Java & JEE Training

Day 10 – OOPs with Java & Exception Handling

MindsMapped Consulting

Java & JEE Training

Review of OOPs concepts (Last class)

MindsMapped Consulting

Difference between method overloading and overriding?

- Explain.

Point to ponder..

Can we override static methods in Java?

Point to ponder:

Can constructors be declared final?

Agenda

- ✓ Overview of OOP continued…
  - ✓ Abstraction – using Abstract Classes and Interfaces.

Abstraction using abstract class and method

```
abstract class Bike{
  abstract void run();
}
class Honda4 extends Bike{
    void run(){System.out.println("running safely..");}
    public static void main(String args[]){
     Bike obj = new Honda4();
     obj.run();
    }
}
```

Example:

```
abstract class Shape{
        abstract void draw();
}
//In real scenario, implementation is provided by others i.e. unknown by end user
class Rectangle extends Shape{
        void draw(){System.out.println("drawing rectangle");}
}
class Circle1 extends Shape{
        void draw(){System.out.println("drawing circle");}
}
//In real scenario, method is called by programmer or user
class TestAbstraction1{
    public static void main(String args[]){
    Shape s=new Circle1();//In real scenario, object is provided through method e.g.
    getShape() method
    s.draw();
    }
}
```

Example:

```
abstract class Bank{
        abstract int getRateOfInterest();
}
class SBI extends Bank{
        int getRateOfInterest(){return 7;}
}
class PNB extends Bank{
        int getRateOfInterest(){return 8;}
}

class TestBank{
    public static void main(String args[]){
    Bank b;
    b=new SBI();
    System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
    b=new PNB();
    System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
}}
```

Abstract class rule...

If there is any abstract method in a class, that class must be abstract.

```
class Bike12{
abstract void run();
}

//Compile time error
```

Abstract class rule

If you are extending any abstract class that have abstract method, you must either provide the implementation of the method or make this class abstract.

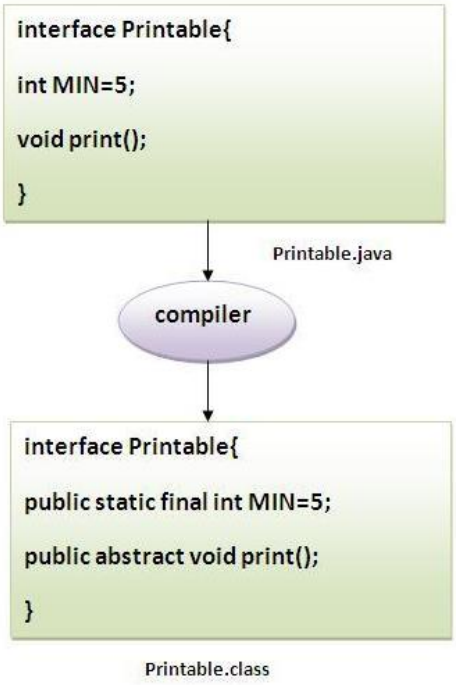
Abstraction with interfaces

```
interface A{
	void a();
	void b();
	void c();
	void d();
}

abstract class B implements A{
		public void c(){System.out.println("I am C");}
}

class M extends B{
	public void a(){System.out.println("I am a");}
	public void b(){System.out.println("I am b");}
	public void d(){System.out.println("I am d");}
}

class Test5{
	public static void main(String args[]){
	A a=new M();
	a.a();
	a.b();
	a.c();
	a.d();
}}
```

Interfaces in Java

- An **interface in java** is a blueprint of a class. It has static constants and abstract methods only.
- The interface in java is **a mechanism to achieve fully abstraction**.

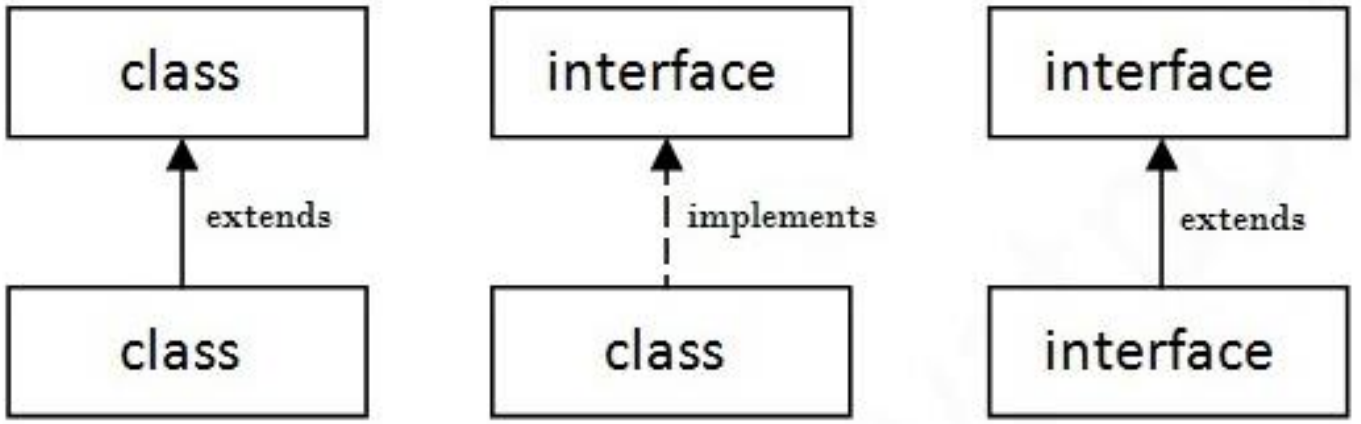
- **The java compiler adds public and abstract keywords before the interface method and public, static and final keywords before data members.**



Why use Java interface?

- It is used to achieve fully abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

Relationship between classes and interfaces



Simple interface example

```
interface printable{
          void print();
}

class A6 implements printable{
        public void print(){System.out.println("Hello");}

        public static void main(String args[]){
        A6 obj = new A6();
        obj.print();
         }
}
```

Multiple inheritance through interfaces

Multiple inheritance example

```
interface Printable{
        void print();
}

interface Showable{
        void show();
}

class A7 implements Printable,Showable{

    public void print(){System.out.println("Hello");}
    public void show(){System.out.println("Welcome");}

    public static void main(String args[]){
    A7 obj = new A7();
    obj.print();
    obj.show();
     }
}
```

Interface inheritance example

```
interface Printable{
        void print();
}
interface Showable extends Printable{
        void show();
}
class Testinterface2 implements Showable{

    public void print(){System.out.println("Hello");}
    public void show(){System.out.println("Welcome");}

    public static void main(String args[]){
    Testinterface2 obj = new Testinterface2();
    obj.print();
    obj.show();
     }
}
```

Point to ponder…

Multiple inheritance is not supported through class in java but it is possible by interface, why?