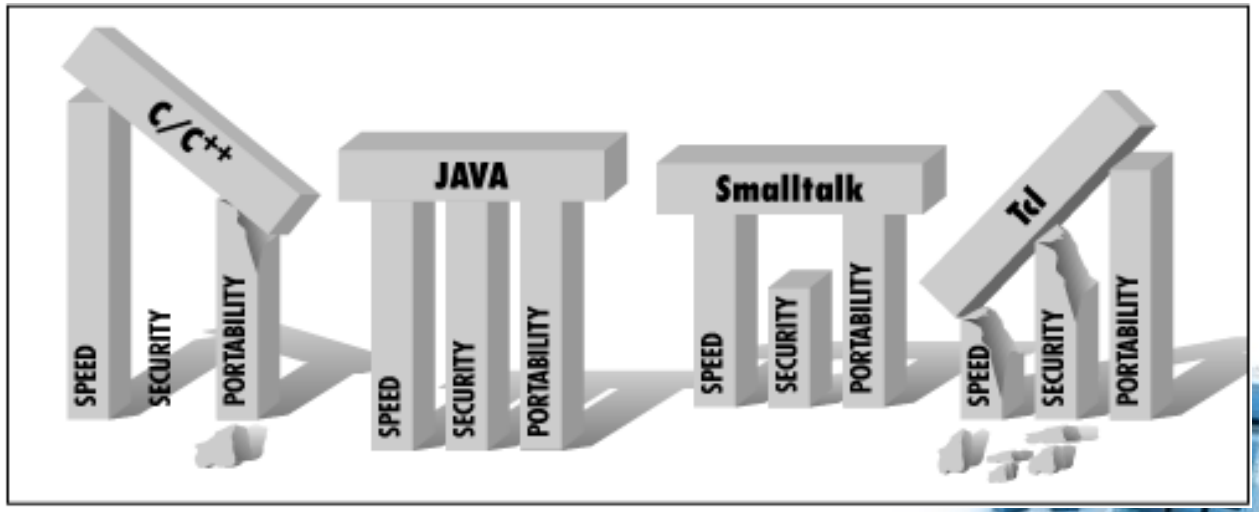


PRATIAN TECHNOLOGIES

# Features of Java

- Simple
- Object-Oriented
- Distributed
- **Portable**
- Interpreted
- High performance
- Robust
- Secure

- Java is platform-independent.
  - 'Write once, run anywhere.'
- The No implementation-dependency feature of Java makes it highly portable
  - The language was originally conceived for portable devices.

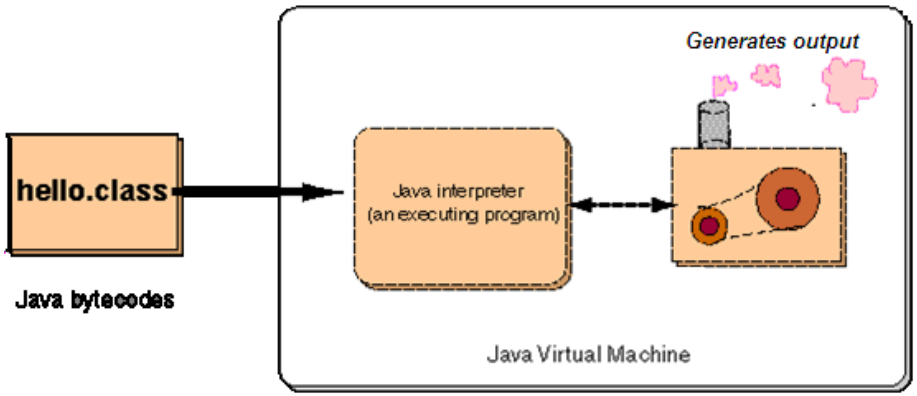


PRATIAN TECHNOLOGIES

# Features of Java

- Simple
- Object-Oriented
- Distributed
- Portable
- **Interpreted**
- High performance
- Robust
- Secure

- The Java compiler does not emit platform-specific executable code.
  - Produces an intermediate code called Bytecode
- Java bytecodes are translated, on the fly, into native machine instructions.
- Each instruction is executed by the JVM one by one (interpreted).




Interpreting Java Bytecode on a Virtual Machine

PRATIAN TECHNOLOGIES

# Features of Java

- Simple
- Object-Oriented
- Distributed
- Portable
- Interpreted
- **High performance**
- Robust
- Secure

**High performance. Guaranteed.**

- To make execution faster and simpler, the Java interpreter makes many optimizations during runtime
  - Virtual memory management and garbage collection
  - Inlining method calls
  - JIT Compilation
- The performance of bytecodes converted to machine code is similar to C or C++.

PRATIAN TECHNOLOGIES

# Features of Java

- Simple
- Object-Oriented
- Distributed
- Portable
- Interpreted
- High performance
- **Robust**
- Secure

What is robust ?

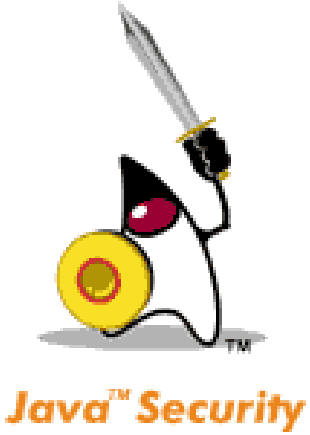
Refers to programs being less error prone and more stable

- Provides features such as exception handling wherein the user can be forced to handle errors without fail
- Is strongly typed and disallows improper conversions or casting
- Contains no pointers and has its own memory-handling mechanism

PRATIAN TECHNOLOGIES

# Features of Java

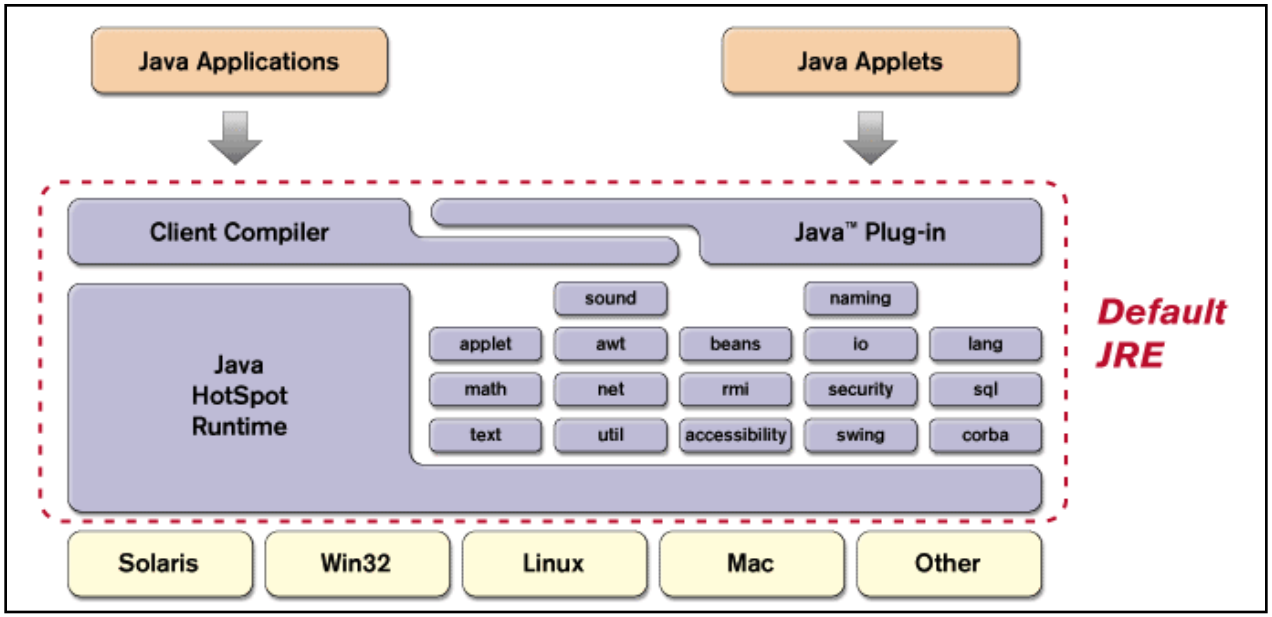
- Simple
- Object-Oriented
- Distributed
- Portable
- Interpreted
- High performance
- Robust
- **Secure**




- Java provides different levels of security, which make Java applications extremely safe.
- Applications cannot forge access to data structures or private data in objects.

PRATIAN TECHNOLOGIES

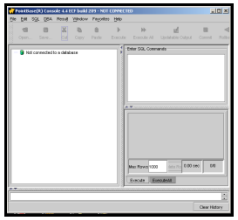
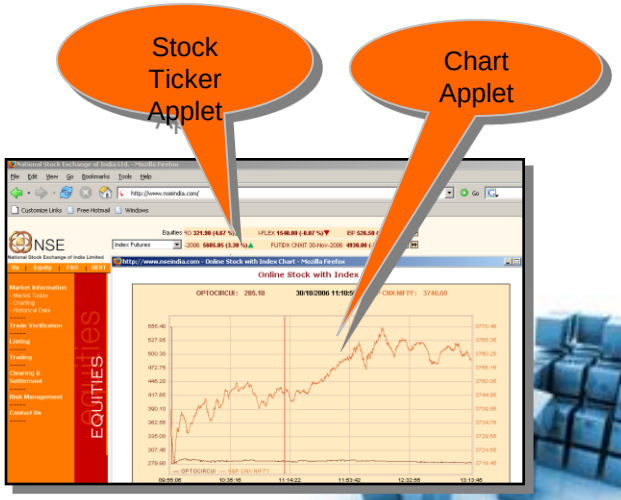
# Java Runtime Environment (JRE)




- **JRE** is the software that needs to be installed on each computer in which you want to execute Java applications.
- In addition to JVM, the JRE also comprises:
  - JIT compiler.
  - Java Plug-in.
  - Built-in APIs.
- For all practical purposes, a lot of people interchangeably use the terms **JRE** and **JVM**.

PRATIAN TECHNOLOGIES

# Java Applications vs Java Applets

| Feature | Application | Applet |
|---|---|---|
| Definition | A Java application is a standalone program that runs directly on JVM. | An applet, contrary to application, is embedded in Web pages. It is executed in AppletViewer or a Java-enabled Browser. |
| Security | It does need any special security restrictions. | It has some security restrictions. |
| Examples | Applications can be GUI-based or console-based:    | Applets are almost always GUI-based:   |

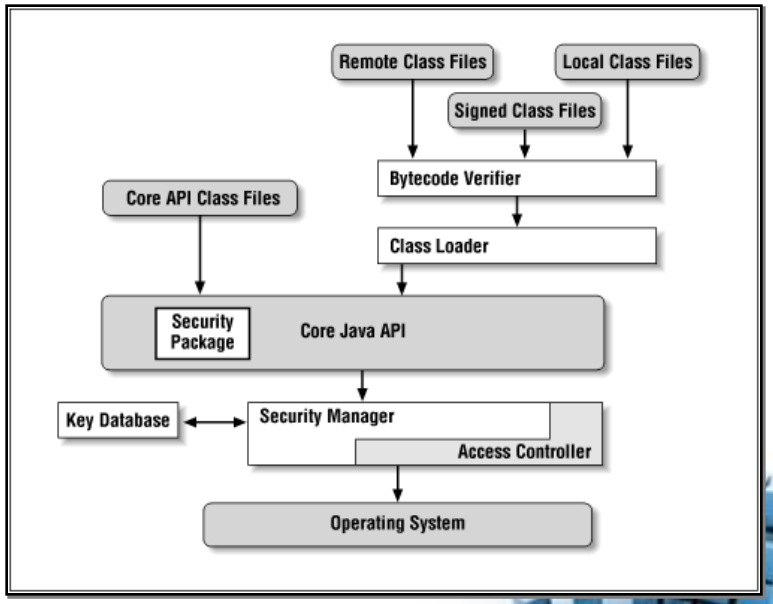
PRATIAN TECHNOLOGIES

# Security in Java

One of the important design goals of Java language: 'Creation of powerful applications that could be distributed easily over a network without jeopardizing the users' local security'

Levels of Security

- The language and the compiler
- Bytecode Verifier
- Class Loader
- SandBox



PRATIAN TECHNOLOGIES

# Security in Java

One of the important design goals of Java language: 'Creation of powerful applications that could be distributed easily over a network without jeopardizing the users' local security'

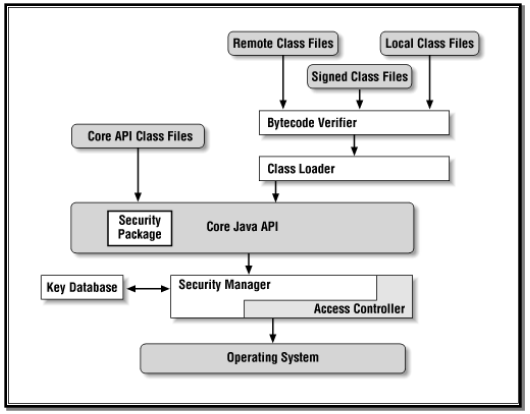
Levels of Security

The language and the compiler

Bytecode Verifier

Class Loader

SandBox




## Level 1: Language and Compiler

- The first line of security is the Java language itself and the compiler.
- Even though Java borrows some features from C++, it does not allow features such as forged access to objects or direct access to memory through pointers.
- Java has true arrays and uses named references instead of pointers.

PRATIAN TECHNOLOGIES

# Security in Java

One of the important design goals of Java language: 'Creation of powerful applications that could be distributed easily over a network without jeopardizing the users' local security'

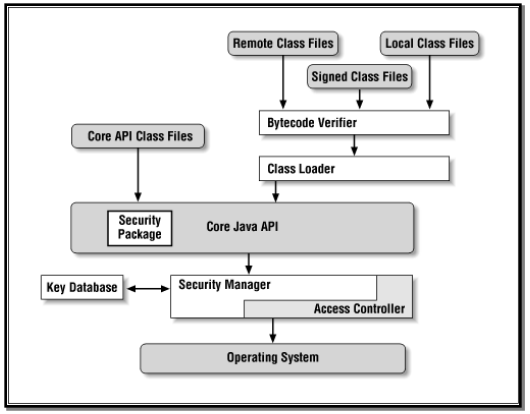
Levels of Security

The language and the compiler

Bytecode Verifier

Class Loader

SandBox




## Level 2: Bytecode Verifier

The compiler ensures that Java source code does not violate any of the safety rules.

If a Java complier that violates these rules, is written, the bytecode verifier performs these checks:

- Does not forge pointers
- Does not violate access restrictions
- Always uses objects in the way they were intended to be used
- Calls methods with appropriate arguments of the appropriate types
- Causes no stack overflows
- Does not try to execute any privileged or insecure instructions

PRATIAN TECHNOLOGIES

# Security in Java

One of the important design goals of Java language: 'Creation of powerful applications that could be distributed easily over a network without jeopardizing the users' local security'

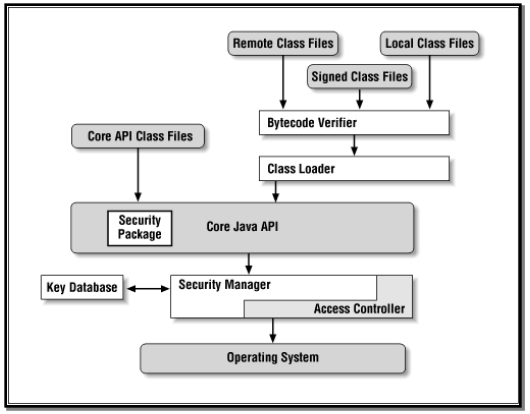
Levels of Security

The language and the compiler

Bytecode Verifier

Class loader

SandBox




## Level 3: Class Loader

- The class loader allocates memory space for each class and makes sure that the code is not attempting to deceive or dislodge (spoof) a built-in class.
- For example, a programmer cannot write his/ her own version of the string class and have it executed instead of the built-in class. The built-in classes are always checked first.

PRATIAN TECHNOLOGIES

# Security in Java

One of the important design goals of Java language: 'Creation of powerful applications that could be distributed easily over a network without jeopardizing the users' local security'

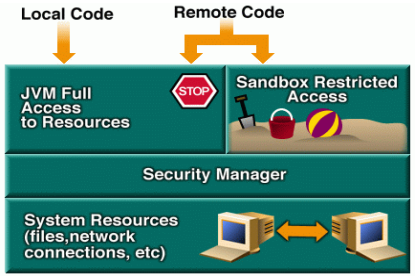
Levels of Security

The language and the compiler

Bytecode Verifier

Class Loader

Sandbox




Level 4: SandBox

The basic rules of SandBox security determine what a Java applet cannot do within a Web browser or in any other container that implements this Java security approach. Java applets cannot:

- delete files on the local system.
- read from or write to local files.
- create new directories on the local system.
- inspect directory contents or check various file attributes.
- execute programs on the local system.
- call DLLs.
- create network connections to machines other than the server from which the applet was loaded.
- create objects from the core packages that manage security, such as security manager and class loader.

PRATIAN TECHNOLOGIES

# Building Applications

- All programs written in the Java language are built from classes.
- Every application needs one class with the main method.
- The class is the entry point for the program and is passed to the java interpreter command to run the application.
- Signature of main() method

```
public static void main(String args[]) {…}
```

PRATIAN TECHNOLOGIES

# Command line arguments

- Passing Command line arguments
  - These arguments are then parsed according to requirements.
  - The java program must handle the arguments within it.

```
class CmdLine
{
  public static void main(String args[])
  {
  String name=args[0];
  System.out.println("Hello "+name);
  }
}
```

**Execute the program as below**
**c:\> java CmdLine Jane**

PRATIAN TECHNOLOGIES

# Test your skills…

- Write a java program to accept the radius of a circle as a command line input, calculate and display the area of the circle
- Compile your code and debug compile time errors, if any
- Execute the code

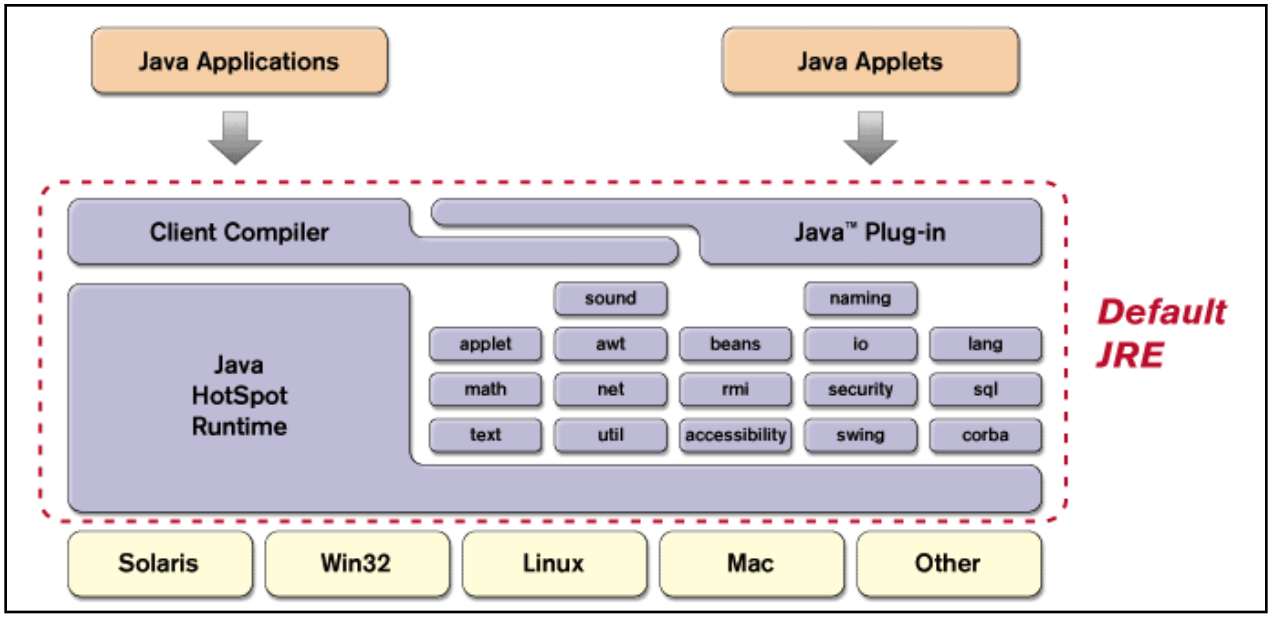
PRATIAN TECHNOLOGIES

# Question time

Please try to limit the questions to the topics discussed during the session. Thank you.

PRATIAN TECHNOLOGIES

# Java Runtime Environment (JRE)



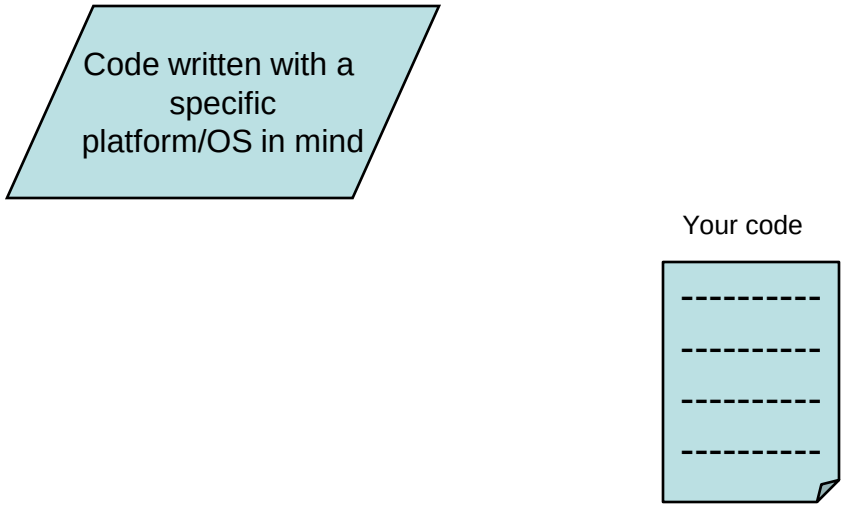

- **JRE** is the software that needs to be installed on each computer in which you want to execute Java applications.
- In addition to JVM, the JRE also comprises:
  - JIT compiler.
  - Java Plug-in.
  - Built-in APIs.

Java APIs provide the core functionality of Java programming language. They are a large collection of readymade software components for use while developing an application.

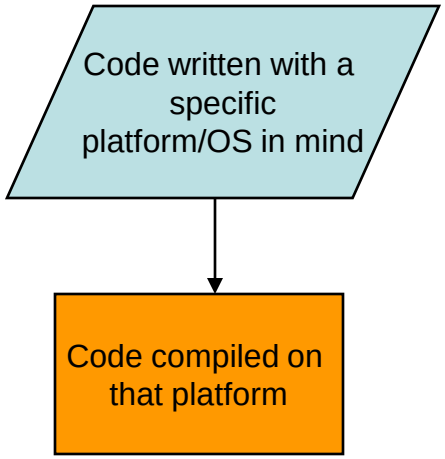
- For all practical purposes, a lot of people interchangeably use the terms **JRE** and **JVM**.

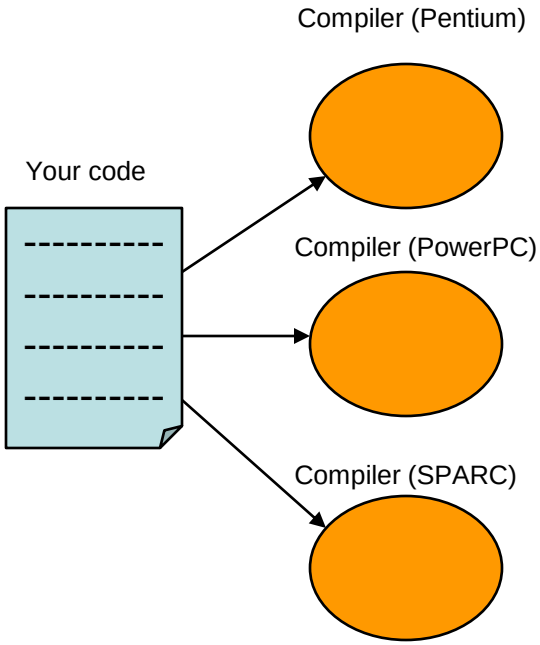
Return

# How traditional programs run

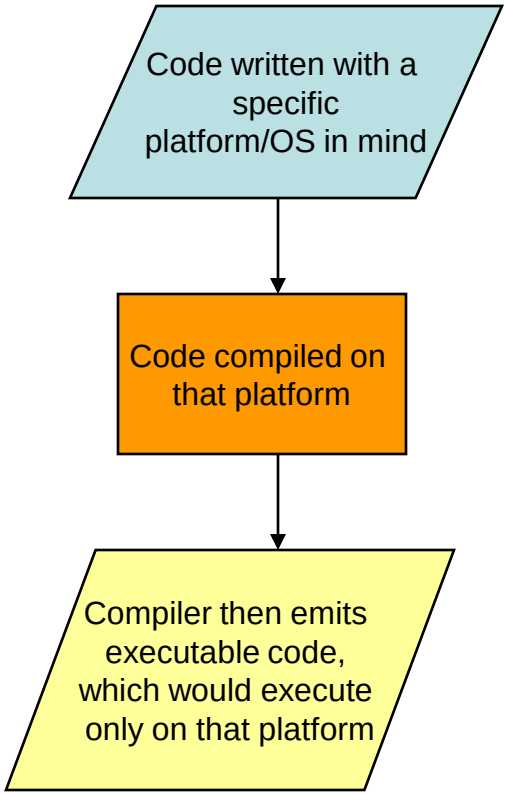


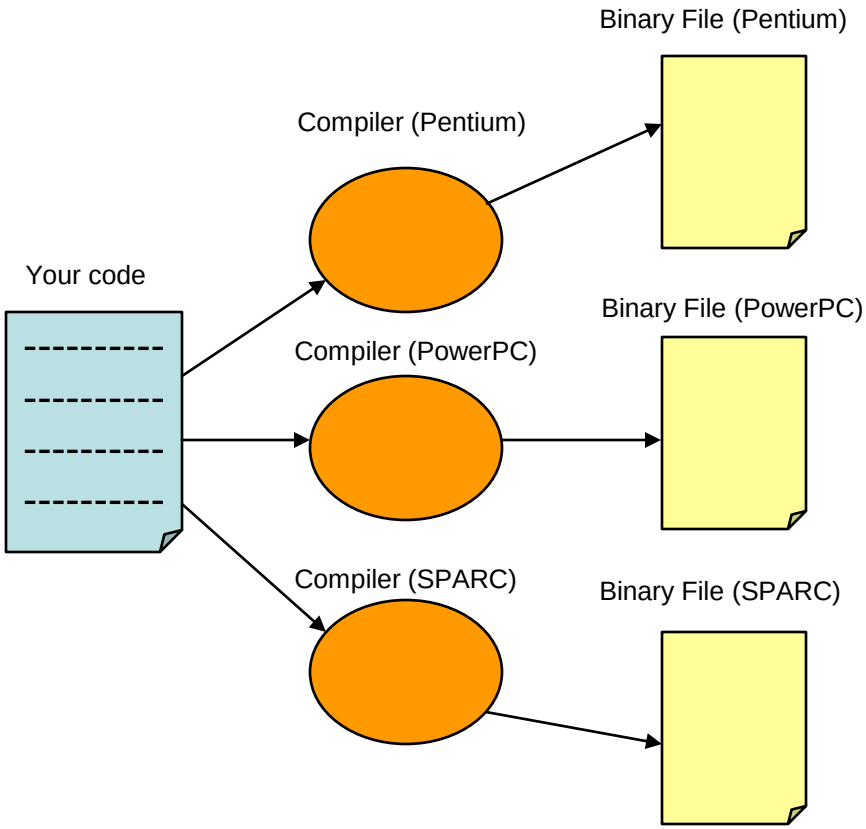
# How traditional programs run



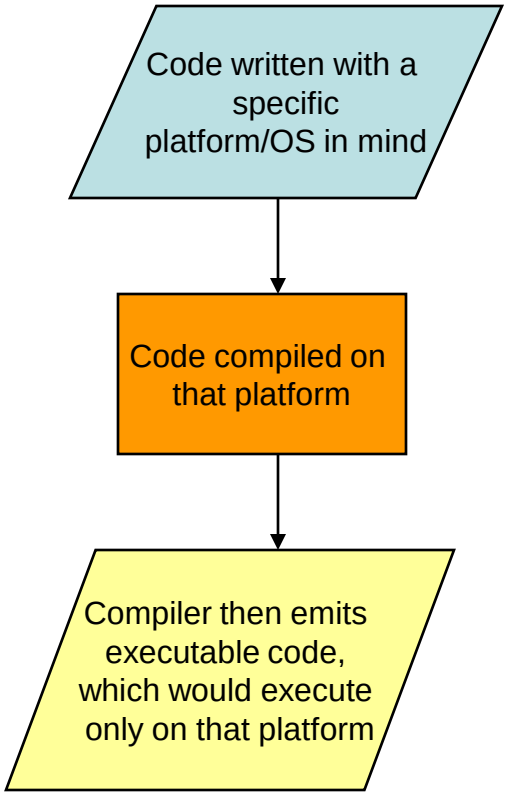



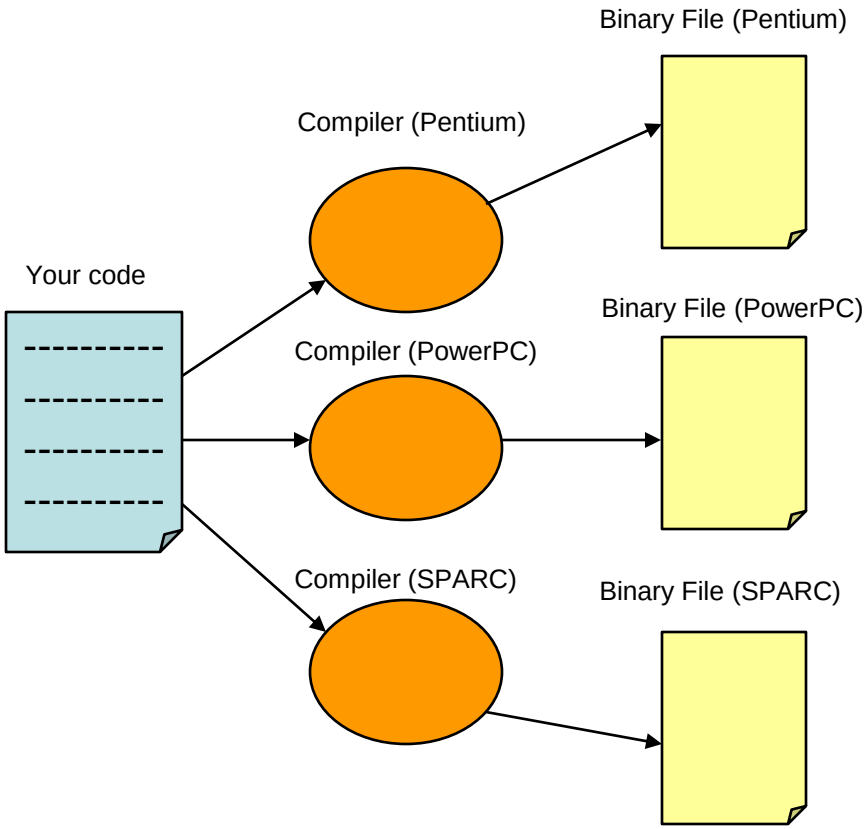
# How traditional programs run



# How traditional programs run

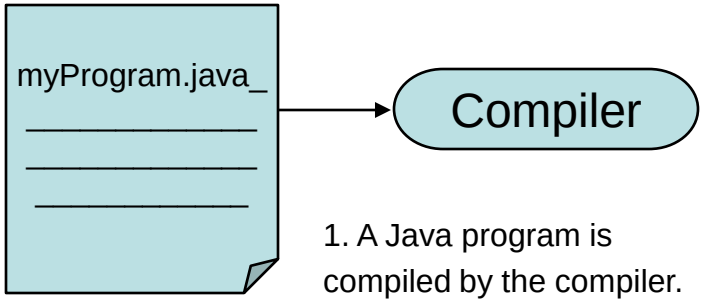



The source files had to be customized and compiled using different compilers for each platform.

Return

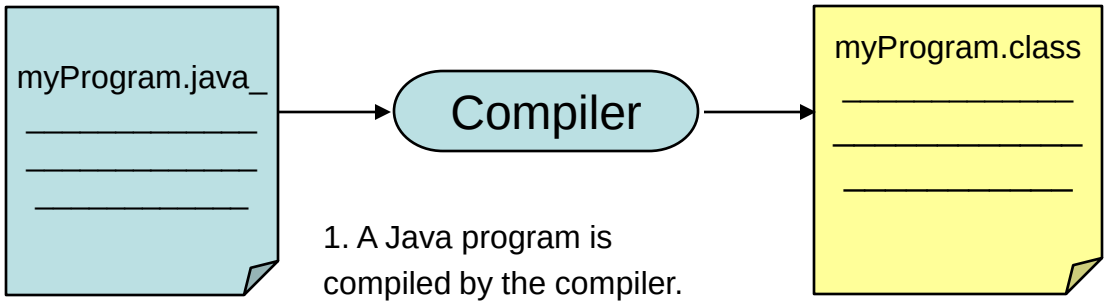
# How Java programs run




1. A Java program is compiled by the compiler.

PRATIAN TECHNOLOGIES

# How Java programs runs

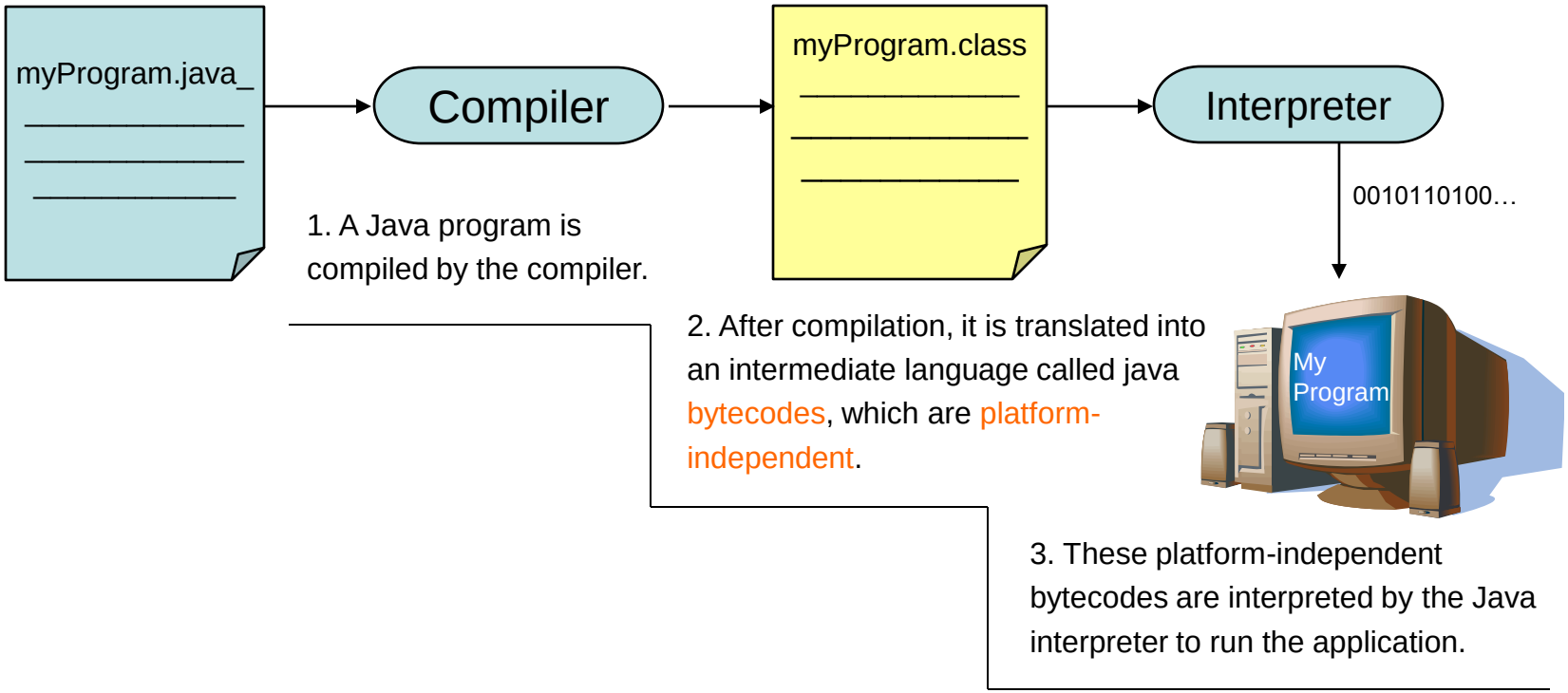



1. A Java program is compiled by the compiler.

2. After compilation, it is translated into an intermediate language called java bytecodes, which are platform-independent.

PRATIAN TECHNOLOGIES

# How Java programs runs



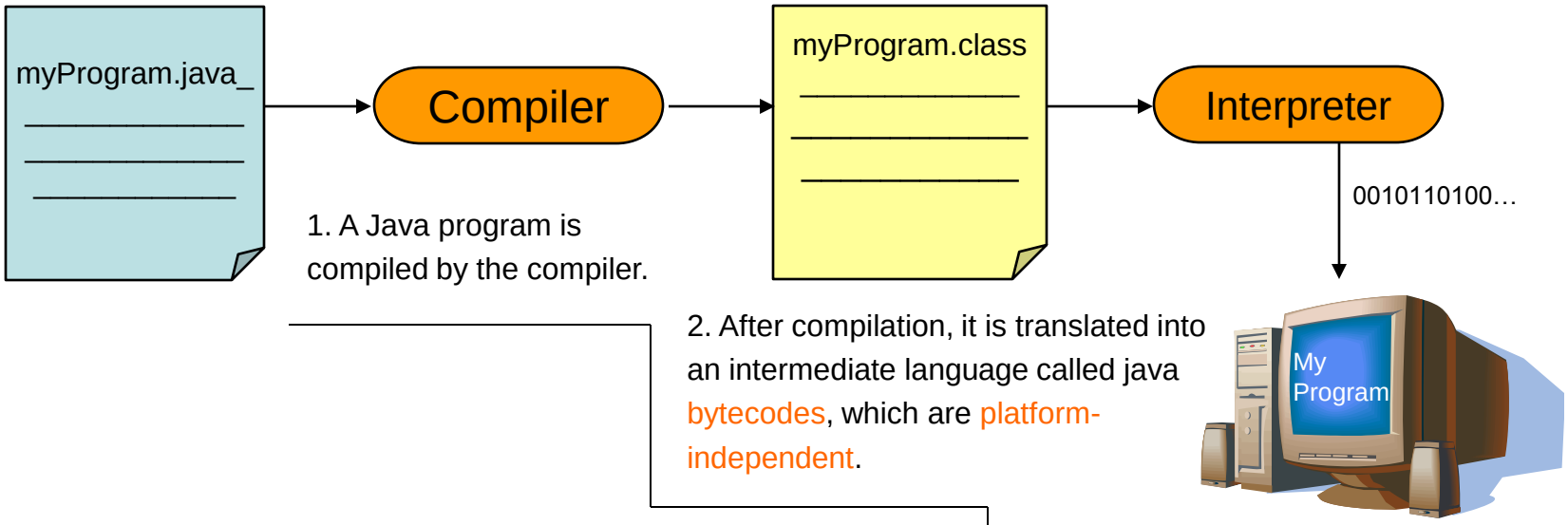

1. A Java program is compiled by the compiler.

2. After compilation, it is translated into an intermediate language called java bytecodes, which are platform-independent.

3. These platform-independent bytecodes are interpreted by the Java interpreter to run the application.

PRATIAN TECHNOLOGIES

# How Java programs runs




1. A Java program is compiled by the compiler.

2. After compilation, it is translated into an intermediate language called java bytecodes, which are platform-independent.

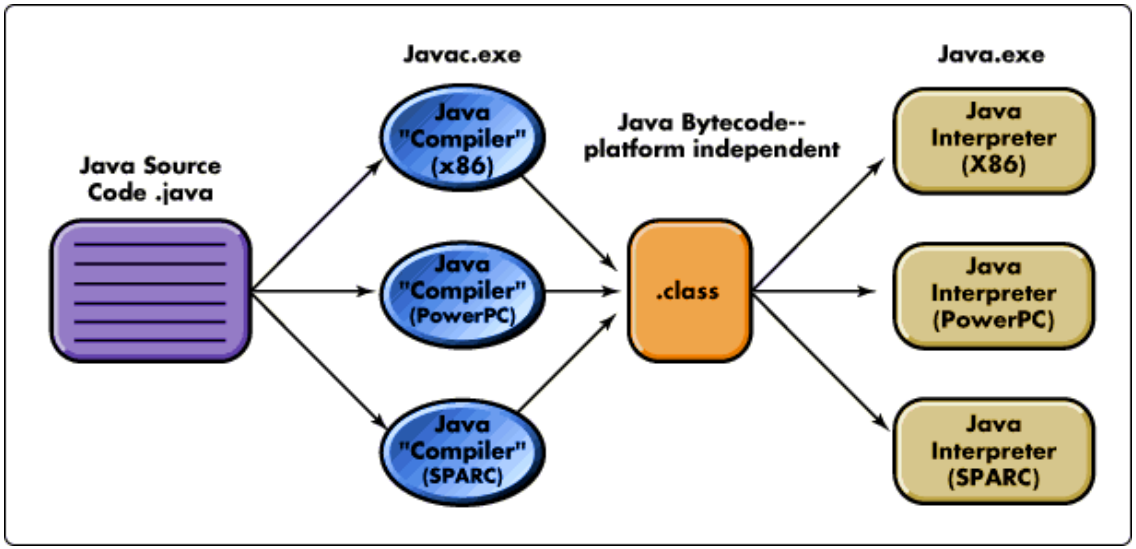
3. These platform-independent bytecodes are interpreted by the Java interpreter to run the application.

The compilation happens only once.
The interpretation occurs each time the program is executed.
Therefore, each Java program is compiled as well as interpreted.

Return

PRATIAN TECHNOLOGIES

# Java Bytecode




- The Java complier compiles the source program **.java* file into an intermediary *.class* file.
- The .class file contains *bytecodes* – the machine language of Java run-time system
- Bytecode is the magic behind
  - "***Write Once, Run Anywhere***"tm
- Bytecode is executed by Java run-time system

Return

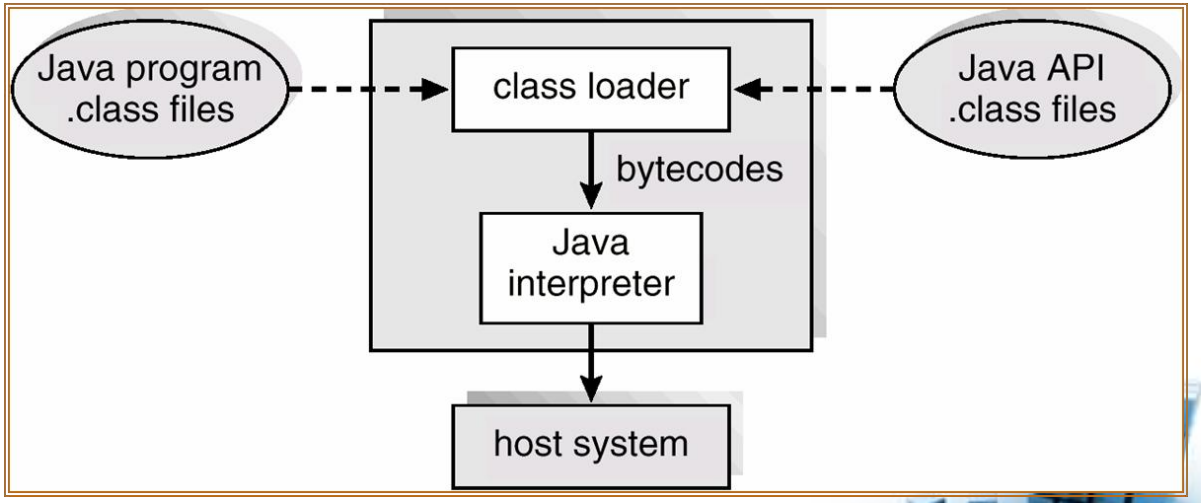
PRATIAN TECHNOLOGIES

# Java Virtual Machine (JVM)

- JVM is an abstract computer on which all Java programs run.
- It interprets the bytecodes to the underlying system.

**JVM:**

- **Allows cross-platform delivery**
- **Results in small size of compiled code**
- **Provides high-level security**

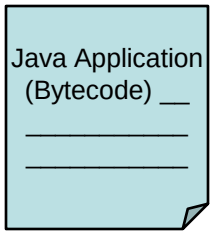


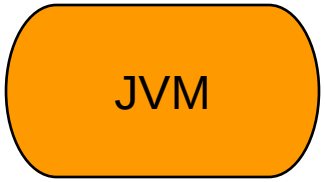

Return

PRATIAN TECHNOLOGIES

# Just In Time (JIT) compilation

Java Virtual Machine (JVM - interpreter)



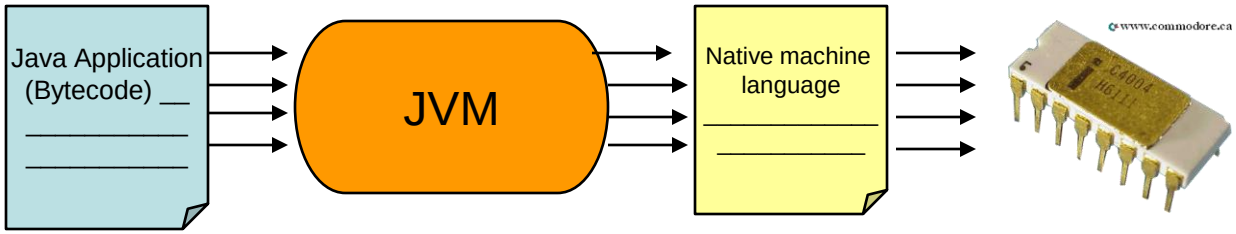




Converts instructions into machine language as each one is encountered

PRATIAN TECHNOLOGIES

# Just In Time (JIT) compilation

Java Virtual Machine (JVM - interpreter)

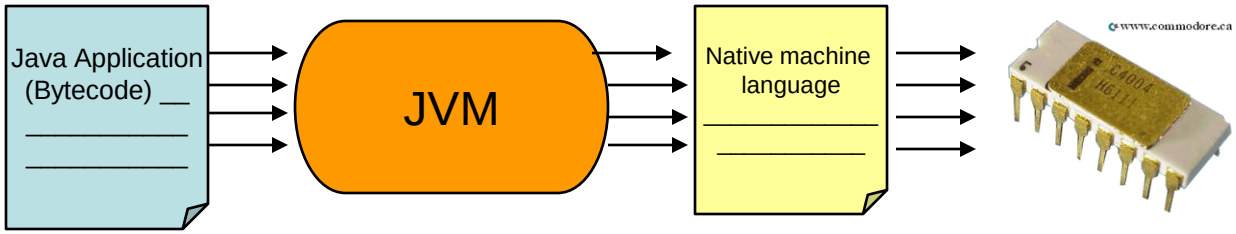



Converts instructions into machine language as each one is encountered

PRATIAN TECHNOLOGIES

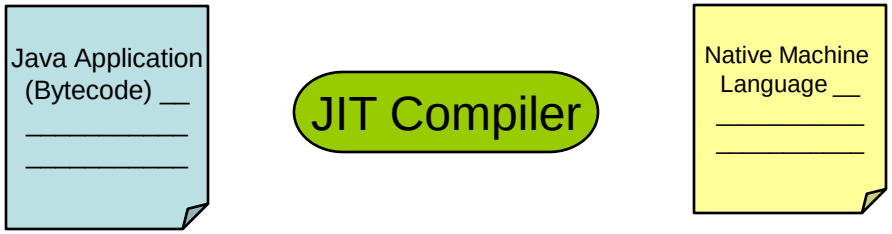
# Just In Time (JIT) compilation

## Java Virtual Machine (JVM - interpreter)




Converts instructions into machine language as each one is encountered

## Just-In-Time (JIT) Compiler (faster)



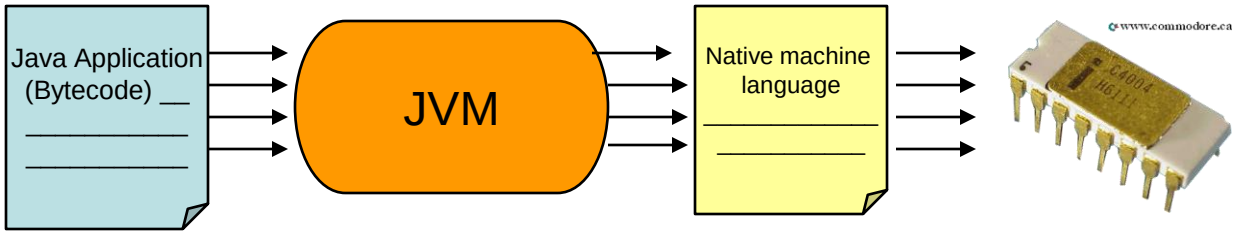

Converts all instructions into machine language as needed

Runs the machine language

PRATIAN TECHNOLOGIES

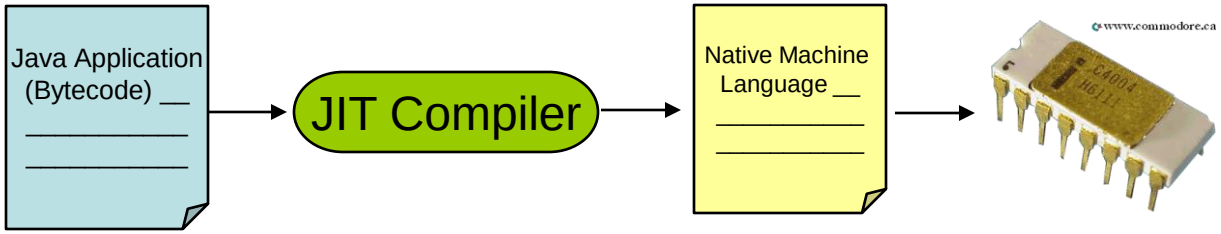
# Just In Time (JIT) compilation

## Java Virtual Machine (JVM - interpreter)




Converts instructions into machine language as each one is encountered

## Just-In-Time (JIT) Compiler (faster)




Converts all instructions into machine language as needed

Runs the machine language

**Different implementations of JVM use different algorithms.**

Return