Course Name - Object Oriented Programming using Java

Lecture 21 – Method Overriding & Dynamic Method dispatch

Presented By
Dr. Sudipta Sahana
Associate Professor
Dept. of CSE
UEM - Kolkata

Topic of Interest

- **Method Overriding**
- **Example**
- **Dynamic Method Dispatch Technique**
- **Example**
- **static methods can’t be overridden**

Method Overriding

- Method overriding is a process through which a particular method can be introduced in parent class as well as it's child class with same name, same return type, same signature but different implementational logic.

- In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

Usage of Method Overriding:

- Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
- Method overriding is used for runtime polymorphism.

Rules of Method Overriding:

- The method must have the same name as in the parent class.
- The method must have the same parameter as in the parent class.
- There must be an IS-A relationship (inheritance).

Example

```
class Vehicle {
        void move()
                {
                        System.out.println("Any vehicle can move");
                }
        }
class Two_wheeler extends Vehicle {
        void move()
                {
                        System.out.println("Two wheeler can move with 2 wheel");
                }
        }
class Test {
        public static void main(String[]  args)
                {
                        Two_wheeler  bicycle=new Two_wheeler();
                        bicycle.move();
                }
        }
```

Output:

Two wheeler can move with 2 wheel

Dynamic Method Dispatch Technique

- Dynamic method dispatch is a technique by which call to a overridden method is resolved at runtime, rather than compile time.
- When an overridden method is called by a reference, then which version of overridden method is to be called is decided at runtime according to the type of object it refers.
- Dynamic method dispatch is performed by JVM not compiler.
- It allows java to support overriding of methods and perform runtime polymorphism.
- It allows subclasses to have common methods and can redefine specific implementation for them.
- This lets the superclass reference respond differently to same method call depending on which object it is pointing.

Advantages of Dynamic method dispatch:

- It allow Java to support overriding of methods which is central for run-time polymorphism.
- It allows a class to specify methods that will be common to all of its derivatives, while allowing subclasses to define the specific implementation of some or all of those methods.
- It also allow subclasses to add its specific methods subclasses to define the specific implementation of some.

Example

```
class Base
{
void show(){System.out.println("Hello");   }}
class Child extends Base
{void show(){   System.out.println("Born!");   }}
class A
{ public static void main(String[] args){
Base ref=new Base();
ref.show(); // output: Hello
ref=new Child(); // the ref to Base can hold obj of Child
//But converse isn't true without explicit type cast.
ref.show(); // output: Born!  it's because, the
// ref although being a reference of Base, holds
//object of Child
// this is runtime( ? ) polymorphism or dynamic method dispatch
}}
```

static methods can’t be overridden

```
class Base
{
static void show(){
        System.out.println("Hello");
   }
}
class Child extends Base
{
static void show(){
        System.out.println("Born!");
   }
}
```

```
class A
{
public static void main(String[] rt){
 Base ref=new Child();
 ref.show();
}
}
```

Output: Hello.

So, static methods are not overridden as they don’t follow the rule of dynamic method dispatch

// either both method should be static or neither should. Well, let’s make both static. Now it compiles. But overriding is not possible. Create an object of Child and store to a Base’s reference. Call the method show(). But Dynamic method dispatch is never seen.

UEM
UNIVERSITY OF ENGINEERING & MANAGEMENT
Good Education, Good Jobs

Thank You