

[Home](#) [Java](#) [Programs](#) [OOps](#) [String](#) [Exception](#) [Multithreading](#) [Collections](#)

OBJECT ORIENTED PROGRAMMING

↑ SCROLL TO TOP

Polymorphism in Java

Polymorphism in Java is a concept by which we can perform a *single action in different ways*. Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

There are two types of polymorphism in Java: compile-time polymorphism and runtime polymorphism. We can perform polymorphism in java by method overloading and method overriding.

If you overload a static method in Java, it is the example of compile time polymorphism. Here, we will focus on runtime polymorphism in java.

Runtime Polymorphism in Java

Runtime polymorphism or **Dynamic Method Dispatch** is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

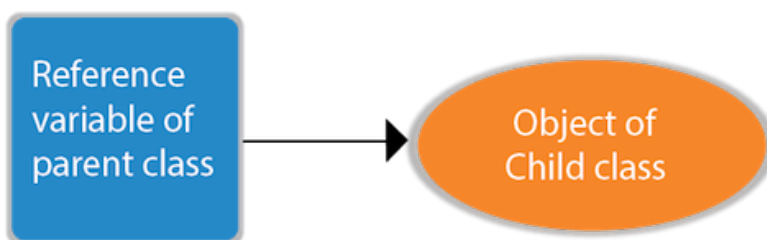


In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

Let's first understand the upcasting before Runtime Polymorphism.

Upcasting

If the 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{}
```

↑ SCROLL TO TOP

... from upcasting

OBJECT ORIENTED PROGRAMMING

For upcasting, we can use the reference variable of class type or an interface type. For Example:

```
interface I{}  
class A{}  
class B extends A implements I{}
```

Here, the relationship of B class would be:

```
B IS-A A  
B IS-A I  
B IS-A Object
```

Since Object is the root class of all classes in Java, so we can write B IS-A Object.

Example of Java Runtime Polymorphism

In this example, we are creating two classes Bike and Splendor. Splendor 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, the subclass method is invoked at runtime.

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

```
class Bike{  
    out.println("running");}
```

↑ SCROLL TO TOP

OBJECT ORIENTED PROGRAMMING

```

class Splendor extends Bike{
    void run(){System.out.println("running safely with 60km");}

    public static void main(String args[]){
        Bike b = new Splendor();//upcasting
        b.run();
    }
}

```

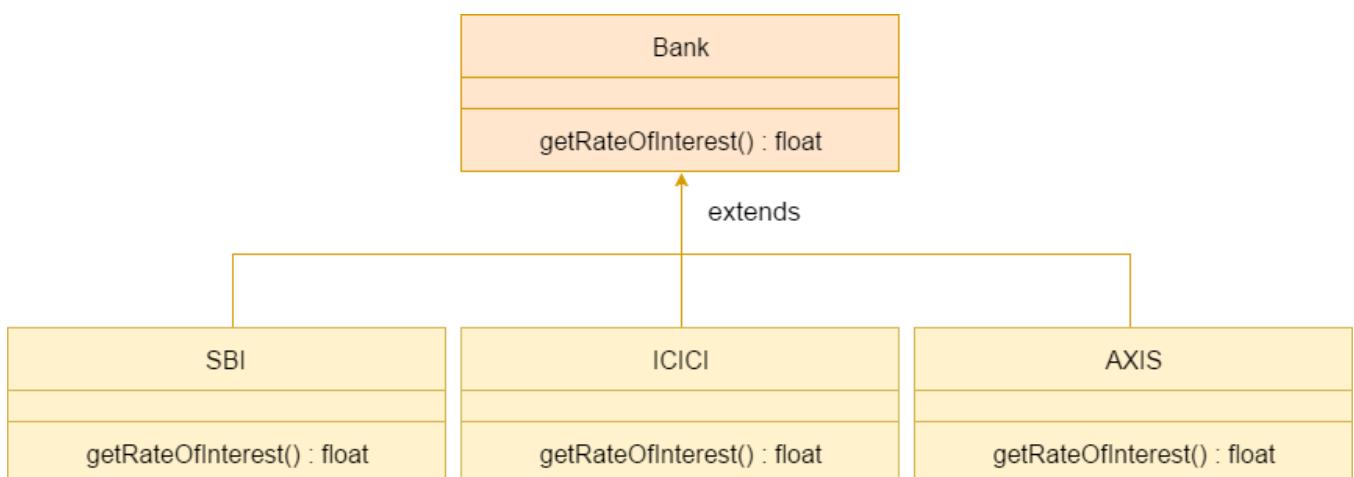
Test it Now

Output:

```
running safely with 60km.
```

Java Runtime Polymorphism Example: Bank

Consider a scenario where Bank is a class that provides a method to get the rate of interest. However, the rate of interest may differ according to banks. For example, SBI, ICICI, and AXIS banks are providing 8.4%, 7.3%, and 9.7% rate of interest.



Note: This example is also given in method overriding but there was no upcasting.

```

class Bank{
    float getRateOfInterest(){return 0;}
}
class SBI extends Bank{
    float getRateOfInterest(){return 8.4f;}
}
class ICICI extends Bank{
    float getRateOfInterest(){return 7.3f;}
}

```

↑ SCROLL TO TOP

Bank{

OBJECT ORIENTED PROGRAMMING

```
float getRateOfInterest(){return 9.7f;}
}
class TestPolymorphism{
public static void main(String args[]){
Bank b;
b=new SBI();
System.out.println("SBI Rate of Interest: "+b.getRateOfInterest());
b=new ICICI();
System.out.println("ICICI Rate of Interest: "+b.getRateOfInterest());
b=new AXIS();
System.out.println("AXIS Rate of Interest: "+b.getRateOfInterest());
}
}
```

Test it Now

Output:

```
SBI Rate of Interest: 8.4
ICICI Rate of Interest: 7.3
AXIS Rate of Interest: 9.7
```

Java Runtime Polymorphism Example: Shape

```
class Shape{
void draw(){System.out.println("drawing...");}
}
class Rectangle extends Shape{
void draw(){System.out.println("drawing rectangle...");}
}
class Circle extends Shape{
void draw(){System.out.println("drawing circle...");}
}
class Triangle extends Shape{
void draw(){System.out.println("drawing triangle...");}
}
class TestPolymorphism2{
public static void main(String args[]){
Shape s;
s=new Rectangle();
s.draw();
s=new Circle();
s.draw();
}
```

↑ SCROLL TO TOP

OBJECT ORIENTED PROGRAMMING

```
s.draw();  
}  
}
```

Test it Now

Output:

```
drawing rectangle...  
drawing circle...  
drawing triangle...
```

Java Runtime Polymorphism Example: Animal

```
class Animal{  
    void eat(){System.out.println("eating...");}  
}  
class Dog extends Animal{  
    void eat(){System.out.println("eating bread...");}  
}  
class Cat extends Animal{  
    void eat(){System.out.println("eating rat...");}  
}  
class Lion extends Animal{  
    void eat(){System.out.println("eating meat...");}  
}  
class TestPolymorphism3{  
    public static void main(String[] args){  
        Animal a;  
        a=new Dog();  
        a.eat();  
        a=new Cat();  
        a.eat();  
        a=new Lion();  
        a.eat();  
    }  
}
```

Test it Now

Output:

A screenshot of a window titled "OBJECT ORIENTED PROGRAMMING". The window has a black border and a close button (a circle with an 'X') in the top right corner. The text "OBJECT ORIENTED PROGRAMMING" is centered in the window in a bold, black, sans-serif font.[↑ SCROLL TO TOP](#)

```
eating bread...  
eating rat...  
eating meat...
```

Java Runtime Polymorphism with Data Member

A method is overridden, not the data members, so runtime polymorphism can't be achieved by data members.

In the example given below, both the classes have a data member speedlimit. We are accessing the data member by the reference variable of Parent class which refers to the subclass object. Since we are accessing the data member which is not overridden, hence it will access the data member of the Parent class always.

Rule: Runtime polymorphism can't be achieved by data members.

```
class Bike{  
    int speedlimit=90;  
}  
class Honda3 extends Bike{  
    int speedlimit= 150;  
  
    public static void main(String args[]){  
        Bike obj=new Honda3();  
        System.out.println(obj.speedlimit);//90  
    }  
}
```

Test it Now

Output:

OBJECT ORIENTED PROGRAMMING

↑ SCROLL TO TOP

Java Runtime Polymorphism with Multilevel Inheritance

Let's see the simple example of Runtime Polymorphism with multilevel inheritance.

```
class Animal{
void eat(){System.out.println("eating");}
}
class Dog extends Animal{
void eat(){System.out.println("eating fruits");}
}
class BabyDog extends Dog{
void eat(){System.out.println("drinking milk");}
public static void main(String args[]){
Animal a1,a2,a3;
a1=new Animal();
a2=new Dog();
a3=new BabyDog();
a1.eat();
a2.eat();
a3.eat();
}
}
```

Test it Now

Output:

```
eating
eating fruits
drinking Milk
```

Try for Output

```
class Animal{
void eat(){System.out.println("animal is eating...");}
}
class Dog extends Animal{
void eat(){System.out.println("dog is eating...");}
}
class BabyDog1 extends Dog{
public static void main(String args[]){
Animal a=new BabyDog1();
}
```

↑ SCROLL TO TOP

OBJECT ORIENTED PROGRAMMING

Test it Now

Output:

```
Dog is eating
```

Since, BabyDog is not overriding the eat() method, so eat() method of Dog class is invoked.

[< Prev](#)[Next >](#)

 **For Videos Join Our Youtube Channel: [Join Now](#)**

Feedback

- Send your Feedback to feedback@javatpoint.com






Help Others, Please Share








OBJECT ORIENTED PROGRAMMING

[↑ SCROLL TO TOP](#)













Learn Latest Tutorials

 Splunk tutorial Splunk	 SPSS tutorial SPSS	 Swagger tutorial Swagger	 T-SQL tutorial Transact-SQL	 Tumblr tutorial Tumblr
 React tutorial ReactJS	 Regex tutorial Regex	 Reinforcement learning tutorial Reinforcement Learning	 R Programming tutorial R Programming	 RxJS tutorial RxJS
 React Native tutorial React Native	 Python Design Patterns Python Design Patterns	 Python Pillow tutorial Python Pillow	 Python Turtle tutorial Python Turtle	 Keras tutorial Keras

Preparation







 Aptitude Aptitude	 Logical Reasoning Reasoning	 Verbal Ability Verbal Ability	 Interview Questions Interview Questions	 Company Interview Questions Company Questions
--	--	--	--	--

Trending Technologies

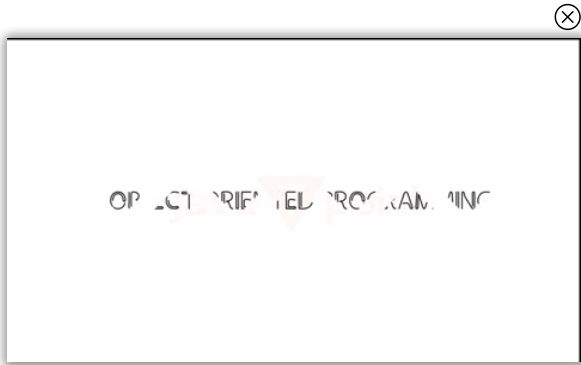
 Artificial Intelligence Tutorial Artificial Intelligence	 AWS Tutorial AWS	 Selenium tutorial Selenium	 Cloud Computing tutorial Cloud Computing	 Hadoop tutorial Hadoop
 ReactJS Tutorial ReactJS	 Data Science Tutorial Data Science	 Angular 7 Tutorial Angular 7	 Blockchain Tutorial Blockchain	 Git Tutorial Git
 Machine Learning Tutorial Machine Learning	 DevOps Tutorial DevOps			

OBJECT ORIENTED PROGRAMMING

B.Tech / MCA

 DBMS tutorial DBMS	 Data Structures tutorial Data Structures	 DAA tutorial DAA	 Operating System tutorial Operating System	 Computer Network tutorial Computer Network
 Compiler Design tutorial Compiler Design	 Computer Organization and Architecture Computer Organization	 Discrete Mathematics Tutorial Discrete Mathematics	 Ethical Hacking Tutorial Ethical Hacking	 Computer Graphics Tutorial Computer Graphics
 Software Engineering Tutorial Software Engineering	 html tutorial Web Technology	 Cyber Security tutorial Cyber Security	 Automata Tutorial Automata	 C Language tutorial C Programming
 C++ tutorial C++	 Java tutorial Java	 .Net Framework tutorial .Net	 Python tutorial Python	 List of Programs Programs
 Control Systems tutorial Control System	 Data Mining Tutorial Data Mining	 Data Warehouse Tutorial Data Warehouse		

OBJECT ORIENTED PROGRAMMING



↑ SCROLL TO TOP