ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΠΛΗΡΟΦΟΡΙΚΗ ΙΙ

Ενότητα 8: Πακέτα (Packages)

Μιχάλης Δρακόπουλος

Σχολή Θετικών επιστημών

Τμήμα Μαθηματικών

ΠΛΗΡΟΦΟΡΙΚΗ ΙΙ (Java)
Ενότητα 8

Πακέτα (Packages)

Τα πακέτα αποτελούν τρόπο ομαδοποίησης κλάσεων συναφούς λειτουργικότητας.

Ένα πακέτο λειτουργεί σαν βιβλιοθήκη κλάσεων που μπορούν να χρησιμοποιηθούν από ένα πρόγραμμα χωρίς να βρίσκονται στον υποκατάλογό του.

Κάθε πακέτο έχει ένα όνομα και όλες του οι κλάσεις βρίσκονται στον ίδιο υποκατάλογο, που έχει το ίδιο όνομα με το πακέτο.

Οι κλάσεις ενός πακέτου είναι κανονικές κλάσεις, άρα ορίζονται:

```
public class <ClassName>
```

αλλά **πριν** τον ορισμό τους έχουν την ακόλουθη δήλωση:

```
package <packageName>;
```

π.χ.
Έστω ότι είμαστε μέσα στον υποκατάλογο με όνομα pack1, ο οποίος έχει μέσα δύο κλάσεις, τα αρχεία ClassOne.java και ClassTwo.java:

Αρχείο ClassOne.java:

```
package pack1;

public class ClassOne
{
   ...
   ...
}
```

Αρχείο ClassTwo.java:

```
package pack1;

public class ClassTwo
{
   ...
   ...
}
```

Επίσης, έστω ότι στον υποκατάλογο pack1 υπάρχει νέος υποκατάλογος με το όνομα smallPack ο οποίος περιέχει τις ακόλουθες κλάσεις:

Αρχείο Class1.java:

```
package pack1.smallPack;

public class Class1
{
  ...
  ...
}
```

Αρχείο Class2.java:

```
package pack1.smallPack;

public class Class2
{
  ...
  ...
}
```

Για να χρησιμοποιήσουμε στο πρόγραμμά μας τις κλάσεις του πακέτου pack1 χρησιμοποιούμε την εντολή **`import`**:

Αρχείο MyClass1.java:

```
import pack1.*;

public class MyClass1
{
  ...
  ...
}
```

ενώ ειδικά για τις κλάσεις του πακέτου smallPack:

Αρχείο MyClass2.java:

```
import pack1.smallPack.*;

public class MyClass2
{
  ...
  ...
}
```

Άρα, με τη δήλωση: `import <packageName>.*;` μπορούμε να χρησιμοποιήσουμε όλες τις κλάσεις ενός πακέτου σε κάποια κλάση του προγράμματός μας. Είναι κλάσεις που βρίσκονται εκτός του υποκαταλόγου του προγράμματός μας.

- Το `.*` δηλώνει όλες τις κλάσεις ενός πακέτου.
- Μπορούμε να κάνουμε import σε μια κλάση μόνο συγκεκριμένες κλάσεις ενός πακέτου, ενώ μπορούμε φυσικά να κάνουμε import κλάσεις από πολλά πακέτα. Π.χ., αν θέλαμε στην

κλάση μας MyClass3 να συμπεριλάβουμε όλο το pack1 και την Class2 του pack1.smallPack:

```
import pack1.*;
import pack1.smallPack.Class2;

public class MyClass3
{
  ...
}
```

Στο παράδειγμα αυτό των pack1 και smallPack πακέτων, το smallPack, το οποίο βρίσκεται σε έναν υποκατάλογο του pack1, ονομάζεται υποπακέτο του pack1. Η δομή αυτή υπάρχει για να μπορούν οι κλάσεις κάθε πακέτου να χωρίζονται σε υπο-ομάδες συναφούς λειτουργικότητας, έτσι ώστε κάποιος να μπορεί εύκολα να συμπεριλάβει στο πρόγραμμά του μια υπο-ομάδα κλάσεων με τη δήλωση:

```
import <πακέτο>.<υποπακέτο>.*;
```

χωρίς να χρειάζεται να κάνει import μία-μία τις κλάσεις αυτές ή να αναγκάζεται να κάνει import όλο το πακέτο, το οποίο μπορεί να περιέχει πολλές κλάσεις που δεν θα χρειαστεί.

Περί ορατότητας μεταβλητών και μεθόδων:

Ορατότητα:	`public`	`private`	χωρίς δήλωση ορατότητας
Από την ίδια την κλάση	ναι	ναι	ναι
Από άλλη κλάση στο ίδιο πακέτο	ναι	όχι	ναι
Από άλλη κλάση έξω από το πακέτο (η οποία κάνει import το πακέτο)	ναι	όχι	**όχι**

δηλαδή,

- οι `public` μεταβλητές (ή μέθοδοι) είναι ορατές από παντού
- οι `private` είναι ορατές μόνο μέσα στην κλάση όπου ορίζονται
- οι "χωρίς δήλωση" είναι ορατές από τις κλάσεις του πακέτου μόνο, δηλαδή αποτελούν κάτι ενδιάμεσο, μεταξύ `public` και `private`.

Άρα: - για κλάσεις του ίδιου πακέτου: «χωρίς δήλωση» ≡ `public`
- για κλάσεις εκτός πακέτου: «χωρίς δήλωση» ≡ `private`

(Υπάρχει και η ορατότητα `protected` στην οποία θα αναφερθούμε στην ενότητα της κληρονομικότητας)

Παράδειγμα:

Έχουμε ένα πακέτο (pack1) με τις ακόλουθες δύο κλάσεις:

```
package pack1;

public class ClassA
{
  private void method1()
   { ... }

  public void method2()
   { ... }

  void method3()
   { ... }

}
```

```
package pack1;

public class ClassB
{

   ClassA obj = new ClassA();

   public void method4()
   {
    obj.method1(); // ΛΑΘΟΣ!
    obj.method2();
    obj.method3();
   }

}
```

Στο πρόγραμμά μας κάνουμε χρήση του πακέτου `pack1`:

```
import pack1.*;

public class MyClass
{

  ClassA objA = new ClassA();

  public void myMethod()
   {
    objA.method1(); // ΛΑΘΟΣ!
    objA.method2();
    objA.method3(); // ΛΑΘΟΣ!!
    objA.method4(); // ΛΑΘΟΣ!! (η method4() ανήκει στην ClassB)
   }
}
```

Κλάσεις με το ίδιο όνομα σε διαφορετικά πακέτα

Αν υπάρχει κλάση με το ίδιο όνομα σε δύο διαφορετικά πακέτα, τότε αναφορές στο όνομα της κλάσης αυτής θα πρέπει να περιέχουν και το όνομα του πακέτου. Π.χ., αν σε συνέχεια του προηγούμενο παραδείγματος με το pack1, θέλουμε να χρησιμοποιήσουμε και το pack2, το οποίο περιέχει και αυτό μια κλάση με το όνομα π.χ. ClassB, τότε:

```
import pack1.*;
import pack2.*;
public class MyNewClass
{
      pack1.ClassB obj1 = new pack1.ClassB();
      pack2.ClassB obj2 = new pack2.ClassB();
      .
```

```
      .
      .
}
```

Στην περίπτωση αυτή, χρησιμοποιούμε το `<πακέτο>.<Κλάση>` για να αναφερθούμε στη συγκεκριμένη κάθε φορά ClassB επειδή υπάρχουν δύο τέτοιες κλάσεις. Επιπλέον, επειδή ακριβώς χρησιμοποιούμε το `<πακέτο>.<Κλάση>`, δεν είναι απαραίτητο να κάνουμε `import` τα πακέτα pack1 και pack2.

Δηλαδή, στις κλάσεις που χρησιμοποιούμε την κλάση `Scanner` για είσοδο από το πληκτρολόγιο και κάνουμε `import` το πακέτο `java.util`:

```
import java.util.*;
public class InputExample
{
   Scanner input = new Scanner(System.in);
   .
   .
}
```

θα μπορούσαμε ισοδύναμα να γράφαμε:

```
public class InputExample
{
   java.util.Scanner input = new java.util.Scanner(System.in);
   .
   .
}
```

→ Σε μια κλάση μπορεί να έχουμε και δήλωση πακέτου και `import` κάποιου πακέτου. Π.χ.

```
package pack3.test;   // η κλάση ανήκει στο πακέτο pack3.test
import javax.swing.*; // η κλάση χρησιμοποιεί το πακέτο javax.swing

public class AskForData
{
  ...
}
```

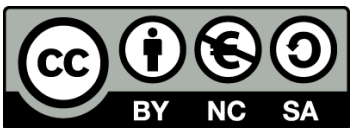
Η δήλωση πακέτου (`package <packageName>;`) είναι υποχρεωτικά η πρώτη εντολή της κλάσης.

Σημειώματα

Σημείωμα Αναφοράς

Μιχάλης Δρακόπουλος. «Πληροφορική ΙΙ. Ενότητα 8: Πακέτα (Packages)». Έκδοση: 1.0. Αθήνα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: http://opencourses.uoa.gr/courses/MATH106/.

Σημείωμα Αδειοδότησης



[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Διατήρηση Σημειωμάτων

- Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.