Polymorphism

POLYMORPHISM

- Polymorphism is derived from two different Greek words ‘Poly’ means Numerous ‘Morphs’ means form which mean “numerous form”.
- Polymorphism is java is ability of an object to exhibit more than one form with same name.

TYPES OF POLYMORPHISM

Compile time polymorphism

Runtime Polymorphism

COMPILE TIME POLYMORPHISM :

- If the binding is achieved at the compile time and the same behaviour is execute it is known as compile time polymorphism.
- It is also said to be static polymorphism.
- It is achieved by

  a)Method overloading

  b)Constructor overloading

  c)Variable shadowing

  d)Method shadowing

  e)Operator overloading (does not supports in java)

METHOD SHADOWING :

If a subclass and superclass have the static method with same signature, it is known as method shadowing.

Which method implementation gets execute, depend on what ?

In method shadowing binding is done at compile time , hence it is compile time polymorphism. The execution of the method is depend on the reference type an object but not the object created.

NOTE :

- Method shadowing is applicable only for the static method.
- It is compile time polymorphism
- Execution of implemented method depends on the reference type of an object.

VARIABLE SHADOWING :

If the superclass and subclass have variables with same name then it is known as variable shadowing.

Which variable is used, depend on what ?

In variable shadowing binding is done at compile time , hence it is a compile time polymorphism. Variable is called depend on the reference type an object but not the object created.

NOTE :

- It is applicable for both static and non static variable.
- It is a compile time polymorphism.
- Variable usage is depend on the reference type of an object

RUNTIME POLYMORPHISM :

★ When the binding is achieved at the runtime then ,it is known as runtime polymorphism.
★ It is also known as dynamic binding.
★ It is achieved by method overriding.

METHOD OVERRIDING :

●If the subclass and superclass have the non static methods with and same signature , it is known as method overriding.

Rule to achieve method overriding :

●Is-A relationship is mandatory.
●It is applicable only for non static methods.
●The signature of the subclass method and superclass method should be same.
●The return type of the subclass and superclass method should be same until 1.4 version but, from 1.5 version covariant return type in overriding method is acceptable (subclass return type should be same or child to the parent class return type.).

RULE TO USE ACCESS MODIFIER

The scope of the access modifier of subclass method should be same or less than the super class method.

Scope of the access modifiers

private<default<protected<public

NOTE :

- Method overriding is used to achieve runtime polymorphism.
- The execution of method implementation is depend on the object created but not depend on the reference type.
- Variable overriding is not applicable.

```
class SuperClass
{
    public void demo()
   {
      System.out.println("Super Class");
    }
}
class SubClass
{
    public void demo()
   {
      System.out.println("Sub Class");
    }
}
```

```
class Driver
{
  public static void  main(String[]args)
   {
     SubClass sub1 = new SubClass();
      sub1.demo(); // from child
     SuperClass super1 = sub1;
     super1.demo(); // from child
```