Interface

Dr. Reema Ajmera

Content

1. Interface
2. How interface are different from abstract
3. Multiple Inheritance
4. Interface creation & implementation
5. Do's and Don'ts
6. Access modifiers in interface
7. Interface vs Abstract class
8. Multiple Inheritance vs. Interface

Interface

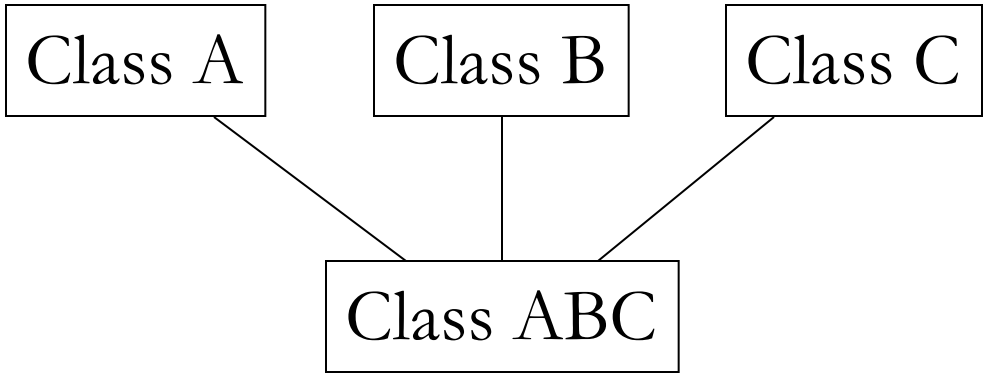
- An interface is a named collection of method definitions and constants ONLY.
- An interface defines a protocol of behavior that can be implemented by any class anywhere in the class hierarchy.
- An interface defines a set of methods but does not implement them.
- A class that implements the interface agrees to implement all the methods defined in the interface, thereby agreeing to certain behaviors

Interface and Abstract Classes

- An interface cannot implement any methods, whereas an abstract class can.

- A class can implement many interfaces but can have only one super class.
- An interface is not part of the class hierarchy. Unrelated classes can implement the same interface.

What is Multiple Inheritance ??



Class ABC inherits all variables and methods from Class A, Class B, and Class C.

Java does NOT support multiple inheritances.

However, we can use **interface** to implement the functionality of multiple inheritance.

Interface Body

- The interface body contains method declarations for **ALL** the methods included in the interface.
- A method declaration within an interface is followed by a semicolon (;) because an interface does not provide implementations for the methods declared within it.
- All methods declared in an interface are implicitly public and abstract.
- All variables declared in an interface are public, static and final.

Java's interface Concept

```
public interface Shape {
        double PI = 3.14; // static and final => upper case
        void draw(); // automatic public
        public void resize(); // automatic public
}
public class Rectangle implements Shape
 {
        public void draw() {
                System.out.println ("Rectangle");
        }
        public void resize() {
                /* do stuff */
        }
}
```

```
public class Square extends Rectangle
{
        public void draw() {
                System.out.println ("Square");
        }
        public void resize()
        { /* do stuff */ }
}
```

Do's and Don'ts

```
interface InterfaceName {
   // "constant" declarations
   // method declarations
}
// inheritance between interfaces interface
  InterfaceName extends InterfaceName
   { ...     }

// not possible
 interface InterfaceName extends
                    interfaceName1, InterfaceName2 { ...
   }
```

```
// not possible
interface InterfaceName extends ClassName
 { ... }

// implements instead of extends
class ClassName implements InterfaceName { ... }

// combine inheritance and interface
   //implementation
class ClassName extends SuperClass implements
   InterfaceName { ... }

// multiple inheritance like again
class ClassName extends SuperClass implements
   InterfaceName1, InterfaceName2 { ... }
```

Access modifiers

- An interface can be public or “friendly” (the default).
- All methods in an interface are default abstract and public.
- Static, final, private, and protected cannot be used with functions.
- All variables (“constants”) are public ,static and final by default .
- Private, protected cannot be used with variables

Interface vs. Abstract

- Methods can be declared
- No method bodies
- “Constants” can be declared
- Has no constructors
- Multiple inheritance possible
- Has no top interface
- Multiple “parent” interfaces

- Methods can be declared
- Method bodies can be defined
- All types of variables can be declared
- Can have constructors Multiple inheritance not possible
- Always inherits from Object
- Only one “parent” class

Multiple Inheritance vs. Interface

Multiple Inheritance

- Declaration and definition is inherited
- Little coding to implement subclass.
- Hard conflict can exist. Very hard to understand (C++ close to Flexible)

Interface

- Only declaration is inherited.
- Must coding to implement an interface.
- No hard conflicts
- Fairly easy to understand
- Very flexible
- Interface totally separated from implementation.

Thank You