INHERITANCE

➢ Generalization VS specialization

- Common thing in Generalisation and Specialisation is they are in the form of **Hierarchy**
- It is Like a Parent Class and Child Class(or)Base Class and Derived Class(or) Super Class and Sub Class.

➢ Generalization

- In Generalisation group of classes are referred with Super class with single name.
- Generalisation means **Bottom Up.**
- In Generalisation A Super Class Is made by Grouping Multiple Sub Classes.
- Generalisation is achieved using **Interfaces.**

For Examples

** Generalization*

1.Smart Phone - I phone
Samsung
Vivo

2.Vehicle -Car
Bike
Ship

3.Shape -Triangle
Rectangle
Circle

- For achieving Generalisation **Abstract Classes** are Used.
-

➢ **Specialization**

In Specialisation a new Sub Class is Generated by borrowing the features of existing concrete class and adding new features to it.

- Specialisation means **Top Down.**
- In specialisation a new Class is derived from an existing Super Class.
- Specialisation is achieved using **Inheritance.**

For Examples

**Specialization*

.I Phone X -> I Phone XS
2.Innova -> Fortuner
3.Circle -> Cylinder
4.Guitar -> Electric Guitar

➢ **What Is Inheritance?**

- Inheritance is the process of acquiring features of an existing Class into a New Class.
- A Class will have Properties and Method.

For Example

A cylinder can acquire all the properties of circle Plus it can have extra feature
Where we can write a class cylinder inheriting from class circle

A Cuboid can Acquire All the Propertied of A Rectangle Plus Extra feature

Example Program

```
class Circle

{
      private double radius;
      public Circle( )
      {
           radius =0.0;
      }
      public double area();{}
      public double perimeter();{}
};

class Cylinder extends Circle
{
      private double height;
      public Cylinder()
      {
          height=0.0;
      }
      public double volume();{}
};
class test
{
      public static void main()
      {
       Circle c1=new Circle();
       Cylinder c2=new Cylinder();
      }
}
```

- Circle and Cylinder are the two classes where circle class is having radius and Cylinder Class acquire the property(radius) of circle with an extra property that is height.
- Cylinder is the Extension of Circle.
- extends is the keyword used for Inheritance.
- Circle Class Is Parent Class.
- Cylinder Class is the Derived Class, Derived Class is also known as Specialised Class.
- In the given example upon the object c1 and c2 the constructors are called automatically.

- In object c1 methods area and perimeter are called
- In object c2 methods area.

➢ Constructors in Inheritance

- Constructors are the **methods** of class which are automatically called when an object is created.
- Constructors are executed from **Top to Bottom Class.**
- To make the child class object firstly the parent class constructor must be created

Example Program

```
class Parent
{
   public Parent()
   {
     System.out.println("Parent Constructor");
   }
}
class Child extends Parent
{
   public Child()
   {
      System.out.println("Child Constructor");
   }
}
class Grandchild extends Child
{
   public Grandchild()
   {
      System.out.println("Grand Child Constructor");
   }
}
class inheritanceDemo
{
  Grandchild g=new Grandchild();
}
```

- In the given example Parent Class have its constructor where **"parent constructor"** is executed, and Child Class have its constructor where "**Child Constructor"**, is executed and Grandchild Class have its constructor where "**Grandchild Constructor"** is executed.

- In the given example program if a Grandchild class object is created then as we know Grandchild class inherits Child class constructor and Child class inherits Parent class constructor so the Grandchild object first executes the top most class then level by level.

For Example

Car companies manufacture a platform (skeleton of a car)
They will give the shape to a car then first they will create a base which has engines ,seats etc. and above that they will put a cabin on it so the platform will be same for more of the cars of that company only the outer body of the cars will be different

Relating to the given program here the platform is the parent class and the base and cabin created is the child class where the base and cabin is inheriting from the platform

Method Overriding

- Redefining the method of the Super Class in the Sub Class.
- Method will be called depending on the object.
- Method overriding is achieved in **Inheritance.**

```
class super

{
  public void display()
  {
    System.out.println("Hello");
  }
}
class sub extends super
{
   public void display()
   {
     System.out.println("Hello Welcome");
   }
}
```

- When the sub class object is called then the display method inherited from the super class is shadowed and the sub class display method is executed.
- Super Class method never be called upon the object of Sub Class.
- In the given example program the super class have a method called display which is saying hello and another class sub class is taken where it inherits the display method from super class and redefines the method.
- When a super class reference holding the object of sub class and overridden method is called then method of object will be called it is **Dynamic Method Dispatch.**

➢ **Dynamic Method Dispatch**

- It is useful for achieving Runtime Polymorphism.

Example Program

```
class Super

{
   Void meth1()
   {
     System.out.println("meth1");
   }
   Void meth2()
   {
     System.out.println("super meth2");
   }
}
class Sub extends Super
{
   Void meth2()
   {
    System.out.println("sub meth2");
   }
   Void meth3()
   {
    System.out.println("meth3");
   }
}
class test
{
   public.static.void.main()
   {
     Super s=new Sub();
   }
}
```

- In the given example meth2 is redefined in the sub class.
- Super Class reference can have Object of Sub Class but a Sub Class reference cannot have Super Class Object.
- A Super Class Reference can hold the Object of Sub Class but it can call only those methods which are present in super class.
- Methods are called depending on the object not the reference then the overridden object is called it is Runtime Polymorphism.
- Dynamic Method Dispatch means calling a Method dynamically because program make the decision at runtime for which object to be called.

➢ Do's and Don'ts of Overriding

- Signature must be same in method overriding.
- If the method name is different the method is not overridden but it is overloaded.
- Argument may be different but the parameter must be same.
- Return type must be same, if it is not same then the method is neither overridden nor overloaded.
- Final and static methods cannot be overridden.
- Method can be overridden with same or lenient (public, protected) access specifiers but the stricter(private) access specifiers cannot be used in sub class.

➢ Polymorphism using Overloading and Overriding

- Polymorphism is one of the principles of Object-oriented-programming, polymorphism means one name different actions.
- Poly means 'many', morphism means 'forms'.
- Polymorphism is achieved using method overriding and overloading.
- In method overloading access specifiers, return types are same but number of parameters or type of parameters are different.
- In overloading number or type of argument will decides which method is to be called.
- Overloading is achieved in same class whereas overriding is achieved in inheritance.
- In method overriding signature is same but in overloading signatures must be different.
- Method calls are different in overriding it depends on object.
- Method overriding is used for **runtime polymorphism** and method overloading is used for **compile time polymorphism.**