Διάλεξη 6: Αφαιρετικότητα, Βιβλιοθήκες

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

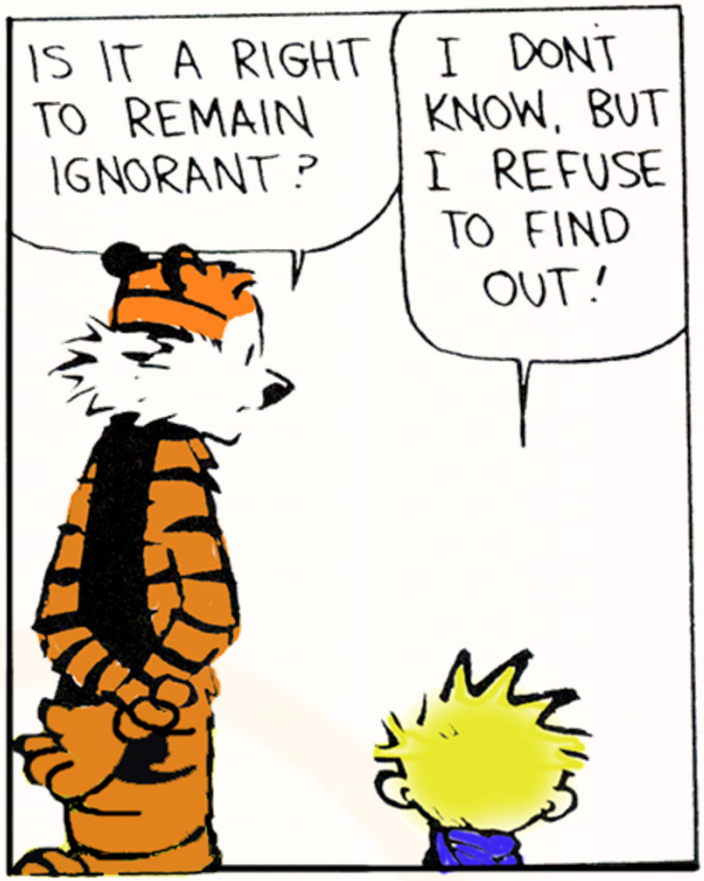
- Αφαιρετικότητα
- Βιβλιοθήκες (packages)

Διδάσκων: Παναγιώτης Ανδρέου

Αφαιρετικότητα (abstraction)

Η αφαιρετικότητα είναι επιλεκτική άγνοια!

- Επιλογή του τι είναι σημαντικό και τι όχι
- Έμφαση και Εξάρτηση στα σημαντικά
- Αγνόηση και Απεξάρτηση από τα ΜΗ σημαντικά
- Χρήση της ενθυλάκωσης (encapsulation) για να επιτύχουμε την αφαιρετικότητα



Edsger Dijkstra: "The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise."

Αφαιρετικότητα (abstraction) (συν.)

Αφαιρετικότητα (Abstraction)

- Κάθε αντικείμενο έχει συγκεκριμένο ρόλο και ευθύνες.
- Παράδειγμα: Ο Γιάννης και ο Άθως είναι μόνο ανθοπώληδες, ο Ανδρέας είναι μόνο αποστολέας.
- Αυτό φυσικά εναπόκειται στις καλές σχεδιαστικές ικανότητες του προγραμματιστή

Το παράδειγμα της αποστολής λουλουδιών

- Ο **Χρίστος** θέλει να στείλει λουλούδια στον **Βασίλη**, ο οποίος βρίσκεται σε άλλη πόλη.
- Ο **Χρίστος** δεν μπορεί να παραδώσει τα λουλούδια ο ίδιος, έτσι χρησιμοποιεί τις υπηρεσίες κάποιου ανθοπώλη (**Άθως**).
- Ο **Χρίστος** λέει στον **Άθω** τη διεύθυνση του Βασίλη, το ποσό που μπορεί να διαθέσει και το είδος των λουλουδιών.
- Ο **Άθως** με τη σειρά του επικοινωνεί με άλλο ανθοπώλη (**Γιάννης**) στην πόλη του **Βασίλη**, του μεταφέρει τις πληροφορίες. Ο **Γιάννης** ετοιμάζει την παραγγελία και ενημερώνει τον αποστολέα (**Ανδρέας**).
- Ο **Ανδρέας** παραδίδει τα λουλούδια.


```
public class Anthopolis{
   private createBouquet(int
         noOfFlowers, Color c){
      …
   }
…
public class Apostoleas{
   private sendBouquet(Flowers
f, Address a){
      …
   }
…
```

Παράδειγμα Αφαιρετικότητας 1

- **Κάθε ζώο** βρίσκεται σε μία τοποθεσία (location)
- **Κάθε ζώο** έχει κάποια ενεργειακά αποθέματα
- **Κάθε ζώο** είναι πεινασμένο ή όχι
- **Κάθε ζώο** μπορεί να φάει κάποιο φαγητό
- **Κάθε ζώο** μπορεί να κινηθεί σε κάποια άλλη τοποθεσία

- **Κάθε ζώο** ανήκει στην κατηγορία των ζωντανών πλασμάτων
  → **Κληρονομικότητα (διαφορετικά επίπεδα αφαιρετικότητας)**

```
public class Animal {
  private Location loc;
  private double energyReserves;

  public boolean isHungry() {
    return energyReserves < 2.5;
  }

  public void eat(Food f) {
    // Consume food
    energyReserves +=
                  f.getCalories();
  }

  public void moveTo(Location l) {
    // Move to new location
    loc = l;
  }
}
```

```
public class Animal extends Living
```

Παράδειγμα Αφαιρετικότητας 2

Κινητά τηλέφωνα

Παρατηρήσεις

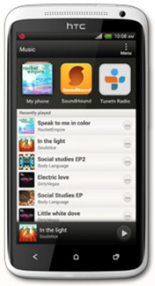
- Πολλά HTC τηλέφωνα
- 1 iPhone
- Μοντέλα για Ψυχαγωγία
- Μοντέλα Δουλειά
- Μοντέλα για Κοινωνικότητα
- Touch ή με keypad
- Διαφορετικά χρώματα, βάρος…
- … και πολλά άλλα

- **Όλα είναι κινητά τηλέφωνα**


HTC CHA CHA


HTC Desire C


HTC One S


HTC One X


HTC Desire C


HTC Incredible S


HTC Desire X


iPhone


HTC Sensation XL


HTC Sensation XE

Παράδειγμα Αφαιρετικότητας 2 (συν.)

Όλα είναι κινητά τηλέφωνα

Παράδειγμα Αφαιρετικότητας 2 (συν.)

Χρειαζόμαστε όμως καλύτερη κατηγοριοποίηση…

Ας τα κατηγοριοποιήσουμε ανάλογα με τον κατασκευαστή τους

Παράδειγμα Αφαιρετικότητας 2 (συν.)

Θα μπορούσαμε όμως να τα κατηγοριοποιήσουμε και σύμφωνα με τη χρήση τους π.χ., εργασία, ψυχαγωγία, κοινωνικότητα.

Παράδειγμα Αφαιρετικότητας 2 (συν.)

Θα μπορούσαμε όμως να τα κατηγοριοποιήσουμε και σύμφωνα με το αν είναι touch ή type.

Βιβλιοθήκες/Πακέτα (packages)

- (Συνήθως) Ομαδοποιούμε τις κλάσεις σύμφωνα με τα αντικείμενα που αναπαριστούν και τις λειτουργίες τους.
- Ένα “πακέτο” (βιβλιοθήκη) JAVA απαρτίζεται από μια ομάδα κλάσεων, οι οποίες ανήκουν στον ίδιο χώρο ονομάτων (namespace).
- Για τη δημιουργία μιας βιβλιοθήκης Java, πρέπει να χρησιμοποιήσουμε την λέξη-κλειδί **package** μαζί με το όνομα της βιβλιοθήκης, στην αρχή κάθε αρχείου, το οποίο θέλουμε να εντάξουμε στην βιβλιοθήκη.
- Έτσι, δηλώνοντας : **package mypackage**
  F στην αρχή κάποιων αρχείων Java, ορίζουμε ότι τα αρχεία αυτά ανήκουν στην ίδια βιβλιοθήκη, η οποία έχει το όνομα mypackage.
- Η δημιουργία μιας βιβλιοθήκης Java («πακέτου») δεν προϋποθέτει ούτε συνεπάγεται την τοποθέτηση αρχείων-κλάσεων σε κάποιο ενιαίο αρχείο.

Βιβλιοθήκες/Πακέτα (packages) (συν.)

Παράδειγμα: package test.animals;

```
package test.animals;

public class Tiger {

  public String name;

  public Tiger(){ }

  public Tiger(String name){
    this.name = name;
  }

  public String getName(){
    return name;
  }

}
```

```
package test.animals;

public class Lion {

  public String name;

  public Lion(){ }

  public Lion(String name){
    this.name = name;
  }

  public String getName(){
    return name;
  }

  public String isKing(){
    return "I am the king";  }
}
```

Βιβλιοθήκες/Πακέτα (packages) (συν.)

- Η Java μας δίνει τη δυνατότητα να επαναχρησιμοποιήσουμε τις κλάσεις μιας βιβλιοθήκης JAVA, με χρήση της εντολής **import** στον κώδικα των αρχείων στα οποία θέλουμε να κάνουμε χρήση των κλάσεων της βιβλιοθήκης.
- **Τρόπος 1 (χρήση import):**

```
import test.animals.Lion;
…
Lion l = new Lion();
```

- **Τρόπος 2 (χωρίς τη χρήση import):**

```
test.animals.Lion l = new
test.animals.Lion();
```

```
package test.animals;

public class Lion {

  public String name;

  public Lion(){ }

  public Lion(String name){
    this.name = name;
  }

  public String getName(){
    return name;
  }

  public String isKing(){
    return "I am the king";  }
}
```

Βιβλιοθήκες/Πακέτα (packages) (συν.)

- Παράδειγμα:

```
import test.animals.Lion;
import test.animals.Tiger;
// OR import test.animals.*;

public class TestAnimals{

  public static void main(String args[]){
    Lion l1 = new Lion("lion");
    // OR test.animals.Lion l1 =
                    new test.animals.Lion("lion");

    Tiger t1 = new Tiger("tiger");
    // OR test.animals.Tiger t1 =
                    new test.animals.Tiger(“tiger");
  }
}
```

Βιβλιοθήκες/Πακέτα (packages) (συν.)

- Ο λόγος της εισαγωγής βιβλιοθηκών στην JAVA, μέσα στον πηγαίο κώδικα, οφείλεται στην ανάγκη διαχείρισης της ονοματολογίας (name spaces) για την αποφυγή συγκρούσεων (name clashes).
- Π.χ. τι θα συμβεί αν ορίσουμε κάποια άλλη κλάση **Lion** και η ίδια κλάση βρίσκεται ήδη στην μηχανή μας ή φορτώνεται αυτόματα από το δίκτυο καθώς τρέχει μία εφαρμογή μας;
- Για την πρόληψη των προβλημάτων αυτών η JAVA μας δίνει (μέσω των βιβλιοθηκών) πλήρη έλεγχο στην ονοματολογία και τη δυνατότητα να δημιουργούμε απολύτως μοναδικά ονόματα, ασχέτως των περιορισμών του Διαδικτύου.

Οργάνωση Πακέτων

- Τα αρχεία των κλάσεων (class files) που έχουν δηλωθεί ότι ανήκουν στην ίδια βιβλιοθήκη, πρέπει να τοποθετούνται σε κοινό υποκατάλογο (sub-directory), με όνομα το οποίο συνάγεται από το όνομα της βιβλιοθήκης.
- Παράδειγμα: package test.animals →<root>/test/animals
- Η διαρρύθμιση αυτή μας επιτρέπει:
  - Να επιλύσουμε το πρόβλημα της ονοματολογίας, χρησιμοποιώντας την ιεραρχική δομή των καταλόγων τού συστήματος αρχείων για να δώσουμε μοναδικά ονόματα σε βιβλιοθήκες - το όνομα της βιβλιοθήκης αντιστοιχεί στην θέση της στο σύστημα αρχείων.
  - Να εντοπίζουμε με ευκολία τις κλάσεις μιας βιβλιοθήκης.

Εντοπισμός Βιβλιοθηκών

Ο διερμηνέας εντοπίζει τις κλάσεις μιας βιβλιοθήκης ως εξής:

- Πρώτα εντοπίζει και διαβάζει την μεταβλητή περιβάλλοντος **CLASSPATH**, η οποία περιέχει μια σειρά από καταλόγους της μηχανής, από τους οποίους μπορεί να ξεκινήσει η αναζήτηση των κλάσεων.
- Στη συνέχεια «επιλύει» το όνομα μιας βιβλιοθήκης η οποία έχει εισαχθεί στον κώδικα, σε όνομα καταλόγου του συστήματος αρχείων. Π.χ.: το **package foo.bar.baz** επιλύεται σε **foo/bar/baz**
- Το όνομα του καταλόγου αυτού επικολλάται στα περιεχόμενα του CLASSPATH, δημιουργώντας έτσι την ακριβή διεύθυνση στο σύστημα αρχείων απ' όπου μπορούν να αναζητηθούν οι κλάσεις της βιβλιοθήκης.
- Σημειώστε ότι στο CLASSPATH πρέπει να έχετε τοποθετήσει και τον κατάλογο «.».

Ονοματολογία στην Java

- Ας υποθέσουμε ότι θέλετε να δημιουργήσετε και να τοποθετήσετε κάποιες κλάσεις στο Διαδίκτυο, οπότε θέλετε να τους δώσετε και μοναδική ονομασία.
- Τότε, μπορείτε να ονομάσετε τις κλάσεις με βάση το δικό σας Πεδίο Διαδικτύου (internet domain), το οποίο θεωρείται μοναδικό. Π .χ.:
  package mypackage.util;
  package cy.ac.ucy.cs.epl233.util;
- Αν υποθέσουμε ότι στο CLASSPATH υπάρχει μόνο ο τρέχων κατάλογος (“.”) και αναζητούμε τη βιβλιοθήκη **cy.ac.ucy.cs.epl233.util**, ο διερμηνέας θα αναζητήσει το αρχείο : **./cy/ac/ucy/cs/epl233/util** σε σύστημα UNIX, και **.\cy\ac\ucy\cs\epl233\util** σε σύστημα DOS-Windows.

Μετάφραση και Βιβλιοθήκες

- Στην περίπτωση δημιουργίας ενός αντικειμένου με βάση κάποια εισαγόμενη κλάση, ο μεταφραστής θα αναζητήσει την κλάση σε αρχείο με το ίδιο όνομα, χρησιμοποιώντας το όνομα της κλάσης. Αν βρεθεί το αρχείο, η κλάση θα φορτωθεί από τον μεταφραστή.
- Στην περίπτωση όμως όπου υπάρχει αντίστοιχο αρχείο **.java** νεώτερης «έκδοσης», το αρχείο αυτό θα μεταφραστεί αυτόματα και θα φορτωθεί η καινούρια έκδοση τού **.class** αρχείου .

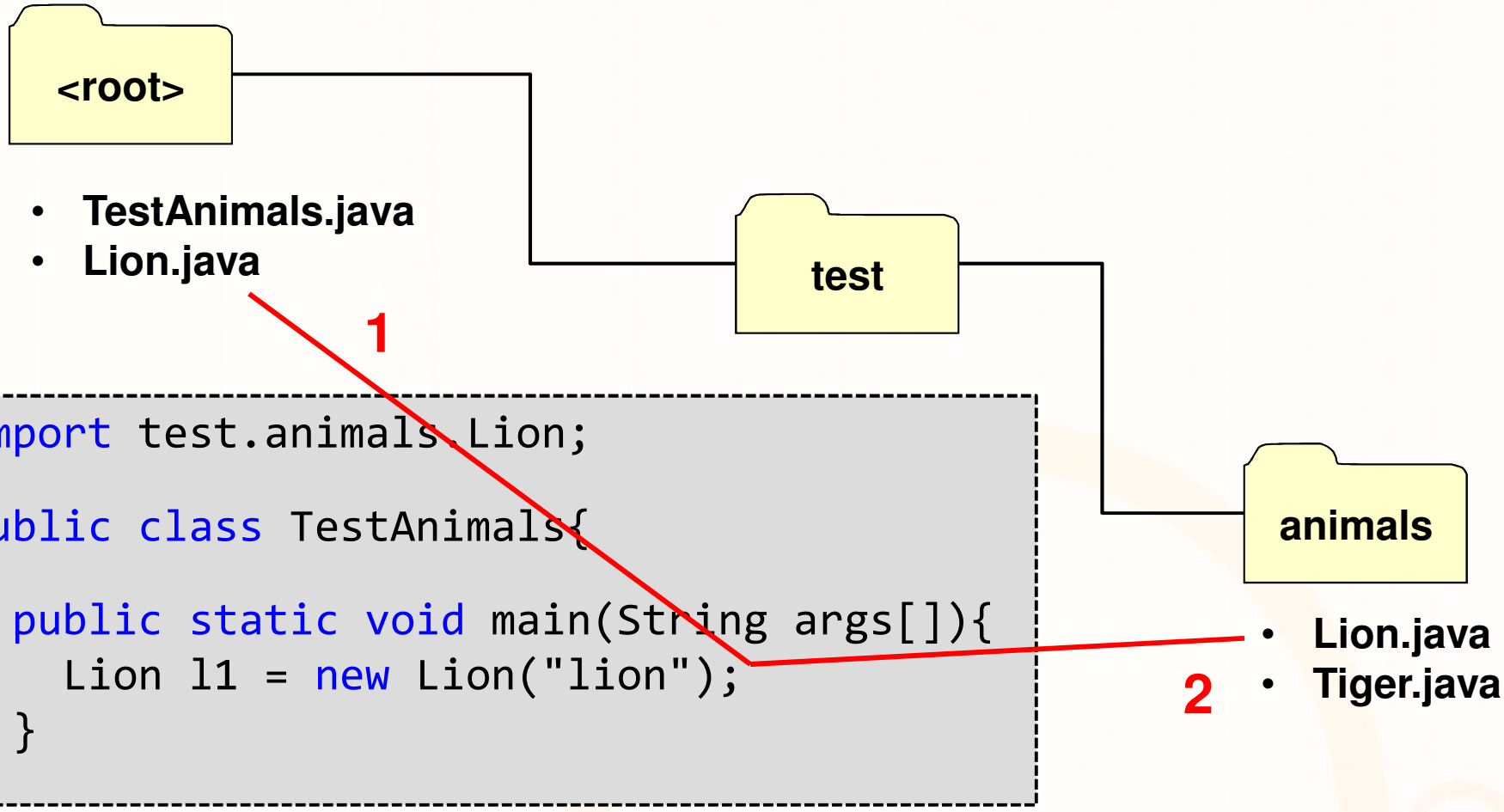
Σύγκρουση Ονομάτων

- Ας υποθέσουμε ότι σε κάποιο πρόγραμμα Java εισάγονται δύο βιβλιοθήκες, οι οποίες περιλαμβάνουν κλάσεις με τα ίδια ονόματα. Π.χ.:
  import mypackage.util.*;
  import java.util.*;
  όπου και οι δύο βιβλιοθήκες ορίζουν την κλάση Vector.
- Δεν υπάρχει πρόβλημα στο βαθμό που δεν χρησιμοποιούμε την Vector. Διαφορετικά ο μεταφραστής βγάζει σχετικό μήνυμα λάθους, διότι δεν έχει τρόπο να γνωρίζει για ποιά κλάση πρόκειται.
- Αν ωστόσο η κλάση Vector ορίζεται μέσα στο αρχείο μας ή στον κατάλογο όπου αποθηκεύεται το αρχείο μας, τότε ο μεταφραστής υποθέτει ότι οι δύο κλάσεις βρίσκονται στην **default** βιβλιοθήκη και άρα δεν υπάρχει πρόβλημα σύγκρουσης (θα χρησιμοποιηθεί η τοπική Vector).

Παράδειγμα: Σύγκρουση Ονομάτων

Παράδειγμα: name clash

- **Ερώτηση:** Ποια κλάση Lion θα χρησιμοποιηθεί;



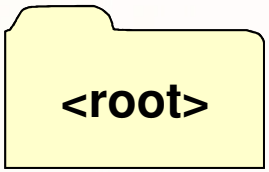
Προκαθορισμένη Βιβλιοθήκη

Αρχεία κλάσεων τα οποία:

- Είναι τοποθετημένα στον ίδιο υποκατάλογο
- Δεν περιέχουν δήλωση package

Θεωρούνται ότι ανήκουν στην ίδια βιβλιοθήκη, το λεγόμενο default package

Παράδειγμα:



- **TestAnimals.java**
- **Lion.java**

Παγίδες με το classpath

- Είναι καλό να αποφεύγουμε να προσθέτουμε πολλούς καταλόγους στο CLASSPATH, ώστε να έχουμε καλύτερο έλεγχο των κλάσεων που χρησιμοποιούμε.
- Η εισαγωγή μιας βιβλιοθήκης σε κάποιο αρχείο Java μέσω της **import**, σηματοδοτεί ότι κάποια κλάση *μπορεί να βρεθεί* μέσα στην βιβλιοθήκη.
- Ωστόσο , ο μεταφραστής ελέγχει ολόκληρο το **CLASSPATH** κι έτσι υπάρχει η πιθανότητα να βρει την αναζητούμενη κλάση κάπου αλλού.
- Αν η «λανθασμένη» κλάση βρεθεί πρώτη, τότε ο μεταφραστής σταματάει την αναζήτηση και χρησιμοποιεί την “λανθασμένη” κλάση.