Interface & Abstraction in Java

MODULE NAME: OBJECT ORIENTED PROGRAMMING WITH JAVA

MODULE CODE: CST 05206

INSTRUCTORS: JAMILA L.

Interface Definition

An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods.

A class implements an interface, thereby inheriting the abstract methods of the interface.

- The interface in Java is *a mechanism to achieve* ***abstraction***.
- There can be only abstract methods in the Java interface, not method body.
- It is used to achieve abstraction and **multiple inheritance in Java.**

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Interface Definition

Interfaces have the following properties:

- An interface is implicitly abstract. You do not need to use the **abstract** keyword while declaring an interface.
- Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.
- Methods in an interface are implicitly public.

There are mainly three reasons to use interface. They are given below.

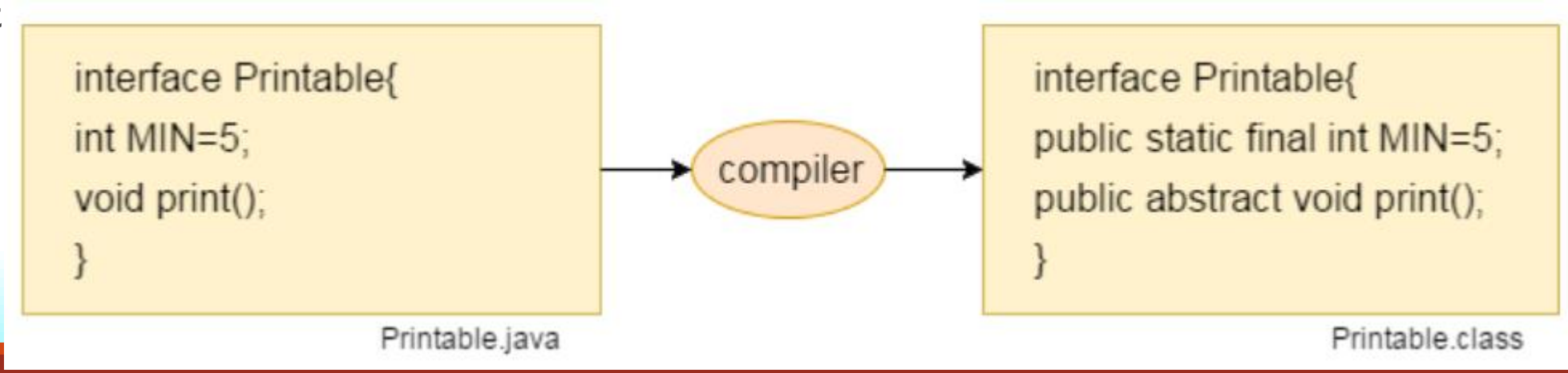
1. It is used to achieve abstraction.
2. By interface, we can support the functionality of multiple inheritance.
3. It can be used to achieve loose coupling.

Interface declaration

How to declare an interface?

- An interface is declared by using the interface keyword.
- It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default.
- A class that implements an interface must implement all the methods declared in the interface.

In other words, Interface fields are public, static and final by default, and the methods are public and abst



Interface declaration cont…

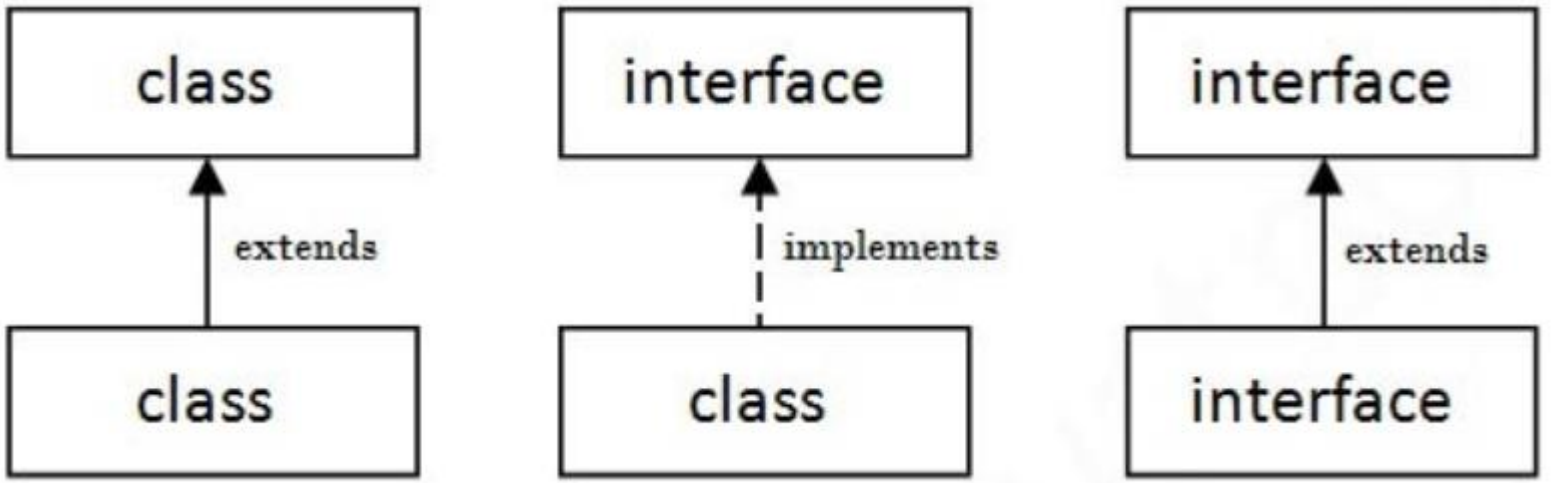
Syntax:

```
interface <interface_name>{

    // declare constant fields
    // declare methods that abstract
    // by default.
}
```

The relationship between classes and interfaces

As shown in the figure given below, a class extends another class, an interface extends another interface, but a **class implements an interface**.



```
//Example


/* File name : Animal.java */
interface Animal {
   public void eat();
   public void travel();
}
```

Implementing Interfaces

- When a class implements an interface, you can think of the class as signing a contract, agreeing to perform the specific behaviors of the interface.
- A class uses the **implements** keyword to implement an interface.

Implementing interface

```
/* File name : Mammal.java */
public class Mammal implements Animal {
   public void eat() {
      System.out.println("Mammal eats");
   }
   public void travel() {
      System.out.println("Mammal travels");
   }
   public int noOfLegs() {
      return 0;
   }
```

Implementing interface cont…

```
/* Main Class */
public static void main(String args[]) {
        Mammal m = new Mammal();
        m.eat();
        m.travel();
    }
}
```

Abstraction in Java

Abstraction definition

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

- Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.
- Abstraction lets you focus on what the object does instead of how it does it.

Abstract declaration

A class which is declared as abstract is known as an **abstract class**.

It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Points to Remember:

- An abstract class must be declared with an **abstract keyword**.
- It can have abstract and non-abstract methods.
- It cannot be **instantiated**.
- It can have **constructors** and **static methods** also.
- It can have **final methods** which will force the subclass not to change the body of the method.

Abstract Method

A method which is declared as abstract and does not have implementation is known as an abstract method.

Example of abstract method:

```
abstract void printStatus();
```

Example of Abstract class

```
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();
}
}
```

Vs.

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.
6) An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
7) An **abstract class** can be extended using keyword "extends".	An **interface** can be implemented using keyword "implements".
8) A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.

9)**Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

Example of Interface and Abstract

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

Example of Interface and Abstract cont…

```
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");}
}
```

Example of Interface and Abstract cont…

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

Thank you!

for listen