INTERFACE IN JAVA

-Compiled by Nikahat Mulla

Agenda

- Single vs Multiple Inheritance
- Java Interface
- Interface Syntax
- Implementing interfaces
- Stack Interface Example

Single vs. Multiple Inheritance

- Some object-oriented languages allow *multiple inheritance,* which allows a class to be derived from two or more classes, inheriting the members of all parents
- The price: collisions, such as the same variable name, same method name in two parents, have to be resolved
- Java decision: *single inheritance*, meaning that a derived class can have only one parent class

Java Interface

- A Java *interface* is a collection of constants and abstract methods
  - abstract method: a method header without a method body; we declare an abstract method using the modifier `abstract`
  - since all methods in an interface are abstract, the `abstract` modifier is usually left off
- Methods in an interface have public visibility by default

Interface: Syntax

interface is a reserved word

```
public interface Doable
{
   public static final String NAME="XYZ";

   public void doThis();
   public int doThat();
   public void doThis2 (float value, char ch);
   public boolean doTheOther (int num);
}
```

A semicolon immediately follows each method header

No method in an interface has a definition (body)

Interfaces (more)

- ✓ An interface is treated like a special class in Java. Each interface is compiled into a separate bytecode file, just like a regular class
- ✓ Can not create instance from an interface using the new operator
- ✓ Can be used as a data type for a reference variable
- ✓ Relationship between the class and the interface is known as *interface inheritance*
- ✓ A constant defined in an interface can be accessed using the syntax InterfaceName. CONSTANT_NAME
- ✓ All data fields are *public final static* and all methods are *public abstract* in an interface, Java allows these modifiers to be omitted.

```java
public interface T {
  public static final int K = 1;

  public abstract void p();
}
```

Equivalent

```java
public interface T {
  int K = 1;

  void p();
}
```

Implementing an Interface

- A class formally implements an interface by
  - stating so in the class header in the `implements` clause
  - a class can implement multiple interfaces: the interfaces are listed in the implements clause, separated by commas
- If a class asserts that it implements an interface, it must define all methods in the interface or the compiler will produce errors

Implementing Interfaces

```
public class Something implements Doable
{
    public void doThis ()
    {
        // whatever
    }

    public void doThat ()
    {
        // whatever
    }

    // etc.
}
```

implements is a reserved word

Each method listed in Doable is given a definition

```
public class ManyThings implements Doable, AnotherDoable
```

Interfaces: An Example

A data structure: Stack for implementing a stack of numbers

Operations on the stack:

push(), pop(),peek()

Problem Statement:

Create the Stack interface which lists the methods push(), pop(), peek()

Write a class called Integer Stack which creates a stack of numbers by implementing the Stack interface.

Interfaces: Examples from Java Standard Class Library

- The Java Standard Class library defines many interfaces:
  - the `Iterator` interface contains methods that allow the user to move through a collection of objects easily
    - `hasNext(), next(), remove()`
  - the `Comparable` interface contains an abstract method called `compareTo`, which is used to compare two objects

```
if(obj1.compareTo(obj2)< 0)
  System.out.println("obj1 is less than obj2");
```

Polymorphism via Interfaces

- Define a polymorphism reference through interface
  - declare a reference variable of an interface type

    `Doable obj;`
  - the `obj` reference can be used to point to any object of any class that implements the `Doable` interface
  - the version of `doThis` depends on the type of object that `obj` is referring to:

    `obj.doThis();`

Example: Polymorphism via Interface

- Consider a SmartPhone class which extends the Phone class and implements MusicPlayer and Camera interfaces

Refer InterfaceExample.java

More Examples

```
Speaker guest;

guest = new Philosopher();
guest.speak();

guest = Dog();
guest.speak();
```

```
Speaker special;
special = new Philosopher();

special.pontificate(); // compiler error
```

```
Speaker special;
special = new Philosopher();

((Philosopher)special).pontificate();
```

```
public interface Speaker
{
   public void speak();
}

class Philosopher extends Human
   implements Speaker
{
   //
   public void speak()
   {…}
   public void pontificate()
   {…}
}

class Dog extends Animal
   implements Speaker
{
   //
   public void speak()
   {
   …
   }
}
```

Extending Interfaces

- ✓ An interface can inherit other interfaces using the extends keyword

```
public interface NewInterface extends Interface1, ..., InterfaceN {
// constants and abstract methods
}
```

- ✓ A class implementing NewInterface must implement the abstract methods defined in NewInterface, Interface1, …. and InterfaceN

- ✓ An interface can extend other interfaces but not classes

- ✓ A class can extend its superclass and implement multiple interfaces.

Interface Summary

- In Java, an *interface* is a reference type, similar to a class
- Can contain constants & method signatures
- There are no method bodies
- Cannot be instantiated — they can only be *implemented* by classes or *extended* by other interfaces
- Variables & methods declared in interfaces are public by default
- An interface can extend another interface
- A class can implement multiple interfaces

Interface Summary (Contd…)

Why Interface?

- To reveal an object's programming interface (functionality of the object) without revealing its implementation
  - ✓ This is the concept of encapsulation
  - ✓ The implementation can change without affecting the caller of the interface
  - ✓ The caller does not need the implementation at the compile time
  - ✓ It needs only the interface at the compile time
  - ✓ During runtime, actual object instance is associated with the interface type

Interface Summary (Contd…)

Why Interface?

- To have unrelated classes implement similar methods (behaviors)
- Example:
  - ✓ Class Line and class MyInteger
  - ✓ They are not related through inheritance
  - ✓ You want both to implement comparison methods
    - checkIsGreater(Object x, Object y)
    - checkIsLess(Object x, Object y)
    - checkIsEqual(Object x, Object y)
  - ✓ Define Comparison interface which has the three abstract methods above

Interface Summary (Contd…)

Why Interface?

- To model multiple inheritance
  - ✓ A class can implement multiple interfaces while it can extend only one class

Interface Summary (Contd…)

- Example of an interface declaration:

```
public interface Interface1
{
    public void set(int i);
    public int get();
}
```

```
Interface1 i1 = new Interface1();
```

Interfaces and Abstract classes can not be instantiated

Thank You