Μεθοδολογία Προγραμματισμού

Abstract Κλάσεις και Interfaces

Νικόλαος Πεταλίδης

Τμήμα Μηχανικών Πληροφορικής και Επικοινωνιών
Διεθνές Πανεπιστήμιο της Ελλάδος

Εισαγωγή
Εαρινό Εξάμηνο

Abstract κλάσεις

- Πολλές φορές είναι χρήσιμο να έχουμε κλάσεις με μερική μόνο υλοποίηση
- Οι κλάσεις αυτές ορίζουν συνήθως κάποιες μεθόδους με υλοποίηση και κάποιες άλλες μεθόδους στις οποίες οι υλοποίηση *πρέπει* να δοθεί από άλλες (υπο)κλάσεις
- Τέτοιες κλάσεις λέγονται `abstract`

Παράδειγμα

```
abstract public class Shape
{
   abstract public void Draw ( );
 }
public class Circle extends Shape
{
   public void Draw ( ) {
   ...
   }
 }
```

Ιδιαιτερότητες των abstract κλάσεων

- Δε μπορούν να αρχικοποιηθούν
- Αν υπάρχει έστω και μία abstract μέθοδος, η κλάση χαρακτηρίζεται abstract
- Αν μια κλάση κληρονομεί από μια abstract κλάση είτε πρέπει να υλοποιήσει τις abstract μεθόδους είτε πρέπει να οριστεί και αυτή ως abstract

Interfaces

- Προσφέρουν ένα τρόπο να περιγράφουν τι πρέπει να κάνει μια κλάση, χωρίς όμως να ορίζουν το πώς
- Δεν είναι κλάσεις, αλλά ένα σύνολο από απαιτήσεις που πρέπει να ικανοποιούν οι κλάσεις

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

Δήλωση interface

- Αποτελείται από τη λέξη `interface`, το όνομα και τα μέλη
- Παρόμοια με τις κλάσεις μπορούν να έχουν
  - Σταθερές (constant fields)
  - Δηλώσεις μεθόδων
  - default μεθόδους (Java 8)
  - static μεθόδους (Java 8)
  - εσωτερικές κλάσεις και άλλα interfaces

Σταθερές

- Ενα interface μπορεί να έχει `public, static` και `final` πεδία (συνήθως παραλείπονται)
- Ενα interface δε μπορεί να έχει άλλα πεδία.
- Τα πεδία πρέπει να είναι σε κεφαλαία

```
interface Verbose {
   int SILENT = 0;
   int TERSE = 1;
   void setVerbosity(int level);
   int getVerbosity();
}
```

Δηλώσεις μεθόδων

- Είναι εξ' ορισμού `abstract` και `public`
- Κάθε δήλωση μεθόδου αποτελείται από το όνομα της μεθόδου, τις παραμέτρους και τον επιστρεφόμενο τύπο
- Οι μέθοδοι δε μπορεί να είναι `final`

default μέθοδοι

- Ορίζονται με τη λέξη κλειδί `default`
- Αν μια κλάση που υλοποιεί ένα `interface` δεν παρέχει υλοποίηση για μια `default` μέθοδο, τότε κληρονομεί την `default` υλοποίηση

```
interface Verbose {
    default void doSomething() {
        // ... do something
    }
}
```

- Δοκιμάστε να δείτε τι θα γίνει αν ορίσετε μια κλάση που υλοποιεί δύο interfaces που έχουν την ίδια default μέθοδο.

static μέθοδοι

- Ορίζονται με τη λέξη κλειδί `static`
- Μια μέθοδος που ορίζετα `static`ι κληρονομείται από όλες τις υλοποιήσεις και δε μπορεί να αλλάξει

```
interface Verbose {
   static void doSomething() {
      // ...do something
   }
   // ...
   Verbose.doSomething();
}
```

Interface modifiers

- Ένα interface μπορεί να είναι `public` ή `package` (το default)
- Όλα τα interfaces είναι abstract

Υλοποίηση ενός interface

- Μια κλάση μπορεί να *υλοποιεί* ένα ή περισσότερα interfaces
- Αυτό σημαίνει ότι πρέπει να παρέχει υλοποιήσεις για όλες τις μεθόδους που ορίζει το interface
- Μια κλάση δηλώνει ότι υλοποιεί ένα interface χρησιμοποιώντας τη λέξη κλειδί `implements`

Παράδειγμα

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

«Δημιουργία» interfaces

- Τα intefaces *δεν είναι κλάσεις*
- Δε μπορείτε να κάνετε αυτό: Comparable n = **new** Comparable();
- Μπορείτε όμως να κάνετε αυτό: Comparable n = **new** Employee();

Επέκταση intefrace

- Τα interfaces υποστηρίζουν την πολλαπλή κληρονομικότητα
- Ένα interface μπορεί να επεκτείνει πάνω από ένα interface

```
public interface SerializableRunnable
    extends java.io.Serializable, Runnable {
  . . .
}
```

Λεπτομέρειες

- Οταν ένα interface επεκτείνει δύο interfaces που έχουν τα ίδια πεδία:

```
interface A {
   int val = 1;
}
interface B {
   int val = 2;
}
interface C extends A, B {
   System.out.println"(A.val = "+ A.val);
   System.out.println"(B.val = "+ B.val);
}
```

Λεπτομέρειες (συνχ)

- Όταν ένα interface επεκτείνει ένα interface και έχει το ίδιο πεδίο

```
interface X {
  int val = 1;
}
interface Y extends X {
  int val = 2;
  int sum = val + X.val;
}
```

Λεπτομέρειες (συνχ)

- Όταν μια κλάση υλοποιεί ένα interface που έχει επεκτείνει ένα άλλο

```
class Z implements Y {
...
 System.out.println"(Z.val":+val);     //Z.val = 2
}
 //outside the class
 System.out.println"(Z.val":+Z.val); //Z.val = 2
 Z v = new Z( );
^^|^^| System.out.print( "v.val = " + v.val ^^|
^^|^^|^^|^^| "    +, ((Y)v).val = " + ((Y)v).val^^|
//2
^^|^^|^^|^^| "    +, ((X)v).val = " + ((X)v).val );
```

Marker interfaces

- Ένα marker (tagging) interface δεν έχει ούτε μεθόδους ούτε σταθερές.
- Ο μόνος του σκοπός είναι να επιτρέψει τη χρήση του τελεστή `instance of` κατά την εκτέλεση του προγράμματος/
- Το πιο γνωστό παράδειγμα ενός τέτοιου interface είναι το `Cloneable`
- *Αποφύγετε* τη χρήση marker interfaces

Διαφορά Interface και Abstract κλάσεων

- Μια κλάση μπορεί να κληρονομήσει μόνο μία abstract αλλά μπορεί να υλοποιήσει πολλά interfaces
- Mia abstract κλάση μπορεί να έχει μερική υλοποίηση
- Τα interfaces έχουν μόνο public σταθερές and public μεθόδους

Απαριθμήσεις (Enumerations)

- Ορισμένες φορές μια μεταβλητή μπορεί να πάρει μόνο ορισμένες τιμές.
- Για παράδειγμα η μεταβλητή `dayOfWeek` μπορεί να πάρει μόνο τις τιμές: `Δευτέρα, Τρίτη, Τετάρτη κτλ`
- Μια τέτοια μεταβλητή δεν ανήκει σε κανέναν από τους συνηθισμένους τύπους: `int, boolean κτλ`

Enums

- Στη Java είναι δυνατόν να ορίσεις τέτοιους τύπους ως `enum`
- Ένας τύπος enum είναι ένας νέος τύπος στον οποίο ορίζουμε εμείς ποιές είναι οι πιθανές τιμές του

```
enum Day {
  MONDAY, TUESDAY, WEDNESDAY,
  THURSDAY, FRIDAY,
  SATURDAY, SUNDAY
  }
```

Enum και Class

- Ένα enum είναι ουσιαστικά ένας νέος τύπος κλάσης
- Μπορούν να οριστούν ως εσωτερικές ή εξωτερικές κλάσεις
- Μπορούν να οριστούν ως `public`, `static` ή final

Περισσότερα Enums

- Ένα enum υλοποιεί το interface `Comparable` και μπορεί να ταξινομηθεί
- Ένα enum επαναορίζει τη μέθοδο `toString()`
- Ένα enum προσφέρει τη μέθοδο `valueOf()`

```
Day day = Day.MONDAY;
System.out.println(day); // prints MONDAY
day = Day.valueOf("SUNDAY");
System.out.println(day); // prints SUNDAY
```

Πλεονεκτήματα των enums

- Προσφέρουν προστασία από την κακή χρήση τύπων (δε μπορείς να πάρεις άλλες τιμές από αυτές που έχεις ορίσει)
- Μπορούν να μπουν σε collections
- Μπορούν να έχουν πεδία και μεθόδους

Παράδειγμα enum

```
public enum Coin {
  PENNY(1), NICKEL(5),
  DIME(10), QUARTER(25);
    private final int value;
    Coin(int value) {
      this.value = value;
    }
}
```