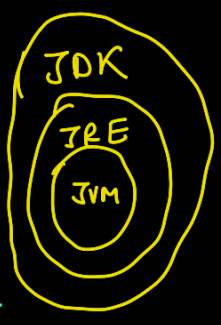
Java is platform independent object oriented programming language

Major advantage of Java is Portability [WORA]

Write Once Run Anywhere

Java program converted to Byte code & Now this Byte code can be run Any where !!



Java has 3 major components

① JVM ② JRE ③ JDK

↓ ↓ ↓

Java Virtual M/C — Java Runtime Environment — Java Development Kit

①

JVM is just an abstract [Virtual] M/C

JVM runs the Bytecode & Gives us M/C code !!

JVM is not Platform Independent.

for MAC OS we have diff exist JVM

for Windows we have different JVM

JVM has JIT compiler [Just In Time compiler] which converts Bytecode to M/C code

How JVM helps in Portability??

Suppose we write code on Windows Generated Bytecode Now This Bytecode I can run in Any M/C JVM

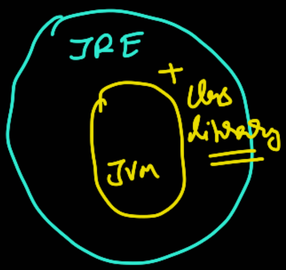
Java Program [Student.java] —javac Student.java (Runs Java compiler)→ Bytecode [Student.class]

② JRE [Java Runtime Environment]

JVM + class library = JRE

class library → like java.Math → if you use Any Math fn

java.util → Any collection or Arrays.sort()



for Math.abs() you haven't provide code for those code we need class library

JRE can run any bytecode or any Java Program But we will not be able to code it.

③ JDK (Java Development Kit) → helps to write code

JDK has programming language (Java), Compiler (javac), Debugger

JDK = JRE + programming lang + Compiler + Debugger

JDK → code hone ke liye

JRE → Everything got provided here through library

JVM → ye he run karta hai code to bytecode

All 3 (JDK, JRE, JVM) are Platform Dependent

So we download JDK for Java as it has Everything

java -version → To check Java Version

Now lets see JSE, JEE, JME

JSE (Java Standard Edition) or Core Java

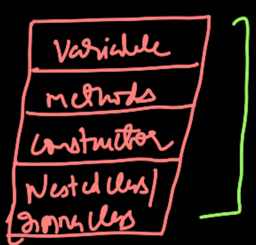
JEE (Java Enterprise Edition) → JSE + Transactional API (rollback, commit, JSP)
→ used to build large Scale Applications

JME (Java Mobile Edition or Jakarta EE)
↓
provide API for mobile Application

First Java Program

① File Name ≡ Class Name & class should be Public

② 1 file can have 1 Public class



A general class structure

③ JVM calls main() method to start the program

public static void main()

public → so that JVM can call it

void → No return type

static → it tells that 9 can not a part of object 9 can a part of class so 9 can be called directly with class Name

```
class A {
    static m1() { }
}
```

m1() can be called by A.m1()
No need of A's object

To Run Java program named Emp.java from cmd
1st compile it javac Emp.java → generates Emp.class
2nd run .class java Emp → no need to put .class

Q Why only 1 Public class in Java file ??

Sol →

```
public class A {
    public static void main() { }
}
```

Entry point of Java Program

main() in Public class → so that it can be called by JVM & main() is static so can be called by class Name

Public class Name ≡ File Name → Employee.java ✓ class Employee { }

Employee.javaa ✗ class Manager { }

Lets suppose we have Multiple Public class then How JVM know which class have main()

By Getting class Name ≡ File Name JVM gets class Name that is Public & can call main() without any confusion