**Polymorphism**

Polymorphism in the context of object-oriented programming, is the “**ability to create a variable, a function, or an object that has more than one form**.”

In more simple words, polymorphism is the ability by which, we **can create functions or reference variables which behaves differently in different programmatic context**.

***Polymorphism is one of the major building blocks of object oriented programming along with inheritance, abstraction and encapsulation***.

In java language, polymorphism is essentially considered into two versions.

1. **Compile time polymorphism** (static binding or method overloading)

**Method Overloading:** When there are multiple functions with the same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in the number of arguments or/and a change in the type of arguments.

As the meaning is implicit, this is used to write the program in such a way, that **flow of control is decided in compile time itself.** It is achieved using method overloading.

In **method overloading**, an object can have **two or more methods with same name**, BUT, with their **method parameters different**. These parameters may be different on two bases:

**1) Parameter type**: Type of method parameters can be different. e.g. java.util.Math.max() function comes with following versions:

public static double Math.max(double a, double b){..}

public static float Math.max(float a, float b){..}

public static int Math.max(int a, int b){..}

public static long Math.max(long a, long b){..}

The actual method to be called is decided on compile time based on parameters passed to function in program.

**2) Parameter count**: Functions accepting different number of parameters. e.g. in employee management application, a factory can have these methods:

EmployeeFactory.create(String firstName, String lastName){...}

EmployeeFactory.create(Integer id, String firstName, String lastName){...}

Both methods have same name “create” but actual method invoked will be based on parameters passed in program.

**Example Program**

package methooverload1;

import java.lang.Math;

//Method overloading (basic- Methodoverloading.java)

class Overload {

double area(float a, float b, float c) {

float s;

s=(a+b+c)/2;

return Math.sqrt(s\*(s-a)\*(s-b)\*(s-c));

}

double area(float r) {

return Math.PI\*r\*r;

}

double area(float l, float b) {

return l\*b;

}

}

public class methodoverload1{

public static void main(String args[]) {

Overload o = new Overload();

double rectangle = o.area(5.1f, 8.12f);

System.out.printf("Area of ractangle is %.2f ",rectangle);

System.out.println("");

double triangle = o.area(5.1f,4.2f,3.5f);

System.out.printf("Area of triangle is %.2f ",triangle);

System.out.println("");

double circle = o.area(5.1f);

System.out.printf("Area of circle is %.2f ", circle);

System.out.println("");

}

}

o/p

Area of ractangle is 41.41

Area of triangle is 7.29

Area of circle is 81.71

**Runtime polymorphism (dynamic binding or method overriding)**

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java. Method overriding is used to provide the specific implementation of a method which is already provided by its superclass. Method overriding is used for runtime polymorphism

**Rules for Java Method Overriding**

* The method must have the same name as in the parent class
* The method must have the same parameter as in the parent class.

//Java Program to demonstrate the real scenario of Java Method Overriding

//where three classes are overriding the method of a parent class.

//Creating a parent class.

class Bank{

int getInterest(){return 0;}

}

//Creating child classes.

class SBI extends Bank{

int getInterest(){return 8;}

}

class ICICI extends Bank{

int getInterest(){return 7;}

}

class AXIS extends Bank{

int getInterest(){return 9;}

}

//Test class to create objects and call the methods

class Override{

public static void main(String args[]){

Bank b= new Bank();

SBI s=new SBI();

ICICI i=new ICICI();

AXIS a=new AXIS();

System.out.println("Bank Interest % is: "+b.getInterest());

System.out.println("SBI Interest % is: "+s.getInterest());

System.out.println("ICICI Interest % is: "+i.getInterest());

System.out.println("AXIS Interest % is: "+a.getInterest());

}

}

o/p

Bank Interest % is: 0

SBI Interest % is: 8

ICICI Interest % is: 7

AXIS Interest % is: 9