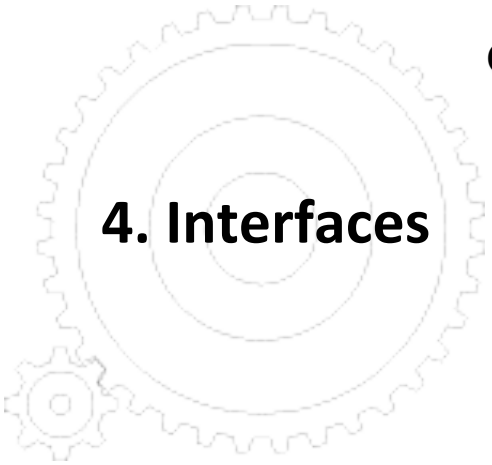
G51OOP



4. Interfaces

Object Oriented Programming

Colin Higgins

4a. Introduction to Java interfaces

- Programming via contracting is good practice
  - Helps in understanding
  - Enforces standards
  - Makes design, debugging, testing etc easier
- Contracting via the concept of responsibilities
- Interfaces takes the idea of responsibilities one step further
  - Enforced by the compiler
- Allows the public methods that define what a class is and does
  - To be easily, clearly visible
  - To be enforced
  - To be used in the design possibly before implementation

Concept

- An `interface` is a way to describe what classes should do, without specifying how they should do it. It's not a class but a set of requirements for classes that want to conform to the interface

```
public interface Comparable {
       int compareTo(Object otherObject);
}
```

this requires that any class implementing the `Comparable` interface contains a `compareTo` method, and this method must take an `Object` parameter and return an integer

Interface declarations

- The declaration consists of a keyword `interface`, its name, and the members
- Similar to classes, interfaces can have three types of members
  - constants (fields)
  - methods
  - (nested classes and interfaces)

Interface member – constants

- An interface can define named constants, which are `public`, `static` and `final` (these modifiers are omitted by convention) automatically. Interfaces never contain instant fields.
- All the named constants MUST be initialized

```
interface Verbose {
      int SILENT = 0;
      int TERSE = 1;
      int NORMAL = 2;
      int VERBOSE = 3;

      void setVerbosity (int level);
      int getVerbosity();
}
```

Interface member – methods

- They are implicitly `abstract` (omitted by convention). So every method declaration consists of the method header and a semicolon.
- They are implicitly `public` (omitted by convention). No other types of access modifiers are allowed.
- They can't be `final`, nor `static`

- Two steps to make a class implement an interface
  1. declare that the class intends to implement the given interface by using the `implements` keyword

```
class Employee implements Comparable { . . . }
```

  2. supply definitions for all methods in the interface

```
public int compareTo(Object otherObject) {
    Employee other = (Employee) otherObject;
    if (salary < other.salary) {
            return -1;
    }
    if (salary > other.salary) {
            return 1;
    }
    return 0;
}
```

- A single class can implement multiple interfaces. Just separate the interface names by comma

```
class Employee implements Comparable, Cloneable{
  . . .}
```

Instantiation properties of interfaces

- Interfaces are not classes. You can never use the `new` operator to instantiate an interface.

```
public interface Comparable {
        . . .
}

Comparable myComparible = new Comparable( );
```

- You can still declare interface variables

```
Comparable myComparible ;
```

  but they must refer to an object of a class that implements the interface

```
class Employee implements Comparable {
        . . .
}
myComparible  = new Employee( );
```

Money Example

```
interface Money {
   void negate(); // change sign of value (+ve <-> -ve)
   int pounds(); // return number of pounds
   int pence(); // return number of pence
   void set(int pounds, int pence); // set to new value
   void increase(Money m); // increase current value by m
   void decrease(Money m); // decrease current value by m
   boolean equals(Money m);     // equal to?
   boolean less(Money m);       // less than?
   boolean greater(Money m); // greater than?
   boolean less_equals(Money m); // less than or equal to?
   boolean greater_equals(Money m); // greater than or equal to?
   boolean is_zero(); // has zero value?
   boolean is_positive(); // is the value +ve or –ve
}
```

Implementing a Money class

```
public class Sterling implements Money {
   private int pnce;
   public Money(int pounds, int pence) {
        …
   }

   void negate() {
        …
   }

   int pounds() {
        …
   }

   //etc
}
```

Using Money…

```
public static void main(String [] argv) {
   Money myMoney = new Sterling(3, 40);
   Money yourMoney = new Sterling(2, 10);

   myMoney.increase(yourMoney);

   //etc
}
```

Summary of Interfaces…

- An interface defines a protocol of communication between two objects.
- An interface declaration contains signatures, but no implementations, for a set of methods, and might also contain constant definitions.
- A class that implements an interface must implement all the methods declared in the interface.
- An interface name can be used anywhere a type can be used.