Interfaces

Shristi Technology Labs

Contents

- Introduction to interfaces
- Syntax
- Why Interface?
- Extending Interface
- Default Methods

Interfaces

- Helps to achieve multiple implementations
- Pure abstract class(can’t be instantiated).
- Can have ***abstract*** methods, ***default*** methods, ***static*** *methods*
- Variables are ***public static final*** *by default*
- Interfaces must be implemented by classes to give functionality to the methods

```
interface interfaceName
interface Runnable{}
```

Syntax

```
interface interfacename{
   // public  static  final variables
   // abstract methods
  // default methods
 //static methods
}
interface flyable{
 int v = 100;
 void fly();
 default void check(){ }
 }
```

```
interface  flyable{
   void fly();
}

 class Man implements Flyable {
     public void fly(){
   System.out.println(“in planes”);
}
```

```
class Plane implements Flyable{
      public  void fly(){
System.out.println (“using engines”);
  }
}
class Bird implements Flyable{
     public void fly(){
 System.out.println(“using wings”);
     }
}
```

Calling methods on interfaces

- The implementation classes are hidden from the user and the methods are called using interface reference.

```
Flyable ref = new Man();
 ref.fly();
ref = new Plane();
 ref.fly();
ref = new Bird();
 ref.fly();
```

More on interfaces

- Using interfaces abstraction is achieved.
- Gives scope for extension in future
- A class can implement multiple interfaces
- An interface can extend another interface.
- An interface without methods is called ***Marker or Tag interface (eg. Serializable, Cloneable)***
- An interface with only one abstract method is called as **Functional Interface**

Why interfaces?

- ***Multiple classes can implement an interface*** to come under a common category

```
class Person implements Flyable{
        public void fly(){    print("in planes");            }
class Plane implements Flyable{
        public void fly(){      print("using engines");  }
class Bird implements Flyable{
        public void fly(){      print("using wings");    }
```

Why interfaces?

- ***A class can implement multiple interfaces*** to get different functionality.

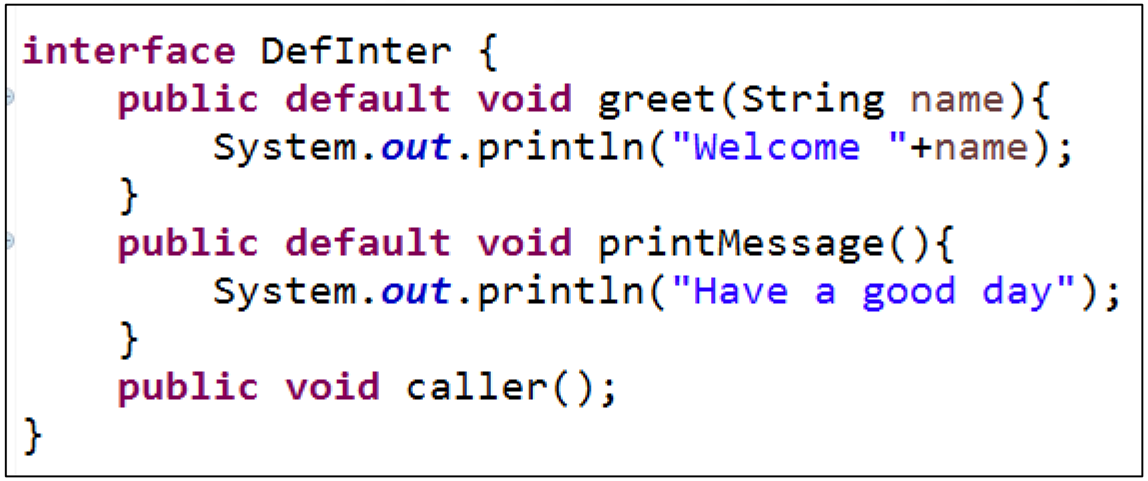
```
interface Swimmable{
   void swim(){ }
}
class Bird implements Flyable, Swimmable{
 public  void fly(){  }
 public  void swim(){ }
}
```

Extending Interfaces

```
interface Calculator {
   void calculate(int x, int y){ }
}
interface  Scientific extends Calculator{
   void square(int x){ }
}
class AdvCalc implements Scientific{
   void calculate(int x, int y){ }
 void square(int x, int y){ }
}
```

Default Methods

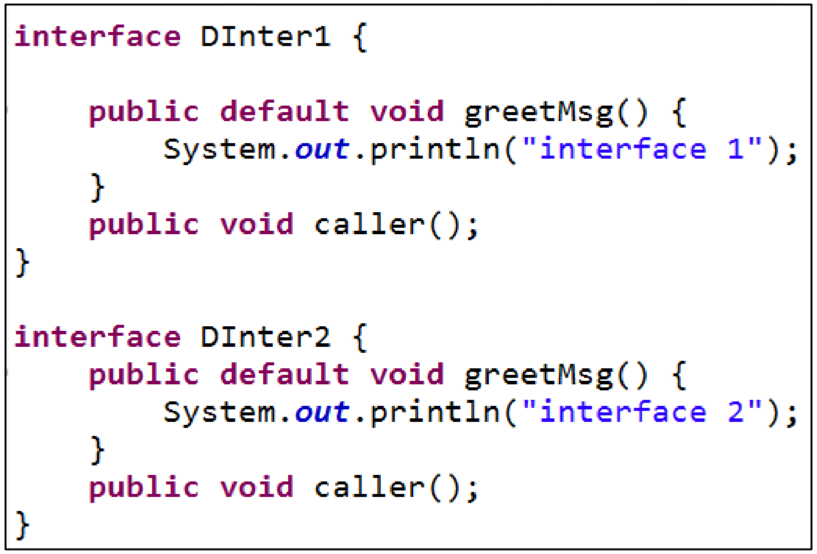
- helps to add new functionality to the existing interfaces of the application
- Adding a default method to an existing interface does not break the contract
- default methods are implicitly public
- Helps to add a common behavior across all implementing classes of the interface



```java
interface DefInter {
    public default void greet(String name){
        System.out.println("Welcome "+name);
    }
    public default void printMessage(){
        System.out.println("Have a good day");
    }
    public void caller();
}
```

Same Default Methods – in two interfaces

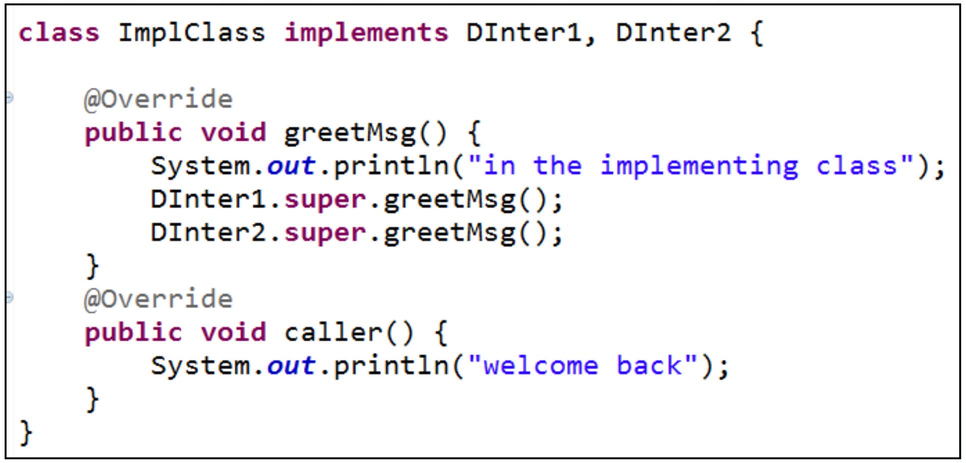
- If default methods have same name in two interfaces, then the implementation class must override them or else will give compiler error



```java
interface DInter1 {

    public default void greetMsg() {
        System.out.println("interface 1");
    }
    public void caller();
}

interface DInter2 {
    public default void greetMsg() {
        System.out.println("interface 2");
    }
    public void caller();
}
```



```java
class ImplClass implements DInter1, DInter2 {

    @Override
    public void greetMsg() {
        System.out.println("in the implementing class");
        DInter1.super.greetMsg();
        DInter2.super.greetMsg();
    }
    @Override
    public void caller() {
        System.out.println("welcome back");
    }
}
```

Summary

- Introduction to interfaces
- Syntax
- Why Interface?
- Extending Interface
- Default Methods

Shristi Technology Labs

Thank you