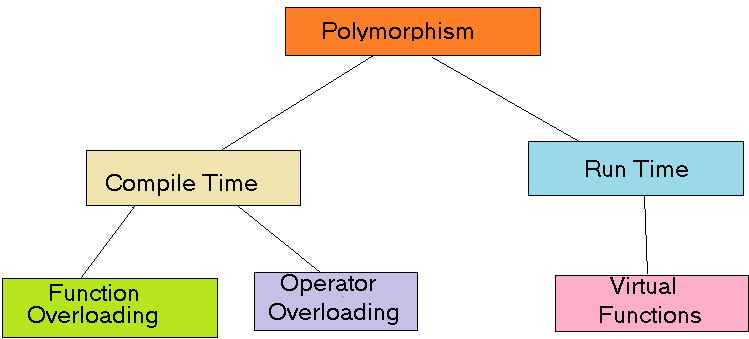
**# Polymorphism**

* + In Polymorphism we can perform a single task/action in different ways.
  + Polymorphism is derived from the Greek words ‘poly’ and ‘morph,’

meaning ‘many’ and ‘forms,’

**There are two types of polymorphism in Java:**

* Compile-time polymorphism
* Run-time polymorphism.



1. **compile-time polymorphism** :-
2. Compile time binding is re-solved by the complier
3. It is also known as static polymorphism or method overloading.
4. It is achieved by Method overloading and operator overloading.
5. When more then one methods have the same name but different parameters in single class .
6. **runtime polymorphism.**
7. **Runtime polymorphism** or **Dynamic Method Dispatch** is a process in which a call to an overridden method is resolved at runtime

**# Inheritance**

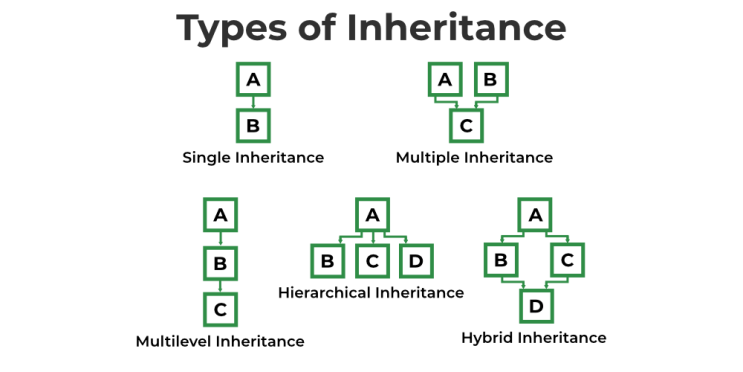
1. **Inheritance** is a mechanism in which one object acquires all the properties and behaviors of a parent object.
2. It is an important part of [OOPs](https://www.javatpoint.com/java-oops-concepts).
3. Inheritance represents the **IS-A relationship** which is also known as a parent-childrelationship*.*
4. A class is derived from another class and uses data and implementation of that other class.
5. The main purpose of Inheritance is to increase code re-usability

### Why use inheritance

1. For [Method Overriding](https://www.javatpoint.com/method-overriding-in-java) (so [runtime polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java) can be achieved).
2. For Code Reusability.

**Types of inheritance :**

* Single Inheritance
* Multiple Inheritance
* Hierarchical Inheritance
* Multilevel Inheritance
* Hybrid Inheritance



1. **Single Inheritance :-** When a class inherits another class, it is known as a single inheritance**.**
2. **Multilevel Inheritance** :- When there is a chain of inheritance, it is known as multilevel inheritance.
3. **Hierarchical Inheritance** :- When two or more classes inherits a single class, it is known as hierarchical inheritance
4. **Multiple Inheritance** :- When one classes inherits two or more classes, it is known as multiple inheritance.
5. **Hybrid Inheritance** :-

### \*\* Abstraction \*\*

**Abstraction** is a process of hiding the implementation details and showing only functionality to the user.

There are two ways to achieve abstraction in java

1. Abstract class
2. Interface

­­­­

### Abstract class :-

1. A class which is declared as abstract keyword is known as an **abstract class**.
2. Abstract class can have abstract and non-abstract methods.
3. To use the Abstract class, we need to inherit the abstract class.
4. We can’t create Object of the Abstract class.
5. Abstract class can have [constructors](https://www.javatpoint.com/java-constructor) and static methods also.
6. Abstract class can have final methods which will force the subclass not to change the body of the method.

### Abstract Method :-

### A method declared using the abstract keyword within abstract class

### and does not have definition is known as an abstract method.

1. To use the Abstract method, it is compulsory to inherit the class.
2. Sub class Object can use super class Methods.
3. If there is an abstract method in a class, that class must be abstract.
4. If you are extending an abstract class that has an abstract method, you must either provide the implementation(Overriding) of the method or make this class abstract.

# \*\* Interface \*\*

**Interface** :

1. An **interface** is a blueprint of a class.
2. It has static constants and abstract methods.
3. The interface is a mechanism to achieve [abstraction](https://www.javatpoint.com/abstract-class-in-java).
4. There can be only abstract methods in the Java interface, not method body.
5. It is used to achieve abstraction and multiple inheritance
6. It cannot be instantiated just like the abstract class.

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Java Interface also **represents the IS-A relationship**.

Since Java 8, we can have **default and static methods** in an interface.

Since Java 9, we can have **private methods** in an interface.

## Why use interface?

Ans : There are mainly three reasons to use interface.

* Interface is used to achieve abstraction.
* Interface can be used to achieve loose coupling.
* By interface, we can support the functionality of multiple inheritance

1. To create Interface we have to use Interface Keyword.
2. We can create only abstract Method in interface, We can’t create Method body in Interface.
3. We can’t create Object of a Interface.
4. To access the data of Interface, It is compulsory to

**#Constructur**

1. Constructor is a special type of member function because its name same as class Name.
2. Constructor have no return type.
3. Constructor is automatically called when Object is created object.

4. It is used to initialize Instance variable.

**Types of constructor's**

5.1). Default Constructor.

5.2). Parameterized Constructor.

1. **Default Constructor**  :-

Constructor with Zero Parameter is Know as defult Constructor.

1. **Parameterized Constructor** :-

A Constructor which is More then one Parameter is Know as parameterized Constructor.