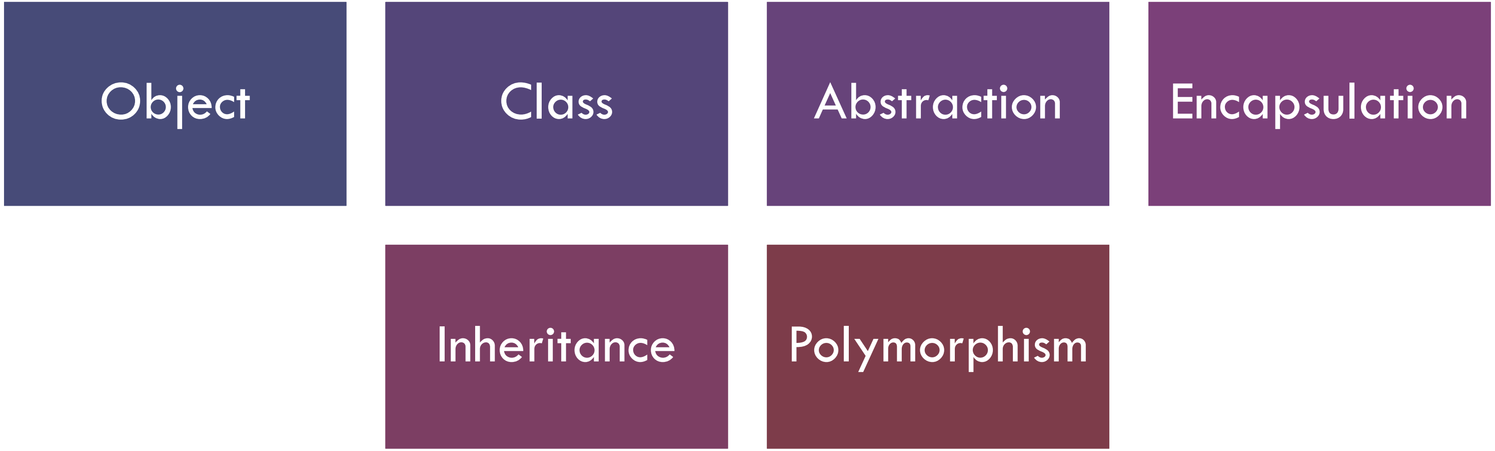
OBJECT ORIENTED PROGRAMMING USING JAVA

Pillars of OOP



Object

Class

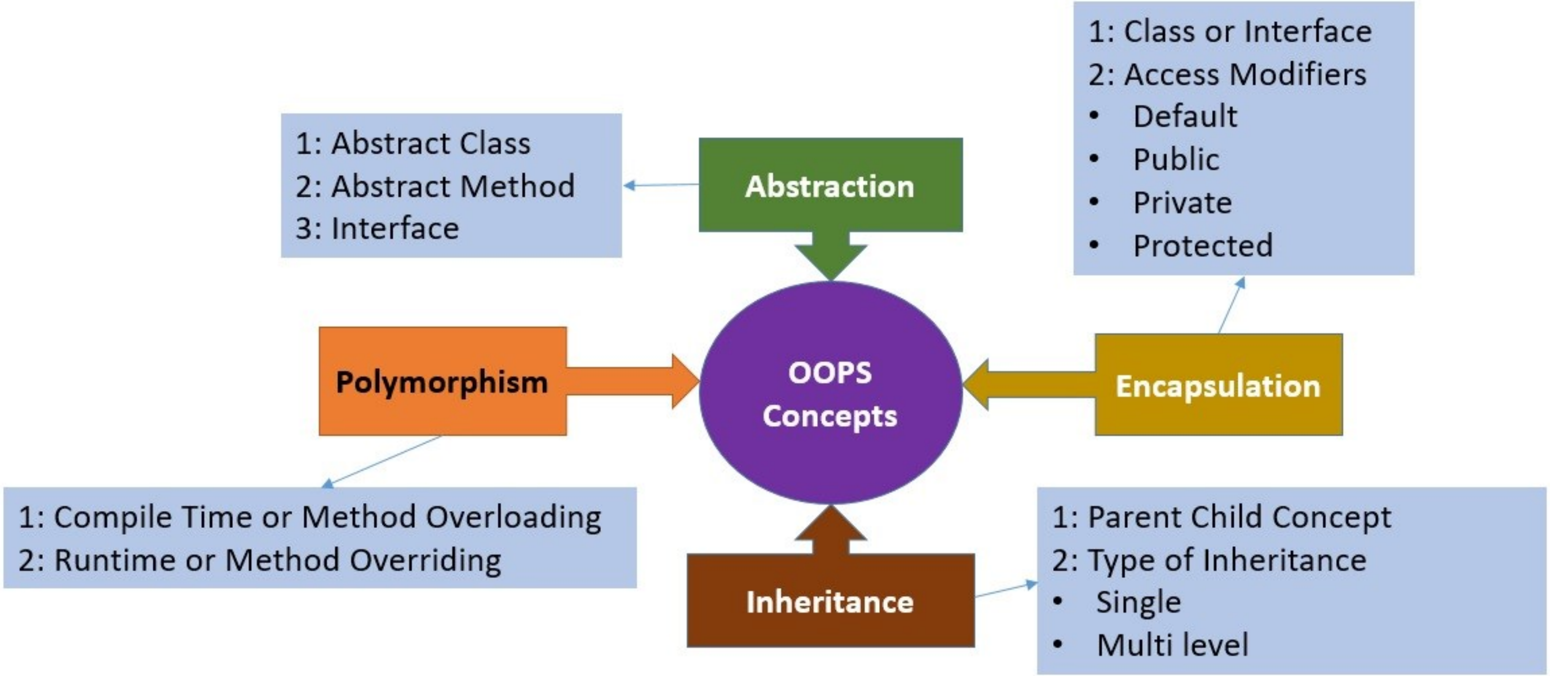
Abstraction

Encapsulation

Inheritance

Polymorphism

Pillars of OOP



OOPS Concepts

Abstraction

1: Abstract Class
2: Abstract Method
3: Interface

Encapsulation

1: Class or Interface
2: Access Modifiers
- Default
- Public
- Private
- Protected

Polymorphism

1: Compile Time or Method Overloading
2: Runtime or Method Overriding

Inheritance

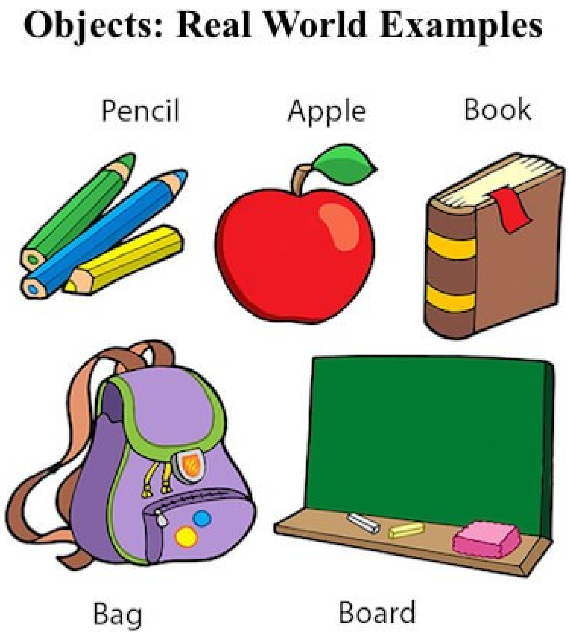
1: Parent Child Concept
2: Type of Inheritance
- Single
- Multi level

Object

- A physical or real-world entity that has state and behavior are known as an object.
- An **Object** is an instance of a class.
- It contains an address and takes some space in heap memory.
- The object is an entity that has state and behavior

Object (Real-world examples)

- Dogs have
  - state (name, color, breed, hungry) and
  - behavior (barking, fetching, wagging tail).
- Bicycles also have
  - state (current gear, current pedal cadence, current speed) and
  - behavior (changing gear, changing pedal cadence applying brakes).
- Chair, Bike, Marker, Pen, Table, Car, Book, Apple, Bag etc It can be physical or logical (tangible), Bank (intangible).



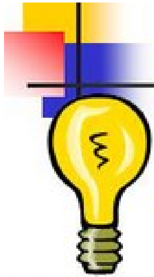
Object (Real-world examples)

BankAccount

- **state/attributes**
  - balance
- **behavior**
  - deposit
  - withdraw
  - check balance

Car

- **state/attributes**
  - # of liters of gas in tank
  - total # of km run so far
  - efficiency (km/liter)
- **behavior**
  - drive
  - load gas
  - change efficiency
  - check gas
  - check odometer reading

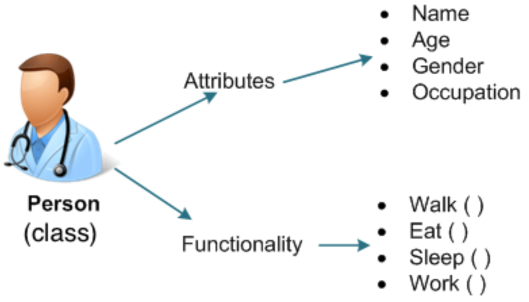


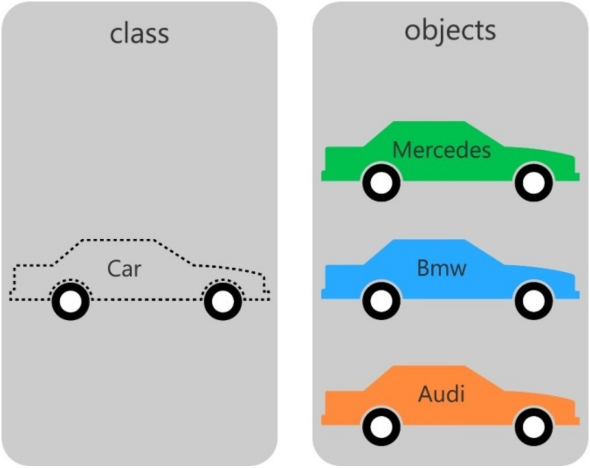
LightBulb

- **state/attributes**
  - on (true or false)
- **behavior**
  - switch on
  - switch off
  - check if on

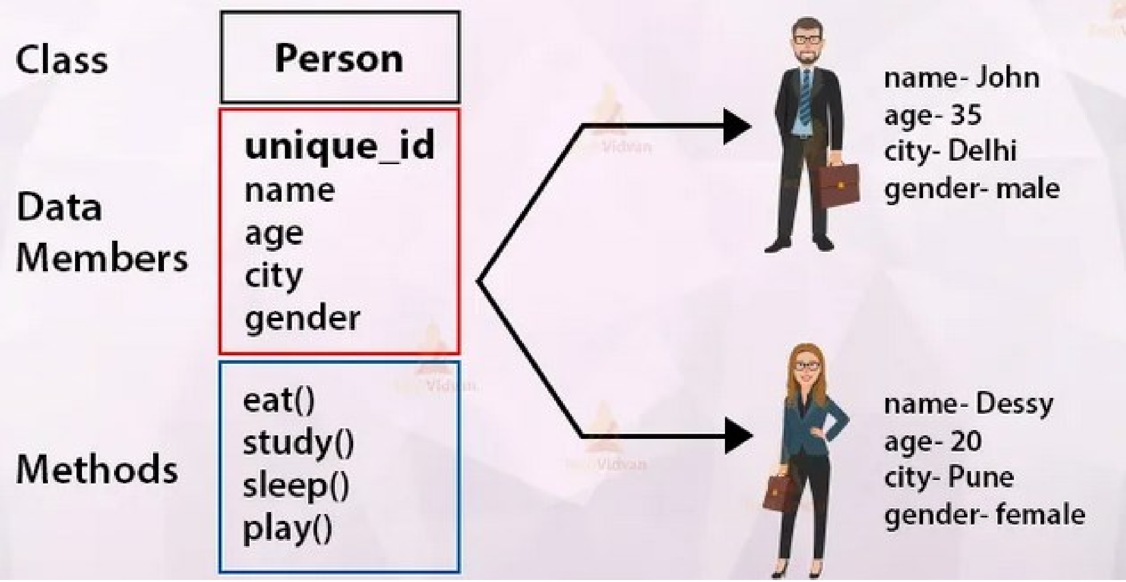
Class

- A class is a group of objects which have common properties.
- It is a template or blueprint from which objects are created.
- A class is the specification or template of an object.
- The collection of objects is called a class.
- Class doesn't consume space.
- For Example, Person is class and "ABC" is the object of that class.

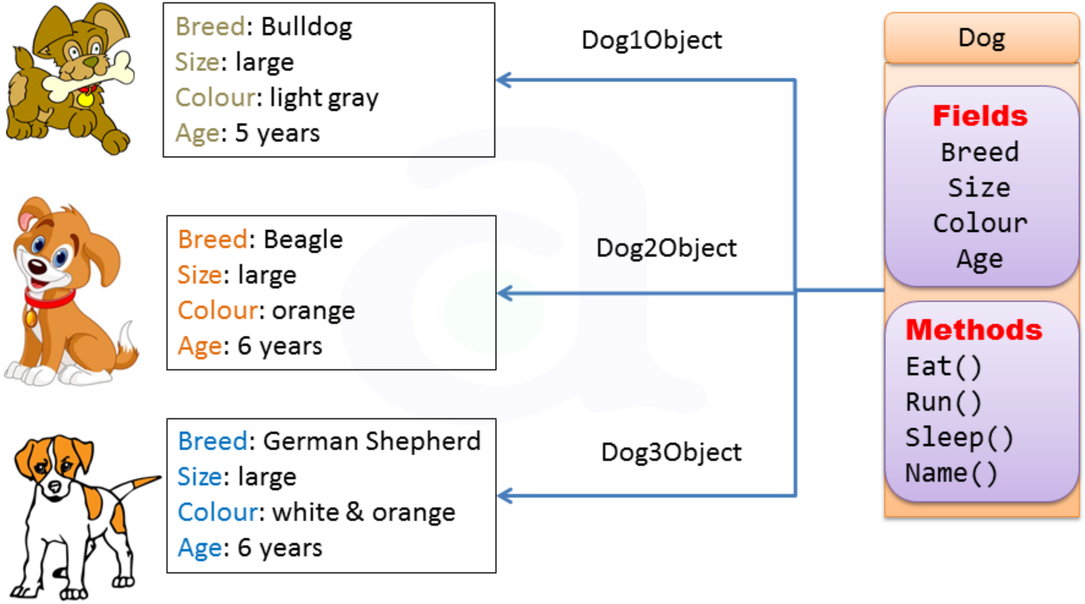




Understanding class & Object



Understanding class & Object



Breed: Bulldog
Size: large
Colour: light gray
Age: 5 years

Breed: Beagle
Size: large
Colour: orange
Age: 6 years

Breed: German Shepherd
Size: large
Colour: white & orange
Age: 6 years

Dog1Object

Dog2Object

Dog3Object

Dog

Fields
Breed
Size
Colour
Age

Methods
Eat()
Run()
Sleep()
Name()

Abstraction

- Abstraction means hiding lower-level details and exposing only the essential and relevant details to the users.
- Real-world examples
- **Example 1:** Let's consider a Car, which abstracts the internal details and exposes to the driver only those details that are relevant to the interaction of the driver with the Car.

Abstraction

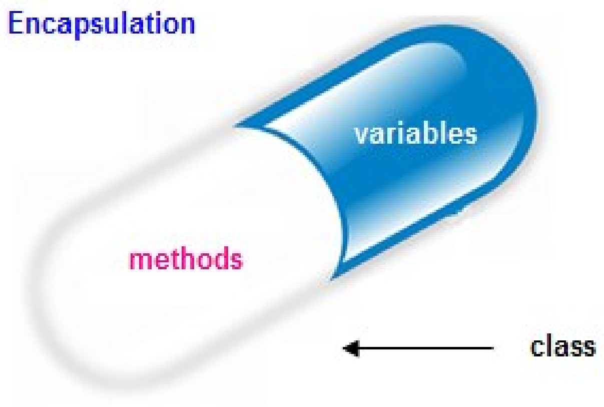
- **Example 2:** Consider an **ATM** Machine; All are performing operations on the ATM machine like cash withdrawal, money transfer, retrieve mini-statement…etc. but we can't know the internal details about ATM.

Real Life Example of Abstraction

Abstraction

Encapsulation

- □ It is a process of wrapping of data and methods in a single unit is called encapsulation.
- □ In OOP, data and methods operating on that data are combined together to form a single unit, which is referred to as a Class.
- □ **Real-world example :**
- □ **Capsule**, it is wrapped with different medicines. In a capsule, all medicine is encapsulated inside a capsule.



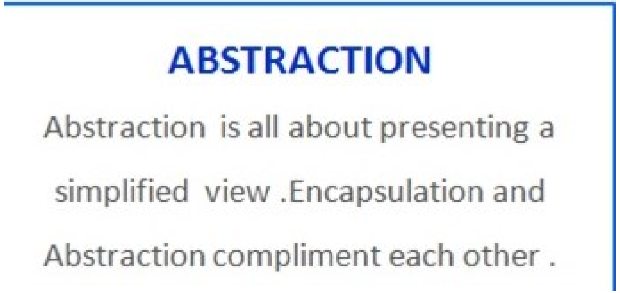
Difference between encapsulation & Abstraction

Difference between Abstraction and Encapsulation

Encapsulation: Information hiding.

Abstraction: Implementation hiding.

Encapsulation
Hiding Unnecessary
Synergy
Abstraction
Showing necessary



ABSTRACTION

Abstraction is all about presenting a simplified view .Encapsulation and Abstraction compliment each other .

Inheritance

- It is a process of obtaining the data members and methods from one class to another class, plus can have its own is known as inheritance.
- Inheritance - IS-A relationship between a superclass and its subclasses.

Inheritance

- Super Class: The class whose features are inherited is known as a superclass (or a base class or a parent class).
- Sub Class: The class that inherits the other class is known as a subclass(or a derived class, extended class, or child class).
- The subclass can add its own fields and methods in addition to the superclass fields and methods.

Inheritance (Real-World Example)

- □ **Real-world example**
- □ The real-life example of inheritance is child and parents, all the properties of a father are inherited by his son

Polymorphism

□ The process of representing one form in multiple forms is known as **Polymorphism**.

Different definitions of Polymorphism are:

□ Polymorphism let us perform a single action in different ways.

□ Polymorphism allows you to define one interface and have multiple implementations

□ We can create functions or reference variables that behave differently in a different programmatic context.

□ Polymorphism means many forms.

Polymorphism (Real-World Example)

In Shopping malls behave like Customer

In Bus behave like Passenger

In School behave like Student

At Home behave like Son

Polymorphism (Real-World Example)

Polymorphism (Real-World Example)

Transform Form of Ben ten
Ben ten
Tutorial4us

Advantage of OOP

Advantage of OOP

- ✓Security - Encapsulation
- ✓Reusability - Inheritance
- ✓Effective communication – Message Passing
- ✓Developing complex software - Inheritance

Advantage of OOP

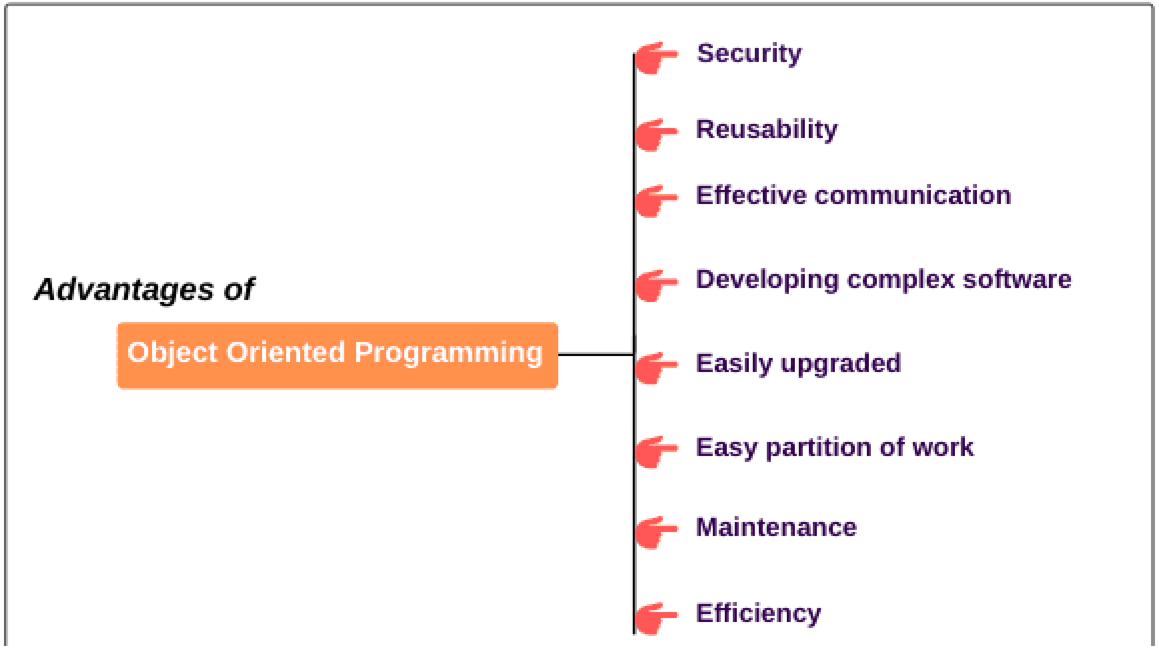
✓Easily upgraded – Bottom-up approach

✓Easy partition of work – Objects

✓Maintenance – Code is Easier

✓Efficiency - Better Efficiency and Easy Development Process.

Advantage of OOP



Advantages of

Object Oriented Programming

- Security
- Reusability
- Effective communication
- Developing complex software
- Easily upgraded
- Easy partition of work
- Maintenance
- Efficiency

Application of OOP

Applications of OOP

- Real-time systems
- Object-oriented database
- Graphical user interface design in the Windows operating system.
- Artificial intelligence and expert systems
- Parallel programming
- CAD/CAM software and in many areas.
- Simulation and Modelling
- Decision support and office automation system

2. JAVA Evolution

What is Java?

- □ Java is a **programming language** and a **platform**.
- □ Java is a high level, robust, object-oriented and secure programming language.
- □ Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995.
- □ *James Gosling* is known as the father of Java.
- □ Before Java, its name was *Oak*. Since Oak was already a registered company, so James Gosling and his team changed the Oak name to Java.

2. JAVA Evolution

What is Java?

□ **Platform**: Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform. Games, etc.

2. JAVA Evolution

Application

There are many devices where Java is currently used. Some of them are as follows:

- **Desktop Applications** such as acrobat reader, media player, antivirus, etc.
- **Web Applications** such as irctc.co.in, javatpoint.com, etc.
- **Enterprise Applications** such as banking applications.
- Mobile
- Embedded System
- Smart Card
- Robotics
- Games, etc.

2. JAVA Evolution

Types of Java Applications

- There are mainly 4 types of applications that can be created using Java programming:

1) Standalone Application

- Standalone applications are also known as desktop applications or window-based applications.
- Examples of standalone application are Media player, antivirus, etc. AWT and Swing are used in Java for creating standalone applications.

2. JAVA Evolution

Types of Java Applications

2) Web Application

□ An application that runs on the server side and creates a dynamic page is called a web application.

□ Currently, Servlet, JSP, Struts, Spring, Hibernate, JSF, etc. technologies are used for creating web applications in Java.

2. JAVA Evolution

Types of Java Applications

3) Enterprise Application

- □ An application that is distributed in nature, such as banking applications, etc. is called enterprise application. It has advantages of the high-level security, load balancing, and clustering. In Java, EJB is used for creating enterprise applications.

4) Mobile Application

- □ An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

2. JAVA Evolution

Java Platforms / Editions

- □ There are 4 platforms or editions of Java:

1) Java SE (Java Standard Edition)

- □ It is a Java programming platform.
- □ It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math etc. It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

2. JAVA Evolution

Java Platforms / Editions

2) Java EE (Java Enterprise Edition)

□ It is an enterprise platform which is mainly used to develop web and enterprise applications. It is built on the top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA, etc.

3) Java ME (Java Micro Edition)

□ It is a micro platform which is mainly used to develop mobile applications.

4) JavaFX

□ It is used to develop rich internet applications. It uses a light-weight user interface API.

2. JAVA Evolution

History of Java

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

2. JAVA Evolution

History of Java

- 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 the father of Java, in 1995. James Gosling and his team members started the project in the early '90s.

2. JAVA Evolution

History of Java

- □ Currently, Java is used in internet programming, mobile devices, games, e-business solutions, etc. There are given significant points that describe the history of Java.

1) **James Gosling** , **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.

2) Initially designed for small, embedded systems in electronic appliances like set-top boxes.

3) Firstly, it was called **"Greentalk"** by James Gosling, and the file extension was .gt.

4) After that, it was called **Oak** and was developed as a part of the Green project.

2. JAVA Evolution

History of Java

Why Java named "Oak"?

5) **Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.

6) In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.

7) **Why had they chosen java name for Java language?** The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA", etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell and fun to say.

- According to James Gosling, "Java was one of the top choices along with **Silk**". Since Java was so unique, most of the team members preferred Java than other names.

2. JAVA Evolution

History of Java

8) Java is an island of Indonesia where the first coffee was produced (called java coffee). It is a kind of espresso bean. Java name was chosen by James Gosling while having coffee near his office.

9) Notice that Java is just a name, not an acronym.

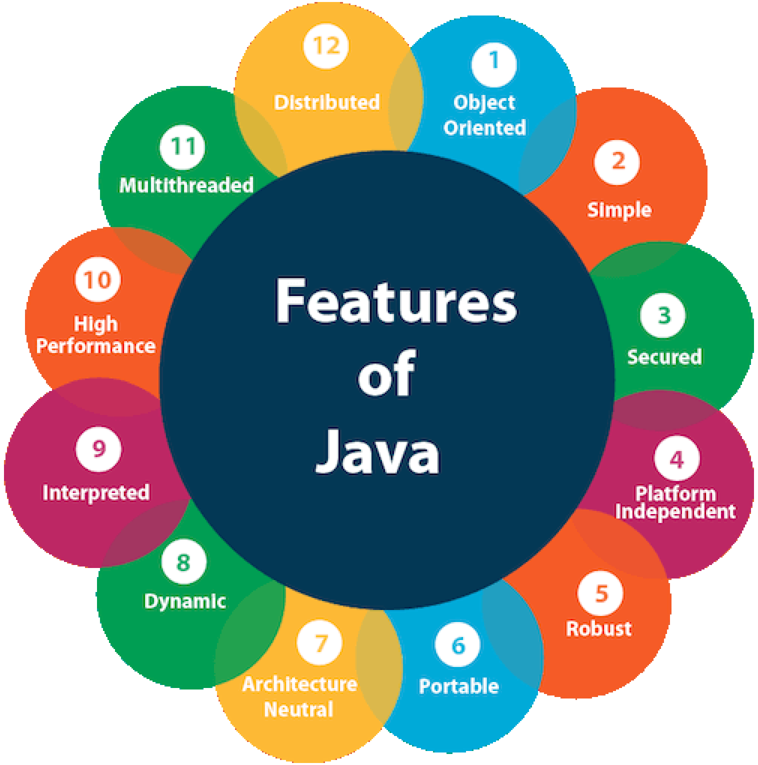
10) Initially developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.

11) In 1995, Time magazine called **Java one of the Ten Best Products of 1995**.

12) JDK 1.0 released in(January 23, 1996). After the first release of Java, there have been many additional features added to the language. Now Java is being used in Windows applications, Web applications, enterprise applications, mobile applications, cards, etc. Each new version adds the new features in Java.

2. JAVA Evolution

Features of Java



2. JAVA Evolution

Features of Java

1. Object-oriented

- It is an object-oriented programming language. Everything in Java is an object.
- Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.
- Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.
- Basic concepts of OOPs are:
- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

2. JAVA Evolution

Features of Java

2. Simple

- Java is very easy to learn, and its syntax is simple, clean and easy to understand.
- Java syntax is based on C++ (so easier for programmers to learn it after C++).
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

2. JAVA Evolution

Features of Java

3. Secured

- Develop virus-free systems. Java is secured because:
- No explicit pointer
- Java Programs run inside a virtual machine sandbox
- Classloader
- Bytecode Verifier
- Security Manager
- Java language provides these securities by default. Some security can also be provided by an application developer explicitly through SSL, JAAS, Cryptography, etc.

2. JAVA Evolution

Features of Java

4. Platform Independent

- Like C,C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.
- There are two types of platforms
  - Software-based
  - Hardware-based
  - Java provides a software-based platform.

2. JAVA Evolution

Features of Java

4. Platform Independent

- It has two components:
  - Runtime Environment
  - API(Application Programming Interface)
- Java code can be run on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc.
- Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere(WORA).

2. JAVA Evolution

Features of Java

5. Robust

- Robust simply means strong. Java is robust because:
- It uses strong memory management.
- There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

2. JAVA Evolution

Features of Java

6. Portable

- Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

7. Architecture-neutral

- Java is architecture neutral because there are no implementation dependent features.
- In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

2. JAVA Evolution

Features of Java

8. 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).

9. Interpreted

- □ Java enables the creation of cross-platform(architecture-neutral) programs by compiling into an intermediate representation called Java bytecode.
- □ Java bytecode can run on a variety of hardware platforms and operating systems. Java source code is compiled into Java bytecode and Java bytecode is interpreted by the JVM (Java Virtual Machine).

2. JAVA Evolution

Features of Java

10. High-performance

□ Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

11. Multi-threaded

□ A thread is like a separate program, executing concurrently.

□ The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area.

□ Threads are important for multi-media, Web applications, etc.

2. JAVA Evolution

Features of Java

12. Distributed

- 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.

3. Overview of Java Language

What is JVM (Java Virtual Machine)?

- □ The Java language is a high-level language.
- □ All language compilers translate source code into machine code.
- □ You save a Java program in a .java file and compile it into a .class file.
- □ The .class file (also called **bytecode**). Output of a Java compiler is not executable code. Rather, it is bytecode.

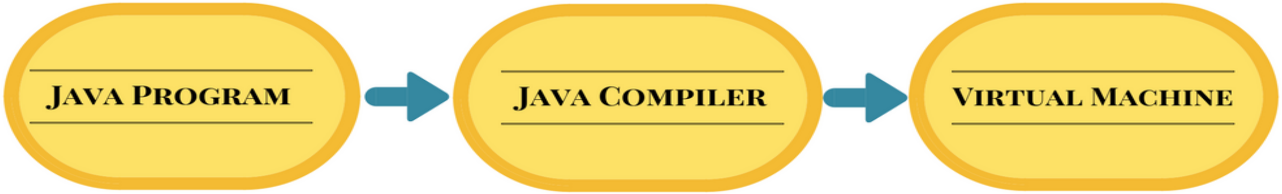
3. Overview of Java Language

What is JVM (Java Virtual Machine)?

- □ Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM).
- □ The bytecode is similar to machine instructions but is architecture neutral and can run on any platform that has a Java Virtual Machine (JVM).The JVM executes our code along with the code in the library.

3. Overview of Java Language

What is JVM (Java Virtual Machine)?



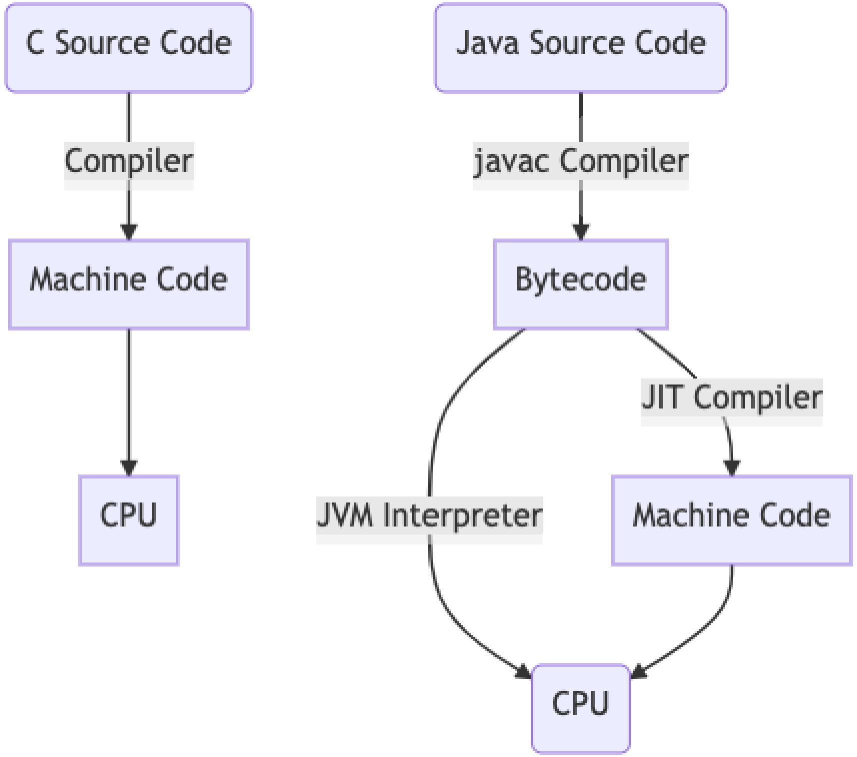
Process of Compilation

3. Overview of Java Language

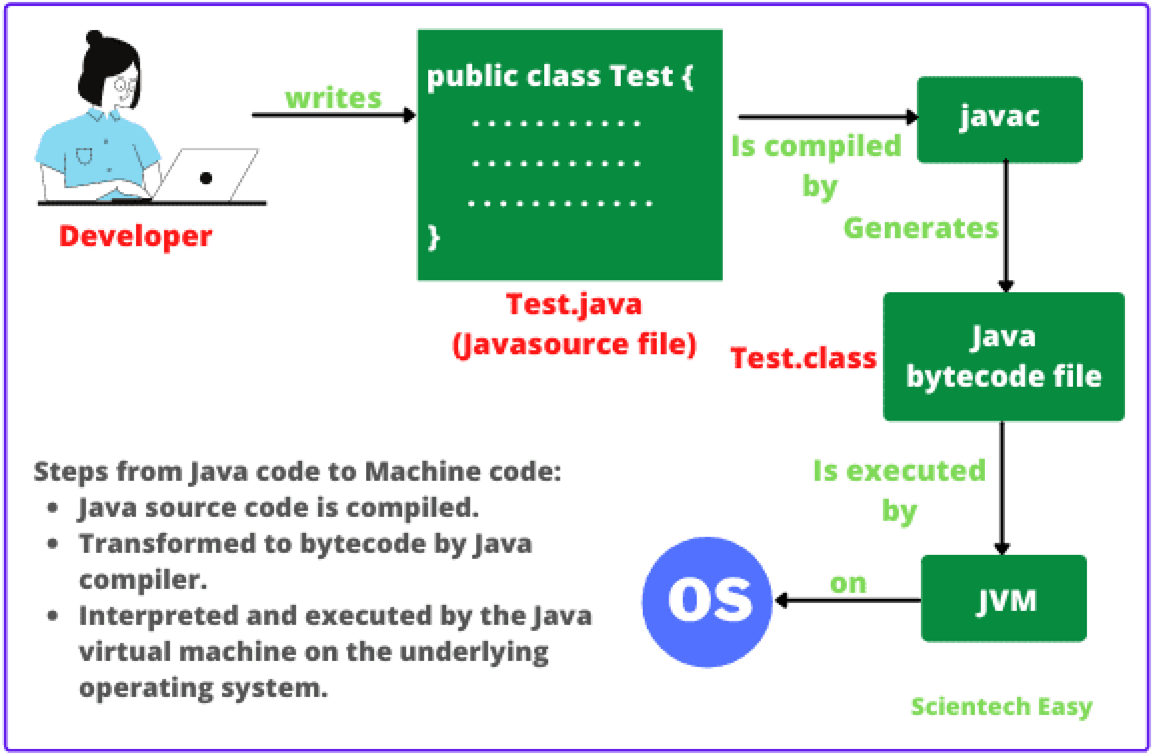
What is JVM (Java Virtual Machine)?

- □ To execute a Java program is to run the program’s bytecode or .class file. You can execute the bytecode on any platform with a JVM, which is an **interpreter** .
- □ The original JVM was designed as an interpreter for bytecode. It translates the individual instructions in the bytecode into the target machine language code one at a time rather than the whole program as a single unit. Each step is executed immediately after it is translated

3. Overview of Java Language



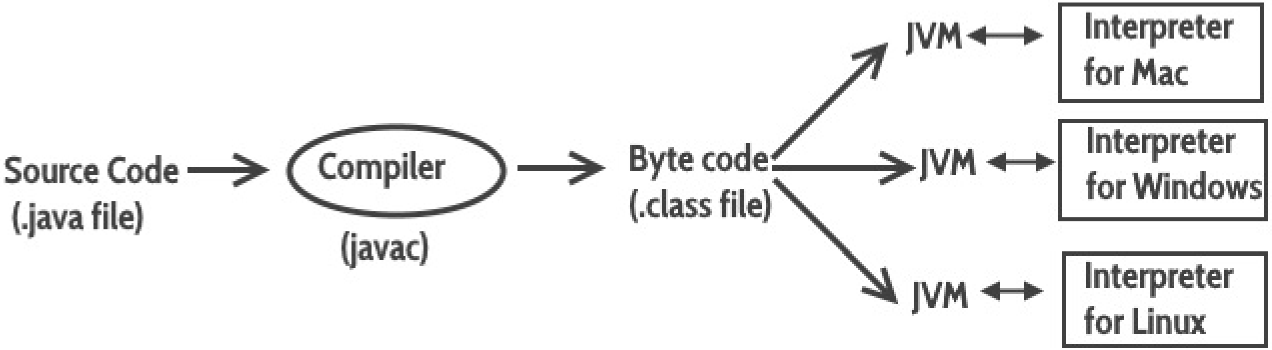
3. Overview of Java Language



Steps from Java code to Machine code:

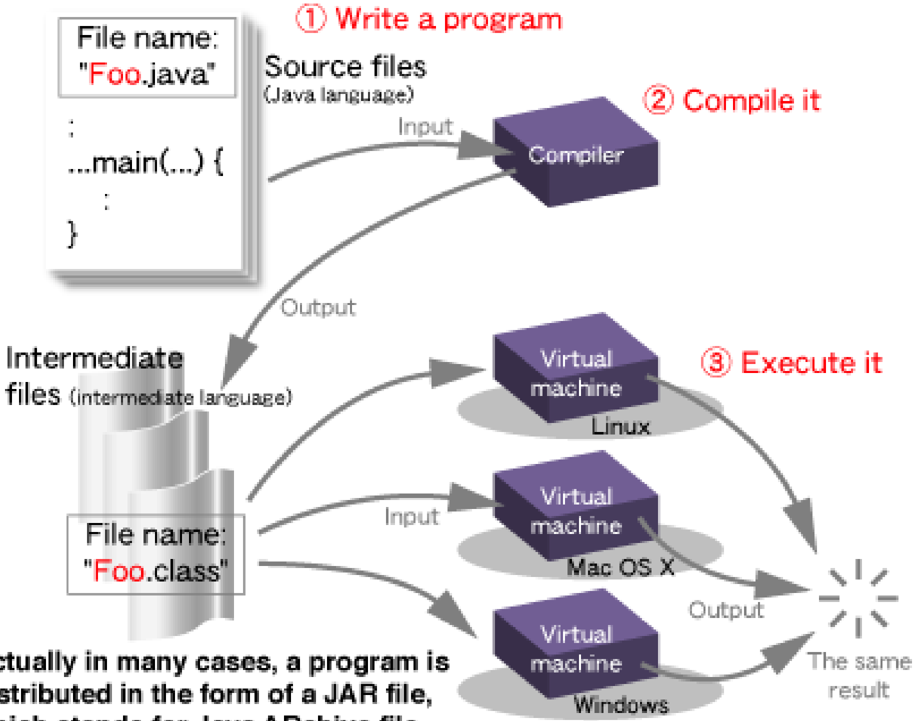
- Java source code is compiled.
- Transformed to bytecode by Java compiler.
- Interpreted and executed by the Java virtual machine on the underlying operating system.

3. Overview of Java Language



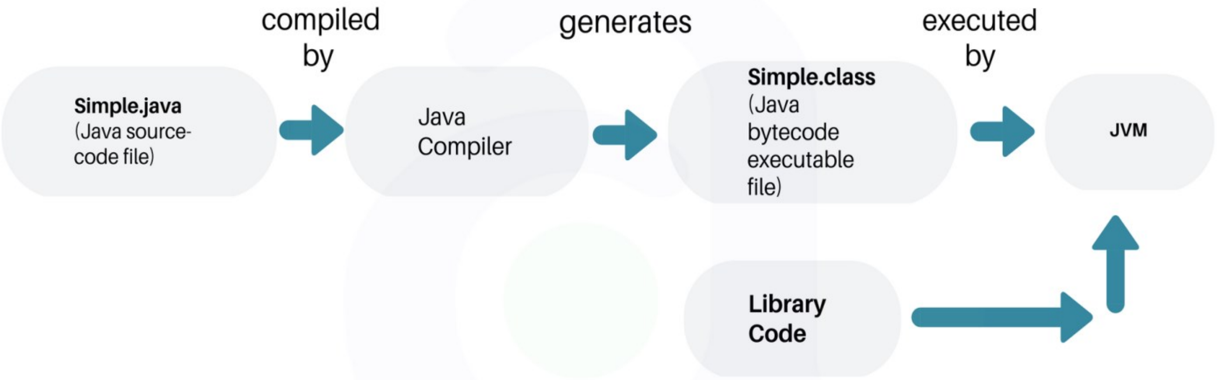
Source Code (.java file) → Compiler (javac) → Byte code (.class file) → JVM ↔ Interpreter for Mac; JVM ↔ Interpreter for Windows; JVM ↔ Interpreter for Linux

3. Overview of Java Language



- Actually in many cases, a program is distributed in the form of a JAR file, which stands for Java ARchive file.

3. Overview of Java Language



Java source code is translated into bytecode.