Interfaces

Agenda

1 Introduction to interfaces

2 Applying Interfaces

Introduction to Interfaces

What is an Interface?

An interface is a named collection of method declarations (without implementations)

- An interface can also include constant declarations
- An interface is syntactically similar to an abstract class
- An interface is a collection of abstract methods and final variables
- A class implements an interface using the **implements** clause

Interface members

- All the methods that are declared within an interface are always, by default, *public* and *abstract*

- Any variable declared within. an interface is always, by default, *public static* and *final.*
    - *That means, all the variables are constant.*

Why interfaces are required ?

- Interfaces allow you to implement common behaviors in different classes that are not related to each other

- Interfaces are used to describe behaviors that are not specific to any particular kind of object, but common to several kind of objects

- Interfaces are implemented by unrelated classes.
    - So, any change in one of the implementing classes **does not affect** the other.
    - This reduces the ripple effect. So these components are loosely coupled.

Why interfaces are required ? (Contd.).

- Thus, an interface is a means of specifying a **consistent specification**, the implementation of which can be different across **many independent and unrelated classes** to suit the respective needs of such classes

- A class can implement more than one interface by giving a comma- separated list of interfaces

- Java does not support multiple inheritance; But, interface concept gives programmers an equivalent to multiple inheritance.

- Interfaces reduce coupling between components in your software

Why interfaces are required ? (Contd.).

- An interfaces allows us to abstract particular behavior in a group of classes.

- If these classes can be grouped in a hierarchy, we can use abstract classes.

- But when the requirement is for implementing common behaviors in different unrelated classes, we require interfaces.

What will you choose..?

What is the behavior which is common among the entities depicted in the pictures above?

Yes..You are right. All of them can fly.

Requirement : You have to develop 3 classes, Bird, Superman and Aircraft with the condition that all these classes must have a method called fly().

What is the mechanism, using which you can ensure that the method fly() is implemented in all these classes?

An Abstract class or An Interface?

Defining an Interface

- An interface is syntactically similar to a class

- It's general form is:

```
public interface FirstIface {
    int addMethod(int x, int y);
    float divMethod(int m, int n);
    void display();
    int N= 10;
    float PI = 3.14f;
}
```

Implementing Interfaces

```java
class FirstImpl implements FirstIface{
    public int  addMethod(int a, int b){
        return(a+b);
    }
    public float divMethod(int i, int j){
        return(i/j);
    }

    public void display(){
        System.out.println("N =" +N);
        System.out.println("PI =" +PI);
    }
}
```

Quiz

Will the following code compile successfully ?

```
interface I1 {
        private int a=100;
        protected void m1();
}

class A1 implements I1 {
        public void m1() {
        System.out.println("In m1 method");
        }
}
```
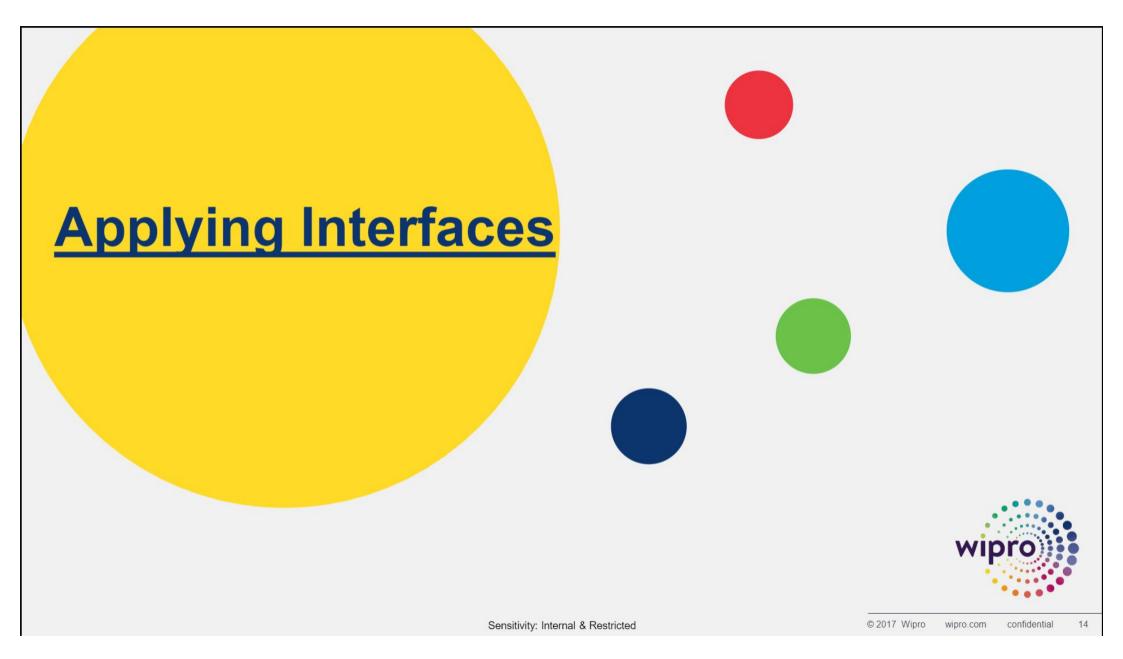
It will throw compilation errors.. Why?

Quiz (Contd.).

Will the following code compile successfully ?

```
interface I1 {
      static int a=100;
      static void m1();
}

class A1 implements I1 {
      public void m1() {
      System.out.println("In m1 method");
      }
}
```

It will throw compilation error.. Why?

Applying Interfaces

Applying Interfaces

- Software development is a process where constant changes are likely to happen

- There can be changes in requirement, changes in design, changes in implementation

- Programming through interfaces helps create software solutions that are reusable, extensible, and maintainable

Applying Interfaces (Contd.).

```java
interface DemoIface{
    void display();

}
class OneImpl implements DemoIface{
    void add(int x, int y){
        System.out.println("The sum is :" +(x+y));

    }
    public void display(){
        System.out.println("Welcome to Interfaces");

    }

}
```

Applying Interfaces (Contd.).

```java
class TwoImpl implements DemoIface{
    void multiply(int i,int j, int k) {
        System.out.println("The result:" +(i*j*k) );
    }
    public void display(){ System.out.println("Welcome to Java ");}
}
class DemoClass{
    public static void main(String args[]) {
        OneImpl c1= new OneImpl();
        TwoImpl c2 = new TwoImpl();
                c1.add(10,20);                      c1.display();
                c2.multiply(5,10,15);               c2.display();
    }
}
```

Interface References

```
interface InterfaceDemo{
    void display();
}
class classOne implements InterfaceDemo{
    void add(int x, int y){
        System.out.println("The sum is :" +(x+y));
    }
    public void display(){
        System.out.println("Class one display method ");
    }
}
```

Interface References (Contd.).

```java
class classTwo implements InterfaceDemo {
    void multiply(int i,int j, int k){
        System.out.println("The result:" +(i*j*k) );
    }
    public void display(){
        System.out.println("Class two display method" );
    }
}
class DemoClass{
    public static void main(String args[]){
        InterfaceDemo c1= new classOne();
        c1.display();
        c1 = new classTwo();
        c1.display();
    }
}
```

Extending Interfaces

- Just as classes can be inherited, interfaces can also be inherited

- One interface can extend one or more interfaces using the keyword **extends**

- When you implement an interface that extends another interface, you should provide implementation for all the methods declared within the interface hierarchy

Marker Interface

- An Interface with no method declared in it, is known as Marker Interface

- Marker Interface is provided as a handle by java interpreter to mark a class, so that it can provide special behavior to it at runtime

- Examples of Marker Interfaces :
    - java.lang.Cloneable
    - java.io.Serializable
    - java.rmi.Remote

Quiz

Will the following code compile successfully ?

```
interface I1 {
        int a=100;
        void m1();
}

class A1 extends I1 {
        public void m1() {
        System.out.println("In m1 method");
        }
}
```

It will throw compilation error.. Why?

Quiz (Contd.).

Will the following code compile successfully ?

```
interface I1 {
        int a=100;
        void m1();
}

interface A1 implements I1 {
        public void m2();
}
```

It will throw compilation error.. Why?

Quiz (Contd.).

Will the following code compile successfully ?

```
interface I1 {
        int a=100;
        void m1();
}

interface A1 extends I1 {
        public void m2();
}


class Aimp implements I1 {
        public void m1() {
        System.out.println("In m1 method");
        }
}
```

This code will compile successfully..!

Abstract Classes v/s Interfaces

Abstract Classes	Interfaces
Abstract classes can have non-final non-static variables.	Variables declared within an interface are always static and final.
Abstract Classes can have abstract methods as well as normal methods.	Interfaces can have only method declarations(abstract methods). You cannot define a normal method.
Abstract class is extended by another class using "extends" keyword.	An interface is "implemented" by a java class using "implements" keyword .

Contd..

Abstract Classes v/s Interfaces (Contd.).

Abstract Classes	Interfaces
An abstract class can extend another class and it can implement one or more interfaces.	An interface can extend one or more interfaces but cannot extend a class. It cannot implement an interface.
An abstract class can have constructors defined within it.	You cannot define constructors within an interface.
An abstract class cannot be instantiated using "new" Keyword	An interface cannot be instantiated.

Summary

- Introduction to interfaces
- Creating interfaces
- Implementing interfaces
- Difference between interfaces and abstract classes

Thank You