Polymorphism

Polymorphism is an oops principle in java, process of one entity showing multiple or different behaviour at different stages of its life cycle is known as polymorphism.

There are two types of polymorphism:
* Complie time
* Run time

Compile time Polymorphism:

The process of binding method declaration and definition at compile time is known as Compile time Polymorphism.
In case of Compile time Polymorphism, method calling is resolved at compile time, hence it is known as static or early binding.
Method overloading is an example for compile time polymorphism.

RunTime Polymorphism

The process of method declaration and definition at runtime is known as Run time Polymorphism
In case of Run time Polymorphism method calling is resolved by JVM at runtime.
Run time Polymorphism can be achieved using method overriding, typecasting and inheritance.

If overridden object of sub class is executed then subclass is only executed. But if it is static then only superclass objects are accessed.

Example:
```
public class Animal
{
	public void sound()
	{
		System.out.println("Animal sound");
	}

}
public class Dog extends Animal
{
	public void sound()
	{
		System.out.println("dog sound");
	}

}
public class Main {

	public static void main(String[] args)
	{
		Animal a=new Dog();
		a.sound();

	}

}
```

Compile Time polymorphism example:

```
public class Demo
{
public static char Print(char ch)
{
	return 'a';
}

public static int Print(int i)
{
	return 10;
}
}
public class Main {

	public static void main(String[] args)
	{
		char b=Demo.Print('a');
		System.out.println(b);
		int i=Demo.Print(10);
		System.out.println(i);
	}

}
```