05

PLYMORPHISOM

OBJECT ORIENTED PROGRAMMING LAB

SESSION - 05

CSE215 | DAFFODIL INTERNATIONAL UNIVERSITY

Polymorphism:

- Polymorphism in java is a concept by which we can perform a *single action by different ways*.
- Polymorphism means many forms.
- It is the ability of an object to take on many forms.

There are two types of polymorphism in java:

- Compile time polymorphism. (method overloading)
- Runtime polymorphism. (method overriding)

*Runtime polymorphism/ Dynamic Method Dispatch:

- a process in which a call to an overridden method is resolved at runtime rather than compile-time.
- In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

Upcasting

When reference variable of Parent class refers to the object of Child class, it is known as upcasting. For example:


```
class A{}
class B extends A{}
```

```
A a=new B();//upcasting
```

Example of Java Runtime Polymorphism

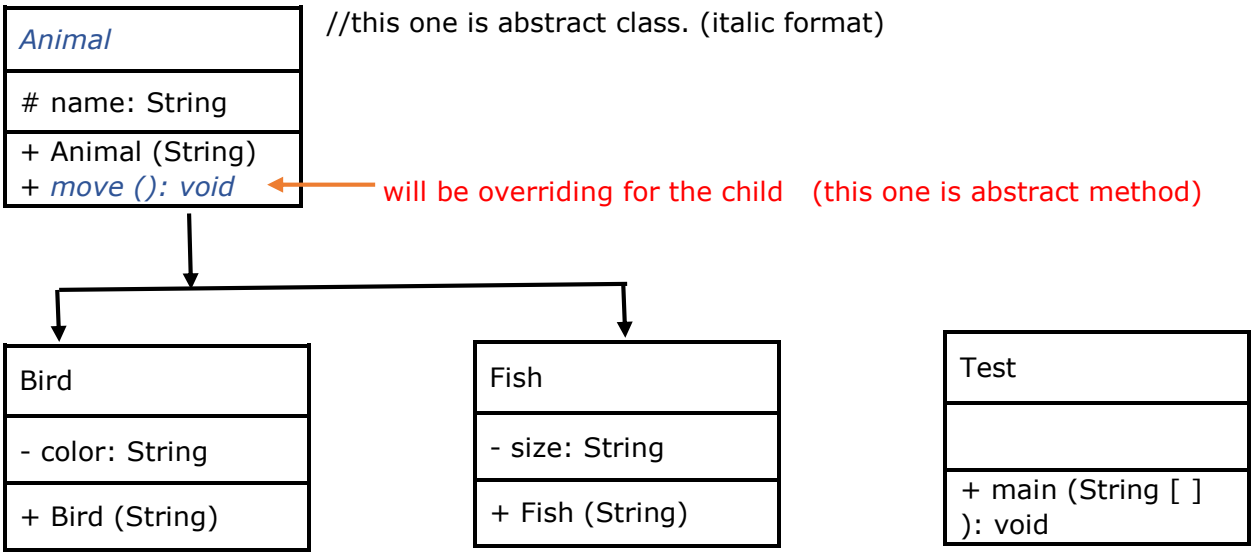
In this example, we are creating two classes Bike and Splendar. Splendar class extends Bike class and overrides its run() method. We are calling the run method by the reference variable of Parent class. Since it refers to the subclass object and subclass method overrides the Parent class method, subclass method is invoked at runtime.

Since method invocation is determined by the JVM not compiler, it is known as runtime polymorphism.

```
class Bike{
  void run(){System.out.println("running");}
}
class Splender extends Bike{
  void run(){
     System.out.println("running safely with 60km");
}
   public static void main(String args[]){
    Bike b = new Splender();  //upcasting
    b.run();
  }
}
```

*Now look the following UML diagram and the implemented code:

***Example:



```
abstract class Animal {
	protected String name;
	public Animal(String name) {
		this.name = name;
	}
	public abstract void move();  //abstract method
}

class Bird extends Animal {
	private String color;
	public Bird(String name, String color) {
		super(name);
		this.color = color;
	}
	public void move() {
		System.out.println("Bird name is " + name + " and color is " + color + ".");
		System.out.println("Fly");
	}
}

class Fish extends Animal {
	private int size;
	public Fish(String name, int size) {
		super(name);
		this.size = size;
	}
	public void move() {
		System.out.println("Fish name is " + name + " and size is " + size + ".");
		System.out.println("Swim");
	}
}

public class Test {
	public static void main(String[] x) {
		Animal a[] = new Animal[2];
		Bird b = new Bird("Dwel", "Black & White");
		Fish f = new Fish("Hilisha", 5);
		a[0] = b;
		a[1] = f;
		for (int i = 0; i < a.length; i++) {
			a[i].move();
		}
	}
}
```

Output is

```
Bird name is Dwel and color is Black & White.
Fly
Fish name is Hilisha and size is 5.
Swim
```

Method overriding:

- If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in java. In other words, If subclass provides the specific implementation of the method that has been provided by one of its parent class, it is known as method overriding.
- Runtime Polymorphism

Rules for Java Method Overriding

1. method must have same name as in the parent class
2. method must have same parameter as in the parent class.
3. must be IS-A relationship (inheritance).

Example:

In this example, we have defined the run method in the subclass as defined in the parent class but it has some specific implementation. The name and parameter of the method is same and there is IS-A relationship between the classes, so there is method overriding.

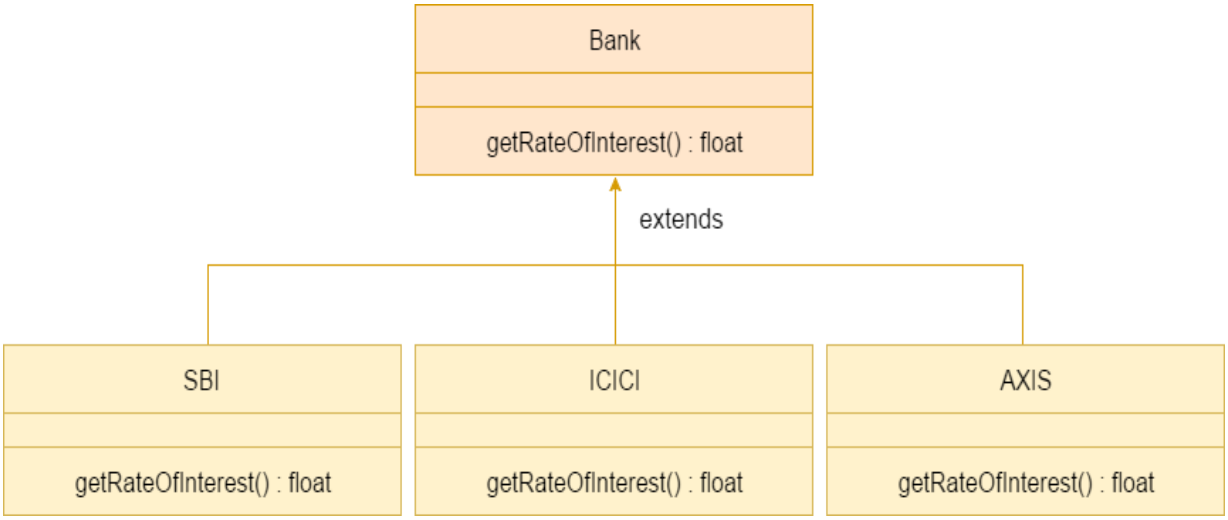
```
class Vehicle
{
   void run()
   {
      System.out.println("Vehicle is running");
   }
}
class Bike2 extends Vehicle
{
   void run()   //overriding run() method from parent class
   {
      System.out.println("Bike is running safely");
   }

   public static void main(String args[])
   {
      Bike2 obj = new Bike2();
      obj.run();
   }
}
```

Output:

```
Bike is running safely
```

Real example of Java Method Overriding & Inheritance:

Consider a scenario, Bank is a class that provides functionality to get rate of interest. But, rate of interest varies according to banks. For example, SBI, ICICI and AXIS banks could provide 8%, 7% and 9% rate of interest.




```
class Bank
{
   int getRateOfInterest()
   {
      return 0;
   }
}

class SBI extends Bank
{
   int getRateOfInterest()
   {
```

```
        return 8;
    }
}

class ICICI extends Bank
{
    int getRateOfInterest()
    {
        return 7;
    }
}
class AXIS extends Bank
{
    int getRateOfInterest()
    {
        return 9;
    }
}

class Test2
{
    public static void main(String args[])
    {
        SBI s=new SBI();
        ICICI i=new ICICI();
        AXIS a=new AXIS();
        System.out.println("SBI Rate of Interest:
"+s.getRateOfInterest());
        System.out.println("ICICI Rate of Interest:
"+i.getRateOfInterest());
        System.out.println("AXIS Rate of Interest:
"+a.getRateOfInterest());
    }
}
```

Output:

```
SBI Rate of Interest: 8
ICICI Rate of Interest: 7
AXIS Rate of Interest: 9
```