Inheritance

- Allows inheriting data member and member method from one class to another
- Parnet/Super/Base is the class from which a new class is inheriting
- Child/Sub/Derived is the one which is inheriting
- Keywords used for inhitence: $extends$ & $implements$
- class $extends$ class
- class $implements$ interface
- interface $extends$ interface
- When creating an object of subclass, it will call default constructor of super class and deafult/Parametrised constrctor subclass (depending on how an object is created).

Types of inheritance in java

- Single : A $\Leftarrow$ B
- Multilevel : A $\Leftarrow$ B $\Leftarrow$ C
- Hierarchical : A $\Leftarrow$ B , B $\Leftarrow$ C
- Hybrid: Combination of any of Single, Mutiple, or Hierarchical
- Multiple: A $\Leftarrow$ B , A $\Leftarrow$ C
  - Java does not support Mutiple or Hybrid Inheritance becuase of ambuiguity
  - Multiple inheritance and Hybrid Inhetance is implemented through $interfaces$

Advantages:

- Plymorphism
- Code reusability
- Method Overrinng (Runtime Polymorphism)

```java
// Inheritance Example
class Parent{
    int a=10;
    public void display(){
        System.out.println("From Parent");
    }
}

class Child extends Parent{
    public void show(){
        System.out.println("From Child");
    }
}

public class Main{
    public static void main(String args[]){
        Child a= new Child();

        a.display();    // "From Parent"
        a.show();       // "From Child"
    }
}
```

Super

Super Method:

- Every subclass constructor (default/parametrised) has a predefined method called $super()$
- $super()$ is used to call the default constructor of parent class
- To call the parametrized constructor of parnet class by passing the argumnets to $super()$ method

Super Keyword:

- Even after overrding the method in base class, the super class method can be called using $super$ keyword. Just add super.methodName().
- $super$ keyword can also be used for accessing super Class variable.

```
class Parent{
    int a=10;
    Parent(){
        System.out.println("From Parent Parametrized Constructor");
    }

    Parent(int a){
        System.out.println("From Parent Parametrized Constructor");
    }
}

class Child extends Parent{
        Child(){
            super(10);
        System.out.println("From Child Default Constructor");
        }
    }

public class Main{
    public static void main(String args[]){
        Child a= new Child();
        //Output: From Parent Parametrised Constructor
        //        From Child Deafult Constructor

    }
}
```

Method Overriding

- Used when both super and sub class have same method definition
- In this case, the sub class overrides the method of super class/parent class.
- We use keyowrd @*override* before a method in sub class, when we want to override a method of super class.
- Using @override gives the advantage that if the method does not exits in super class, it will give a complie time error.
- Even after overrding the method in base class, the super class method can be called using *super* keyword. Just add super.methodName().
- *super* keyword can also be used for accessing super Class variable.
- Method overriding is *RuntimePolymorphism*

Dynamic Method Dispatch

- It is the process of creating an reference of super class and object of subclass.
- Allows acceesing only subclass methods and varibles
- The process of linkning the object to reference is done ar runtime, this is why it is called runtime polymorphism

```
class Parent{
    int a=10;
}

class Child extends Parent{
        public void show(){
            print("Child Show Method");
        }
    }


public class Main{
    public static void main(String args[]){

        Parent a= new Child();  // Runtime Linking
        a.show()                // Dynamic Method Dispatch

        //Output: Child Show Method
    }
}
```