Interfaces
Lecture 16

Prof. Anita Agrawal

BITS- Pilani,K.K.Birla Goa campus

What is an interface?

- Is a core part of java programming.
- It is a reference type in Java.
- It is similar to class.
- It is a collection of abstract methods only???.

Why Interfaces?

- To achieve **total abstraction**.
- To define the **contract** for the subclasses to implement.
- To support **multiple inheritance**.

Interfaces can also contain: (Java 8/9)

- Only constants
- Default methods/defender/virtual extension (consists of body)
- Static methods (consists of body)

Differences between interface and class

- **Interfaces cannot be instantiated.**
- **Classes can be instantiated.**

- **Interfaces cannot contain constructors**
- **Classes contain constructors**

- **Interfaces are used to achieve multiple inheritance**
- **Classes cannot be used to achieve multiple inheritance**

- **Interfaces cannot be declared as private, protected. They are by default public.**
- **Classes can be.**

Differences between Abstract class and Interface

- Variables in interface are by default public, static and final.

An abstract class may contain variables which are final, non-final, static, nonstatic, private, protected etc.

- A Java interface can be implemented using keyword “implements

Abstract class can be extended using keyword “extends”

- Interface can’t provide the implementation of abstract class

Abstract class can provide the implementation of interface.

- Interface can have only abstract methods. From Java 8, it can have default and static methods also.

Abstract class can have abstract and non-abstract methods.

- An interface can extend another Java interface only, it cannot extend any class.

An abstract class can extend another class and implement multiple interfaces.

- While providing implementation of any method in class of an interface, it needs to be mentioned as public.

This is not the requirement in Abstract classes

Some more properties of Interfaces

- An interface is implicitly abstract. We need not explicitly declare it as such.
- Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.
- Methods in an interface are implicitly public
- If a class implements an interface and does not provide method bodies for all functions specified in the interface, then class must be declared abstract.

Implementing Interfaces

- A class formally implements an interface by
  - stating so in the class header in the implements clause
  - a class can implement multiple interfaces: the interfaces are listed in the implements clause, separated by commas

- If a class asserts that it implements an interface, it must define all methods in the interface or the compiler will produce errors

Multiple inheritance in Java by interface

- If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.

- Multiple inheritance is not supported in the case of class because of ambiguity. However, it is supported in case of an interface because there is no ambiguity. It is because its implementation is provided by the implementation class

Extending interfaces

- An interface can extend another interface in the same way that a class can extend another class.
- **Implements** keyword is used when a class extends an interface and **extend** keyword is used when an interface extends an interface
- The child interface/class inherits the methods of the parent interface.
- An interface can extend more than one parent interface and the class can extend more than one parent interface too.(multiple inheritance)
- The extends keyword is used once, and the parent interfaces are declared in a comma-separated list.

Interface Hierarchies

- Inheritance can be applied to interfaces as well as classes
- One interface can be used as the parent of another
- The child interface inherits all abstract methods of the parent
- A class implementing the child interface must define all methods from both the parent and child interfaces
- Note that class hierarchies and interface hierarchies are distinct (they do not overlap)

Nested Interface

- An interface can be declared inside another interface also.
- We mention the interface as i_**name1.i_name2**

i_name1 is the name of the interface in which it is nested and

i_name2 is the name of the interface to be implemented.

Static methods in interfaces

- **Static Methods** in **Interface** are those methods, which are defined in the interface with the keyword static.
- The static method in an interface can be defined in the interface, but these methods cannot be overridden in Implementation Classes.
- To use a static method, Interface name should be instantiated with it, as it is a part of the Interface only.

Default methods

- Java 8 introduces the “Default Method” or (Defender methods) feature, which allows the developer to add new methods to the interfaces without breaking their existing implementation
- ***Default methods*** have introduced as a mechanism to extend interfaces in a backward-compatible way
- ***Default methods*** can be provided to an interface without affecting implementing classes as it includes an implementation