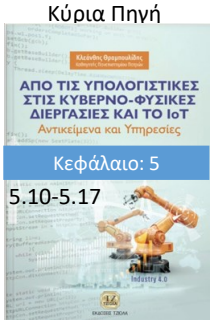
Αντικειμενοστρεφής Προγραμματισμός (Object-Oriented Programming)

(CEID_NNY106)

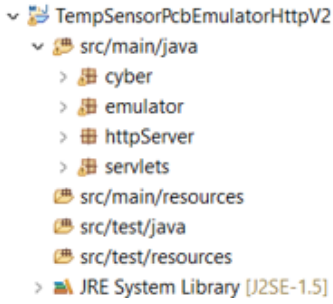
Java as an OOP Language

Miscellaneous

Κύρια Πηγή



Kleanthis Thramboulidis
Prof. of Software and System Engineering
University of Patras
https://sites.google.com/site/thramboulidiskleanthis/

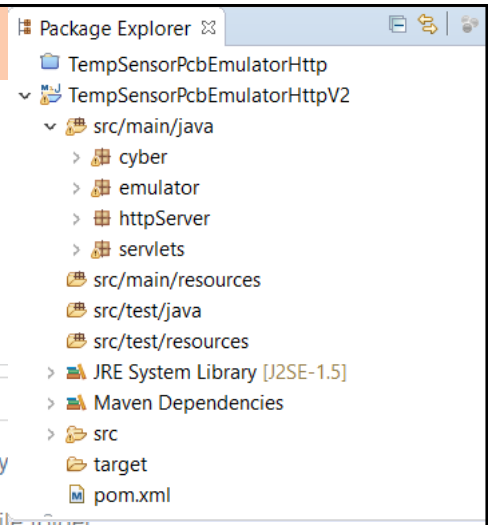


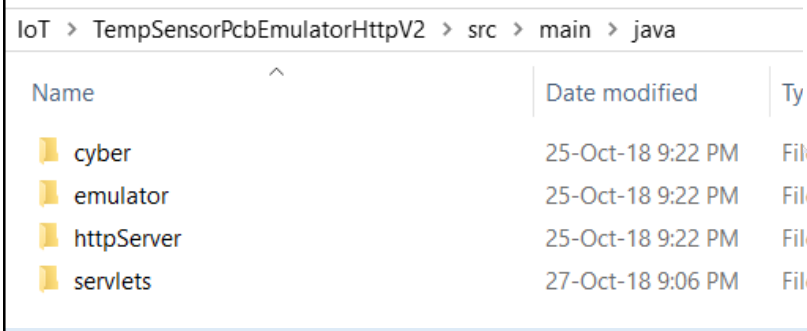
Οργάνωση Διάλεξης

- **Πακέτα (Packages)**
- Προσδιοριστές ορατότητας (visibility modifiers)
- Απόδοση αρχικών τιμών (field initialization)
- Autoboxing
- Απαριθμητικός τύπος (enumeration type)
- Κατηγορίες κλάσεων
  - Nested, Local, Anonymous Classes
- Functional Interfaces, Lambda Expressions
- Module

Packages

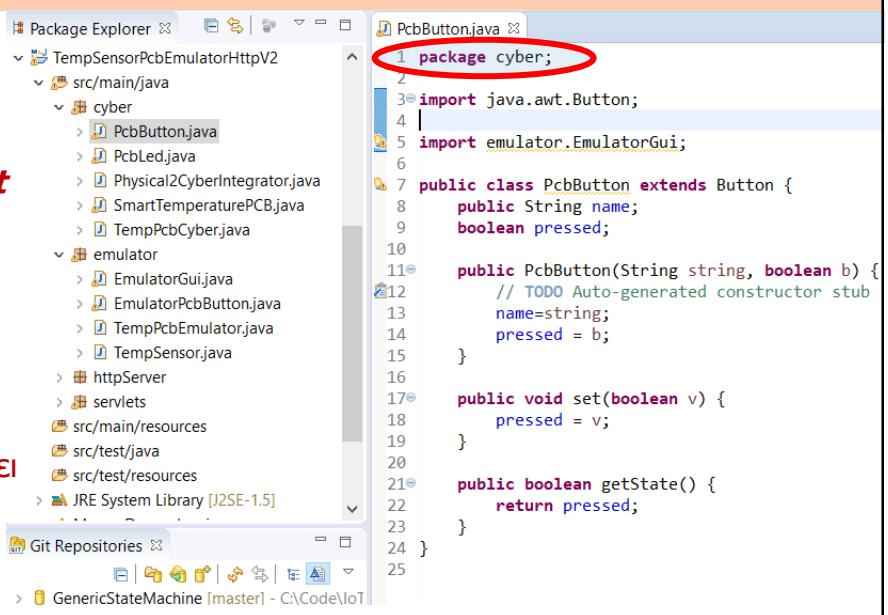
- Είναι ένας μηχανισμός που επιτρέπει την ομαδοποίηση σχετιζόμενων κλάσεων και interfaces σε επώνυμες ομάδες.
- Βοηθούν στην οργάνωση και δόμηση των κλάσεων και των συσχετίσεων τους.
- Are used
  - to bundle classes and interfaces into packages, and
  - to **arrange your file system** so that the compiler can find your source files.





Package

- Εάν δεν χρησιμοποιήσεις πρόταση package statement, αυτομάτα οι κλάσεις πάνε στο***default package*** (**unnamed package**).
- Το πακέτο default είναι μόνο για μικρές εφαρμογές και για την εκπαιδευτική διαδικασία.
- Οι τύποι αναφοράς πρέπει να ομαδοποιούνται σε επώνυμα πακέτα.

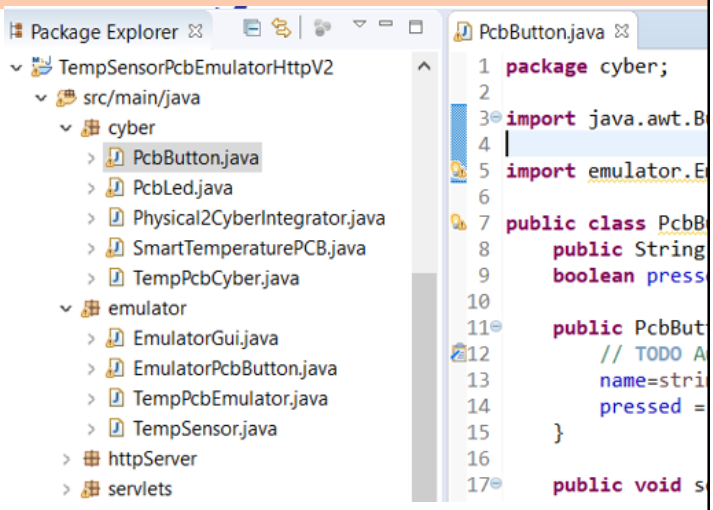


Δημιουργία πακέτου

- Επιλέγουμε το όνομα του πακέτου και βάζουμε την πρόταση package με το όνομα του πακέτου ως πρώτη γραμμή σε κάθε αρχείο που περιέχει τύπους που αναθέτουμε στο πακέτο. π.χ.

  package operators;

- Μόνο μια πρόταση package σε κάθε αρχείο πηγαίου κώδικα.
- Αυτή ισχύει για όλους τους τύπους του πακέτου.



Απόδοση Ονόματος σε πακέτο

- Ο μεταγλωττιστής επιτρέπει σε δύο κλάσεις να έχουν το ίδιο όνομα με την προϋπόθεση να βρίσκονται σε διαφορετικά πακέτα.
- Το πλήρες όνομα (**fully qualified name**) μιας κλάσης περιλαμβάνει και το όνομα του πακέτου
  - java.util.Stack
  - graphics.Rectangle (Rectangle class in the graphics package)
  - java.awt.Rectangle (Rectangle class in the java.awt package)

Legalizing Package Names

Domain Name	Package Name Prefix
hyphenated-name.example.org	org.example.hyphenated_name
example.int	int_.example
123name.example.com	com.example._123name

- Χρησιμοποιούμε μικρούς χαρακτήρες για να διακρίνεται από ονόματα τύπων.
- Συνηθίζεται να χρησιμοποιείται το domain name ανάστροφα ως πρώτο συνθετικό του ονόματος πακέτου. Παράδειγμα

  gr.upatras.mypackage

  προσδιορίζει το πακέτο mypackage που δημιουργεί ένας προγραμματιστής στην εταιρεία με domain name upatras.gr
- Για την αποφυγή συγκρούσεων ονομάτων ορίζονται συμβάσεις εντός της επιχείρησης όπως π.χ. συμπερίληψη ονόματος παραρτήματος, τομέα, προτζεκτ.
- Παράδειγμα

  gr.upatras.ceid.sseg.cpus.gcas

Οργάνωση Διάλεξης

- Πακέτα (Packages)
- **Προσδιοριστές ορατότητας (visibility modifiers)**
- Απόδοση αρχικών τιμών (field initialization)
- Autoboxing
- Απαριθμητικός τύπος (enumeration type)
- Κατηγορίες κλάσεων
  - Nested, Local, Anonymous Classes
- Functional Interfaces, Lambda Expressions
- Module

Προσδιοριστές ορατότητας

- Προσδιορίζουν την ορατότητα κλάσεων και των συνθετικών τους (data members and methods) που αφορά τη δυνατότητα χρήσης τους από άλλα αντικείμενα της εφαρμογής κατά τον χρόνο εκτέλεσης.
- Η καθολική ορατότητα παρότι διευκολύνει την συνεργασία των αντικειμένων δημιουργεί πολλά προβλήματα, οπότε την αποφεύγουμε.
- Αν δεν ορισθεί σαφώς η ορατότητα τότε ισχύει το **default access level**.

to gain control **explicitly control access** by specifying access levels

Προσδιοριστές ορατότητας

- **Κλάσης**
  - public : η κλάση έχει καθολική ορατότητα

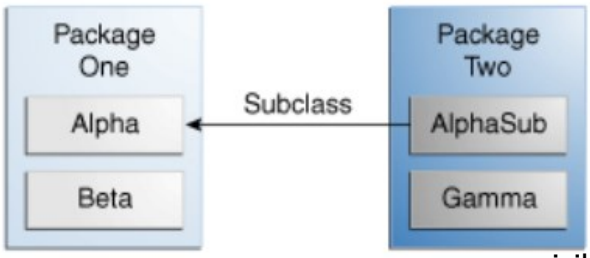
    public class Operand {
  - *package-private* (μη χρήση προσδιοριστή) : η κλάση είναι ορατή μόνο μέσα στο πακέτο της.

    class Operand {
- **Συνθετικών** (μεταβλητών και μεθόδων)
  - public, private, protected, or *package-private* (no explicit modifier)

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Προσδιοριστές ορατότητας - Παράδειγμα

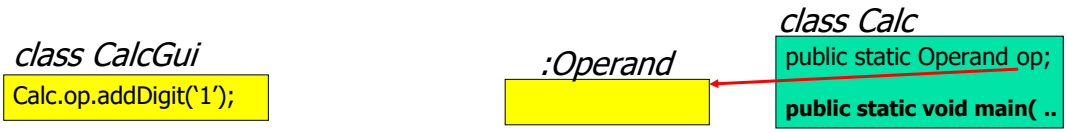


visibility of the members of the Alpha class

Modifier	Alpha	Beta	Alphasub	Gamma
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Επιλογή επιπέδου ορατότητας

- Χρησιμοποιούμε το πιο περιοριστικό επίπεδο ορατότητας που ικανοποιεί τις ανάγκες της εφαρμογής. **Ξεκινάμε από private** εκτός και αν υπάρχει λόγος να μην χρησιμοποιηθεί το επίπεδο αυτό.
- **Αποφεύγουμε** την καθολική ορατότητα (public) για μεταβλητές (information hiding). Την χρησιμοποιούμε μόνο σε σταθερές.
  - Το public χρησιμοποιείταιι σε πολλά παραδείγματα (για διάφορους λόγους) αλλά συνίσταται η αποφυγή σε επαγγελματικό κώδικα.
  - Το public πεδία βάζουν **ισχυρές εξαρτήσεις** με την συγκεκριμένη υλοποίηση και περιορίζουν την ευελιξία (flexibility) σε αλλαγές στον κώδικα.



Οργάνωση Διάλεξης

- Πακέτα (Packages)
- Προσδιοριστές ορατότητας (visibility modifiers)
- **Απόδοση αρχικών τιμών (field initialization)**
- Autoboxing
- Απαριθμητικός τύπος (enumeration type)
- Κατηγορίες κλάσεων
  - Nested, Local, Anonymous Classes
- Functional Interfaces, Lambda Expressions
- Module

Απόδοση αρχικών τιμών

- **Απλή όταν**
  - Όταν έχουμε την τιμή στο edit time, και
  - η απόδοση της μπορεί να γίνει σε μία γραμμή κώδικα

```
public class Test {
    public int x = 1;
    public static int max = 20;
```

```
public class Window {
    private int width;
    private int height;
    private String title;

    private ArrayList<Shape> shapes = new ArrayList<>();
```

- Η απλότητα αυτή βάζει **περιορισμούς**
  - Όπως για παράδειγμα όταν η απόδοση βασίζεται σε κάποια λογική π.χ. διαχείριση λαθών ή επανάληψη (for loop).
- Μεταβλητές στιγμιοτύπου συνήθως αρχικοποιούνται στον δημιουργό.
- Μεταβλητές κλάσης μπορούν να αρχικοποιηθούν σε ***static initialization blocks***.

Απόδοση αρχικών τιμών σε μεταβλητές Στιγμιοτύπου

- Συνήθως γίνεται στον Δημιουργό.
- Εναλλακτικά με
  - initializer blocks και
  - Final μεθόδους
- Initializer Block (IB)

```
{
// initialization code
}
```

```
class Whatever {
    private varType myVar = initInstanceVar();
    protected final varType initInstanceVar( ) {
        // initialization code goes here
    }
}
```

  - Ο μεταγλωττιστής αντιγράφει τα initializer blocks σε κάθε Δημιουργό.
  - Με βάση αυτό τα IB μπορούν να χρησιμοποιηθούν για την αποφυγή επανάληψης του ίδιου κώδικα σε πολλούς δημιουργούς.
- Μέθοδος *final* δεν μπορεί να γίνει overridde σε υποκλάση.

Απόδοση αρχικών τιμών σε μεταβλητές Κλάσης

```
public class Test {
        private static int[ ] sqr;
        private static final int N = 100;
        static {
                sqr = new int[N];
                for (int i = 0; i < N; i++)          sqr[i] = i * i;
        }
        public static void main(String args[]) {
                System.out.println("17 squared is " + sqr[17]);
        }
}
```

Private static method
Advantage: can be reused later if you need to reinitialize the class variable

- Μια κλάση μπορεί να έχει πολλά static initialization blocks
  - Μπορεί να εμφανίζονται οπουδήποτε μέσα στον κώδικα
  - Το περιβάλλονν υλοποίησης διασφαλίζει ότι τα static initialization blocks καλούνται με την σειρά με την οποία εμφανίζονται στον πηγαίο κώδικα.
- you can use static initialization blocks to do one-time per-class initialization
  - Static fields are initialized when the class is loaded.
- Alternative?

Οργάνωση Διάλεξης

- Πακέτα (Packages)
- Προσδιοριστές ορατότητας (visibility modifiers)
- Απόδοση αρχικών τιμών (field initialization)
- **Autoboxing**
- Απαριθμητικός τύπος (enumeration type)
- Κατηγορίες κλάσεων
  - Nested, Local, Anonymous Classes
- Functional Interfaces, Lambda Expressions
- Module

Autoboxing

- Οι wrapper κλάσεις όπως οι Integer, Double, Boolean επιτρέπουν στον προγραμματιστή να μετατρέπει τιμές πρωτογενών τύπων σε αντικείμενα (στιγμιότυπα των αντίστοιχων wrapper κλάσεων).

```
7           Double d1 = new Double(10.2);
8           int num = 12;
9           Integer i1 = new Integer(num);
```

- **Autoboxing** είναι η **αυτόματη μετατροπή** μιας τιμής πρωτογενούς τύπου π.χ. int, double, σε στιγμιότυπο της αντίστοιχης wrapper κλάσης, π.χ. Integer, Double, την οποία εκτελεί ο μεταγλωττιστής.

```
11          Double d1 = 10.2;
12          int num = 12;
13          Integer i1 = num;
```


Autoboxing - Unboxing

- Υποστηρίζεται από την Java 1.5 και μετά.
- Δίνει ευελιξία στον προγραμματιστή όσον αφορά την χρήση πρωτογενών τύπων και των αντίστοιχων τύπων αναφοράς.
- Για παράδειγμα επιτρέπει την εύκολη εισαγωγή τιμών πρωτογενών τύπων σε συλλογές

```
17          List<Integer> li = new ArrayList<>();
18          for (int i = 1; i < 50; i += 2)
19              li.add(i);
```

- Χωρίς το autoboxing ο προγραμματιστής θα έπρεπε να κάνει αυτή την μετατροπή μόνος του

```
20              li.add(Integer.valueOf(i));
```

Unboxing

- **Unboxing** είναι η **αυτόματη μετατροπή ενός αντικειμένου** (στιγμιότυπου wrapper κλάσης, όπως π.χ. Integer, Double, σε τιμή του αντίστοιχου πρωτογενούς τύπου δηλαδή int, double, την οποία εκτελεί ο μεταγλωττιστής.
- είναι η ακριβώς αντίθετη μετατροπή που κάνει το autoboxing

```
System.out.println("Unboxing");
var d = Double.parseDouble("10.2");
double d1 = d;
var i1 = Integer.parseInt("10");
int num = i1;
System.out.println("d1: "+d1+" i1: " + i1);
```

```
29
30          List<Integer> li = new ArrayList<>();
31          for (int i = 1; i < 10;i++)
32              li.add(i);
33          int sum=0;
34          for (Integer i: li)
35              sum+=i;
            System.out.println("sum is  "+sum);
```