**Types of polymorphism in java- Runtime and Compile time polymorphism**

Earlier we saw all about Polymorphism in Java. In this tutorial we will cover**types of polymorphism in java**. There are two types of polymorphism in java- **Runtime polymorhism( Dynamic polymorphism)**and**Compile time polymorphism (static polymorphism)**.

**Runtime Polymorhism( or Dynamic polymorphism)**

[**Method overriding**](http://beginnersbook.com/2014/01/method-overriding-in-java-with-example/) is a perfect example of  runtime polymorphism. In this kind of polymorphism, reference of class X can hold object of class X or an object of any sub classes of class X. For e.g. if class Y extends class X then both of the following statements are valid:

Y obj = new Y();

//Parent class reference can be assigned to child object

X obj = new Y();

Since in method overriding both the classes(base class and child class) have same method, compile doesn’t figure out which method to call at compile-time. In this case JVM(java virtual machine) decides which method to call at runtime that’s why it is known as runtime or dynamic polymorphism.

Lets see an example to understand it better.

public class X

{

public void methodA() //Base class method

{

System.out.println ("hello, I'm methodA of class X");

}

}

public class Y extends X

{

public void methodA() //Derived Class method

{

System.out.println ("hello, I'm methodA of class Y");

}

}

public class Z

{

public static void main (String args []) {

X obj1 = new X(); // Reference and object X

X obj2 = new Y(); // X reference but Y object

obj1.methodA();

obj2.methodA();

}

}

**Output:**

hello, I'm methodA of class X

hello, I'm methodA of class Y

As you can see the methodA has different-2 forms in child and parent class thus we can say methodA here is polymorphic.

**Compile time Polymorhism( or Static polymorphism)**

Compile time polymorphism is nothing but the method overloading in java. In simple terms we can say that a class can have more than one methods with same name but with different number of arguments or different types of arguments or both. To  know more about it refer [**method overloading**](http://beginnersbook.com/2013/05/method-overloading/) in java.

Lets see the below example to understand it better-

class X

{

void methodA(int num)

{

System.out.println ("methodA:" + num);

}

void methodA(int num1, int num2)

{

System.out.println ("methodA:" + num1 + "," + num2);

}

double methodA(double num) {

System.out.println("methodA:" + num);

return num;

}

}

class Y

{

public static void main (String args [])

{

X Obj = new X();

double result;

Obj.methodA(20);

Obj.methodA(20, 30);

result = Obj.methodA(5.5);

System.out.println("Answer is:" + result);

}

}

**Output:**

methodA:20

methodA:20,30

methodA:5.5

Answer is:5.5

As you can see in the above example that the class has three variance of methodA or we can say methodA is polymorphic in nature since it is having three different forms. In such scenario, compiler is able to figure out the method call at compile-time that’s the reason it is known as compile time polymorphism.