




SDET Fundamental Training

Java Programming

A Brief History of Java



- 
- Creation of Java Language, by team named “Green” with members lead by James Arthur Gosling
 - Originally called Oak (1991)
 - First version of Java was released: 1995
 - FOSS (GPL): FOSS is a free and open source software and GNU General Public License
 - The GNU General Public License is a free, copy-left license for software and other kinds of works
 - Netscape Navigator Internet browser was the first Java enabled browser

Versions

- JDK 1.0 (1995)
- JDK 1.1 (1997)
- J2SE 1.2 (1998)) - Playground
- J2SE 1.3 (2000) - Kestrel
- J2SE 1.4 (2002) - Merlin
- J2SE 5.0 (2004) - Tiger
- Java SE 6 (2006) - Mustang
- Java SE 7 (2011) - Dolphin
- Java SE 8 (2014) – version for our session

Java 2

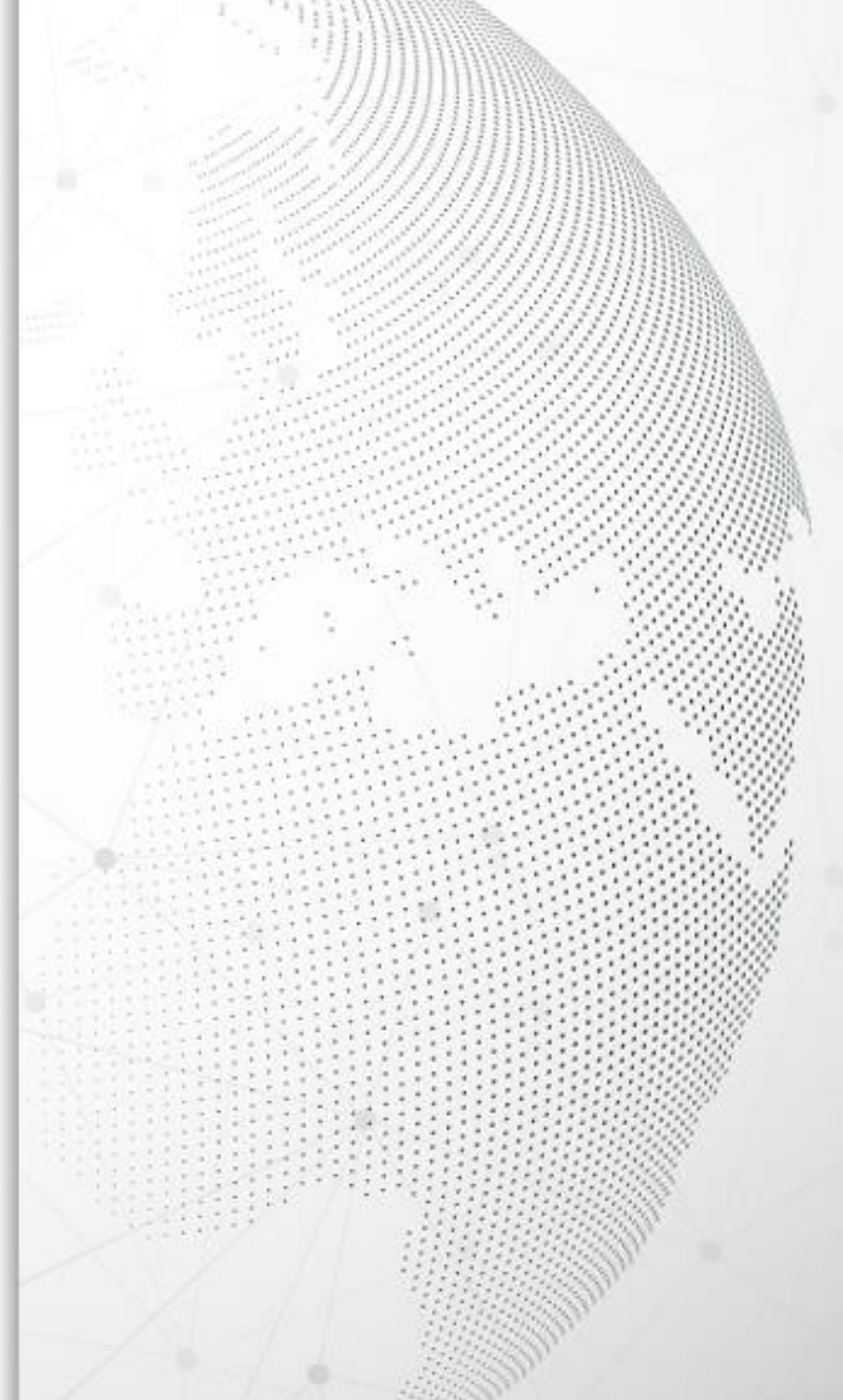
Why so many versions!?

Why so many versions?

Java started as a language originally targeting the digital cable television industry but it was ahead of its time for this industry. The Internet was beginning to boom at that time and Java turned out to be just right for it.

Eventually more APIs were added and Java (just like any other language) progressively evolved in terms of providing more features for security, reflection mechanism, newer utility classes etc.

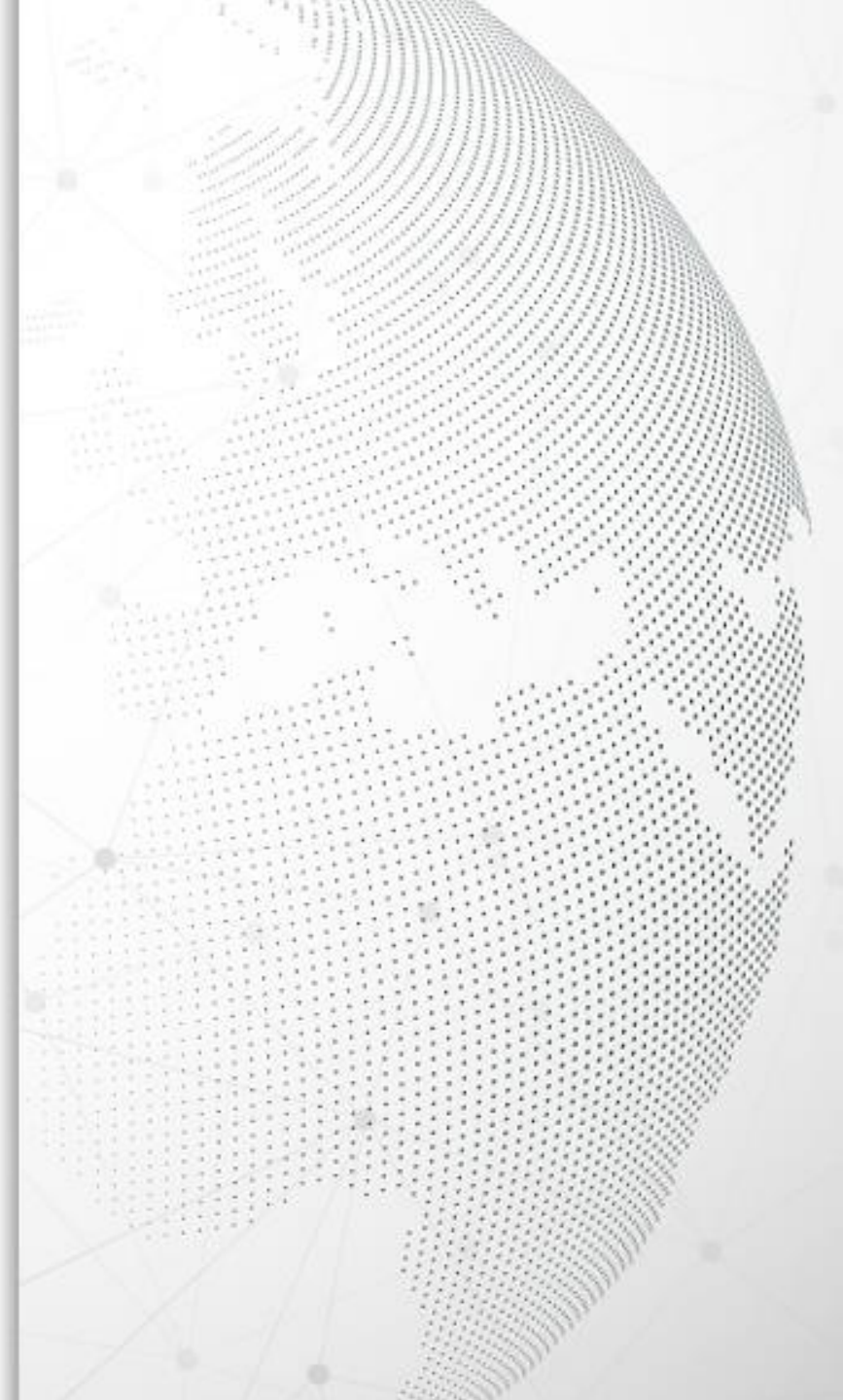
Therefore with each version, the scope of the language has increased in terms of features.



Features

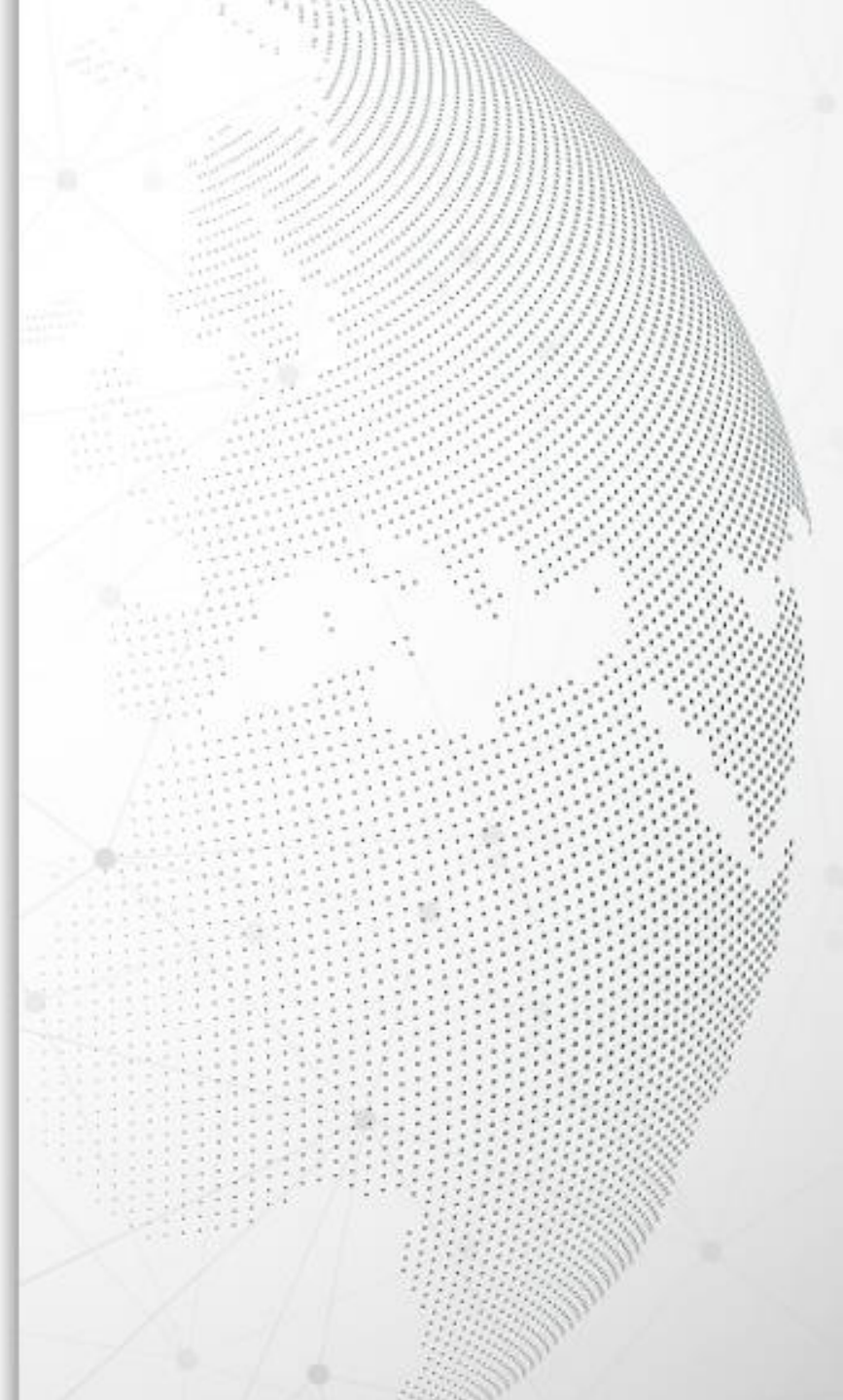
Simple

- Syntax similar to C/C++
 - Same types of loops
 - Same data types
- Error-free code
 - no pointers
 - no array index out of bounds problems
 - no explicit memory allocation & de-allocation



Features Simple

Java is able to overcome memory related problems by the virtue of Garbage collection which is a tool that attempts to free unreferenced memory (memory occupied by objects that are no longer in use by the program) also called garbage, in program



Object-Oriented Programming

“Object-oriented programming is a method of implementation in which programs are organized as cooperative collections of **objects**, each of which represents an **instance** of some **class**, and whose classes are all **members** of a **hierarchy of classes** united via **inheritance** relationships.”

GRADY BOOCH

Programming Approaches

Structured Approach

- Based on functions
- goto branching
- C, C++, COBOL, Pascal

Some disadvantages: No constructs for encapsulation, chances of code repetition, No strong data hiding concept, difficult to debug

Object-Oriented Approach

- Smalltalk, Java, C#

Object-Oriented Programming

Basic Concepts:

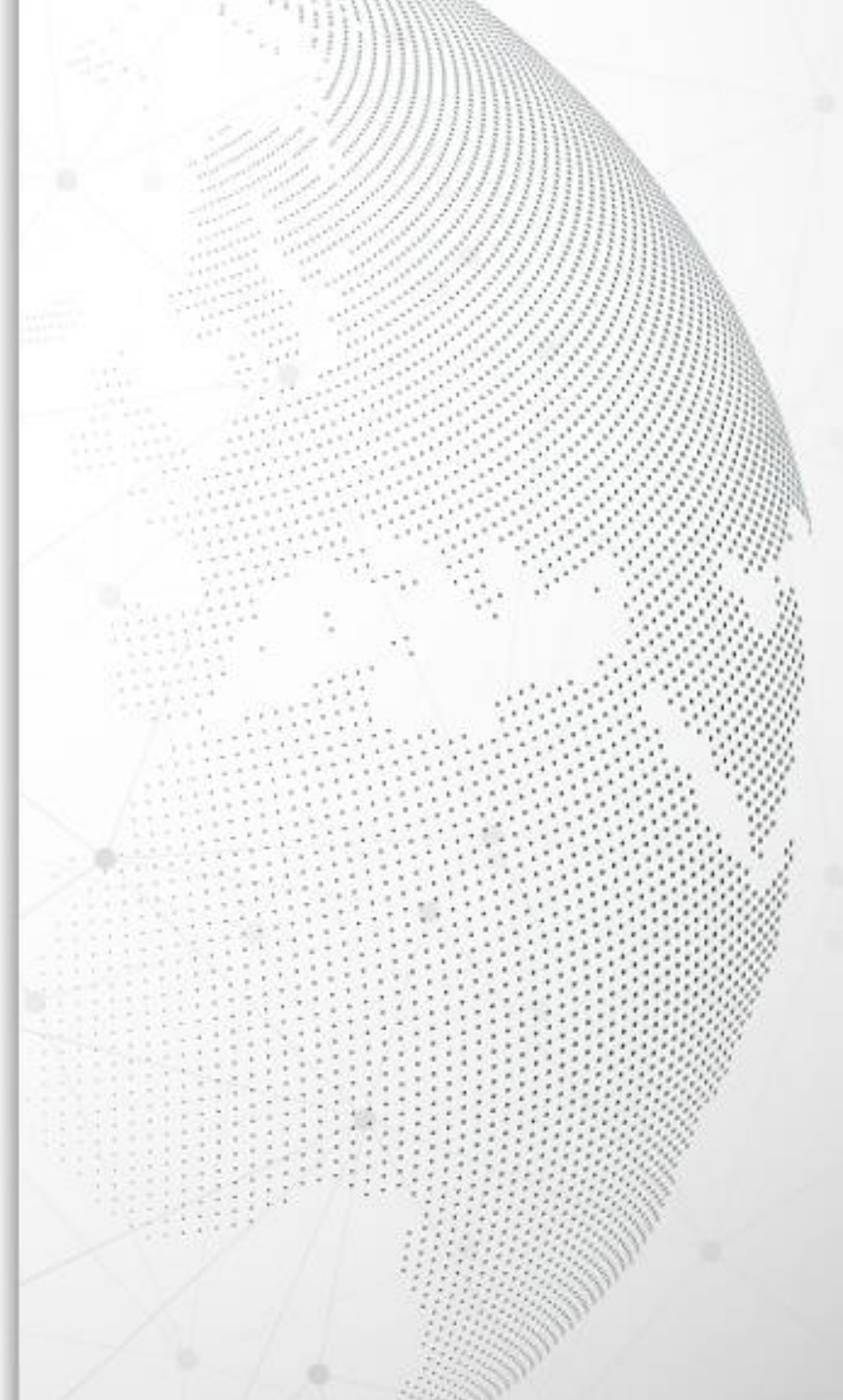
1. Object
2. Class
3. Abstraction
4. Encapsulation
5. Inheritance

Object

- A thing in a real world that can be either physical or conceptual. An object in object oriented programming can be physical or conceptual.
- Conceptual objects are entities that are not tangible in the way real-world physical objects are.

Bulb is a physical object. While college is a conceptual object.

- Conceptual objects may not have a real world equivalent. For instance, a Stack object in a program.
- Object has state and behavior.

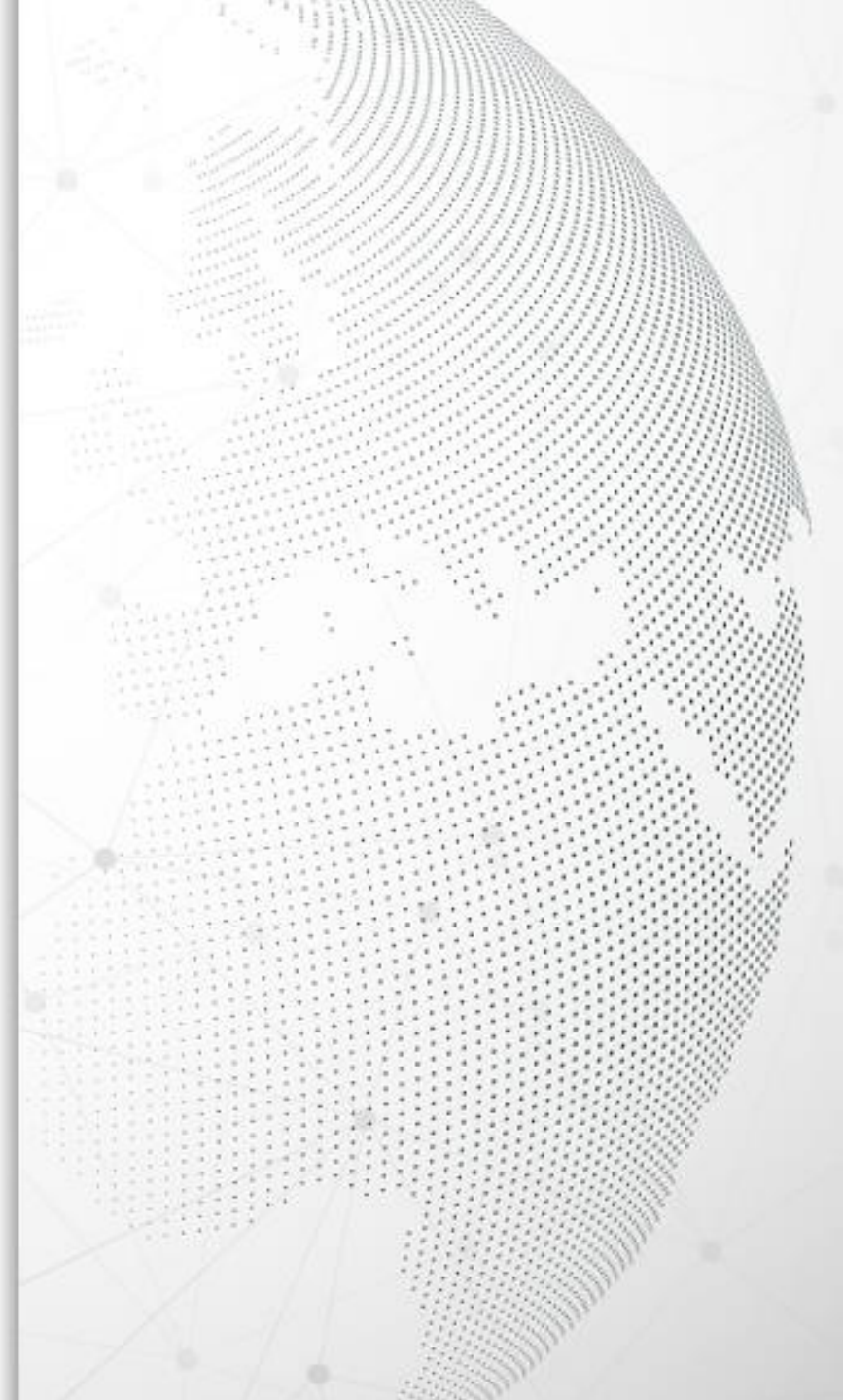


What is the state and behavior
of this bulb?



Objects: Attributes & Operations

- The object's state is determined by the value of its properties or attributes.
- Properties or attributes → member **variables** or data members
- The object's behavior is determined by the operations that it provides.
- Operations → member functions or **methods**



Example: A Light Bulb in Java Terms

OBJECT

It's a real-world thing.

METHOD

Can be switched on to generate light and switched off.

VARIABLES

It has real features like the glass covering, filament and holder.

It also has conceptual features like power.

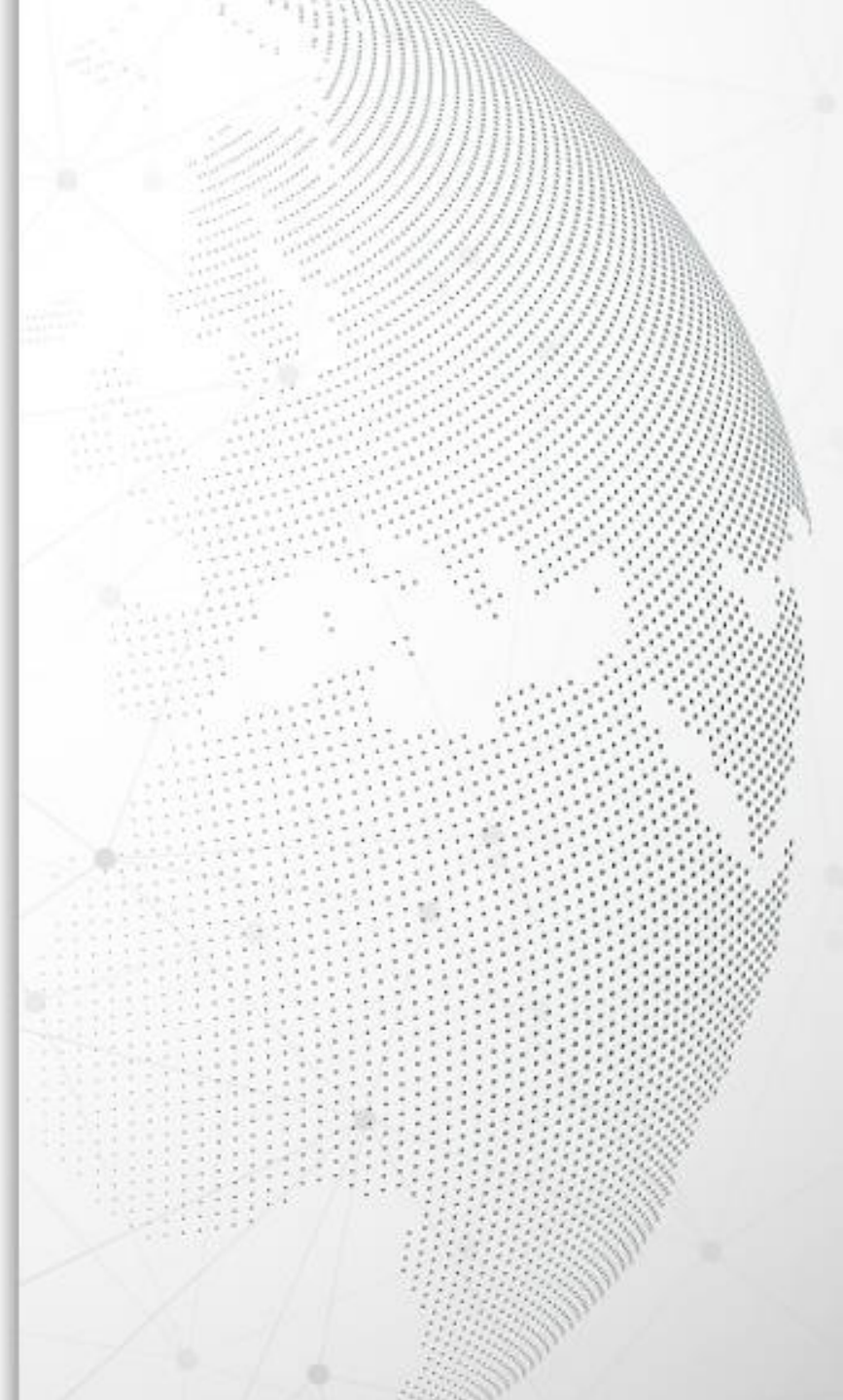
CLASS

A bulb manufacturing factory produces many bulbs based on a basic description or pattern of what a bulb is.



Class

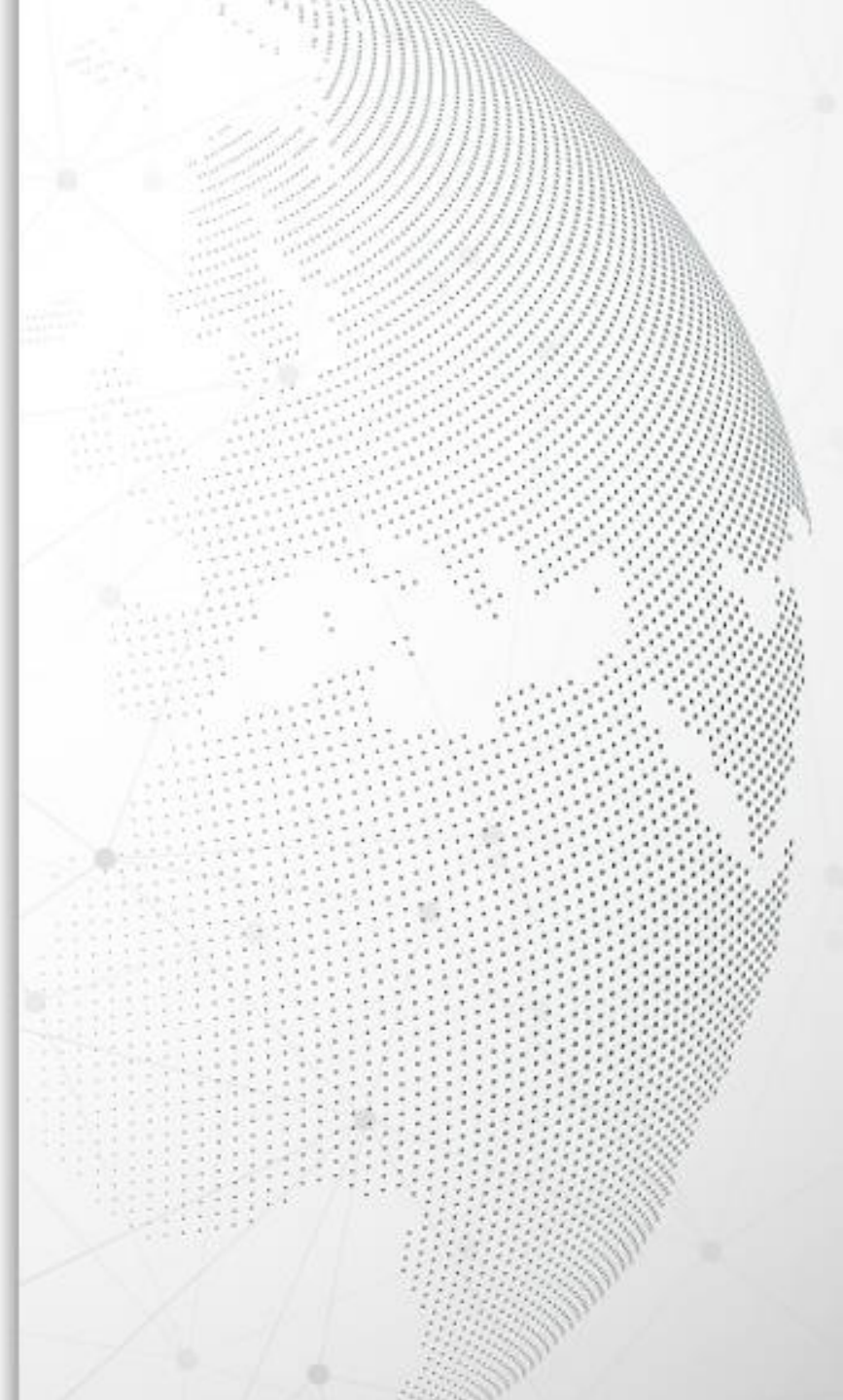
- A class is a construct created in object-oriented programming languages that enables creation of objects.
- Also sometimes called blueprint or template or prototype from which objects are created.
- It defines members (variables and methods).
- A class is an **abstraction**.



Abstraction

“Abstraction denotes essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.”

GRADY BOOCH



Abstraction

Abstraction is the process of taking only a set of essential characteristics from something.

Example:

For a Doctor → you are a Patient

Name, Age, Old medical records

For a Teacher → you are a Student

Name, Roll Number/RegNo, Education background

For HR Staff → you are _____

_____, _____, _____



Java Class

```
public class Student{  
    public int regno;  
    public String name;  
    public void display()  
    {  
        //display statements  
    }  
}
```

} Public Data Members

→ Public Member Function

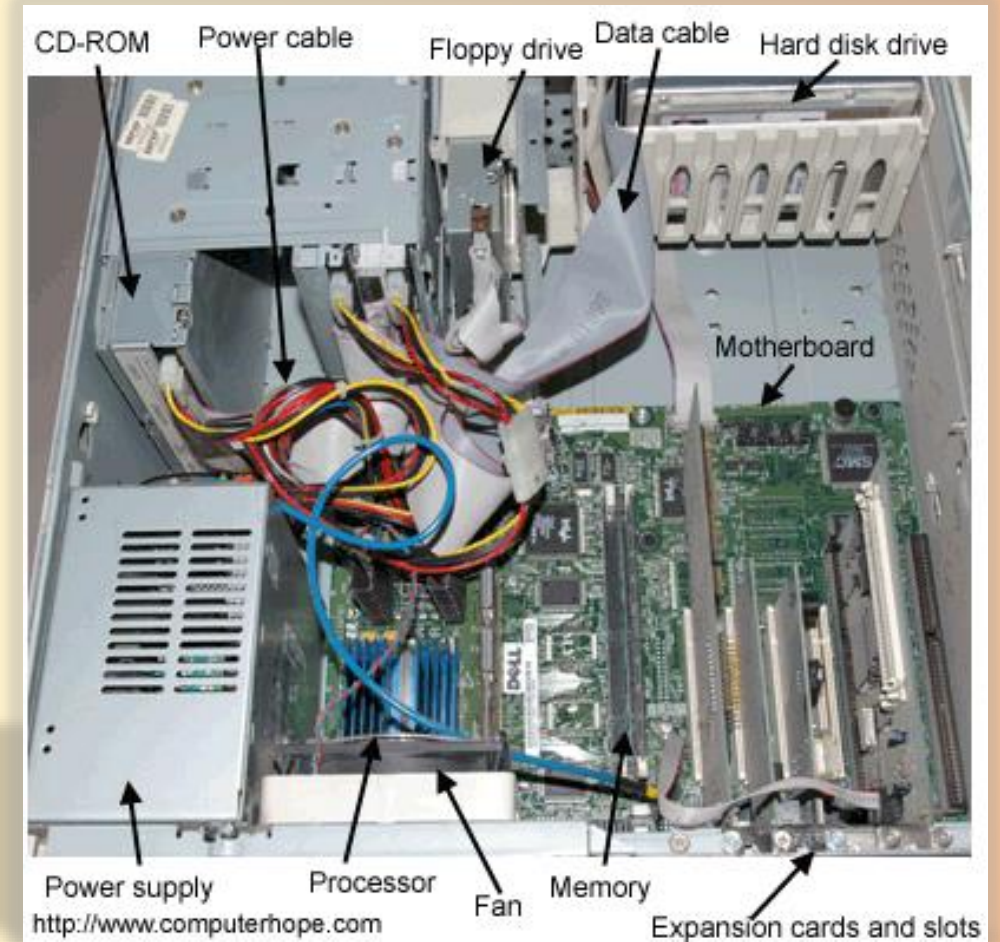
Creating Student Object:

```
Student s= new Student();  
s.display(); → access members using ‘.’
```

Encapsulation

Would you like it if your CPU is given to you like this?

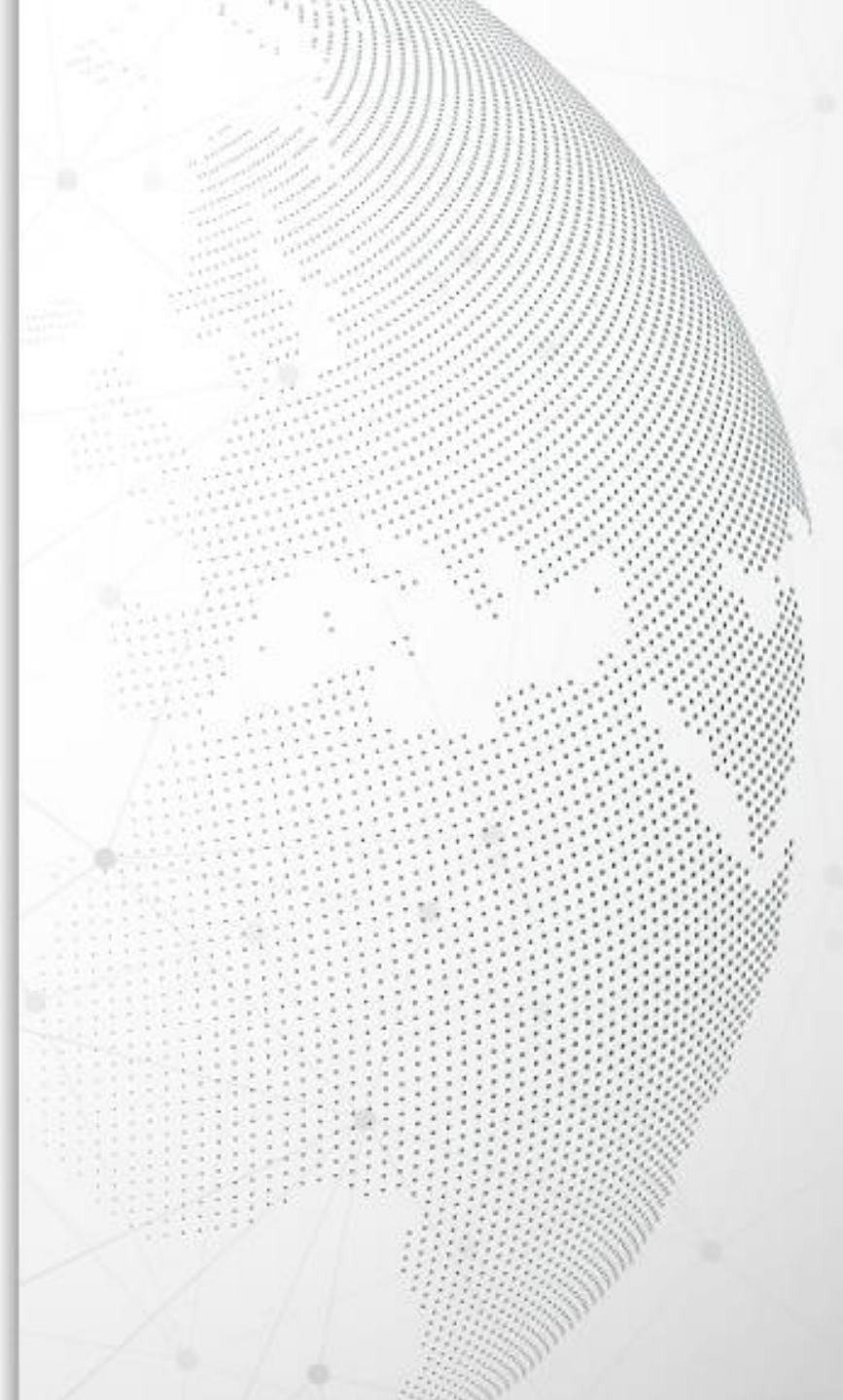
What are the problems if it were given to you like this?



Encapsulation

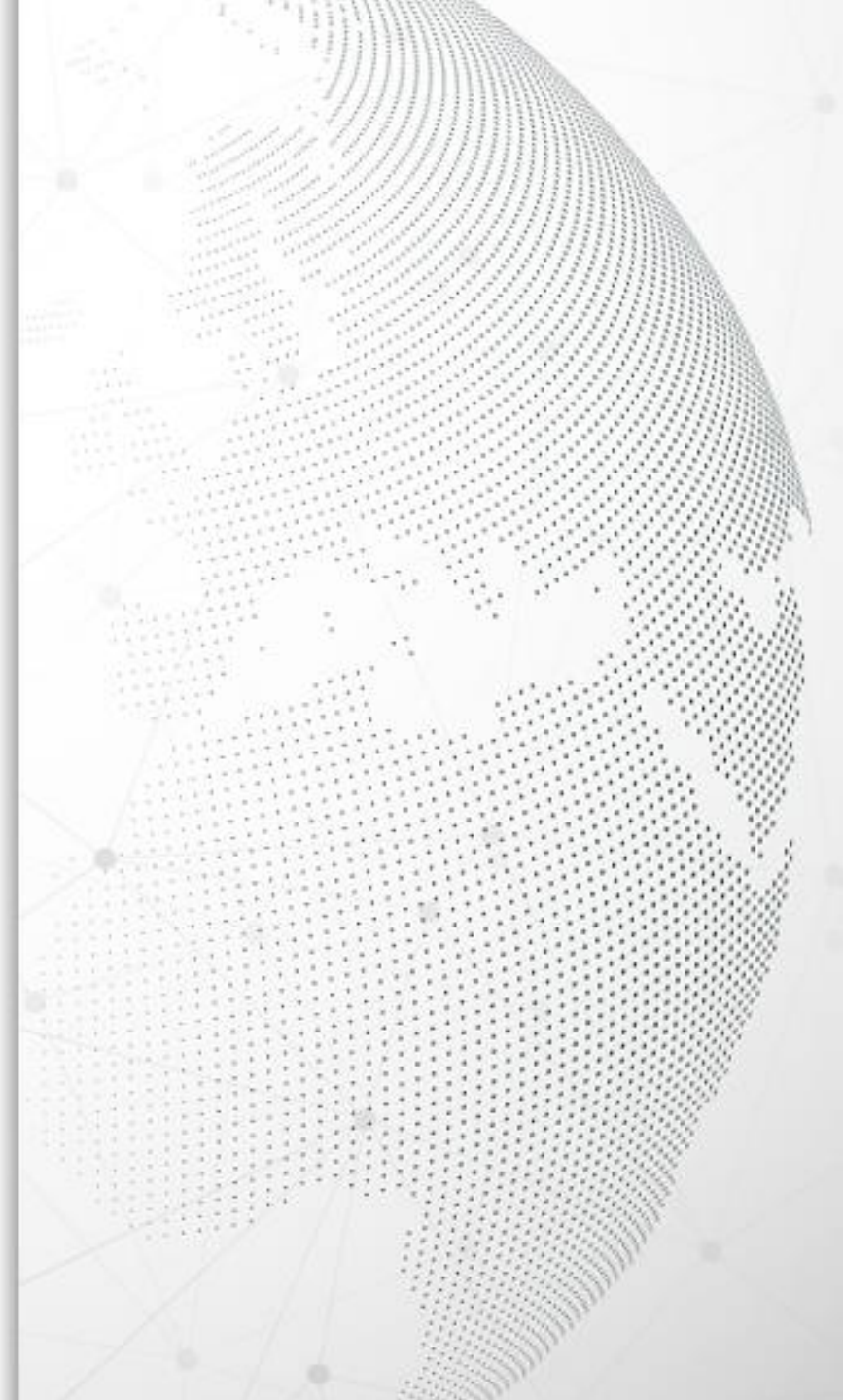
“Encapsulation is the process of compartmentalizing the elements of abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation.”

GRADY BOOCH



Encapsulation

- Encapsulation is binding data and operations that work on data together in a construct.
- Encapsulation involves Data and Implementation Hiding.





Data and Implementation Hiding in Java Classes

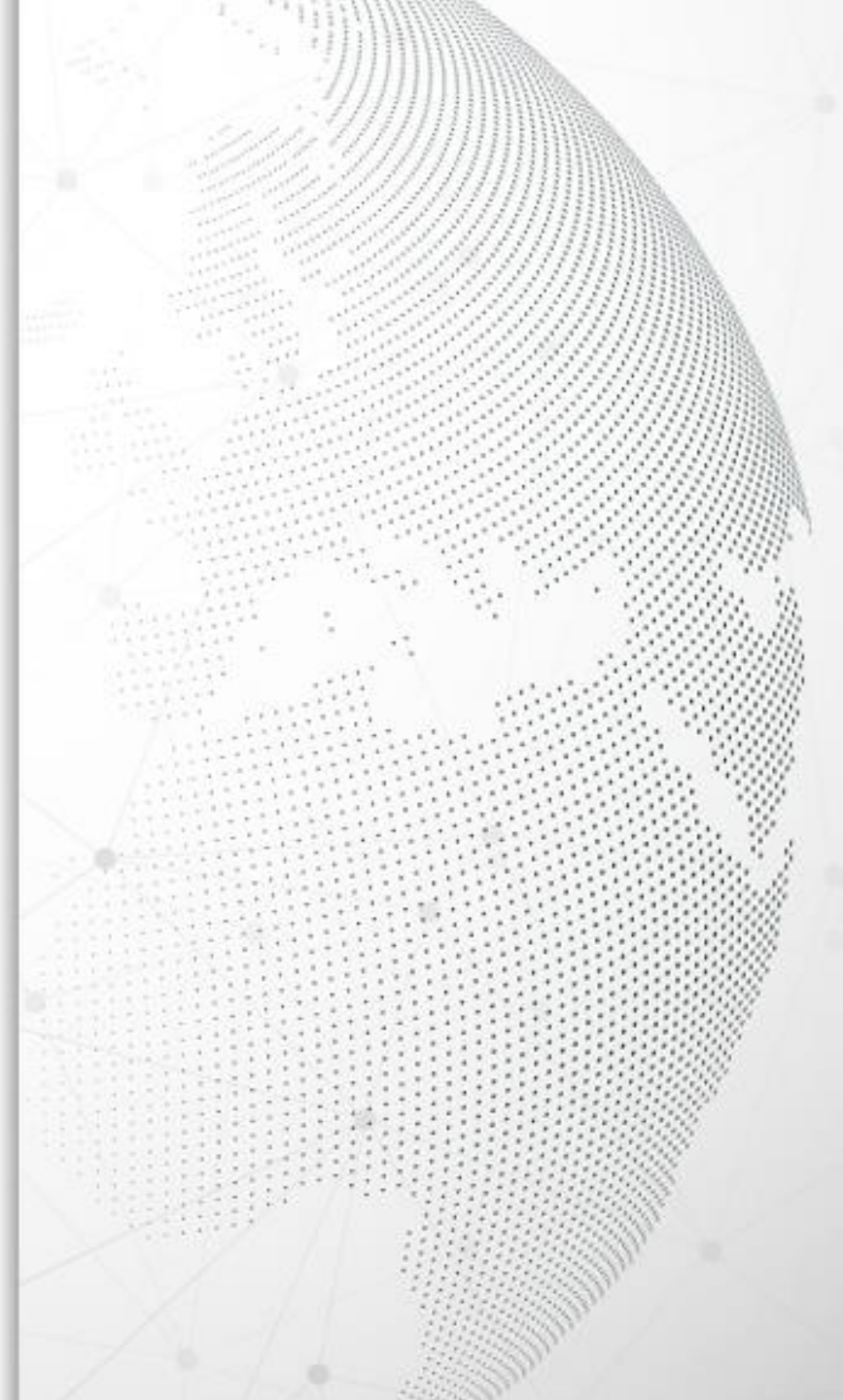
Java Supports Four Access Specifiers:

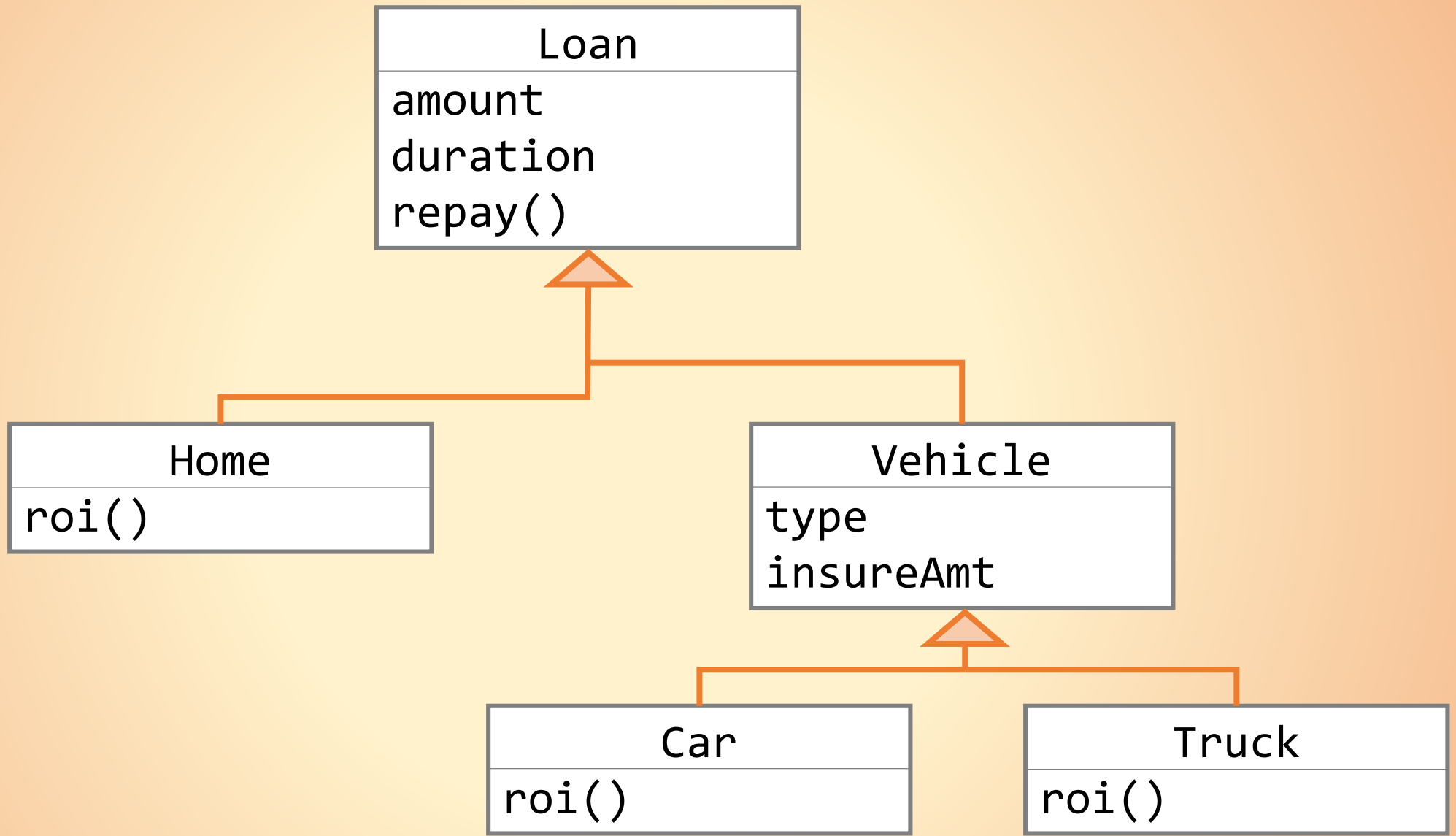
1. Public
2. Private
3. Default
4. Protected

Inheritance & Polymorphism

“Inheritance defines relationship among classes, wherein one class share structure or behavior defined in one or more classes.”

GRADY BOOCH





Portable & Platform Independent

Before we understand portability and platform independence, we need to understand a few concepts.

- Java Code can be compiled anywhere
 - Bytecode can be executed anywhere
- } Write Once / Run Anywhere

Now let's begin by writing a simple Java program!

Simple Hello World in Java

Hello.java

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

Save the file as Hello.java. A public class must be saved in the same name as class name.

Features

Portable

- Java source code can be compiled in any java-aware system. Also when java code executes, its behavior is exactly same in any java-aware system.
- There are no platform-specific code in java programs that causes compilation problems in any other OS. (For example in C if you include conio.h library, this will work only in Window OS and will not work in linux).
- Also some features that were considered not portable in C/C++ (like sizes of int, right shift operator behavior etc.) were eliminated for Java.
- So, Java programs are portable, which implies that they behave the same way when executed in any system and produce the same result.

But what is Platform Independence then? Let us compile our code first which will take us there.



Environment to Compile & Execute

Compile java programs

1. From command prompt
2. Through an IDE (Integrated development Environment)
 - Eclipse → Apache
 - NetBeans → Oracle SDN
 - JBuilder → Borland
 - Integrated Development Environment → IBM

But what is an IDE?



IDE: Integrated Development Environment

Integrated development environment (sometimes also called integrated debugging environment or interactive development environment or integrated design environment) is an GUI interface that allows programmers to build and test (and sometimes design) their application.

Typically IDE consists of:

- An editor where code can be written.
- A compiler: Some IDEs (like all of them listed in previous slide) compile as and when the code is written. In eclipse, redline underline is used to indicate compilation errors and yellow underline is used to indicate warnings.
- Run and Debug features
- Tools to create language specific components
- Context sensitive help (In eclipse, ctrl spacebar)
- Tools for auto-build (packaging a JEE application in eclipse)

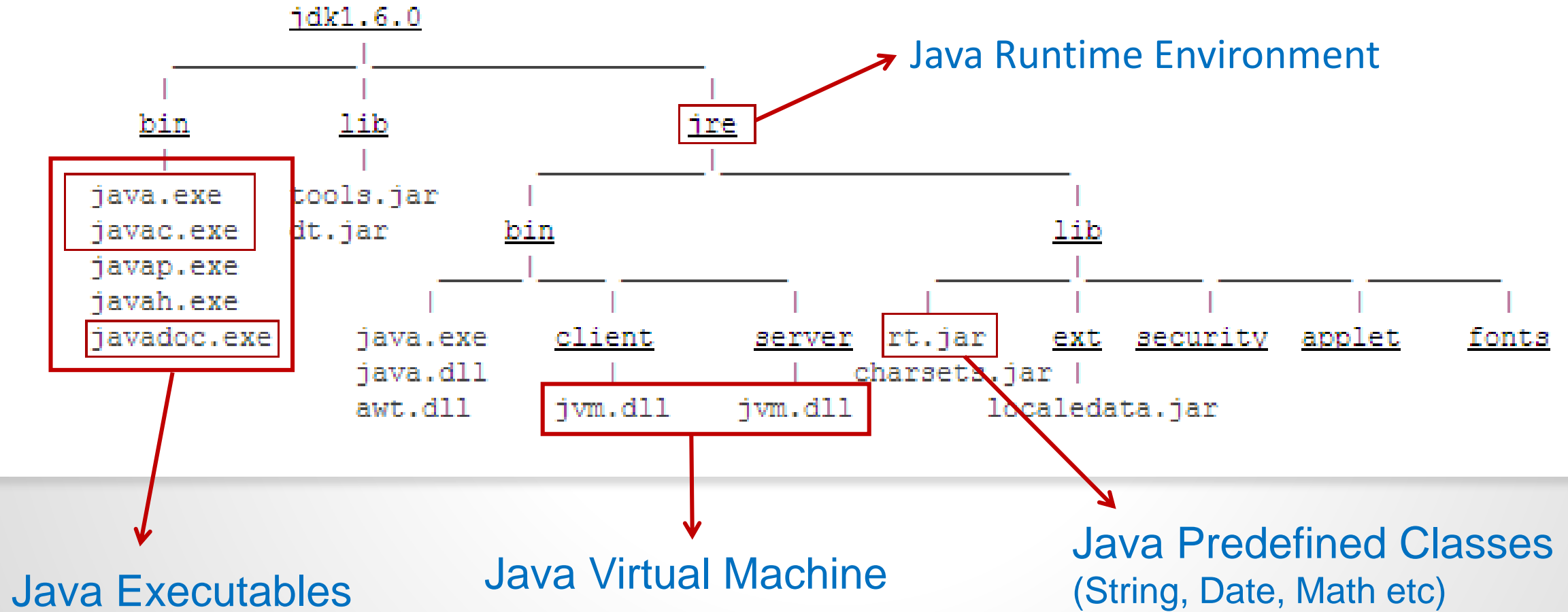


Setup Java Developer Kit (JDK)

<http://java.com/en/download/index.jsp> or find appropriate link in <http://www.oracle.com/technetwork/indexes/downloads>

- Download Java based on the type of OS
 - Windows
 - Linux
 - Mac OS
 - Solaris
- Install JDK

JDK Installation Directory



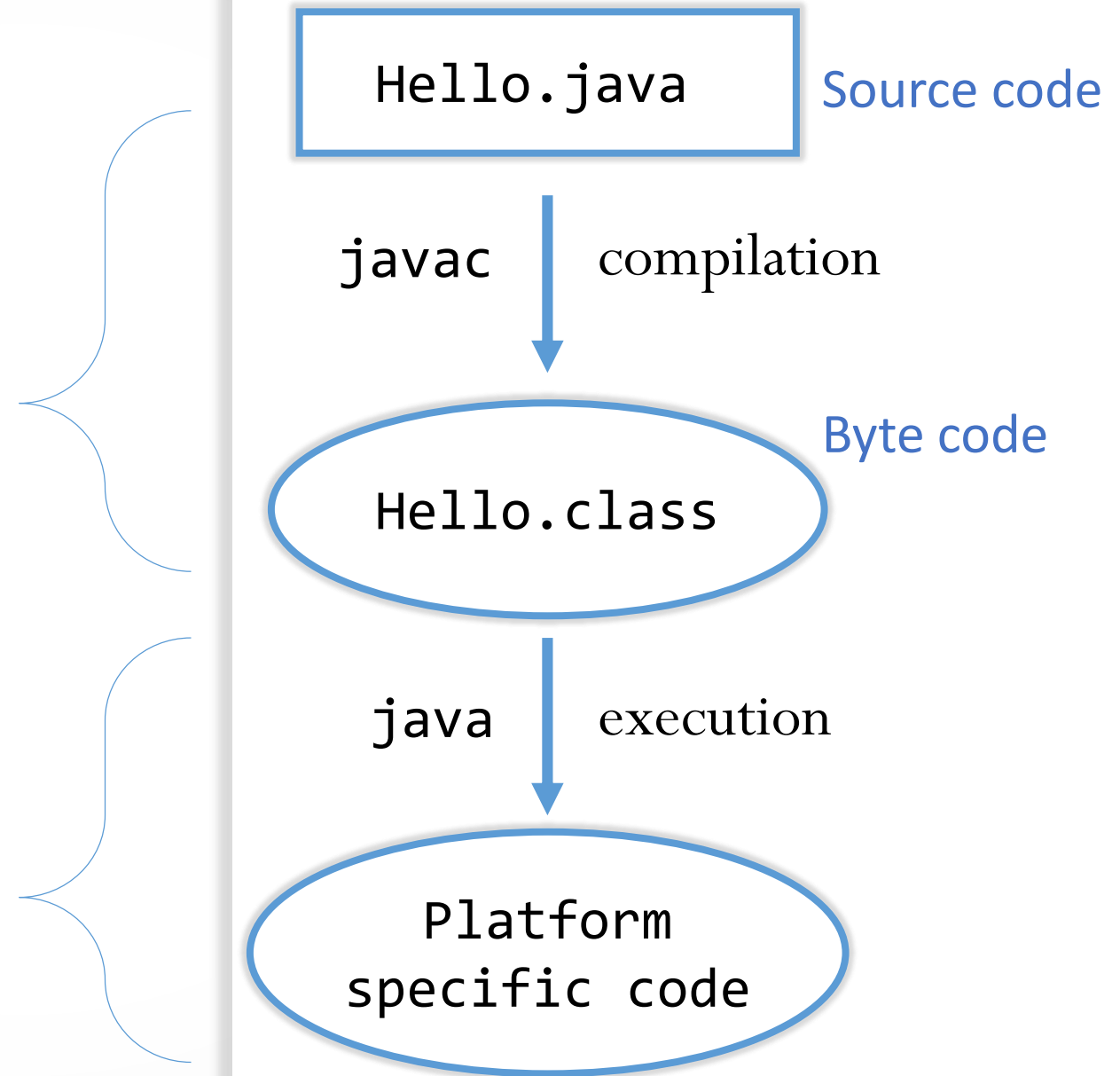
Compilation & Execution from command prompt

Compile:

```
C:\MyJava>javac Hello.java
```

Execute:

```
C:\MyJava>java Hello
```





About Eclipse 4.4 IDE

- IDE where Java programs can be written, compiled and executed.
- Open source
- JRE needs to be installed before eclipse is installed.
- Available for different OS(Windows, Mac OS, Linux, Solaris, IBM AIX, HP-UX)

Activity: Writing code in Eclipse IDE

- To start Eclipse double-click on the file "eclipse.exe".
- Enter the workspace (path) where you want store your java files.
- Close the Welcome screen(You can view that later).
- There are many perspective that Eclipse provides. By default Java Perspective is opened.
- Select from the menu File -> New-> Java project. Enter the project name as Java1. Select "Create separate source and output folders".
- Right click on src folder on the left under Java1 and select New -> Class
- Enter the name of the class as "Hello". Make sure "public static void main" is ticked. Click finish.
- Enter the print statement in the source code file that opens.



Feature

Platform Independent

- A Java program requires JVM (part of JRE) to execute Java code. When java application starts to executes, that **Java Virtual Machine** also starts.
- Bytecode has instructions that **Java Virtual Machine** can understand and execute.
- JVM converts the Bytecode to machine specific code.
- Java Bytecode can be copied on to any machine that has JVM and executed. This is what makes Java **Platform Independent**.
- **“Write Once, Run Anywhere”**

Is JVM platform independent?





Is JVM Platform Independent?

No, since JVM needs to convert the byte code to machine specific code, it is different for each machine or OS, since each OS has its own native language.

That is the reason why JDK/JRE is available for different platforms.

Sun Microsystems claims that there are over 4.5 billion JVM-enabled devices!

Automatic Garbage Collection

- The garbage collector is a tool that attempts to free unreferenced memory (memory occupied by objects that are no longer in use by the program) also called garbage, in program.
- **Automatic garbage collection** is an integral part of Java and its run-time system.
- Java technology has no pointers. So there is no question of allocation and freeing by programmers.
- Does that mean Java does not support dynamic memory allocation?
- No. It means that Java takes care of memory and relieves the programmers from memory-related hazards.
- `java -verbose:gc` can be used to get more information about garbage collection process.



Feature: Multi-threaded

- At the OS level, smallest unit of work that can be scheduled, is a thread.
- Thread is a sequence of execution of code.
- A process consists of one or more threads.
- Multiple threads in the same program share same resources.
- Unlike a multiple process, multiple threads in a process share same memory location. That is why threads are sometimes called Light weight process.
- Java Standard API has rich set of classes that allows us to work with multiple threads simultaneously



Feature: Security

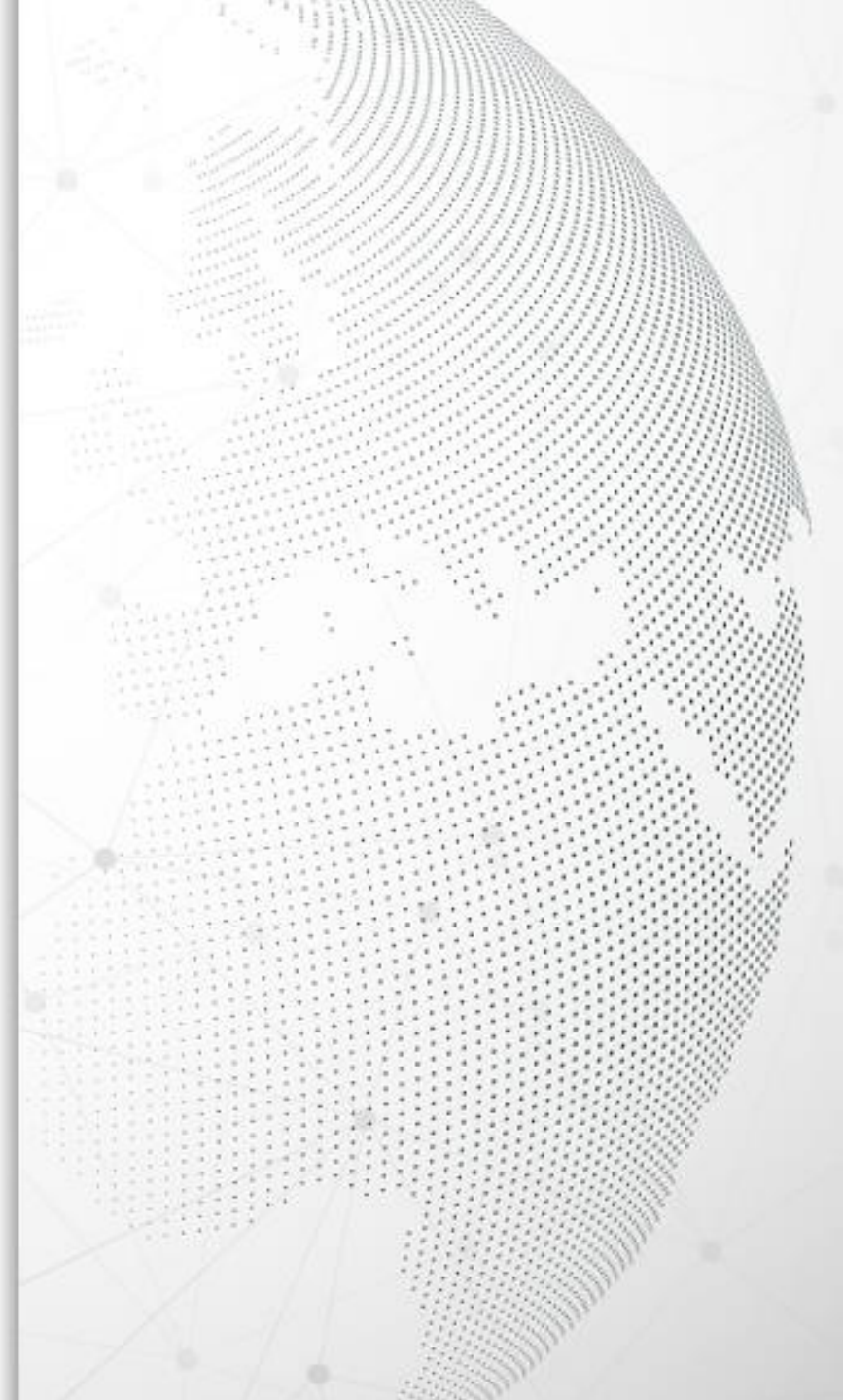
- Java's language rules
 - Has no syntax for pointers. So it is impossible to access illegal memory
 - Has extensive syntax (like private, protected) to secure data from being accessed by illegal objects
 - Comprehensive API with support for a wide range of cryptographic services and for Authentication and Access Control
 - Secure communication
- Java Compiler
 - Flags error on illegal conversions
 - Also makes sure that all java's language rules are adhered to

Interpreter vs. JIT

- Java Bytecodes were originally designed to be interpreted by JVM meaning bytecode are translated to machine code without it being stored anywhere.
- Since **bytecode verifier** (which is part of JVM) performs runtime checks, line by line execution was important.
- Since speed became an issue, Just-in-Time Compilation (JIT) came into being. JIT converts chunks of code, stores it temporarily in memory and then executes the converted code.
- JIT compilers are typically bundled with or are a part of a virtual machine and do the conversion to native code at runtime, on demand.
- The compiler also does automatic register allocation and some optimization when it produces the bytecodes.
- Therefore, JIT is hybrid compiler.

Summary of Features in Java

- Simple
- Object oriented language
- Portable and platform independent
- Robust
- Multithreaded
- Dynamic Linking
- Secure
- Performance



Flavors of Java

JSE

- Java Standard Edition formerly known as J2SE.
- This forms the core part of Java language.

JEE

- Java Enterprise Edition formerly known as J2EE.
- These are the set of packages that are used to develop distributed enterprise-scale applications.
- These applications are deployed on JEE application servers.

Our
Focus

JME

- Java Micro Edition formerly known as J2ME.
- These are the set of packages is used to develop application for mobile devices and embedded systems.