Java Programming

INTRODUCTION TO JAVA PROGRAMMING

Anu Arora, Assistant Professor, GNA University

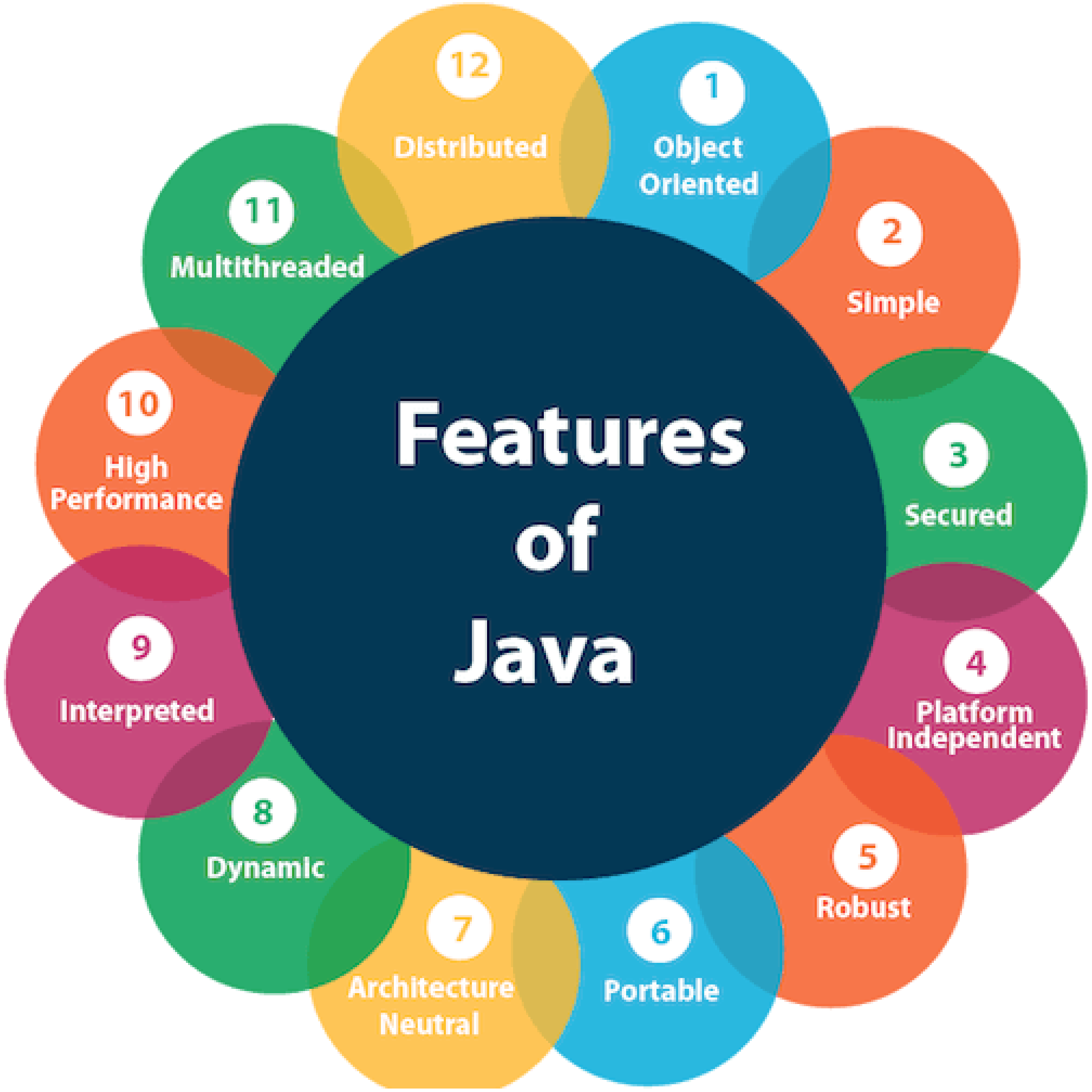
Introduction

- 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 name from Oak to Java.

- The latest release of the Java Standard Edition is Java SE 19. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms.
- For example: J2EE for Enterprise Applications, J2ME for Mobile Applications. The new J2 versions were renamed as Java SE, Java EE, and Java ME respectively.
- Java is guaranteed to be WORA-Write Once, Run Anywhere.

Feature of Java Language

- Object Oriented
- Platform Independent
- Secure
- Simple
- Architecture-neutral
- Portable
- Robust
- Multi threaded
- Interpreted
- High Performance
- Dynamic
- Distributed


Features of Java
1 Object Oriented
2 Simple
3 Secured
4 Platform Independent
5 Robust
6 Portable
7 Architecture Neutral
8 Dynamic
9 Interpreted
10 High Performance
11 Multithreaded
12 Distributed

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. These are traditional software that we need to install on every machine. Examples of standalone application are Media player, antivirus, etc. AWT and Swing are used in Java for creating standalone 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.

3) Enterprise Application: An application that is distributed in nature, such as banking applications, etc. is called an enterprise application. It has advantages like 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.

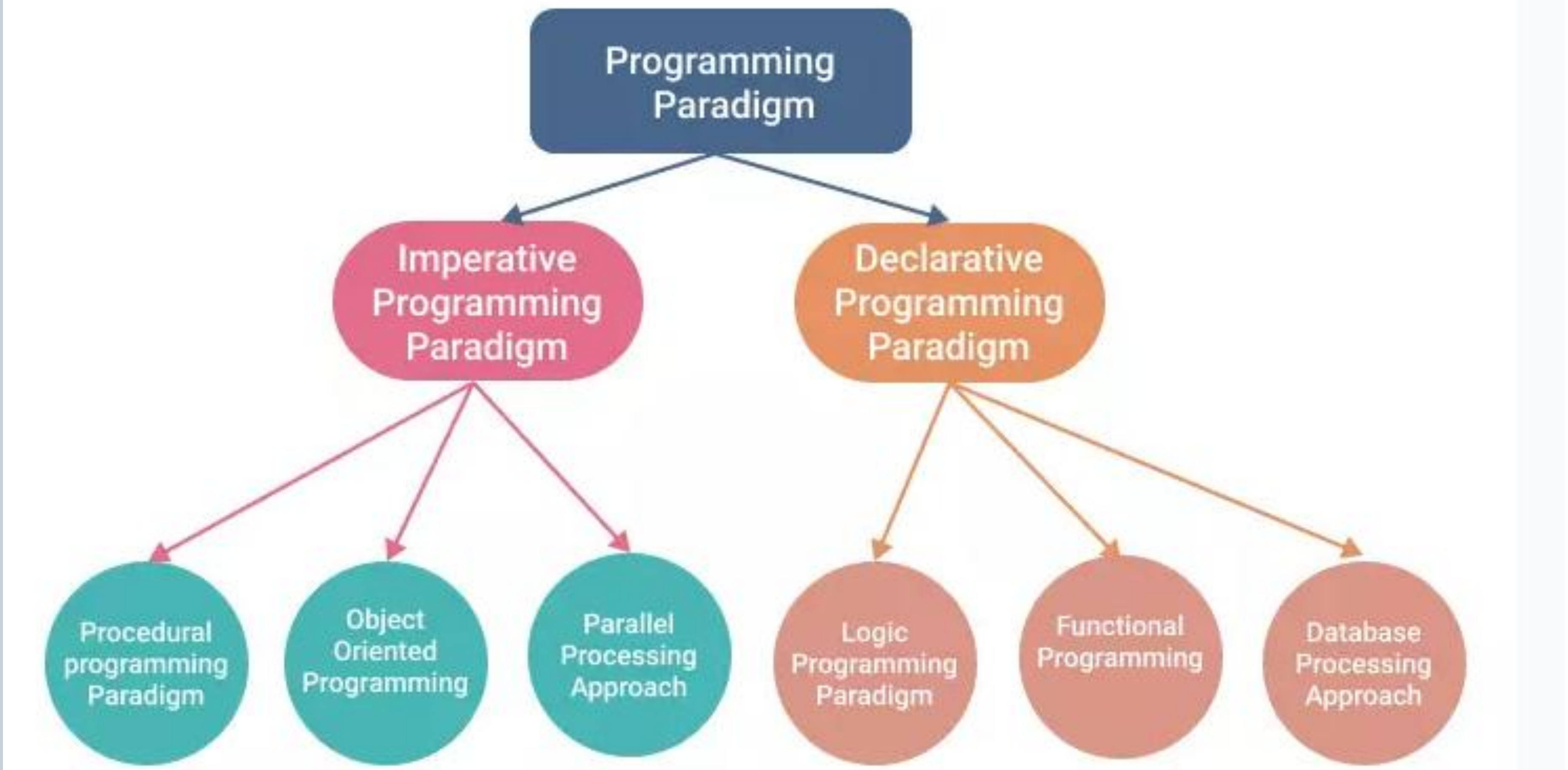
Programming Paradigm

- A ***programming paradigm*** is a way, an approach, a style by which we write programs in a specific programming language to solve some problem.
- A good programming language should support multiple paradigms because different programming problems require a different way to approach.

Types Of Programming Paradigms

The programming paradigms are categorized in multiple categories yet most significant are only two:

- **Imperative programming paradigm**
- **Declarative programming paradigm**


Programming Paradigm
Imperative Programming Paradigm
Declarative Programming Paradigm
Procedural programming Paradigm
Object Oriented Programming
Parallel Processing Approach
Logic Programming Paradigm
Functional Programming
Database Processing Approach

Imperative programming paradigm

- The imperative programming paradigm is the oldest paradigm and follows the most basic idea of programming. The programs in these paradigms are executed step by step by **changing the current state of the program**.
- An imperative program consists of some command that computer operates and change the state of the program.
- Imperative programming mainly focuses on how a program operates, unlike declarative programming which focuses on what the result of the program is going to achieve.
- Example: C, C++, Java, Ruby, PHP, Pascal etc.

Imperative programming: Subcategories

The imperative is broadly divided into 3 subcategories:

- **Procedural programming paradigm:** It is the same as imperative programming but with a procedure call that lets you reuse the code. And this feature was an amazing advancement at that time. Example **C**, **C++**, **Java**, etc.
- **Object Oriented Programming (OOP):** This is used to work on real-world entities in form of class and objects. Class is the blueprint of the object and you can replicate as many as the object you want. These classes contain some properties and method which all are replicated in objects. Example **C++**, **Java, Python**, etc.
- **Parallel processing:** In this type of programming, a program is processed by dividing it into multiple processors. The system contains multiple processors to solve the problem in less time.

Advantages & Disadvantages

Advantage	**Disadvantage**
Simple to implement and manually visualize	Unable to solve complex problems
Use of control flow statements	Programs can't execute parallelly
Easy to read and learn	Less productive, can't difficult to tackle modern day problems
	More risk of error

Declarative Programming Paradigm

A ***declarative programming paradigm*** are those paradigms in which the programmer describes the property of the result **without focusing on how to achieve it** (imperative programming focuses on how to achieve the task by changing the state of the program).

- The approach of this programming is to create some object or elements within the program.
- The main focus in this kind of programming is what is to be done rather than how it should be done.
- It does not talk about the work process of the result and only describes what the program must accomplish.
- Declarative programming is used in programming languages used in a database query, regular expression, functional programming, logical programming, etc.
- Example: Prolog, JavaScript, Scala, SQL, Lisp etc.

Declarative programming: Subcategories

- Logic programming paradigm
- Functional programming paradigm
- Database programming approach

Logic programming paradigm

- Logic programming uses sentences in logical form and creates an expression by using symbols.
- In machine learning and artificial intelligence, there are many models that use these programs.
- The programs are executed very much like some mathematical statement. It is mainly based on forming logic.

Functional programming paradigm

- Functional programming is the programming in which a program is constructed by creating and using functions.
- Rather than a series of statements functional programming use function to map and change one value to another value.
- Functional programming is the key feature of JavaScript.

Database programming approach

- This programming is based on enquiring data, its modification, its movement, etc. The most famous programming language that supports this is SQL.

Question/Answers

Q1. What is meant by programming paradigm?

A programming paradigm is a style of programming in any programming language. It is also used to classify programming languages on the basis of paradigms.

Q2. What are the 4 programming paradigms?

The 4 programming paradigms are: Procedural, Object-Oriented, Functional and Logical

Q3. What programming paradigm is SQL?

The programming paradigm supported by SQL is declarative programming.

Q4. What paradigm is C?

The programming paradigm of C is a procedural paradigm.

OOPs concepts in Java

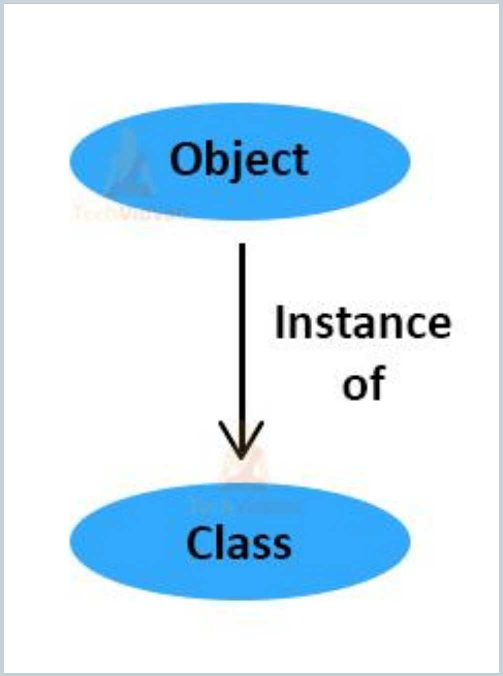
- Object-oriented programming System(OOPs) is a programming paradigm based on the concept of “objects” that contain data and methods.
- The primary purpose of object-oriented programming is to increase the flexibility and maintainability of programs.
- Object oriented programming brings together data and its behaviour (methods) in a single location(object) makes it easier to understand how a program works.

Object Oriented Paradigms

- Objects & Classes
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Object

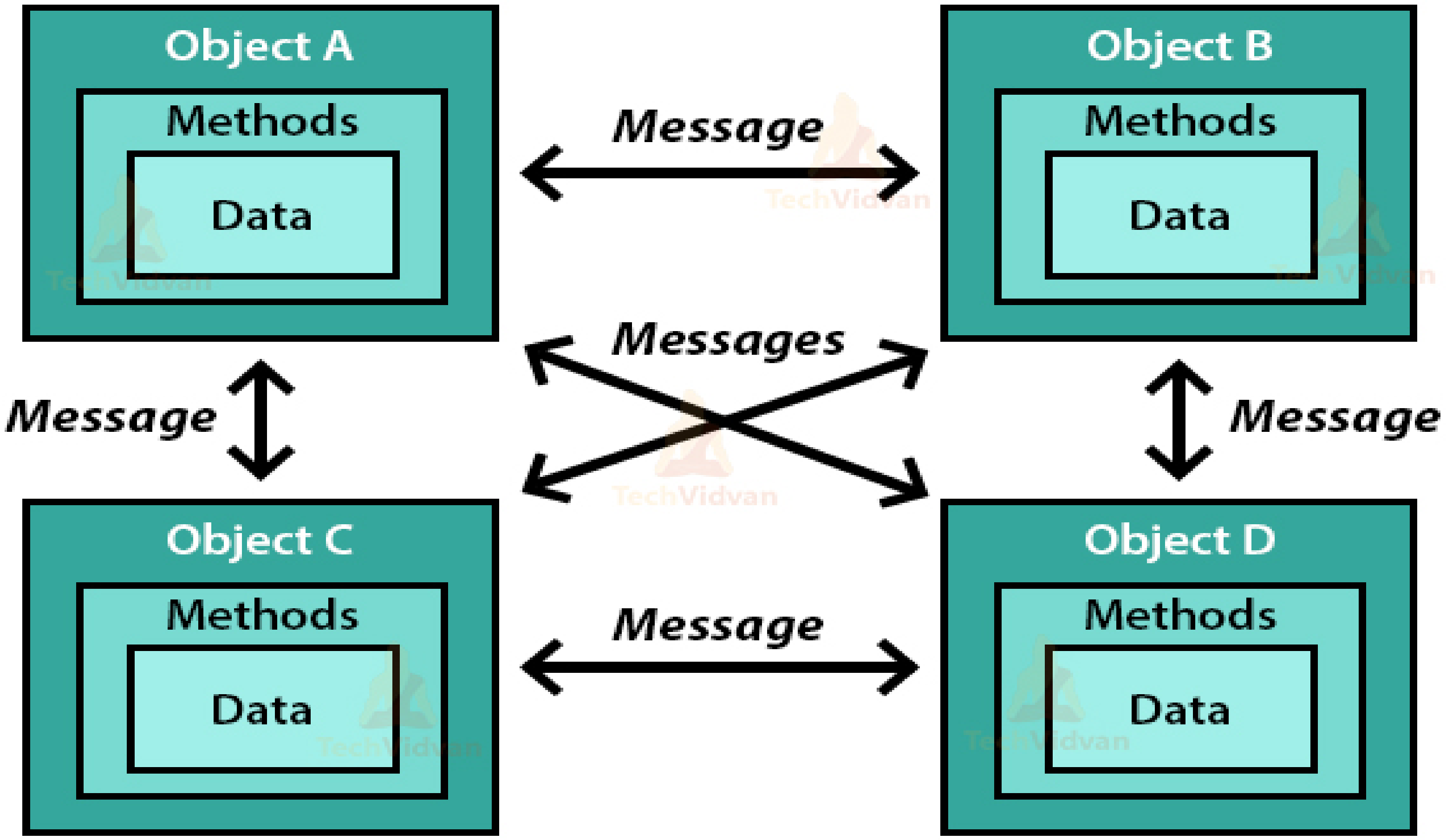
- **Object:** is a bundle of data and its behaviour(often known as methods).
- Objects have two characteristics: They have **states** and **behaviors**.
- **Characteristics of Objects:**
    - Abstraction
    - Encapsulation
    - Message passing



Characteristics of Objects

- **Abstraction**: Abstraction is a process where you show only "relevant" data and "hide" unnecessary details of an object from the user.
- **Encapsulation**: Encapsulation simply means binding object state(fields) and behaviour(methods) together. If you are creating class, you are doing encapsulation.
- **Message passing:** A single object by itself may not be very useful. An application contains many objects. One object interacts with another object by invoking methods on that object. It is also referred to as **Method Invocation**.

Message Passing



Ways to create object of a class

- **Using new keyword:** It is the most common and general way to create object in java.

  Example: Test t = new Test();

- **Using clone() method:** clone() method is present in Object class. It creates and returns a copy of the object.

  Example: Test t2 = (Test)t1.clone();

Ways to create object of a class

- **Using Class.forName(String className) method:** There is a pre-defined class in java.lang package with name Class. The forName(String className) method returns the Class object associated with the class with the given string name.We have to give the fully qualified name for a class. On calling new Instance() method on this Class object returns new instance of the class with the given string name.

  Example:// creating object of public class Test

  // consider class Test present in *com.p1* package

  Test obj = (Test)Class.forName("com.p1.Test").newInstance();

Ways to create object of a class

- **Deserialization:** De-serialization is technique of reading an object from the saved state in a file.

  Example:

  ```
  FileInputStream file = new FileInputStream(filename);
  ObjectInputStream in = new ObjectInputStream(file);
  Object obj = in.readObject();
  ```

Class

- A class is a user defined blueprint or prototype from which objects are created.
- It represents the set of properties or methods that are common to all objects of one type.
- In general, class declarations can include these components, in order:
  - Modifiers
  - Class name
  - Superclass
  - Interfaces
  - Body

- **Modifiers**: A class can be public or has default access
- **Class name:** The name should begin with a initial letter (capitalized by convention).
- **Superclass(if any):** The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
- **Interfaces(if any):** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
- **Body:** The class body surrounded by braces, { }.

Syntax to create a Class

```
modifier  class <classname>
{
   Body of class
}
e.g.
public class Add2Numbers
{
// body of class
}
```

- **Method**: The behavior of an object is the method.

  Example: The fuel indicator indicates the amount of fuel left in the car.
- **Instance variables**: Every object has its own unique set of instance variables. The state of an object is generally created by the values that are assigned to these instance variables.

  Example: Steps to compile and run a java program in a console

Data Abstraction

- Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essentials units are not displayed to the user.
- Ex: A car is viewed as a car rather than its individual components.
- Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details.
- The properties and behaviours of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.

Encapsulation

- It is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates.
- Another way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield.

- Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared.
- As in encapsulation, the data in a class is hidden from other classes, so it is also known as **data-hiding**.
- Encapsulation can be achieved by Declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

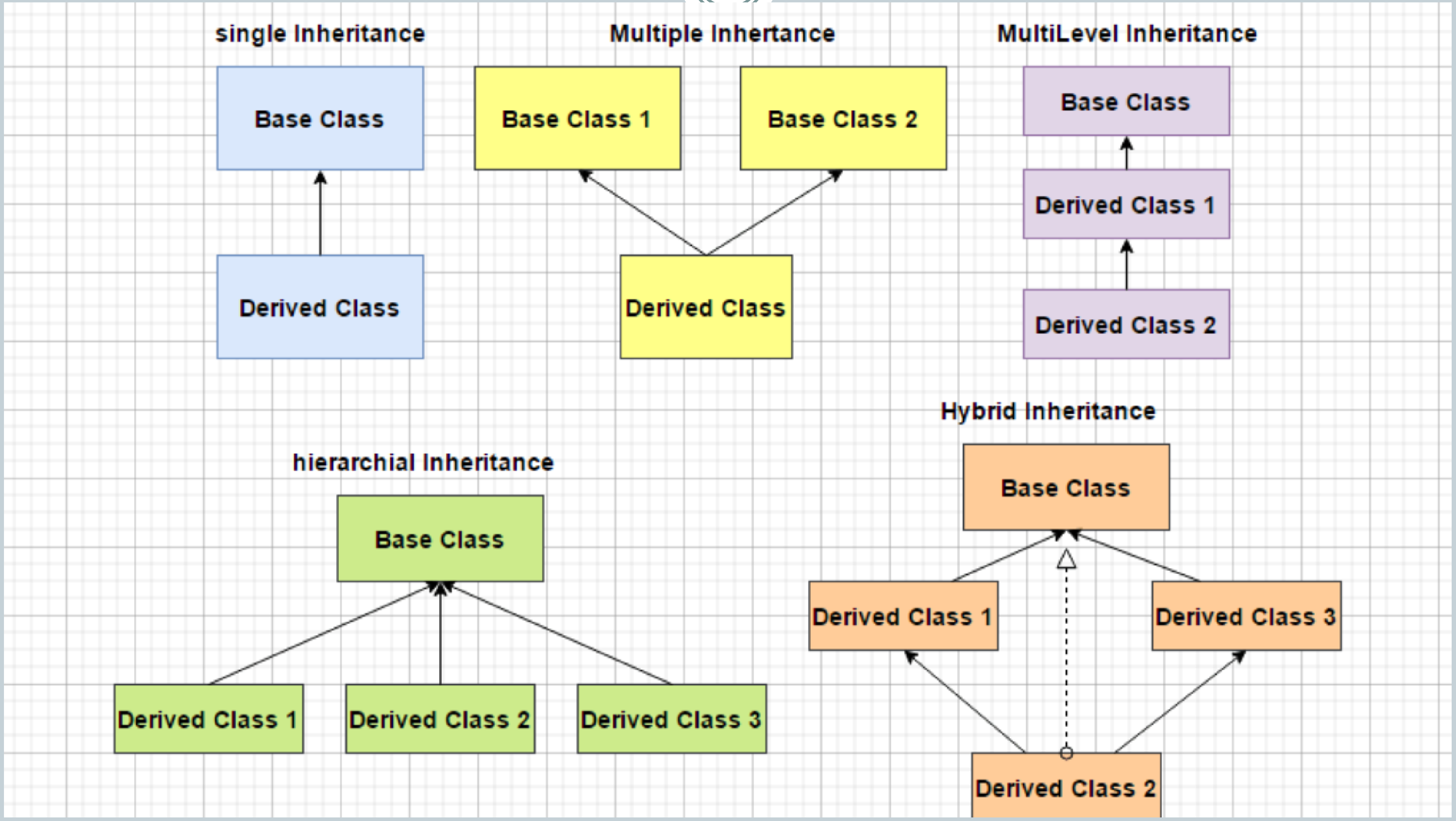
Inheritance

- Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class.
- Let us discuss some of frequent used important terminologies:
- **Super Class:** The class whose features are inherited is known as superclass(or a base class or a parent class).
- **Sub Class:** The class that inherits the other class is known as 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.
- **Reusability:** Inheritance supports the concept of "reusability", i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

Types of Inheritance

- **Single Inheritance**: refers to a child and parent class relationship where a class extends the another class.
- **Multilevel inheritance**: refers to a child and parent class relationship where a class extends the child class. For example class A extends class B and class B extends class C.
- **Hierarchical inheritance**: refers to a child and parent class relationship where more than one classes extends the same class. For example, class B extends class A and class C extends class A.
- **Multiple Inheritance**: refers to the concept of one class extending more than one classes, which means a child class has two parent classes. Java doesn’t support multiple inheritance

Types of Inheritance



Polymorphism

- It refers to the ability of OOPs programming languages to differentiate between entities with the same name efficiently. This is done by Java with the help of the signature and declaration of these entities.

Note: Polymorphism in Java are mainly of 2 types:

1) Static Polymorphism
2) Dynamic Polymorphism

Types of Polymorphism

Static Polymorphism:

Polymorphism that is resolved during compiler time is known as static polymorphism. Method overloading can be considered as static polymorphism example. **Method Overloading**: This allows us to have more than one methods with same name in a class that differs in signature.

Dynamic Polymorphism

It is also known as Dynamic Method Dispatch. Dynamic polymorphism is a process in which a call to an overridden method is resolved at runtime rather, that's why it is called runtime polymorphism.

References

- E.Balaguruswamy, Programming with JAVA, A primer, 3e, TATA McGraw-Hill Company.
- https://www.javatpoint.com/operators-in-java
- https://www.geeksforgeeks.org/operators-in-java/

Any Questions?