Training On Java

Lecture – 7
Concept Of Polymorphism

Polymorphism In Java

The term “Polymorphism” means “One Thing Many Forms”. There are two types of polymorphism in Java:-
1. Compile Time Polymorphism (Overloading)
2. Run Time Polymorphism (Overriding)

Compile time polymorphism [Overloading]:-

1. If java class allows two methods with same name but different number of arguments such type of methods are called overloaded methods.
2. We can overload the methods in two ways in java language
   a. By passing different number of arguments to the same methods.
   void m1(int a){}
   void m1(int a,int b){}
   b. Provide the same number of arguments with different data types.
   void m1(int a){}
   void m1(char ch){}
3. If we want achieve overloading concept one class is enough.
4. It is possible to overload any number of methods in single java class.

Method Overloading

In java programming language you can give same name to multiple methods but their arguments should be different. Based on method arguments it is decided at compilation time that which method call from where. It is called method overloading.

Example Application - 1

```
/*Create a class with names Shape in Shape class make three method with same name area (method
overloading). First method find the area of square, second method find the area of circle and third method find
the area of rectangle. Now test the class Shape.*/
import java.util.*;
class Shape {
public int area(int s) {
return (s*s);
}
public int area(int l,int b) {
return (l*b);
}
public double area(float r) {
return (3.14*r*r);
}
}
```

Example Application – 1 (cont..)

```
class Test {
public static void main(String [] args) {
int s,l,b,a1,a2;
double a3;
float r;
Scanner sc=new Scanner(System.in);
Shape sh=new Shape();
System.out.print(“Enter side of square : ”);
s=sc.nextInt();
System.out.print(“Enter length and breadth of rectangle : ”);
l=sc.nextInt();
b=sc.nextInt();
System.out.print(“Enter radius of circle : ”);
r=sc.nextFloat();
a1=sh.area(s);
a2=sh.area(l,b);
a3=sh.area( r);
System.out.println(“Area of square=”+a1);
System.out.println(“Area of rectangle=”+a2);
System.out.println(“Area of circle=”+a3);
}}
```

Example Application – 2

/* Constructor Overloading:-The class contains more than one constructors with same name but different arguments is called constructor overloading. */

```
class Test  {
//overloaded constructors
Test()  {
System.out.println("0-arg constructor");
}
Test(int i) {
System.out.println("int argument constructor");
}
Test(char ch,int i) {
System.out.println(ch+"-----"+i);
}
public static void main(String[] args)   {
Test t1=new Test(); //zero argument constructor executed.
Test t2=new Test(10); // one argument constructor executed.
Test t3=new Test('a',100);//two argument constructor executed.
}
}
```

Method Overriding

The re-writing of base class method to derived class is called method overriding.

1) If we want to achieve method overriding we need two class with parent and child relationship.
2) The parent class method contains some implementation (logics).
    a. If child is satisfied use parent class method.
    b. If the child class not satisfied (required own implementation) then override the method in child class.
3) A subclass has the same method as declared in the super class it is known as method overriding.

The parent class method is called ===> overridden method

The child class method is called ===> overriding method

Rules For Method Overriding

1) While overriding child class method signature & parent class method signatures must be same otherwise we are getting compilation error.

2) The return types of overridden method & overriding method must be same.

3) While overriding the methods it is possible to maintains same level permission or increasing order but not decreasing order, if you are trying to reduce the permission compiler generates error message “attempting to assign weaker access privileges ”.

4) You are unable to override final methods. (Final methods are preventing overriding).

5) While overriding check the covariant-return types.

6) Static methods are bounded with class hence we are unable to override static methods.

7) It is not possible to override private methods because these methods are specific to class.

Example Application - 3

```
// Develop a program to demonstrate concept of method overriding.
class A {
public void m1() {
System.out.println(“m1 of A”);
}
public void m2() {
System.out.println(“m2 of A”);
}
}
class B extends A {
public void m2() {
System.out.println(“m2 of B”);
}
public void m3() {
System.out.println(“m3 of B”);
}
}
```

Example Application – 3 (cont..)

```
class Test
{
public static void main(String [] args)
{
A a1=new A();
a1.m1(); //m1 of A
a1.m2(); //m2 of A
B b1=new B();
b1.m1();//m1 of A
b1.m2();//m2 of B
b1.m3();//m3 of B
A a2=new B(); //Up-casting
a2.m1(); //m1 of A
a2.m2(); //m2 of B
}
}
```

Difference Between Method Overloading & Overriding

Method Overloading	**Method Overriding**
Method overloading is used to increase the readability of the program.	Method overriding is used to provide the specific implementation of the method that is already provided by its super class.
Method overloading is performed within class.	Method overriding occurs in two classes that have IS-A (inheritance) relationship.
In case of method overloading, parameter must be different.	In case of method overriding, parameter must be same.
Method overloading is the example of compile time polymorphism.	Method overriding is the example of run time polymorphism.
In java, method overloading can't be performed by changing return type of the method only. Return type can be same or different in method overloading. But you must have to change the parameter.	Return type must be same or covariant in method overriding.