Βιβλιοθήκες και Προσδιοριστές Πρόσβασης στην JAVA

«Μονάδα Μετάφρασης»

ΕΠΛ233

«Μονάδα Μετάφρασης»

ΕΠΛ233

- Όταν δημιουργείται ένα αρχείο πηγαίου κώδικα στην Java, το αρχείο καλείται **μονάδα μετάφρασης (compilation unit)** και πρέπει να έχει όνομα που να τελειώνει σε «.java».

«Μονάδα Μετάφρασης»

ΕΠΛ233

- Όταν δημιουργείται ένα αρχείο πηγαίου κώδικα στην Java, το αρχείο καλείται **μονάδα μετάφρασης (compilation unit)** και πρέπει να έχει όνομα που να τελειώνει σε «.java».
- Μέσα στη μονάδα μετάφρασης μπορεί να ορίζεται **το πολύ** μια δημόσια κλάση (public class), η οποία **πρέπει** να έχει το ίδιο ακριβώς όνομα με το όνομα του αρχείου (με αποκοπή του επιθέματος .java).

«Μονάδα Μετάφρασης»

ΕΠΛ233

- Όταν δημιουργείται ένα αρχείο πηγαίου κώδικα στην Java, το αρχείο καλείται **μονάδα μετάφρασης (compilation unit)** και πρέπει να έχει όνομα που να τελειώνει σε «.java».
- Μέσα στη μονάδα μετάφρασης μπορεί να ορίζεται **το πολύ** μια δημόσια κλάση (public class), η οποία **πρέπει** να έχει το ίδιο ακριβώς όνομα με το όνομα του αρχείου (με αποκοπή του επιθέματος .java).
- Οι υπόλοιπες κλάσεις που ορίζονται στο αρχείο παραμένουν «κρυμμένες» από τον «έξω κόσμο», αφού δεν είναι *δημόσιες* και ρόλος τους είναι να *υποστηρίξουν* την δημόσια κλάση.

«Μονάδα Μετάφρασης»

ΕΠΛ233

- Όταν δημιουργείται ένα αρχείο πηγαίου κώδικα στην Java, το αρχείο καλείται **μονάδα μετάφρασης (compilation unit)** και πρέπει να έχει όνομα που να τελειώνει σε «.java».
- Μέσα στη μονάδα μετάφρασης μπορεί να ορίζεται **το πολύ** μια δημόσια κλάση (public class), η οποία **πρέπει** να έχει το ίδιο ακριβώς όνομα με το όνομα του αρχείου (με αποκοπή του επιθέματος .java).
- Οι υπόλοιπες κλάσεις που ορίζονται στο αρχείο παραμένουν «κρυμμένες» από τον «έξω κόσμο», αφού δεν είναι *δημόσιες* και ρόλος τους είναι να *υποστηρίξουν* την δημόσια κλάση.
- Μετά την μετάφραση της μονάδας μετάφρασης, ο μεταφραστής επιστρέφει από ένα αρχείο με επίθεμα **.class** για **κάθε κλάση** που ορίζεται μέσα στην μονάδα μετάφρασης.

«Μονάδα Μετάφρασης»

ΕΠΛ233

- Όταν δημιουργείται ένα αρχείο πηγαίου κώδικα στην Java, το αρχείο καλείται **μονάδα μετάφρασης (compilation unit)** και πρέπει να έχει όνομα που να τελειώνει σε «.java».
- Μέσα στη μονάδα μετάφρασης μπορεί να ορίζεται **το πολύ** μια δημόσια κλάση (public class), η οποία **πρέπει** να έχει το ίδιο ακριβώς όνομα με το όνομα του αρχείου (με αποκοπή του επιθέματος .java).
- Οι υπόλοιπες κλάσεις που ορίζονται στο αρχείο παραμένουν «κρυμμένες» από τον «έξω κόσμο», αφού δεν είναι *δημόσιες* και ρόλος τους είναι να *υποστηρίξουν* την δημόσια κλάση.
- Μετά την μετάφραση της μονάδας μετάφρασης, ο μεταφραστής επιστρέφει από ένα αρχείο με επίθεμα **.class** για **κάθε κλάση** που ορίζεται μέσα στην μονάδα μετάφρασης.
- Τα ονόματα αυτών των αρχείων είναι τα ίδια με τα ονόματα των αντιστοίχων κλάσεων.

Δήλωση Βιβλιοθηκών JAVA με την package

ΕΠΛ233

Δήλωση Βιβλιοθηκών JAVA με την package

- Ένα “πακέτο” (βιβλιοθήκη) JAVA απαρτίζεται από μια ομάδα κλάσεων, οι οποίες εντάσσονται στον ίδιο *χώρο ονομάτων* (namespace).

Δήλωση Βιβλιοθηκών JAVA με την package

- Ένα “πακέτο” (βιβλιοθήκη) JAVA απαρτίζεται από μια ομάδα κλάσεων, οι οποίες εντάσσονται 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.

Δήλωση Βιβλιοθηκών JAVA με την package

- Ένα “πακέτο” (βιβλιοθήκη) JAVA απαρτίζεται από μια ομάδα κλάσεων, οι οποίες εντάσσονται 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 :

Δήλωση Βιβλιοθηκών JAVA με την package

- Ένα “πακέτο” (βιβλιοθήκη) JAVA απαρτίζεται από μια ομάδα κλάσεων, οι οποίες εντάσσονται 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

Δήλωση Βιβλιοθηκών JAVA με την package

- Ένα "πακέτο" (βιβλιοθήκη) JAVA απαρτίζεται από μια ομάδα κλάσεων, οι οποίες εντάσσονται 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

  στην αρχή κάποιων αρχείων Java, ορίζουμε ότι τα αρχεία αυτά εντάσσονται στην ίδια βιβλιοθήκη, η οποία έχει το όνομα **mypackage.**

Δήλωση Βιβλιοθηκών JAVA με την package

- Ένα "πακέτο" (βιβλιοθήκη) JAVA απαρτίζεται από μια ομάδα κλάσεων, οι οποίες εντάσσονται 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

  στην αρχή κάποιων αρχείων Java, ορίζουμε ότι τα αρχεία αυτά εντάσσονται στην ίδια βιβλιοθήκη, η οποία έχει το όνομα **mypackage.**

Η δημιουργία μιας βιβλιοθήκης Java («πακέτου») δεν προϋποθέτει ούτε συνεπάγεται την τοποθέτηση αρχείων-κλάσεων μέσα στο ίδιο *αρχείο.*

package: η μονάδα βιβλιοθήκης στην JAVA

package: η μονάδα βιβλιοθήκης στην JAVA

- 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).

package: η μονάδα βιβλιοθήκης στην JAVA

- 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 κλάση **stack** και η ίδια κλάση βρίσκεται ήδη στην μηχανή μας ή φορτώνεται αυτόματα από το δίκτυο καθώς τρέχει μία εφαρμογή μας;

package: η μονάδα βιβλιοθήκης στην JAVA

- 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 κλάση **stack** 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.

Αξιοποίηση βιβλιοθηκών

Αξιοποίηση βιβλιοθηκών

ΕΠΛ233

- 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.

Αξιοποίηση βιβλιοθηκών

ΕΠΛ233

- 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.
- Αν υποθέσουμε ότι στο **mypackage** έχουμε συμπεριλάβει τις δημόσιες κλάσεις : **foo, pub , publ2**, τότε μπορούμε να τις χρησιμοποιήσουμε από κάποιο άλλο πρόγραμμα είτε με:
  - επίκληση του **mypackage.foo...** κοκ
  - χρήση του **import mypackage;** και απευθείας χρήση του foo(), publ () κλπ.

Οργάνωση κλάσεων βιβλιοθήκης JAVA

ΕΠΛ233

Οργάνωση κλάσεων βιβλιοθήκης JAVA

- Τα αρχεία των κλάσεων (class files) που έχουν δηλωθεί ότι ανήκουν στην ίδια βιβλιοθήκη, πρέπει να τοποθετούνται σε κοινό υποκατάλογο (subdirectory), με όνομα το οποίο συνάγεται από το όνομα της βιβλιοθήκης.

Οργάνωση κλάσεων βιβλιοθήκης JAVA

- Τα αρχεία των κλάσεων (class files) που έχουν δηλωθεί ότι ανήκουν στην ίδια βιβλιοθήκη, πρέπει να τοποθετούνται σε κοινό υποκατάλογο (subdirectory), με όνομα το οποίο συνάγεται από το όνομα της βιβλιοθήκης.
- Η διαρύθμιση αυτή μας επιτρέπει:

Οργάνωση κλάσεων βιβλιοθήκης JAVA

- Τα αρχεία των κλάσεων (class files) που έχουν δηλωθεί ότι ανήκουν στην ίδια βιβλιοθήκη, πρέπει να τοποθετούνται σε κοινό υποκατάλογο (subdirectory), με όνομα το οποίο συνάγεται από το όνομα της βιβλιοθήκης.
- Η διαρύθμιση 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.

Οργάνωση κλάσεων βιβλιοθήκης JAVA

- Τα αρχεία των κλάσεων (class files) που έχουν δηλωθεί ότι ανήκουν στην ίδια βιβλιοθήκη, πρέπει να τοποθετούνται σε κοινό υποκατάλογο (subdirectory), με όνομα το οποίο συνάγεται από το όνομα της βιβλιοθήκης.
- Η διαρύθμιση 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.

Παράδειγμα


```
package mypackage;
   public class MyClass {
   public void test() {
                    System.out.println("test pack");
   }
}
```

- Ο κώδικας αυτός πρέπει να σωθεί σε αρχείο **MyClass.java**.
- Αφού μεταφραστεί το αρχείο, το αντίστοιχο αρχείο κλάσης πρέπει να αποθηκευθεί σε κάποιον υποκατάλογο με όνομα **mypackage.**
- Ο μεταφραστής (κι αργότερα ο διερμηνέας) θα ψάξουν μέσα στον κατάλογο **mypackage** για να βρούν την **MyClass.class.**

Παράδειγμα

Κατά συνθήκη, το όνομα μιας βιβλιοθήκης γράφεται με μικρά γράμματα

```
package mypackage;
   public class MyClass {
   public void test() {
                  System.out.println("test pack");
   }
}
```

- Ο κώδικας αυτός πρέπει να σωθεί σε αρχείο **MyClass.java**.
- Αφού μεταφραστεί το αρχείο, το αντίστοιχο αρχείο κλάσης πρέπει να αποθηκευθεί σε κάποιον υποκατάλογο με όνομα **mypackage.**
- Ο μεταφραστής (κι αργότερα ο διερμηνέας) θα ψάξουν μέσα στον κατάλογο **mypackage** για να βρούν την **MyClass.class.**

Παράδειγμα (συνέχεια)

ΕΠΛ233

```
import mypackage.MyClass;
class PackageTestApp {
        public static void main(String args[]) {
                MyClass theClass = new MyClass();
                theClass.test();
        }
}
```

- Τα αρχεία που θα δημιουργηθούν από τον μεταφραστή της Java, στο παράδειγμά μας, πρέπει να τα τοποθετήσουμε ως εξής :

classes
PackageTestApp.java
PackageTestApp.class
mypackage
MyClass.java
MyClass.class

Εντοπισμός κλάσεων βιβλιοθηκών

Εντοπισμός κλάσεων βιβλιοθηκών

- Ο διερμηνέας εντοπίζει τις κλάσεις μιάς βιβλιοθήκης ως εξής:

Εντοπισμός κλάσεων βιβλιοθηκών

- Ο διερμηνέας εντοπίζει τις κλάσεις μιάς 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.

Εντοπισμός κλάσεων βιβλιοθηκών

- Ο διερμηνέας εντοπίζει τις κλάσεις μιάς 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**

Εντοπισμός κλάσεων βιβλιοθηκών

ΕΠΛ233

- Ο διερμηνέας εντοπίζει τις κλάσεις μιάς 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.

Εντοπισμός κλάσεων βιβλιοθηκών

ΕΠΛ233

- Ο διερμηνέας εντοπίζει τις κλάσεις μιάς 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

Ονοματολογία στην Java

- Ας υποθέσουμε ότι θέλετε να δημιουργήσετε και να τοποθετήσετε κάποιες κλάσεις στο Διαδίκτυο, οπότε θέλετε να τους δώσετε και μοναδική ονομασία.

Ονοματολογία στην Java

ΕΠΛ233

- 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.:

Ονοματολογία στην Java

ΕΠΛ233

- 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 com.bruceeckel.util;

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 com.bruceeckel.util;

  package cy.ac.ucy.cs.epl233.util;

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 com.bruceeckel.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 :

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 com.bruceeckel.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

Ονοματολογία στην Java

ΕΠΛ233

- 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 com.bruceeckel.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:

Ονοματολογία στην Java

ΕΠΛ233

- 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 com.bruceeckel.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

Ονοματολογία στην Java

ΕΠΛ233

- 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 com.bruceeckel.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

ΕΠΛ233

- 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 .

Συγκρούσεις

Συγκρούσεις

- Ας υποθέσουμε ότι σε κάποιο πρόγραμμα Java εισάγονται δύο βιβλιοθήκες, οι οποίες περιλαμβάνουν κλάσεις με τα ίδια ονόματα. Π.χ.:

```
import com.bruceeckel.util.*;
import java.util.*;
```

Συγκρούσεις

- Ας υποθέσουμε ότι σε κάποιο πρόγραμμα Java εισάγονται δύο βιβλιοθήκες, οι οποίες περιλαμβάνουν κλάσεις με τα ίδια ονόματα. Π.χ.:

  import com.bruceeckel.util.*;

  import java.util.*;

  όπου και οι δύο βιβλιοθήκες ορίζουν την κλάση Vector.

Συγκρούσεις

- Ας υποθέσουμε ότι σε κάποιο πρόγραμμα Java εισάγονται δύο βιβλιοθήκες, οι οποίες περιλαμβάνουν κλάσεις με τα ίδια ονόματα. Π.χ.:

  import com.bruceeckel.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.

Συγκρούσεις

ΕΠΛ233

- Ας υποθέσουμε ότι σε κάποιο πρόγραμμα Java εισάγονται δύο βιβλιοθήκες, οι οποίες περιλαμβάνουν κλάσεις με τα ίδια ονόματα. Π.χ.:

  import com.bruceeckel.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).

Προκαθορισμένη Βιβλιοθήκη
(default package)

ΕΠΛ233

- Αρχεία κλάσεων τα οποία:
  - Είναι τοποθετημένα στον ίδιο υποκατάλογο
  - Δεν περιέχουν δήλωση package

  Θεωρούνται ότι ανήκουν στην ίδια βιβλιοθήκη, το λεγόμενο default package.

Παγίδες με το CLASSPATH

Παγίδες με το CLASSPATH

- Είναι καλό να αποφεύγουμε να προσθέτουμε πολλούς καταλόγους στο CLASSPATH, ώστε να έχουμε καλύτερο έλεγχο των κλάσεων που χρησιμοποιούμε.

Παγίδες με το CLASSPATH

ΕΠΛ233

- 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.

Παγίδες με το CLASSPATH

ΕΠΛ233

- 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.

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.

Static import (Java5)

ΕΠΛ233

- Παράδειγμα πρόσβασης σε στατικά μέλη μιας κλασης:
  - double r = Math.cos(Math.PI * theta);
- Με τη χρήση στατικής εισαγωγής μπορούμε να κάνουμε χρήση των στατικών πεδίων μιας κλάσης, χωρίς να χρησιμοποιούμε το όνομα της κλάσης στην οποία ανήκουν:
  - import static java.lang.Math.PI;
  - import static java.lang.Math.*;
  - double r = cos(PI * theta);
- Η δήλωση στατικής εισαγωγής εισαγει τα στατικά μέλη μιας κλάσης (ή κλάσεων) στο πρόγραμμά μας.

ΠΡΟΣΔΙΟΡΙΣΤΕΣ ΠΡΟΣΒΑΣΗΣ

Access Specifiers

Προσδιοριστές Πρόσβασης

ΕΠΛ233

- Βασική αρχή:
  - Μια κλάση ελέγχει την πρόσβαση που μπορούν να αποκτήσουν άλλες κλάσεις στα μέλη της.

- Οι προσδιοριστές πρόσβασης **public, protected** και **private** της Java χρησιμοποιούνται μπροστά από τον ορισμό κάθε *μέλους* μιας κλάσης (είτε μεθόδου είτε δεδομένου) και καθορίζουν τις δυνατές προσβάσεις προς αυτό.

Πρόσβαση βιβλιοθήκης

ΕΠΛ233

- Σε περίπτωση που δεν χρησιμοποιείται προσδιοριστής πρόσβασης, έχουμε τον προκαθορισμένο (default) προσδιοριστής πρόσβασης, ο οποίος αναφέρεται σαν **«φιλικός»** (friendly):
  - Ο φιλικός προσδιοριστής καθορίζει ότι όλες οι κλάσεις σε ένα πακέτο (βιβλιοθήκη) έχουν πρόσβαση στα «φιλικά» μέλη, τα οποία εμφανίζονται ως «ιδιωτικά» σε όλες τις κλάσεις έξω από την βιβλιοθήκη.
  - Αφού μια μονάδα μετάφρασης (αρχείο) μπορεί να ανήκει σε μια μόνο βιβλιοθήκη, όλες οι κλάσεις μέσα σε αυτή θεωρούνται αυτομάτως φιλικές η μια προς την άλλη.

Παροχή Πρόσβασης

ΕΠΛ233

- Για να αποκτηθεί πρόσβαση προς τα μέλη μιας κλάσης, από άλλες κλάσεις, υπάρχουν οι εξής εναλλακτικές δυνατότητες:
  - Να καταστεί το μέλος της κλάσης που μας ενδιαφέρει **δημόσιο**. Όλοι μπορούν να έχουν πρόσβαση σε αυτό.
  - Να γίνει το μέλος της κλάσης που μας ενδιαφέρει **φιλικό**, οριζόμενο χωρίς κανέναν προσδιοριστή – και οι υπόλοιπες κλάσεις να τοποθετηθούν στην ίδια βιβλιοθήκη.
  - Στην περίπτωση κληρονομικότητας, με χρήση του προσδιοριστή **protected.**
  - Με δήλωση μεθόδων πρόσβασης/τροποποίησης (accessor/mutator), οι οποίες διαβάζουν και γράφουν κάποιες τιμές-μέλη της κλάσης που μας αφορά.

Η χρήση του public

ΕΠΛ233

```
package c05.dessert;
public class Cookie {
    public Cookie() {
                System.out.println("Cookie constructor");
   }
   void bite() {
        System.out.println("bite"); }
}

import c05.dessert.*;
public class Dinner {
    public Dinner() { System.out.println("Dinner constructor"); }

public static void main(String[] args) {
    Cookie x = new Cookie();
    //! x.bite(); // Can't access
} }
```

Το προκαθορισμένο (default) πακέτο

ΕΠΛ233

```
class Cake {
 public static void main(String[] args) {
     Pie x = new Pie();
     x.f();
 }}
```

Cake.java file

```
class Pie {
 void f() { System.out.println("Pie.f()");}}
```

Pie.java file

- Η κλάση **Cake** μπορεί να χρησιμοποιήσει τη μέθοδο **f()** της **Pie**, εφόσον η Cake και η Pie βρίσκονται στον ίδιο κατάλογο, **δεν** ανήκουν ρητά σε κάποια βιβλιοθήκη και άρα ο μεταφραστής θεωρεί ότι ανήκουν στο προκαθορισμένο «πακέτο».

Ιδιωτικά μέλη

ΕΠΛ233

- Ο προσδιοριστής πρόσβασης **private** υποδηλοί ότι κανείς δεν έχει πρόσβαση στο «ιδιωτικό» μέλος μιας κλάσης, **εκτός** από τις μεθόδους που είναι εσωτερικές σε αυτή την κλάση.
- Άλλες κλάσεις της ίδιας βιβλιοθήκης επίσης **δεν** έχουν πρόσβαση σε ιδιωτικά μέλη της κλάσης. Π .χ.:

```
class Sundae {
  private Sundae() {}
  static Sundae makeASundae() {
       return new Sundae();
  }}

public class IceCream {
  public static void main(String[] args) {
       Sundae y = new Sundae();
       Sundae x = Sundae.makeASundae();
     }
}
```

Διαπροσωπεία και Υλοποίηση

Διαπροσωπεία και Υλοποίηση

- Ο έλεγχος πρόσβασης αποτελεί τον μηχανισμό της JAVA για επίτευξη της απόκρυψης πληροφορίας (information hiding) ή απόκρυψης της υλοποίησης.

Διαπροσωπεία και Υλοποίηση

- Ο έλεγχος πρόσβασης αποτελεί τον μηχανισμό της JAVA για επίτευξη της απόκρυψης πληροφορίας (information hiding) ή απόκρυψης της υλοποίησης.
- Η ενσωμάτωση δεδομένων και μεθόδων σε συνδυασμό με την απόκρυψη υλοποίησης αποκαλείται συχνά ενθυλάκωση (encapsulation) και μας επιτρέπει να ορίζουμε νέους τύπους δεδομένων με χαρακτηριστικά και συμπεριφορές.

Διαπροσωπεία και Υλοποίηση

ΕΠΛ233

- Ο έλεγχος πρόσβασης αποτελεί τον μηχανισμό της JAVA για επίτευξη της απόκρυψης πληροφορίας (information hiding) ή απόκρυψης της υλοποίησης.
- Η ενσωμάτωση δεδομένων και μεθόδων σε συνδυασμό με την απόκρυψη υλοποίησης αποκαλείται συχνά ενθυλάκωση (encapsulation) και μας επιτρέπει να ορίζουμε νέους τύπους δεδομένων με χαρακτηριστικά και συμπεριφορές.
- Ο έλεγχος πρόσβασης περιορίζει την πρόσβαση σε έναν τύπο δεδομένων, για δύο λόγους:

Διαπροσωπεία και Υλοποίηση

ΕΠΛ233

- Ο έλεγχος πρόσβασης αποτελεί τον μηχανισμό της JAVA για επίτευξη της απόκρυψης πληροφορίας (information hiding) ή απόκρυψης της υλοποίησης.
- Η ενσωμάτωση δεδομένων και μεθόδων σε συνδυασμό με την απόκρυψη υλοποίησης αποκαλείται συχνά ενθυλάκωση (encapsulation) και μας επιτρέπει να ορίζουμε νέους τύπους δεδομένων με χαρακτηριστικά και συμπεριφορές.
- Ο έλεγχος πρόσβασης περιορίζει την πρόσβαση σε έναν τύπο δεδομένων, για δύο λόγους:
  - Για να προσδιορίσει τι μπορούν και τι δεν μπορούν να χρησιμοποιήσουν «τρίτοι» προγραμματιστές (client programmers): ο προγραμματιστής μιας κλάσης μπορεί να υλοποιήσει εσωτερικούς μηχανισμούς χωρίς να φοβάται ότι κάποιος τρίτος προγραμματιστής θα τους χρησιμοποιήσει κατά τρόπο μη ενδεδειγμένο.

Διαπροσωπεία και Υλοποίηση

ΕΠΛ233

- Ο έλεγχος πρόσβασης αποτελεί τον μηχανισμό της JAVA για επίτευξη της απόκρυψης πληροφορίας (information hiding) ή απόκρυψης της υλοποίησης.
- Η ενσωμάτωση δεδομένων και μεθόδων σε συνδυασμό με την απόκρυψη υλοποίησης αποκαλείται συχνά ενθυλάκωση (encapsulation) και μας επιτρέπει να ορίζουμε νέους τύπους δεδομένων με χαρακτηριστικά και συμπεριφορές.
- Ο έλεγχος πρόσβασης περιορίζει την πρόσβαση σε έναν τύπο δεδομένων, για δύο λόγους:
  - Για να προσδιορίσει τι μπορούν και τι δεν μπορούν να χρησιμοποιήσουν «τρίτοι» προγραμματιστές (client programmers): ο προγραμματιστής μιας κλάσης μπορεί να υλοποιήσει εσωτερικούς μηχανισμούς χωρίς να φοβάται ότι κάποιος τρίτος προγραμματιστής θα τους χρησιμοποιήσει κατά τρόπο μη ενδεδειγμένο.
  - Για να διαχωρίσει τη διαπροσωπεία από την υλοποίηση. Εφόσον η πρόσβαση τρίτων προγραμματιστών περιορίζεται στα δημόσια μέλη της κλάσης (τη διαπροσωπεία της), ο προγραμματιστής της κλάσης μπορεί να αλλάξει ελεύθερα ο,τιδήποτε δεν είναι δημόσιο χωρίς να επηρεάζει τον κώδικα των τρίτων προγραμματιστών.

Πρόσβαση κλάσης (class access)

Πρόσβαση κλάσης (class access)

ΕΠΛ233

- Οι προσδιοριστές πρόσβασης μπορούν να χρησιμοποιηθούν και για τον καθορισμό των κλάσεων μιας βιβλιοθήκης που είναι προσβάσιμες είτε μέσα από την ίδια βιβλιοθήκη είτε από άλλες βιβλιοθήκες.

Πρόσβαση κλάσης (class access)

- Οι προσδιοριστές πρόσβασης μπορούν να χρησιμοποιηθούν και για τον καθορισμό των κλάσεων μιας βιβλιοθήκης που είναι προσβάσιμες είτε μέσα από την ίδια βιβλιοθήκη είτε από άλλες βιβλιοθήκες.
- Μια κλάση μπορεί να προσδιορισθεί **μόνο** σαν φιλική ή σαν δημόσια.

Πρόσβαση κλάσης (class access)

ΕΠΛ233

- Οι προσδιοριστές πρόσβασης μπορούν να χρησιμοποιηθούν και για τον καθορισμό των κλάσεων μιας βιβλιοθήκης που είναι προσβάσιμες είτε μέσα από την ίδια βιβλιοθήκη είτε από άλλες βιβλιοθήκες.
- Μια κλάση μπορεί να προσδιορισθεί **μόνο** σαν φιλική ή σαν δημόσια.
- Κανονικές (μη εσωτερικές) κλάσεις **δεν** μπορούν να συνδυαστούν με προσδιοριστές πρόσβασης **private** ή **protected.**

Περιορισμοί στο Class Access

Περιορισμοί στο Class Access

- Το πολύ μια **public** κλάση σε μια μονάδα μετάφρασης (αρχείο Java).

Περιορισμοί στο Class Access

- Το πολύ μια **public** κλάση σε μια μονάδα μετάφρασης (αρχείο Java).
- Το όνομα της **public** κλάσης πρέπει να είναι το ίδο με το όνομα του αρχείου που την περιλαμβάνει.

Περιορισμοί στο Class Access

ΕΠΛ233

- Το πολύ μια **public** κλάση σε μια μονάδα μετάφρασης (αρχείο Java).
- Το όνομα της **public** κλάσης πρέπει να είναι το ίδο με το όνομα του αρχείου που την περιλαμβάνει.
- Είναι αποδεκτό (αλλά όχι συνηθισμένο)να έχουμε μια μονάδα μετάφρασης χωρίς δημόσια κλάση στο εσωτερικό της. Στην περίπτωση αυτή, το όνομα του αρχείου μπορεί να είναι οποιοδήποτε (καλό είναι, όμως, να μην χρησιμοποιούμε άσχετα ονόματα).

Καλές προγραμματιστικές πρακτικές

Καλές προγραμματιστικές πρακτικές

- Όταν κατασκευάζετε μια κλάση είναι σκόπιμο να δηλώσετε τα πεδία δεδομένων της σαν ιδιωτικά, για να τα προστατεύσετε από ανεπιθύμητες αλλαγές από άλλες κλάσεις.

Καλές προγραμματιστικές πρακτικές

- Όταν κατασκευάζετε μια κλάση είναι σκόπιμο να δηλώσετε τα πεδία δεδομένων της σαν ιδιωτικά, για να τα προστατεύσετε από ανεπιθύμητες αλλαγές από άλλες κλάσεις.
  - Αυτό ισχύει ακόμα κι αν η κλάση σας είναι προσβάσιμη από τη βιβλιοθήκη της μόνο (όχι δημόσια).

Καλές προγραμματιστικές πρακτικές

- Όταν κατασκευάζετε μια κλάση είναι σκόπιμο να δηλώσετε τα πεδία δεδομένων της σαν ιδιωτικά, για να τα προστατεύσετε από ανεπιθύμητες αλλαγές από άλλες κλάσεις.
  - Αυτό ισχύει ακόμα κι αν η κλάση σας είναι προσβάσιμη από τη βιβλιοθήκη της μόνο (όχι δημόσια).
- Επίσης, είναι συνήθως λογικό να δώσετε στις μεθόδους της κλάσης την ίδια προσβασιμότητα με την κλάση, δηλαδή είτε δημόσια είτε φιλική.

Καλές προγραμματιστικές πρακτικές

- Όταν κατασκευάζετε μια κλάση είναι σκόπιμο να δηλώσετε τα πεδία δεδομένων της σαν ιδιωτικά, για να τα προστατεύσετε από ανεπιθύμητες αλλαγές από άλλες κλάσεις.
  - Αυτό ισχύει ακόμα κι αν η κλάση σας είναι προσβάσιμη από τη βιβλιοθήκη της μόνο (όχι δημόσια).
- Επίσης, είναι συνήθως λογικό να δώσετε στις μεθόδους της κλάσης την ίδια προσβασιμότητα με την κλάση, δηλαδή είτε δημόσια είτε φιλική.
- Αν θέλετε να αποκλείσετε τη χρήση μιας κλάσης από οποιονδήποτε τρίτο προγραμματιστή, μπορείτε να καταστήσετε όλους τους κατασκευαστές της **private**. Έτσι, κανείς δεν μπορεί να δημιουργήσει αντικείμενα της κλάσης, εκτός από εσάς (μέσα από ένα στατικό μέλος της κλάσης).

Παράδειγμα

ΕΠΛ233

```
class Soup1 {
  private Soup1() {}
  public static Soup1 makeSoup() {
        return new Soup();
  }
}
class Soup2 {// singleton
  private Soup2() { }
  private static Soup2 ps1 = new Soup2();
  public static Soup2 access() {
        return ps1;
  }
  public void f() {}
}
```

Παράδειγμα (συνέχεια)

ΕΠΛ233

```
public class Lunch {
  void test1() {
        // Soup priv1 = new Soup();
  }

  void test2() {
        Soup1 soup = Soup1.makeSoup();
        Soup2.access().f();
   }
}
```