Topics

- Interfaces in Java

Interfaces in Java

- Java does not support Multiple-Inheritance directly. Multiple inheritance can be achieved in java by the use of Interfaces.
- We need interfaces when we want functionality to be included but does not want to impose implementation.
- Implementation issue is left to the individual classes implementing the interfaces.
- Interfaces can have only abstract methods and final fields.
- Every method in an interface is by default public abstract
- Every variable in an interface is by default public final
- You can declare a variable to be of type interface. But you can not create an object belonging to type interface.
- Interface variable can point to objects of any class implementing the interface.
- Another way of implementing Run Time Polymorphism.

Class vs Interfaces (Similarities)

- is compiled into byte code file
- can be either public, protected, private or package accessibility
- can not be public unless defined in the file having same name as interface name
- serve as a type for declaring variables and parameters

Class vs Interfaces (Differences)

- Declares only Method Headers and public constants
- Has no constructors [So, an object never belongs to an interface].
- Can be implemented by a class. A class can implement multiple interfaces.
- Can not extend a class.
- Can extend several other interfaces.

Interface Syntax : General Form

- Syntax :

```
<scope> interface <interface-name> extends [ <interface1> ,… ,<interface-N>]
{
        [public][final]      <type>  variable-name-1  = value;
                                        .
                                        .
        [public][final]      <type>  variable-name-N = value;

        [public][abstract] <return type> method-name-1(<parameter lis>);
                                        .
                                        .
        [public][abstract] <return type> method-name-N(<parameter lis>);
}
```

Interface Example 1


```
public interface  A
{
    double PI = 3.14156;
    void show();
    void display();
} // End of Interface  A
class   X    implements    A
{
    public void show() { }
    public void display() { }
}// End of class X
```

Name of source file must be A.java

public final PI = 3.1456;

public abstract void show();

public abstract void display();

Implemented Methods of Interfaces should have public scope

Interface Example 2

```
public interface  A
{
    double PI = 3.14156;
    void show();
    void display();
} // End of Interface        A
abstract class    X    implements    A
{
    public void    show() { }
}
class    Y    extends    X
{
    public void    display() { }
}// End of class X
```

A class should either fully implement an interface or it should be declared as abstract

Interface Example 3


```
interface  A
{
    void show-1();
    void display-1();
} // End of Interface A
interface  B
{
    void show-2();
    void display-2();
} // End of Interface B
interface  C extends A, B
{
    void show-3();
    void display-3();
} // End of Interface C
```

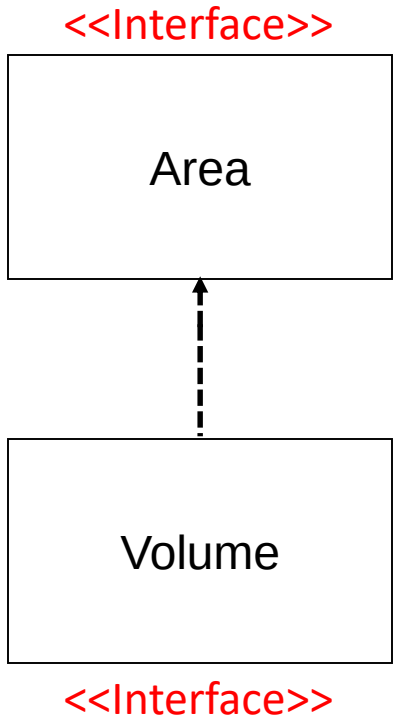
An interface can extend multiple interfaces

```
class     X          implements          C
{
          void show-1()       { }
          void display-1()    { }
          void show-2()       { }
          void display-2()    { }
          void show-3()       { }
          void display-3()    { }
} // End of class X
```

If a class implements a sub-interface then it also implements its super interfaces

Interface Example 4

```
interface Area
{
        double PI = 3.1456;
        double area();
        double perimeter();
} // End of Interface Area
interface Volume extends Area
{
        double volume();
} // End of Interface Volume
```



Interface Example 4 ….

```
class Circle implements Area
{
        private double radius;
        Circle(double radius)
        {
                this.radius = radius;
        }
        double getRadius() { return radius;}
        public double area()
        {
                return PI * radius * radius;
        }
        public double perimeter()
        {
                return 2 * PI * radius;
        }
}// End of class Circle
```

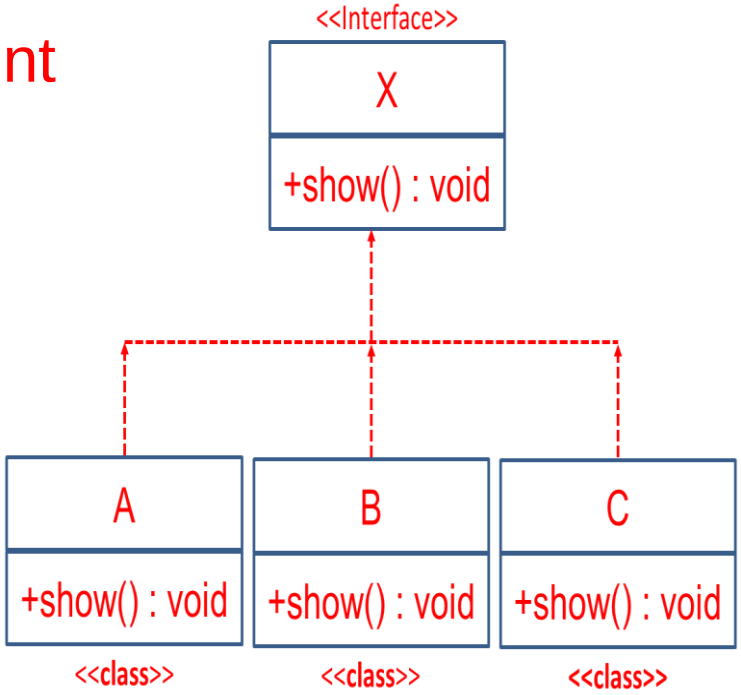
Interface Example 4 ….

```
class BOX implements Volume
{
        private double length;
        private double width;
        private double height;
        BOX(double l, double b, double h)
        {
                length = l;
                width = b;
                height = h;
        }
        double getLength() { return length ;}
        double getWidth() { return width ;}
        double getHeight() { return height ;}
        public double area()
        {
        return 2 * (length * width + width * height + height * length);
        } // End of Method
        public double volume()
        {
                return length * width * height ;
        } // End of Method
        public double perimeter()
        {
                double p = length+width+height;
                return  4 * p;
        }// End of Method
        } // End of class BOX
```

Runtime Polymorphism Through Interfaces

- Suppose ‘X’ is an interface and three concrete classes namely ‘A’, ‘B’, and ‘C’ implements ‘X’ interface

Any Interface Type Variable Can Point to Any Instance of a Class That Implements the Interface




```
X  x1 = new A();
x1.show();
```

Invokes show() of class A

```
x1 = new B();
x1.show();
```

Invokes show() of class B

Thank You