Java - Interfaces

By
Basharat Jehan
Lecturer Agriculture University
Peshawar, Pakistan

- An interface is a collection of abstract methods. A class implements an interface

- An interface is not a class. Writing an interface is similar to writing a class, but they are two different concepts. A class describes the attributes and behaviors of an object. An interface contains behaviors that a class implements.

- An interface is similar to a class in the following ways:
- An interface can contain any number of methods.
- An interface is written in a file with a **.java** extension, with the name of the interface matching the name of the file.
- The bytecode of an interface appears in a **.class** file.

Declaring Interfaces

- The **interface** keyword is used to declare an interface. Here is a simple example to declare an interface:
- **Example:**
- Let us look at an example that depicts encapsulation:
- /* File name : NameOfInterface.java */
- import java.lang.*;

  //Any number of import statements
- public interface NameOfInterface
- { //Any number of final, static fields //Any number of abstract method declarations\ }

Example:

```
interface Animal
{
 public void eat();
public void travel();
 }
```

Implementing Interfaces:

- A class uses the **implements** keyword to implement an interface. The implements keyword appears in the class declaration following the extends portion of the declaration.

```
public class MammalInt implements Animal
{
public void eat()
{
 System.out.println("Mammal eats");
 }
 public void travel()
{
 System.out.println("Mammal travels");
 }
public static void main(String args[])
{
MammalInt m = new MammalInt();
 m.eat();
m.travel();
 }
}
```

Output

- Mammal eats
- Mammal travels

Extending Interfaces:

- An interface can extend another interface, similarly to the way that a class can extend another class. The **extends** keyword is used to extend an interface, and the child interface inherits the methods of the parent interface.

Example

```
public interface Sports
{
 public void setHomeTeam(String name);
 public void setVisitingTeam(String name);
}
 //Filename: Football.java
public interface Football extends Sports
{
 public void homeTeamScored(int points);
public void visitingTeamScored(int points);
public void endOfQuarter(int quarter);
} //Filename: Hockey.java
public interface Hockey extends Sports
{
 public void homeGoalScored();
public void visitingGoalScored();
public void endOfPeriod(int period);
public void overtimePeriod(int ot);
}
```

- The Hockey interface has four methods, but it inherits two from Sports; thus, a class that implements Hockey needs to implement all six methods. Similarly, a class that implements Football needs to define the three methods from Football and the two methods from Sports.

Extending Multiple Interfaces:

- A Java class can only extend one parent class. Multiple inheritance is not allowed. Interfaces are not classes, however, and an interface can extend more than one parent interface.

- The extends keyword is used once, and the parent interfaces are declared in a comma-separated list.
- For example, if the Hockey interface extended both Sports and Event, it would be declared as:

public interface Hockey extends Sports, Event

Tagging Interfaces:

- The most common use of extending interfaces occurs when the parent interface does not contain any methods. For example, the MouseListener interface in the java.awt.event package extended java.util.EventListener, which is defined as:

  package java.util;

  public interface EventListener {}

  An interface with no methods in it is referred to as a **tagging** interface.