Bahria University
Discovering Knowledge

Object Oriented Programming

Topic: Polymorphism,

Late binding and Early binding

Bahria University
Discovering Knowledge

Polymorphism

The word polymorph is a combination of two words namely, ‘poly’ which means ‘many’ and ‘morph’ which means ‘forms’.

Thus, polymorph refers to an object that can have many different forms.

This principle can also be applied to subclasses of a class that can define their own specific behaviors as well as derive some of the similar functionality of the super class.

The concept of method overriding is an example of polymorphism in object-oriented programming in which the same method behaves in a different manner in super class and in subclass.

Bahria University
Discovering Knowledge

Method overriding

Java allows creation of an instance method in a subclass having the same signature and return type as an instance method of the super class.
This is called method overriding.
Method overriding allows a class to inherit behavior from a super class and then, to modify the behavior as needed.
Rules to remember when overriding:

The overriding method must have the same name, type, and number of arguments as well as return type as the super class method.

An overriding method cannot have a weaker access specifier than the access specifier of the super class method.

Bahria University
Discovering Knowledge

Example

```
class Animal
 {
public void move()
{
System.out.println("Animals can move");
}
}
class Dog extends Animal
{
 public void move()
{
 System.out.println("Dogs can walk and run");
}
 }
public class TestDog {
public static void main(String args[])
{
 Animal a = new Animal();
Animal b = new Dog();
a.move();
b.move();
}
```

- Some important differences between static and dynamic binding are listed in the following table:

Static Binding	Dynamic Binding
Static binding occurs at compile time.	Dynamic binding occurs at runtime.
Private, static, and final methods and variables use static binding and are bounded by compiler.	Virtual methods are bounded at runtime based upon the runtime object.
Static binding uses object type information for binding. That is, the type of class.	Dynamic binding uses reference type to resolve binding.
Overloaded methods are bounded using static binding.	Overridden methods are bounded using dynamic binding.

Bahria University
Discovering Knowledge

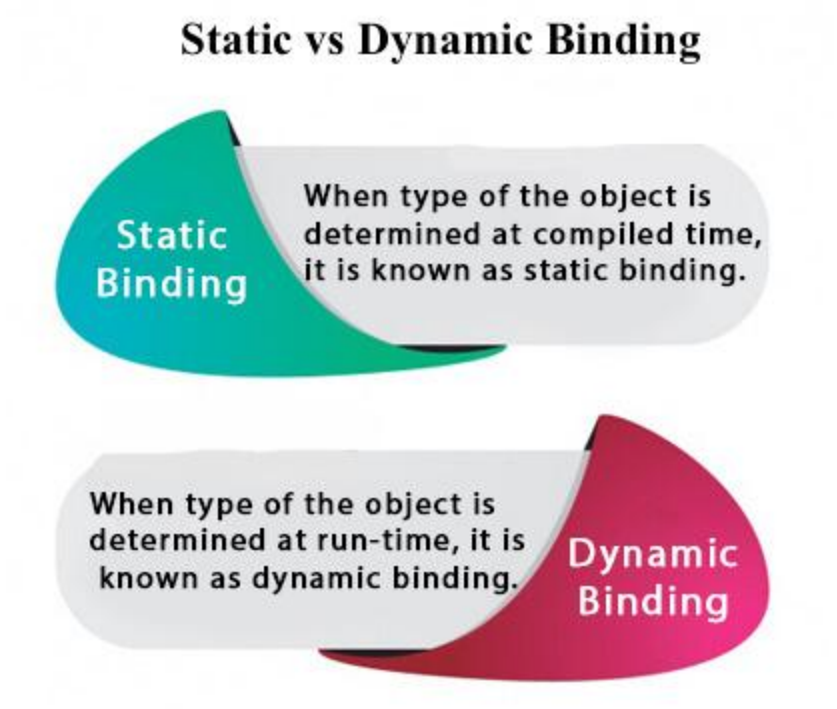
Example

```
class FourWheeler extends Vehicle{
    private boolean powerSteer;
    public FourWheeler(String vId, String vName, int numWheels, boolean pSteer){
        vehicleNo=vId;
        vehicleName=vName;
        wheels=numWheels;
        powerSteer=pSteer;
    }
    public void showDetails() {
        System.out.println(“Vehicle no:”+ vehicleNo);
        System.out.println(“Vehicle Name:”+ vehicleName);
        System.out.println(“Number of Wheels:”+ wheels);
        if(powerSteer==true){
            System.out.println(“Power Steering:Yes”);
        else
           System.out.println(“Power Steering:No”);
    }
    public void accelerate(int speed) {
        System.out.println(“Maximum acceleration:”+ speed + “ kmph”);
    }
}
public class TestVehicle {
    public static void main(String[] args) {
        FourWheeler objFour = new FourWheeler(“LA-09 CS-1406”, “Volkswagen”,4, true);
        objFour.showDetails();
        objFour.accelerate(200);
    }
}
```

Connecting a method call to the method body is known as binding.

There are two types of binding

1. Static Binding (also known as Early Binding).
2. Dynamic Binding (also known as Late Binding).


Static vs Dynamic Binding
Static Binding
When type of the object is determined at compiled time, it is known as static binding.
When type of the object is determined at run-time, it is known as dynamic binding.
Dynamic Binding

Bahria University
Discovering Knowledge

Understanding Type

Let's understand the type of instance.

1) variables have a type

Each variable has a type, it may be primitive and non-primitive.

```
int data=30;
```

Here data variable is a type of int.

2) References have a type

```
class Dog{
 public static void main(String args[])
{
  Dog d1;//Here d1 is a type of Dog
 }
}
```

3) Objects have a type

An object is an instance of particular java class,but it is also an instance of its superclass.

```
class Animal{}

class Dog extends Animal
{
 public static void main(String args[]){
  Dog d1=new Dog();
 }
}
```

Here d1 is an instance of Dog class, but it is also an instance of Animal.

static binding

When type of the object is determined at compiled time(by the compiler), it is known as static binding.
If there is any private, final or static method in a class, there is static binding.

Example of static binding

```
class Dog{
 private void eat()
{
System.out.println("dog is eating...");
}
 public static void main(String args[])
{
  Dog d1=new Dog();
  d1.eat();
 }
}
```

Bahria University
Discovering Knowledge

Dynamic binding

When type of the object is determined at run-time, it is known as dynamic binding.

```
class Animal{
 void eat()
{
System.out.println("animal is eating...");}
}

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

In the above example object type cannot be determined by the compiler, because the instance of Dog is also an instance of Animal.So compiler doesn't know its type, only its base type.

Bahria University
Discovering Knowledge

Static Binding Example

```
class Human{ public static void walk()
 {
System.out.println("Human walks");
 }
}
 class Boy extends Human
{
 public static void walk()
{
 System.out.println("Boy walks");
}
 public static void main( String args[])
 {
Human obj = new Boy();
Human obj2 = new Human();
obj.walk();
obj2.walk();
 }
 }
```

OUTPUT

```
Human walks
Human walks
```

Bahria University
Discovering Knowledge

Dynamic binding example

```
class Human
{
public void walk()
{
System.out.println("Human walks");
} }
class Demo extends Human
{
public void walk(){
System.out.println("Boy walks");
 } public static void main( String args[])
{
 Human obj = new Demo();
 Human obj2 = new Human();
 obj.walk();
 obj2.walk();
}
 }
```

Output:

```
Boy walks
 Human walks
```

Bahria University
Discovering Knowledge

Static Binding vs Dynamic Binding

Lets discuss the **difference between static and dynamic binding in Java**.

•Static binding happens at compile-time while dynamic binding happens at runtime.

•Binding of private, static and final methods always happen at compile time since these methods cannot be overridden. When the method overriding is actually happening and the reference of parent type is assigned to the object of child class type then such binding is resolved during runtime.

•The binding of **overloaded methods** is static and the binding of overridden methods is dynamic.