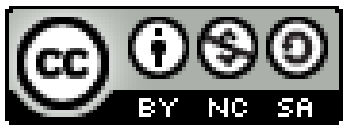
ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Αντικειμενοστρεφής Προγραμματισμός

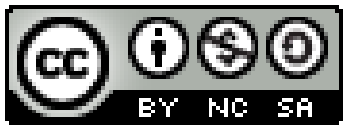
Ενότητα 7: Βελτίωση Δομής με Κληρονομικότητα

Γρηγόρης Τσουμάκας, Επικ. Καθηγητής
Τμήμα Πληροφορικής



Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο την 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.

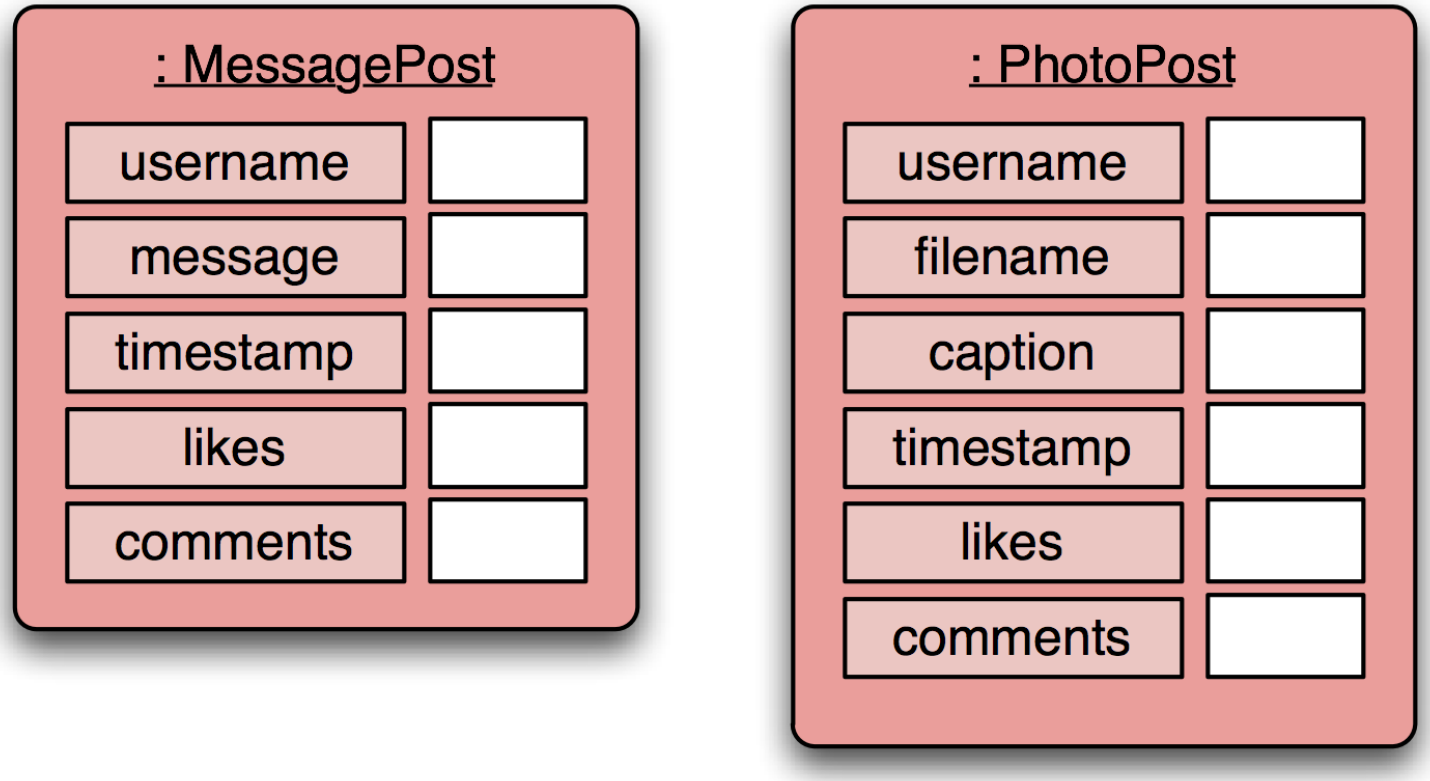
Βελτίωση Δομής με Κληρονομικότητα

Τα παραδείγματα κώδικα που χρησιμοποιούνται σε κάποιες από τις ακόλουθες διαφάνειες μπορούν να βρεθούν στον παρακάτω σύνδεσμο:
http://users.auth.gr/greg/oop.zip

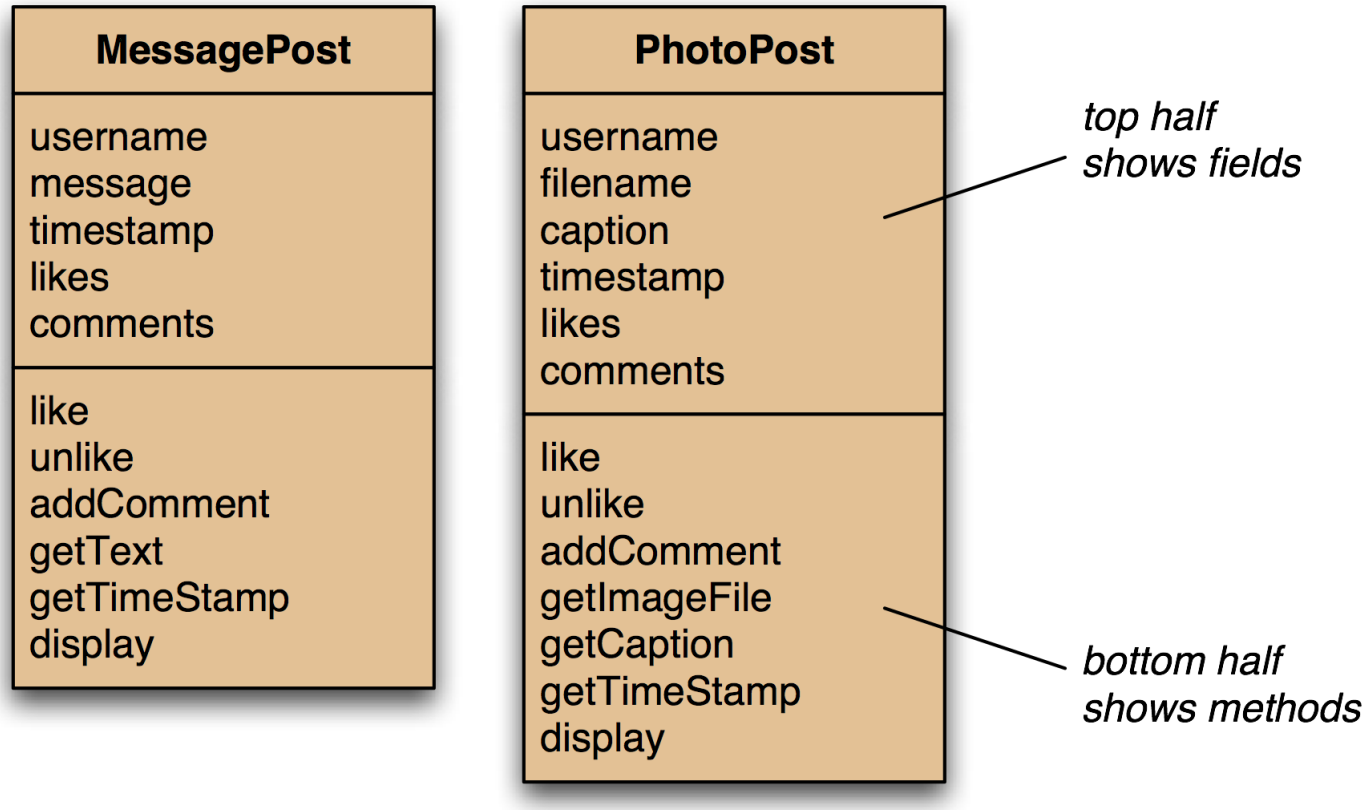
Το Παράδειγμα

- Ένα απλό αρχέτυπο κοινωνικό δίκτυο.
  - Υλοποιεί μια ροή με αναρτήσεις από διάφορα μέλη του κοινωνικού δικτύου.
- Τύποι αναρτήσεων.
  - Απλό κείμενο.
  - Φωτογραφία με λεζάντα.
- Επιτρέπει (αργότερα) την αναζήτηση, εμφάνιση και διαγραφή αναρτήσεων.

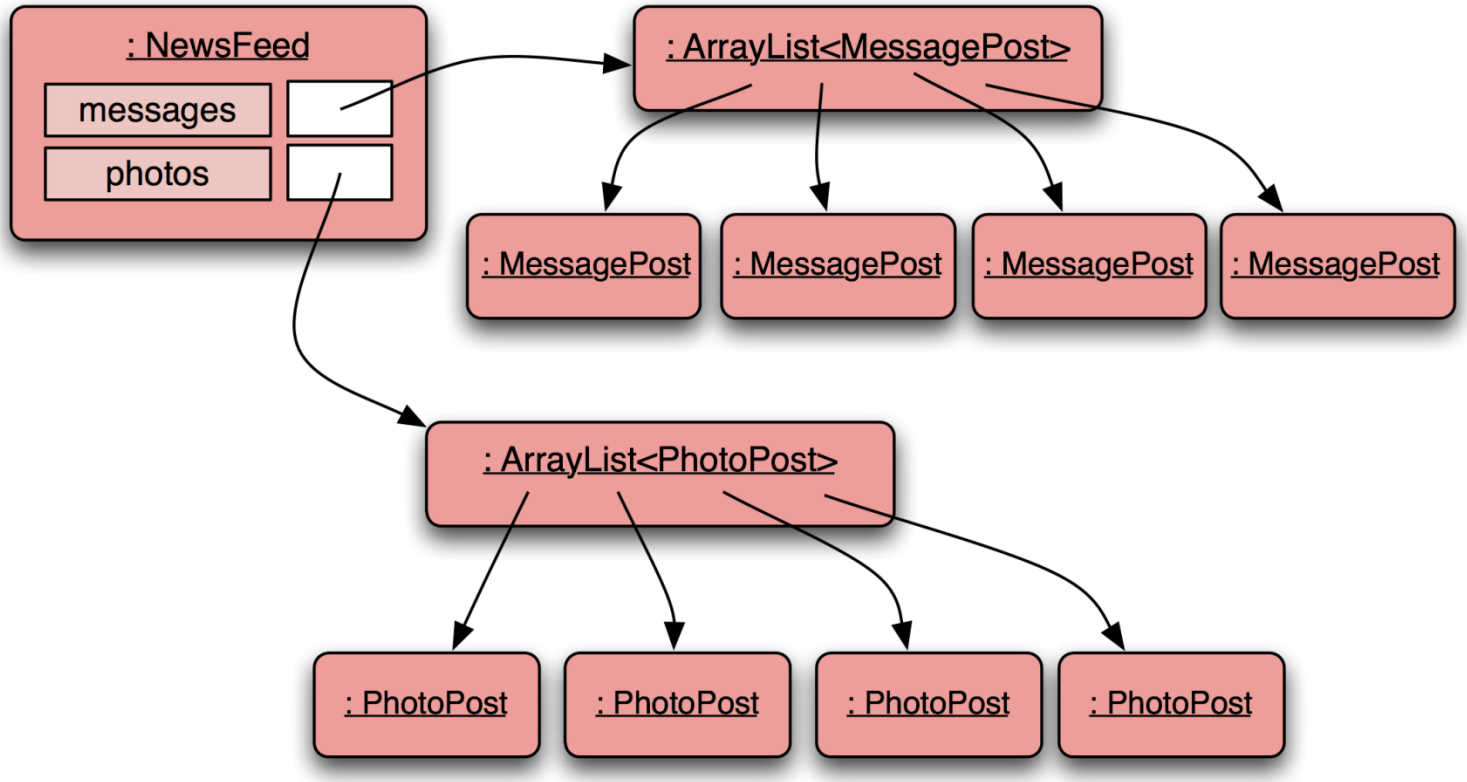
Αντικείμενα



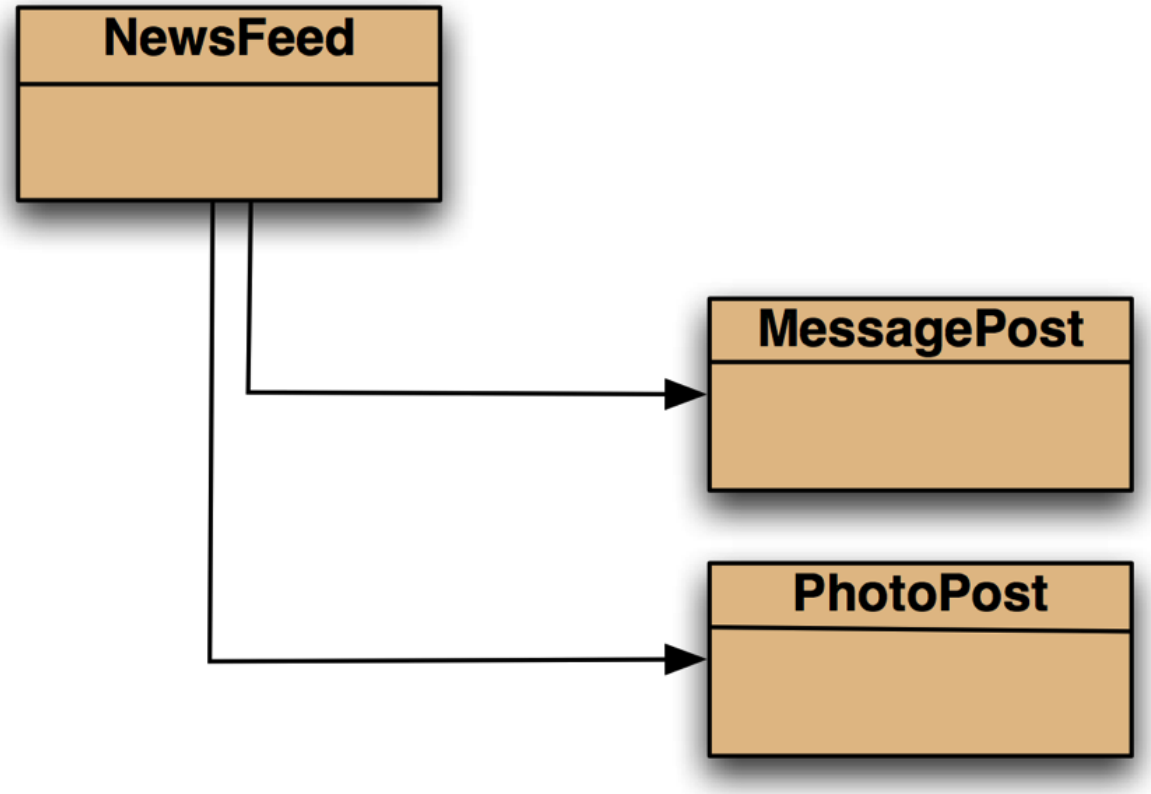
Κλάσεις (σε UML)



Διάγραμμα Αντικειμένων



Διάγραμμα Κλάσεων



Εξέταση του Κώδικα

- Κλάση MessagePost.
- Κλάση PhotoPost.
- Κλάση NewsFeed.

version1

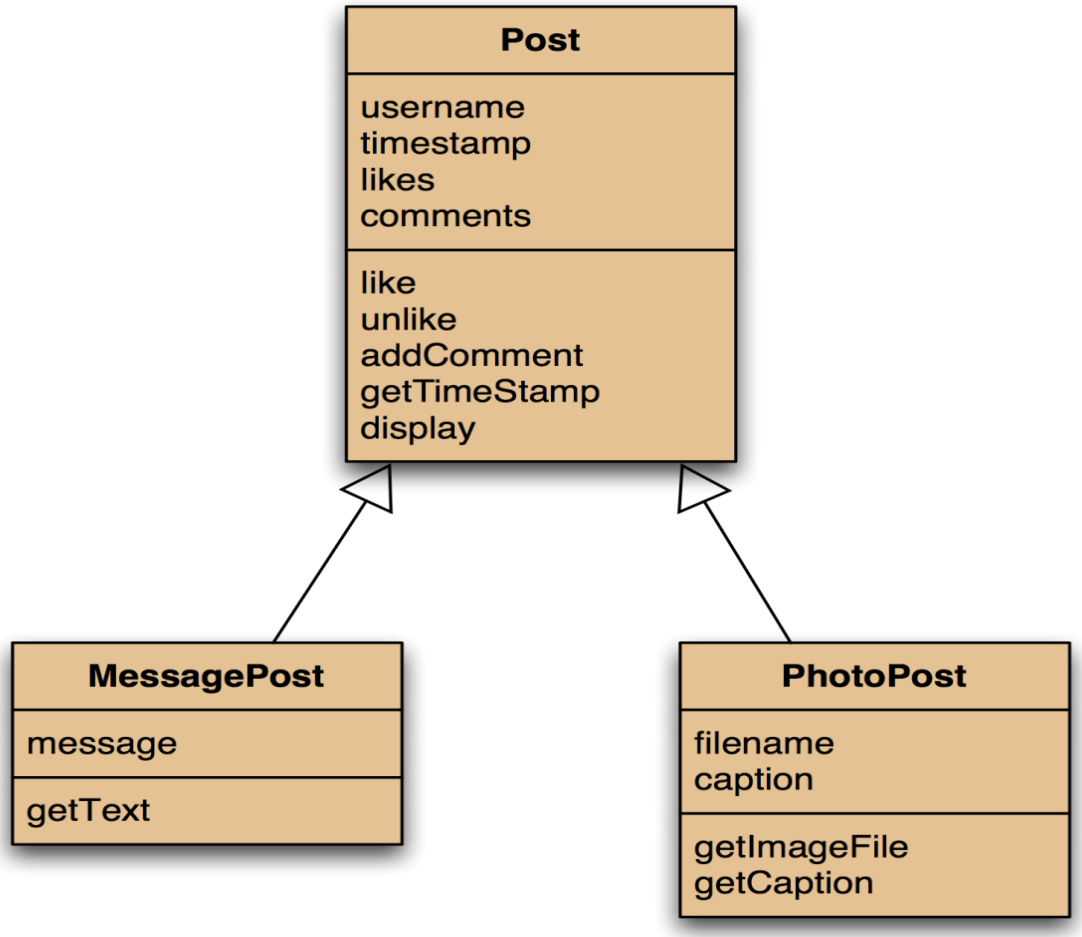
Μειονεκτήματα

- Πανομοιότυπα τμήματα κώδικα στις κλάσεις. MessagePost και PhotoPost.
  - Η συγγραφή και η συντήρηση του κώδικα απαιτεί τον διπλάσιο κόπο.
  - Ελλοχεύει ο κίνδυνος εισχώρησης σφαλμάτων κατά τη συντήρηση.
- Πανομοιότυπα τμήματα κώδικα και στην κλάση NewsFeed.

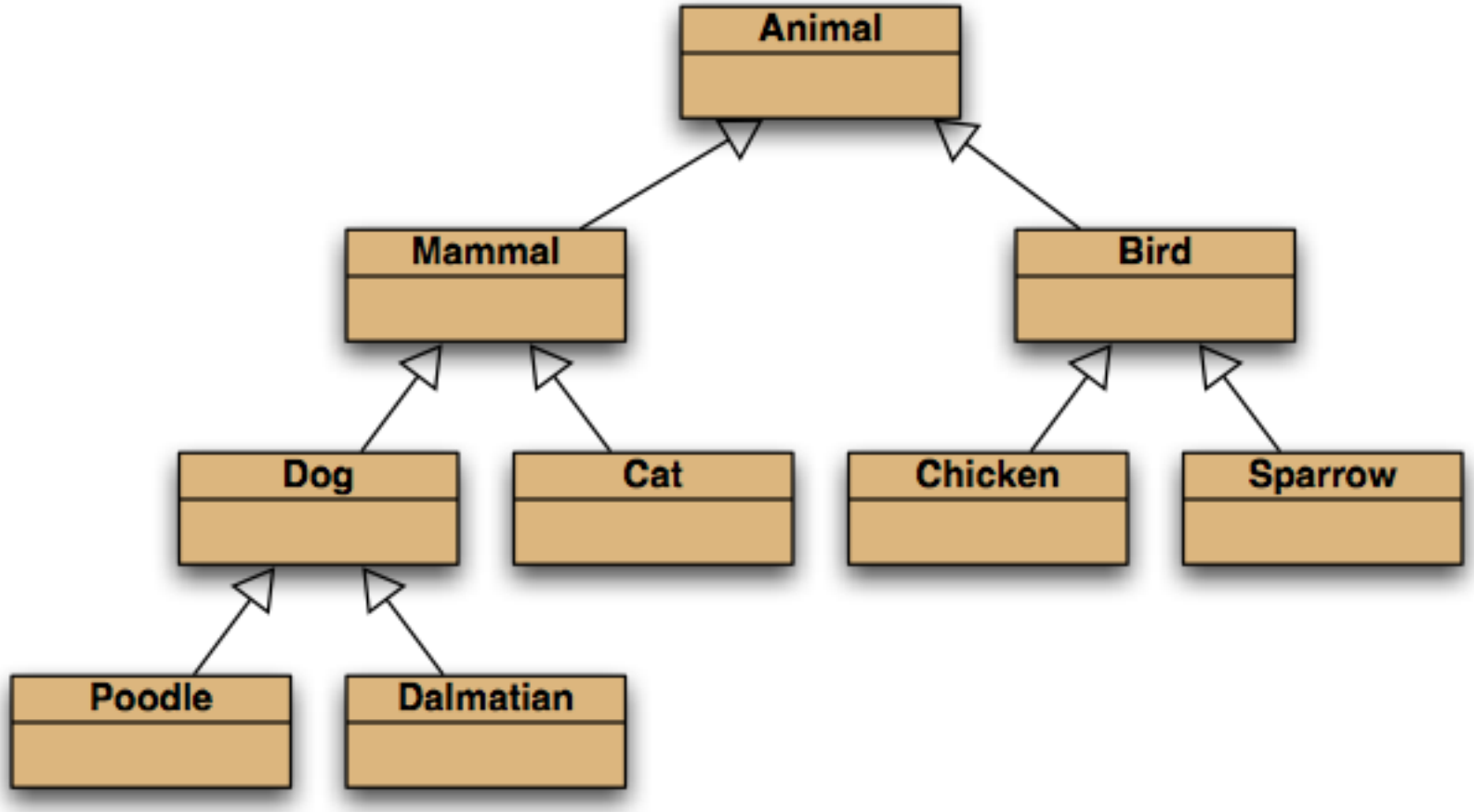
Χρήση Κληρονομικότητας (1/2)

- Ορίζουμε την υπερκλάση ή γονική κλάση Post.
  - Περιλαμβάνει οτιδήποτε κοινό έχουν οι κλάσεις MessagePost και PhotoPost.
- Ορίζουμε υποκλάσεις MessagePost, PhotoPost.
  - Λέμε ότι κληρονομούν ή επεκτείνουν την Post.
  - Προσθέτουμε επιπλέον πεδία και μεθόδους.
- Κληρονομικότητα ως σχέση είναι (is-a).
  - Μια υποκλάση αποτελεί εξειδίκευση μιας υπερκλάσης.

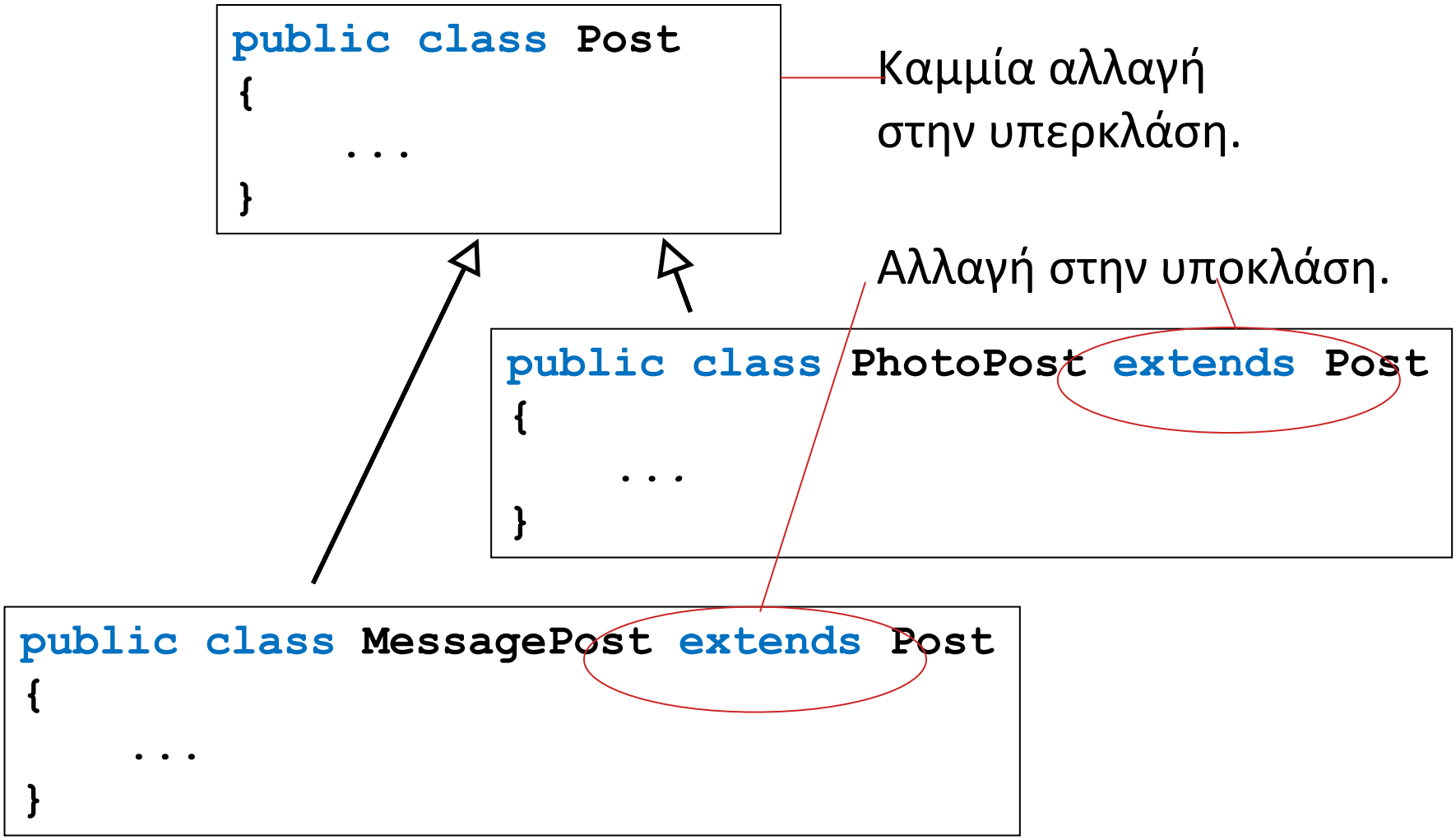
Χρήση Κληρονομικότητας (2/2)



Ιεραρχίες Κληρονομικότητας



Κληρονομικότητα στη Java



Υπερκλάση

```
public class Post {
    private String username;
    private long timestamp;
    private int likes;
    private ArrayList<String> comments;

    // constructor and methods omitted
}
```

version2

Υποκλάσεις

```
public class MessagePost extends Post {
    private String message;

    // constructor and methods omitted.
}
```

```
public class PhotoPost extends Post {
    private String filename;
    private String caption;

    // constructor and methods omitted
}
```

Κατασκευαστές: Υπερκλάση

```
public class Post {
  private String username;
  private long timestamp;
  private int likes;
  private ArrayList<String> comments;

  public Post(String author) {
    username = author;
    timestamp = System.currentTimeMillis();
    likes = 0;
    comments = new ArrayList<>();
  }

  // methods omitted
}
```

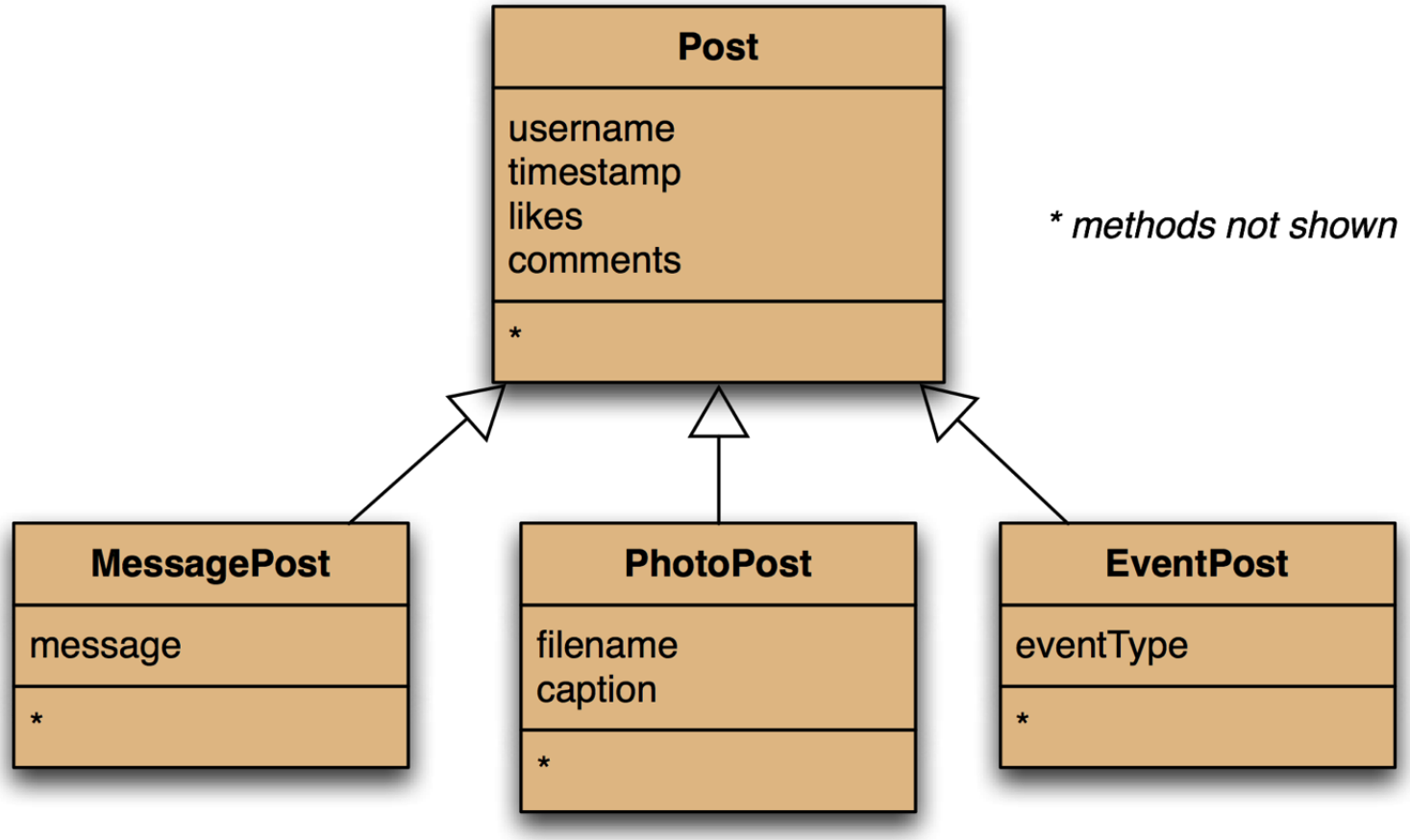
Κατασκευαστές: Υποκλάση

```
public class MessagePost extends Post {
   private String message;

   public MessagePost(String who, String txt) {
      super(who);
      message = txt;
   }

   // methods omitted
}
```

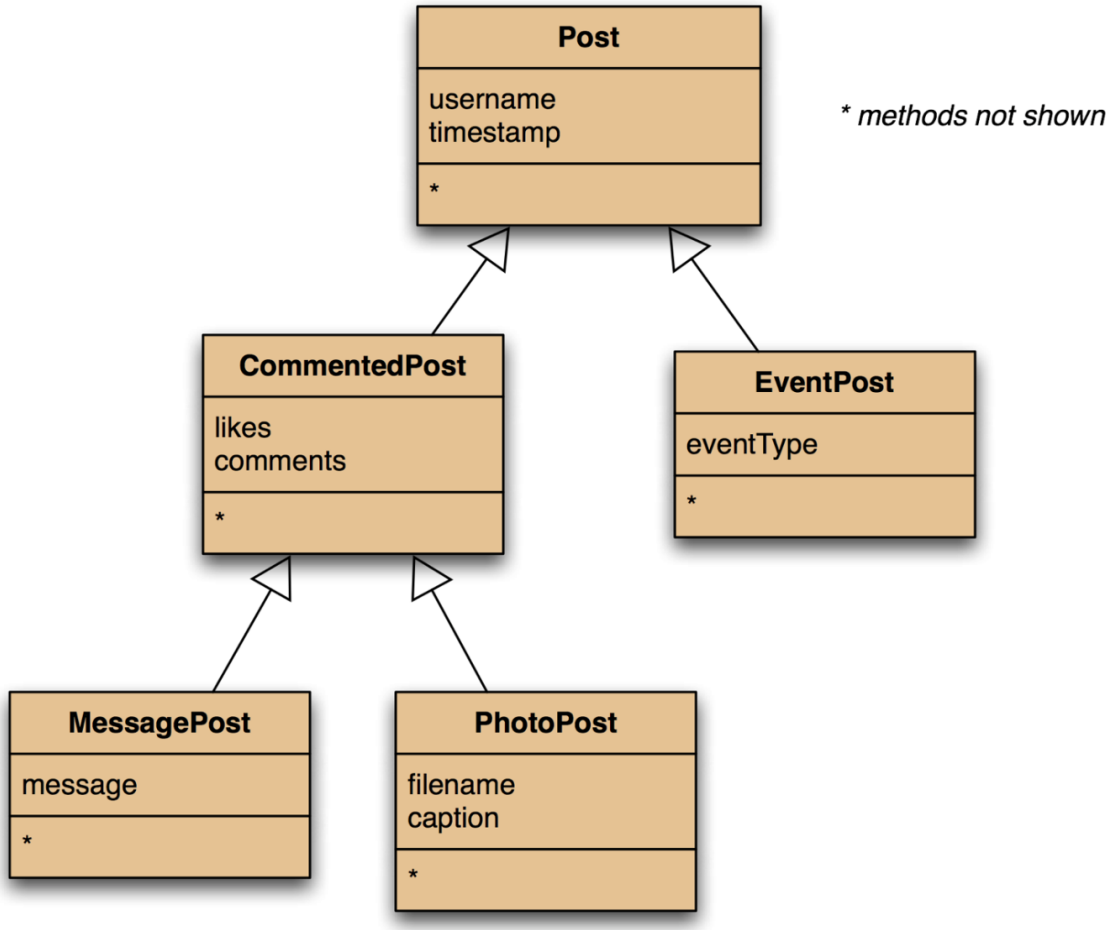
Κλήση Κατασκευαστή Υπερκλάσης

- Οι κατασκευαστές υποκλάσης πρέπει πάντα να περιλαμβάνουν μια κλήση super.
  - Διαφορετικά προστίθεται αυτόματα μια κλήση super χωρίς παραμέτρους από τον μεταγλωττιστή.
  - Θα πρέπει όμως η υπερκλάση να διαθέτει κατασκευαστή χωρίς παραμέτρους.
  - Είναι καλή πρακτική να υπάρχει ρητή κλήση super.
- Η κλήση super πρέπει να είναι η πρώτη πρόταση στον κατασκευαστή υποκλάσης.

Προσθήκη Επιπλέον Τύπων



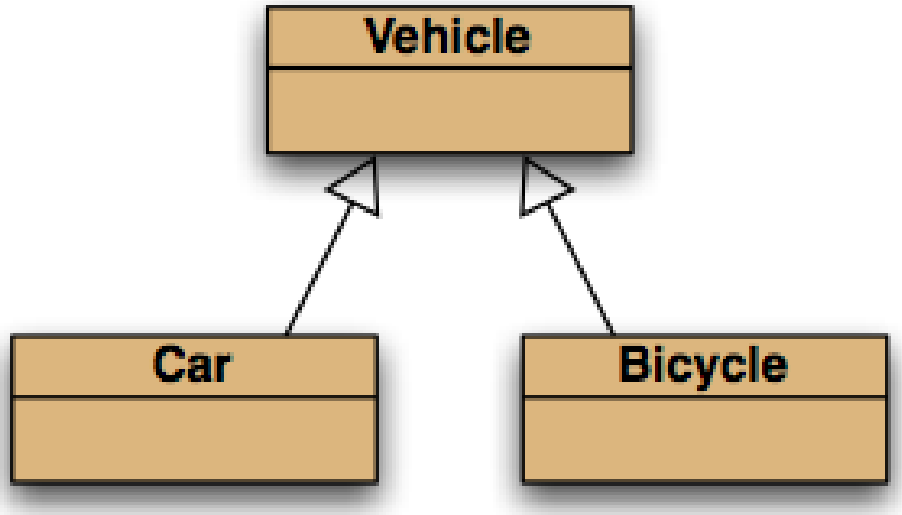
Βαθύτερες Ιεραρχίες



Μεταβλητές και Υποτύποι (1/3)

- Πολυμορφισμός.
  - Οι μεταβλητές που αποθηκεύουν αντικείμενα μπορεί να περιέχουν αντικείμενα της δηλωμένης κλάσης ή οποιασδήποτε υποκλάσης της.
  - Κλάση == τύπος, υποκλάση == υποτύπος.
- Ο στατικός τύπος της μεταβλητής.
  - Ο δηλωμένος τύπος της μεταβλητής.
- Ο δυναμικός τύπος της μεταβλητής.
  - Ο τύπος του αντικειμένου στο οποίο δείχνει.

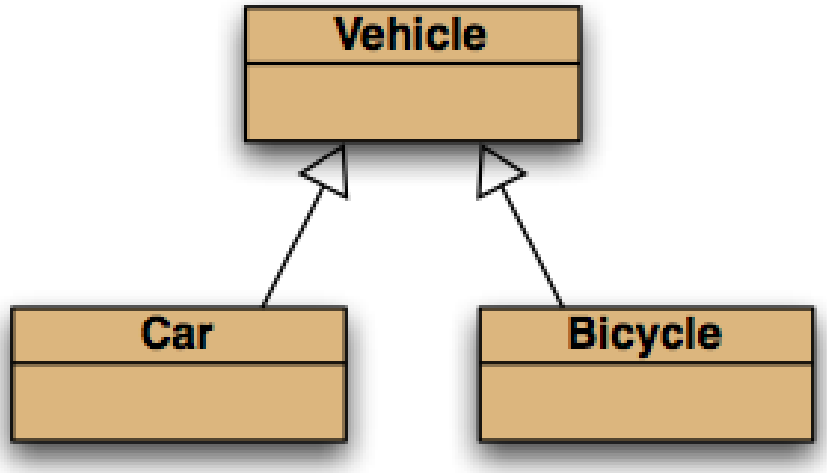
Μεταβλητές και Υποτύποι (2/3)



Αντικείμενα υποκλάσεων μπορούν να ανατεθούν σε μεταβλητές υπερκλάσης (υποκατάσταση).

```
Vehicle v1 = new Vehicle();
Vehicle v2 = new Car();
Vehicle v3 = new Bicycle();
```

Μεταβλητές και Υποτύποι (3/3)



Αντικείμενα υπερκλάσεων ΔΕΝ μπορούν να ανατεθούν σε μεταβλητές υποκλάσης.

```
Car c1 = new Vehicle(); // σφάλμα
Bicycle b1 = new Vehicle(); // σφάλμα
Car c2 = new Bicycle(); // σφάλμα
Vehicle v1 = new Car();
Car c3 = v1; // σφάλμα (απώλεια τύπου)
```

Νέα Έκδοση Κλάσης NewsFeed 1/2

```
public class NewsFeed {
    private ArrayList<Post> posts;

    public NewsFeed() {
        posts = new ArrayList<>();
    }

    public void addPost(Post post) {
        posts.add(post);
    }
    ...
}
```

Νέα Έκδοση Κλάσης NewsFeed 2/2

```
public void show() {

    for (Post post : posts) {
        post.display();
        System.out.println();
    }
}
```

Παράδειγμα Υποκατάστασης

- Αρχικά είχαμε

```
public void addMessagePost(MessagePost p)
public void addPhotoPost(PhotoPost p)
```

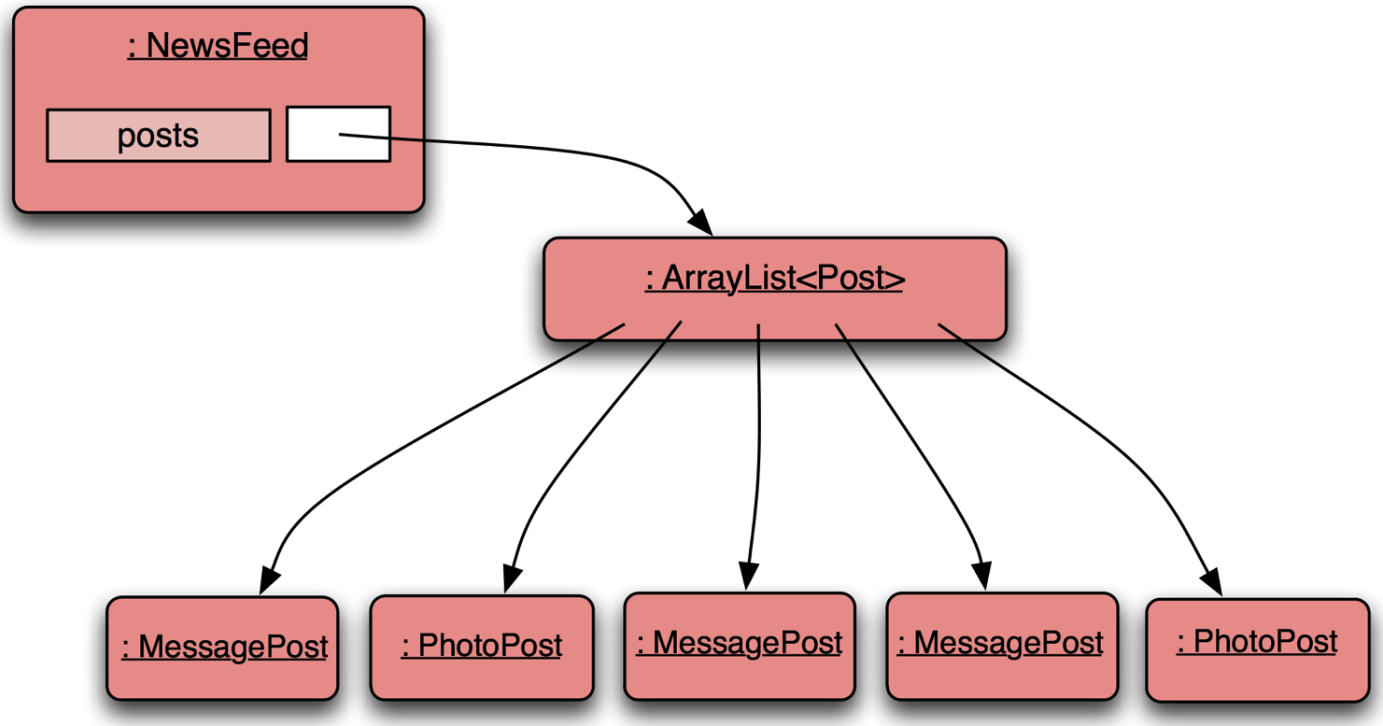
- Τώρα έχουμε

```
public void addPost(Post p)
```

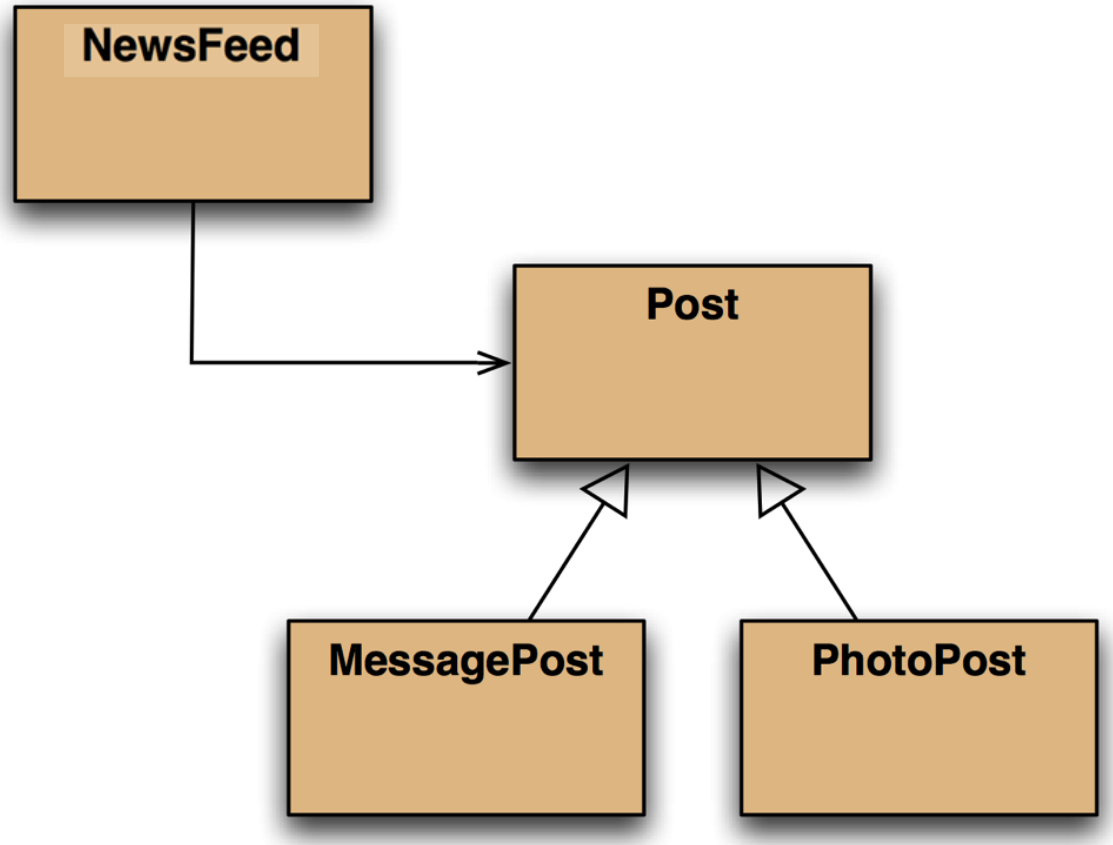
- Καλούμε αυτή τη μέθοδο ως εξής.

```
PhotoPost photo = new PhotoPost(…);
feed.addPost(photo);
MessagePost message = new MessagePost(…);
feed.add(message);
```

Διάγραμμα Αντικειμένων



Διάγραμμα Κλάσεων

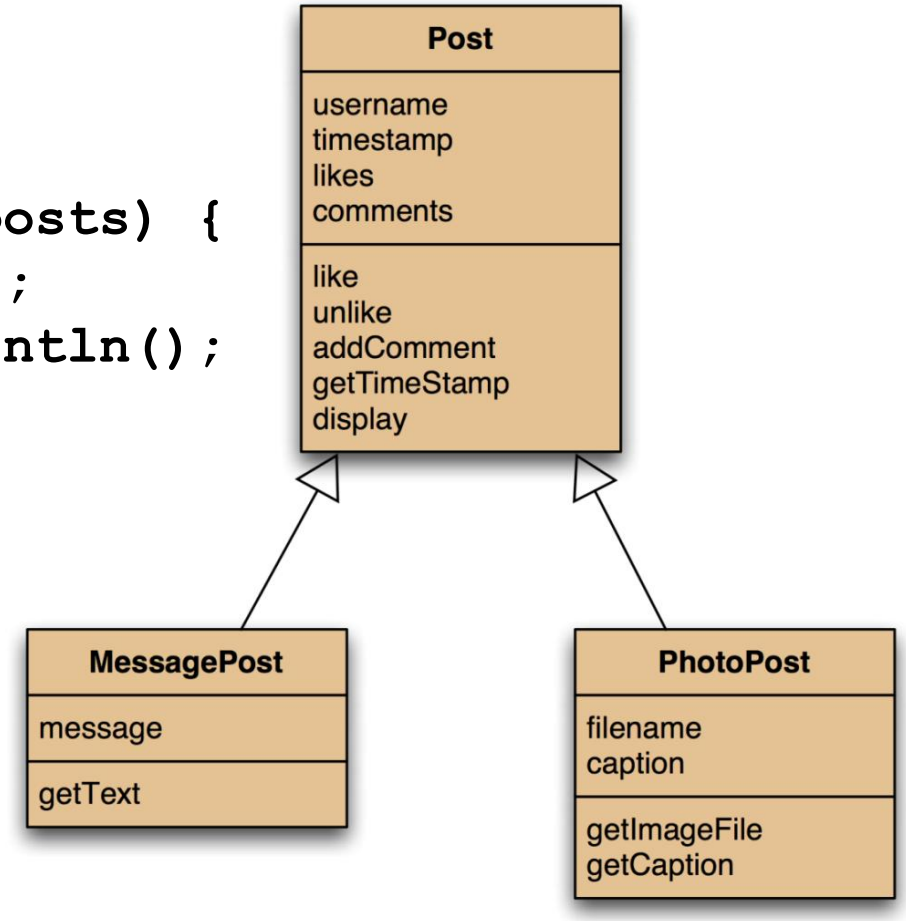


Άσκηση

- Υποθέστε ότι
  - Έχουμε κλάση Person με υποκλάσεις Teacher και Student, καθώς και υποκλάση PhDStudent της κλάσης Student.
- Ποιες από τις παρακάτω εντολές είναι έγκυρες;
  - `Person p1 = new Student();`
  - `Person p2 = new PhDStudent();`
  - `PhDStudent phd1 = new Student();`
  - `Teacher t1 = new Person();`
  - `Student s1 = new PhDStudent();`

Αποτέλεσμα Μεθόδου display (1/2)

```
public void show() {
    for (Post post : posts) {
        post.display();
        System.out.println();
    }
}
```



Αποτέλεσμα Μεθόδου display (2/2)

```
Leonardo da Vinci
Had a great idea this morning.
But now I forgot what it was. Something to do with flying ...
40 seconds ago - 2 people like this.
  No comments.

Alexander Graham Bell
[experiment.jpg]
I think I might call this thing 'telephone'.
12 minutes ago  -  4 people like this.
  No comments.
```

Τι θα θέλαμε

```
Leonardo da Vinci
40 seconds ago  -  2 people like this.
  No comments.

Alexander Graham Bell
12 minutes ago  -  4 people like this.
  No comments.
```

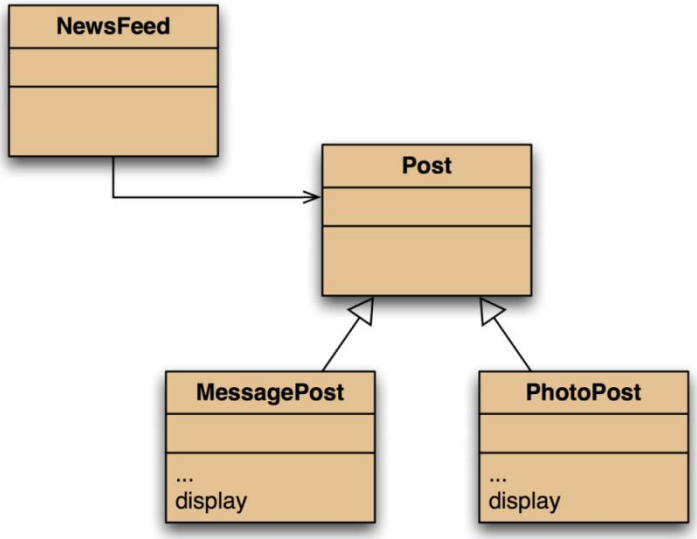
Τι έχουμε

Το Πρόβλημα

- Η μέθοδος *display* στην κλάση *Post* τυπώνει μόνο τα κοινά πεδία.
- Η κληρονομικότητα δεν είναι δρόμος διπλής κατεύθυνσης.
  - Η υποκλάση κληρονομεί τα πεδία της υπερκλάσης.
  - Η υπερκλάση δεν γνωρίζει τι πεδία υπάρχουν στις υποκλάσεις.

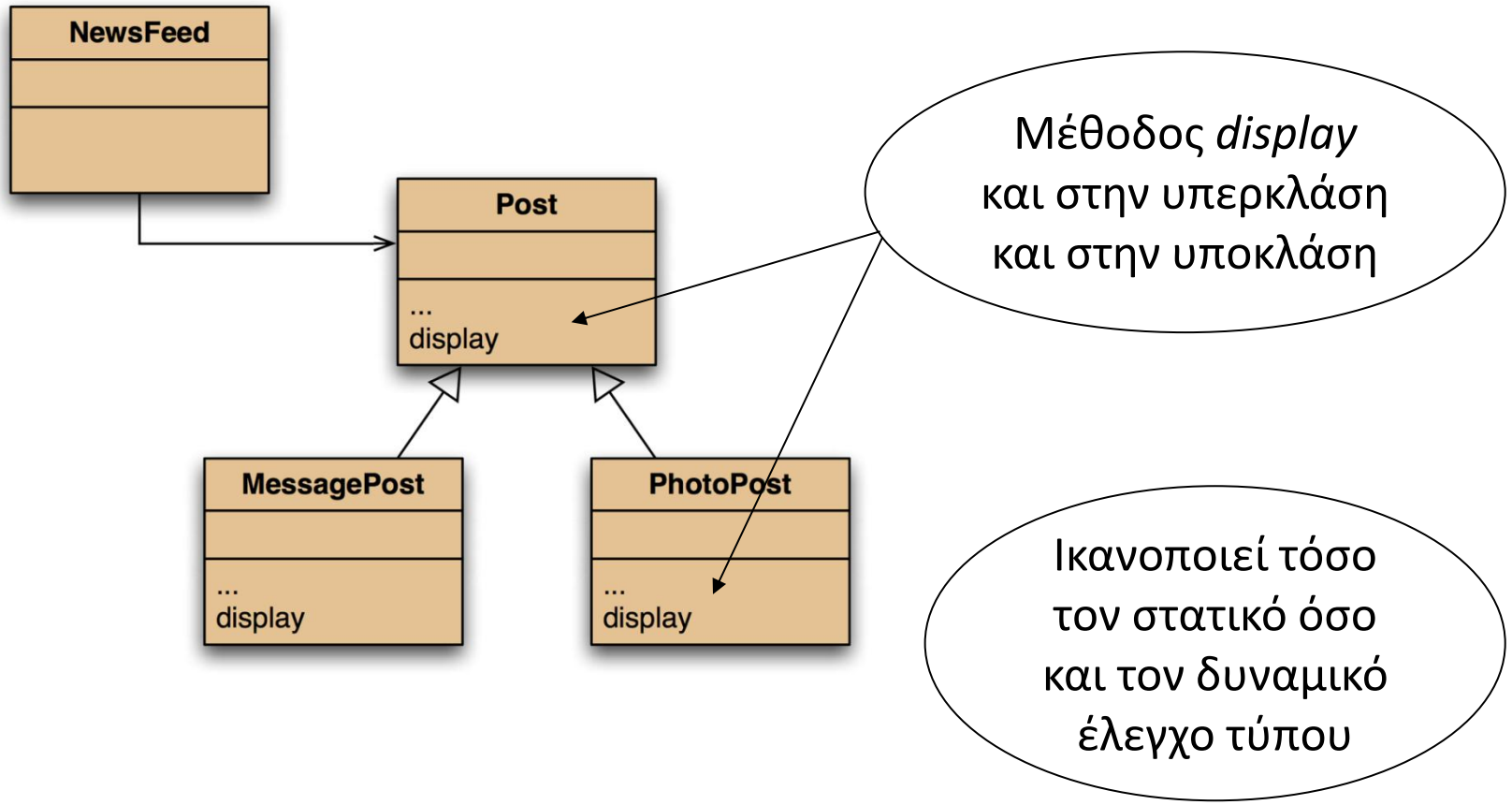
Προσπάθεια Αντιμετώπισης

- Τοποθέτηση *display* εκεί όπου έχει πρόσβαση στις απαραίτητες πληροφορίες.
  - Διαφορετική έκδοση σε κάθε υποκλάση.
- Όμως τα πεδία της *Post* είναι ιδιωτικά.
  - Μέθοδοι πρόσβασης.
  - Πρόσβαση protected.
- Η *NewsFeed* δεν βρίσκει μέθοδο *display* στην *Post*.




```
public void show() {
    for (Post post : posts) {
        post.display();
        System.out.println();
    }
}
```

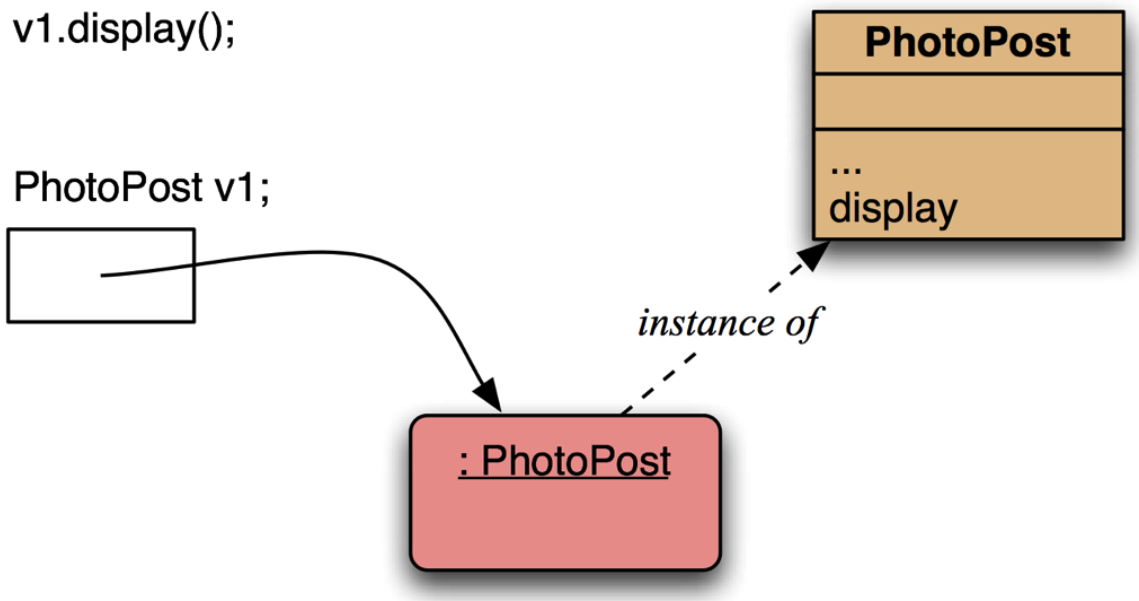
Η Λύση της Υποσκέλισης



Υποσκέλιση

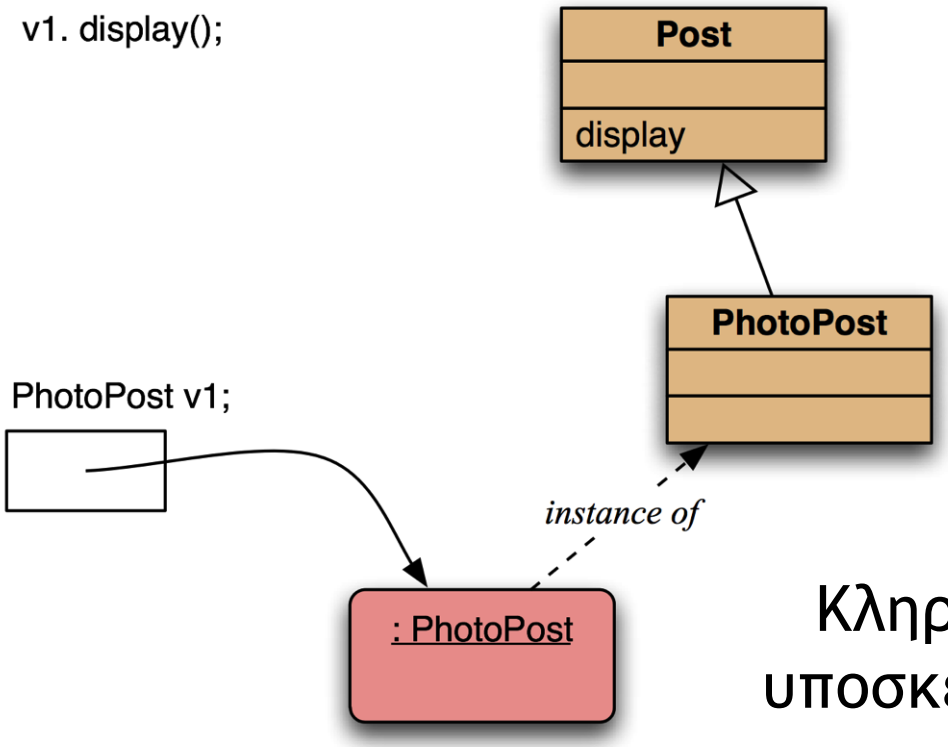
- Η υπερκλάση και η υποκλάση ορίζουν μεθόδους με την ίδια υπογραφή.
  - Κάθε μία έχει πρόσβαση στα πεδία της κλάσης της.
- Η μέθοδος της υπερκλάσης ικανοποιεί τον έλεγχο του στατικού τύπου.
- Κατά το χρόνο εκτέλεσης καλείται η μέθοδος της υποκλάσης.
  - Υποσκελίζει την μέθοδο της υπερκλάσης.
  - Τι γίνεται η μέθοδος της υπερκλάσης;

Αναζήτηση Μεθόδου (1/4)



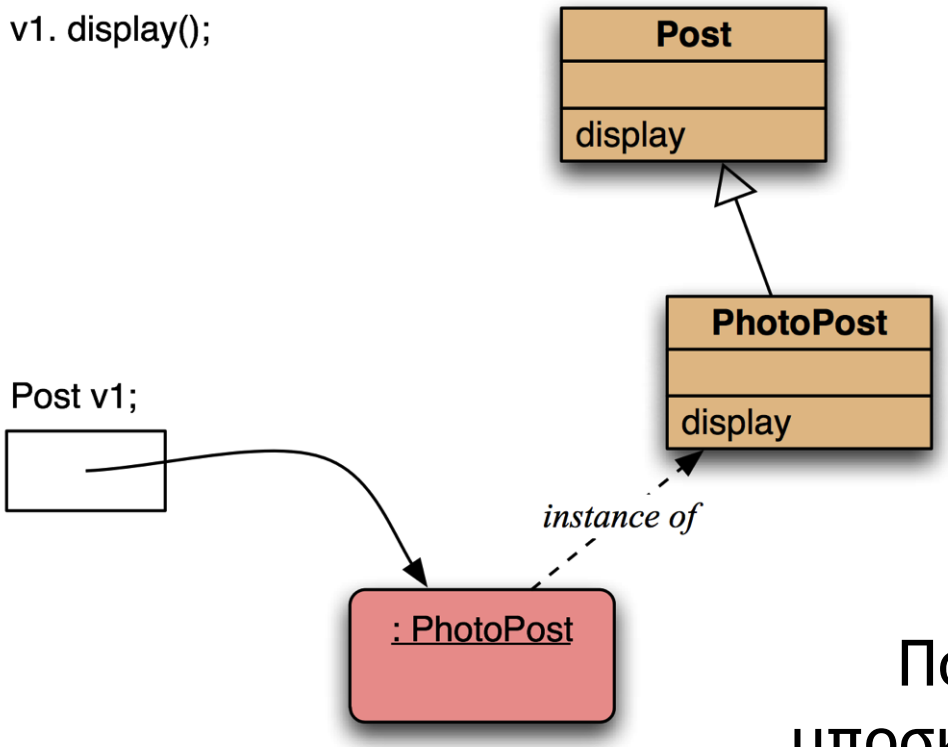
Δίχως κληρονομικότητα ή πολυμορφισμό.
Καλείται η προφανής μέθοδος.

Αναζήτηση Μεθόδου (2/4)



Κληρονομικότητα δίχως υποσκέλιση. Ανεβαίνουμε την ιεραρχία αναζητώντας για κατάλληλη μέθοδο.

Αναζήτηση Μεθόδου (3/4)



Πολυμορφισμός και υποσκέλιση. Η "πρώτη" έκδοση που θα βρεθεί χρησιμοποιείται.

Αναζήτηση Μεθόδου (4/4)

- Εύρεση του δυναμικού τύπου της μεταβλητής.
  - Δηλαδή του τύπου του αντικειμένου που περιέχει.
- Αναζήτηση μεθόδου στην κλάση αυτή.
- Αν δεν βρεθεί εκεί ψάχνουμε στην υπερκλάση.
- Αυτό συνεχίζεται μέχρι να βρεθεί μέθοδος ή να εξαντληθεί η ιεραρχία κλάσεων.

version3

Πολυμορφισμός Μεθόδων

- Πολυμορφικές μεταβλητές.
  - Μπορούν να αποθηκεύουν αντικείμενα διαφορετικών κλάσεων.
- Πολυμορφικές κλήσεις μεθόδων.
  - Η μέθοδος που θα κληθεί εξαρτάται από τον δυναμικό τύπο κατά το χρόνο εκτέλεσης.
  - Μπορεί να κληθεί η display() της MessagePost ή αυτή της PhotoPost.

Χρήση *super* σε Μεθόδους

- Οι υποσκελισμένες μέθοδοι είναι *κρυμμένες*, αλλά θέλουμε να μπορούμε να τις καλούμε.
- Μια υποσκελισμένη μέθοδος μπορεί να κληθεί από τη μέθοδο που την υποσκελίζει.

```
super.method(...)
```

- Διαφορές με *super* στους κατασκευαστές.
  - Δεν απαιτείται να είναι η πρώτη γραμμή κώδικα.
  - Δεν καλούνται αυτόματα (προαιρετική χρήση).

Κλήση Υποσκελισμένης Μεθόδου

```
public void display() {
   super.display();
   System.out.println(" [" + filename + "]");
   System.out.println(" " + caption);
}
```

PhotoPost

```
public void display() {
   super.display();
   System.out.println(message);
}
```

MessagePost

version4

Πλεονεκτήματα Κληρονομικότητας

- Αποφυγή πανομοιότυπων τμημάτων κώδικα.
- Επαναχρησιμοποίηση κώδικα.
- Απλοποίηση του κώδικα.
- Ευκολότερη συντήρηση.
- Επεκτασιμότητα.

Ένα Ακόμα Παράδειγμα (1/2)

```
public class Parent {
  public void foo() { System.out.println("Parent"); }
  public void bar() { foo(); }
}
-----------------------------------------------------
public class Child extends Parent {
  public void foo() { System.out.println("Child"); }
}
-----------------------------------------------------
Parent p = new Child();
p.bar();
```

Ένα Ακόμα Παράδειγμα (2/2)

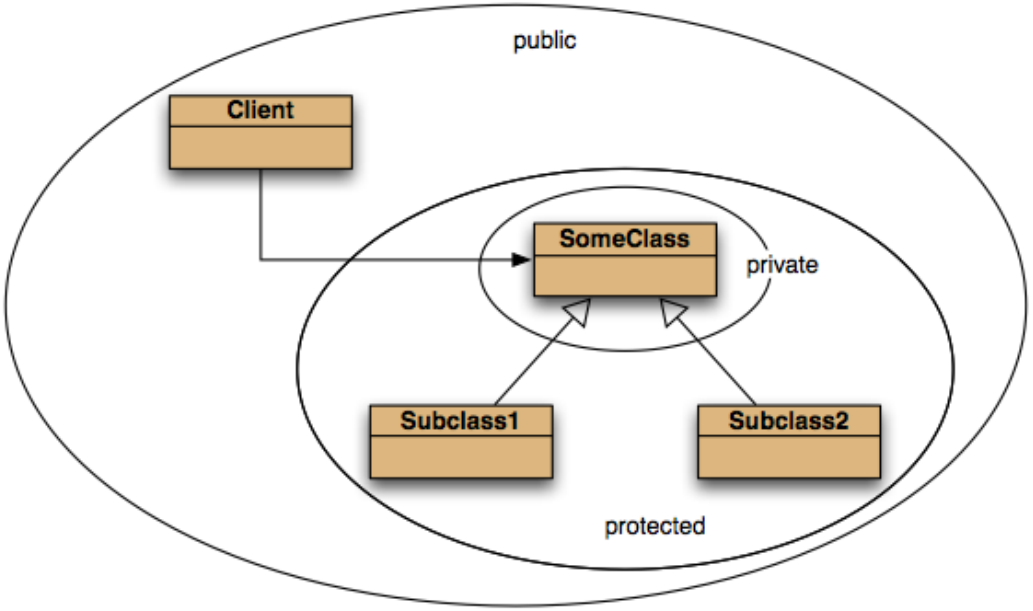
```
public class Parent {
  private void foo() { System.out.println("Parent"); }
  public void bar() { foo(); }
}
-----------------------------------------------
public class Child extends Parent {
  private void foo() { System.out.println("Child"); }
}
-----------------------------------------------
Parent p = new Child();
p.bar();
```

Προστατευμένη Πρόσβαση

- Περιοριστική η απαγόρευση πρόσβασης σε πεδία της υπερκλάσης από μια υποκλάση.
  - Οι σχέσεις κληρονομικότητας είναι στενότερες από τις απλές αλληλεπιδράσεις μεταξύ κλάσεων.
- Χρήση του τελεστή πρόσβασης *protected.*
  - Πιο περιορισμένη πρόσβαση από *public.*
- Προτείνεται ωστόσο και πάλι η χρήση *private.*
  - Πρόσβαση στην υπερκλάση μέσω *protected* μεθόδων πρόσβασης και μετάλλαξης.

Επίπεδα Πρόσβασης

Γενικότερα στον αντικειμενοστρεφή προγραμματισμό

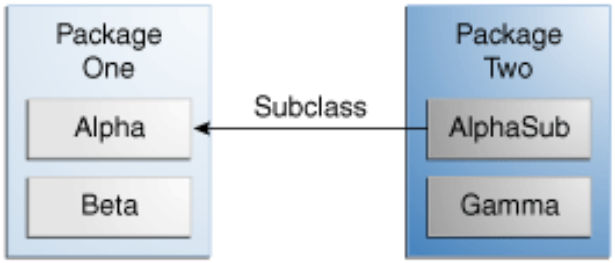


Ειδικότερα στη Java

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

Άσκηση

- Συμπληρώστε στον παρακάτω πίνακα αν είναι ορατό ή όχι κάποιο μέλος της κλάσης *Alpha* από τις κλάσεις *Alpha*, *Beta*, *AlphaSub* και *Gamma* δεδομένου του τροποποιητή πρόσβασής του.



Modifier	Alpha	Beta	Alphasub	Gamma
public				
protected				
no modifier				
private				

Προσαρμογή Τύπου

- Προσαρμογή τύπου (type casting).
  - Μετατροπή τύπου σε άλλον "συμβατό" τύπο.
  - Ο επιθυμητός τύπος προηγείται εντός παρένθεσης.

Προσαρμογή Θεμελιωδών Τύπων

```
int sum=78, length=10;
double mean1, mean2;
mean1 = sum / length;
System.out.println(mean1);
mean2 = (double) sum / length;
System.out.println(mean2);
int x = (int) mean2;
System.out.println(x);
// 7.0
// 7.8
// 7
```

Casting.java

Προσαρμογή Υπερτύπου

- Μετατροπή μεταβλητής υπερτύπου στον δυναμικό υποτύπο του αντικειμένου της.

```
Vehicle v1 = new Bicycle();
Bicycle b1 = v1; // σφάλμα μεταγλώττισης
Bicycle b2 = (Bicycle) v1; // σωστό
Car c1 = (Car) b2; // σφάλμα μεταγλώττισης
Car c2 = (Car) v1; // σφάλμα εκτέλεσης
```

- Θα πρέπει να αποφεύγεται.
  - Ελλοχεύει ο κίνδυνος σφαλμάτων εκτέλεσης.
  - Χρήση πολυμορφικών μεθόδων στη θέση της.

Ο Τελεστής instanceof

- anObject instanceof SomeClass.
- Επιστρέφει *true* αν ο δυναμικός τύπος μιας μεταβλητής (αριστερός τελεστέος) είναι μιας συγκεκριμένης κλάσης (δεξιός τελεστέος) ή οποιασδήποτε υποκλάσης της.

Παράδειγμα 1

```
ArrayList<MessagePost> messages;

messages = new ArrayList<>();

for (Post post : posts) {

   if (post instanceof MessagePost) {
     MessagePost msg = (MessagePost) post;

     messages.add(msg);
   }

}
```

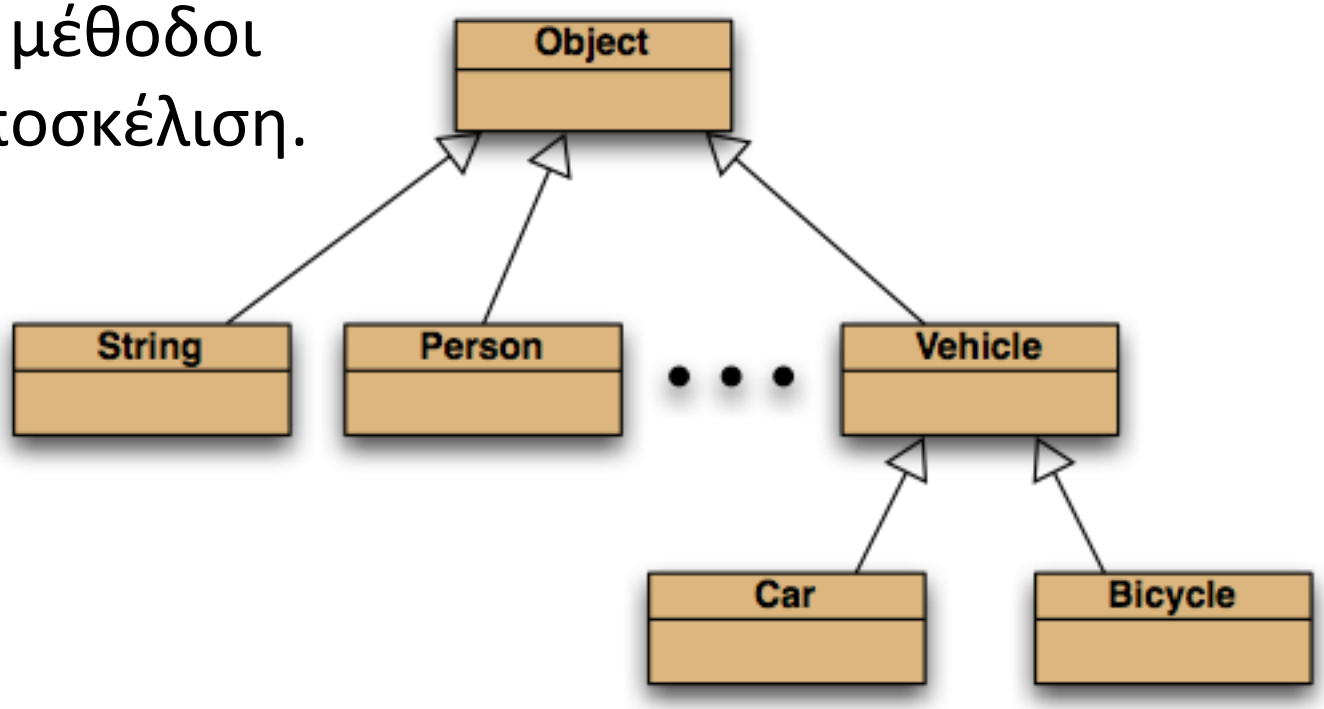
Παράδειγμα 2

```
public class Parent {}
-----------------------------------------
public class Child extends Parent {}
-----------------------------------------
public static void main(String[] args) {
   Parent o1 = new Parent();
   Parent o2 = new Child();
   System.out.println(o1 instanceof Parent);
   System.out.println(o1 instanceof Child);
   System.out.println(o2 instanceof Parent);
   System.out.println(o2 instanceof Child);
}
// true false true true
```

Parent.java, Child.java

Η Κλάση *Object*

- Όλες οι κλάσεις κληρονομούν από την *Object.*
  - Πολυμορφικές μεταβλητές Object.
  - Έτοιμες μέθοδοι προς υποσκέλιση.



Μέθοδος *toString*

- Σκοπός.
  - Αναπαράσταση αντικειμένου ως συμβολοσειρά.
- Προκαθορισμένη συμπεριφορά.
  - *ΌνομαΚλάσης@ΔιεύθυνσηΜνήμης.*
  - Υποσκέλιση για επιστροφή αναπαραστάσεων μεγαλύτερου ενδιαφέροντος.
- Παρατηρήσεις.
  - Αντικατάσταση μεθόδων εμφάνισης αντικειμένων.
  - Αυτόματη κλήση όταν αναμένεται συμβολοσειρά.

Χρήση *toString* στην *Post*

```
public String toString() {
  String text = username + "\n" + timeString();
  if (likes > 0) {
     text += "  -  " + likes + " people like this.\n";
  } else {
     text += "\n";
  }
  if (comments.isEmpty()) {
    text += "   No comments.\n";
  } else {
    text += "   " + comments.size() +
            " comment(s). Click here to view.\n";
  }

  return text;
}
```

version5

Χρήση *toString* στις Υποκλάσεις

```
public String toString() {
   String text = super.toString();
   text += message + "\n";
   return text;
}
```

MessagePost

```
public String toString() {
   String text = super.toString();
   text += "   [" + filename + "]\n";
   text += "   " + caption;
   return text;
}
```

PhotoPost

version5

Αυτόματη Κλήση στην *NewsFeed*

```
public void show() {
   for (Post post : posts) {
      // post.display();
      System.out.println(post);
   }
}
```

version5

Ισότητα Αντικειμένων

- Ο τελεστής ==
  - Ελέγχει τη θέση μνήμης των αντικειμένων.
- Υλοποίηση μεθόδου για έλεγχο ισότητας.
  - Εναπόκειται σε εμάς, συγκρίνοντας τις τιμές των ιδιοτήτων που εμείς θεωρούμε ότι χαρακτηρίζουν τη μοναδικότητα των αντικειμένων.
  - Αντί για την υλοποίηση μιας οποιασδήποτε μεθόδου, υποσκελίζουμε την *equals* της *Object.*
    - Εξ' ορισμού κάνει ίδιο έλεγχο με τον τελεστή ==
    - Ευρύτατη χρήση, π.χ. HashSet, HashMap, ArrayList.

Ιδιότητες της Object.equals (1/2)

- Συμμετρική.
  - Για x και y που δεν είναι *null*, η κλήση x.equals(y) θα πρέπει να επιστρέφει true αν και μόνο αν η y.equals(x) επιστρέφει επίσης true.
- Μεταβατική.
  - Για x, y και z που δεν είναι null, αν η κλήση x.equals(y) επιστρέφει true και η y.equals(z) επιστρέφει true, τότε και η x.equals(z) θα πρέπει να επιστρέφει true.

Ιδιότητες της Object.equals (2/2)

- Ανακλαστική.
  - Για x και y που δεν είναι null, η κλήση x.equals(x) θα πρέπει να επιστρέφει true.
- Συνέπεια.
  - Για x και y που δεν είναι null, πολλαπλές κλήσεις της x.equals(y) θα πρέπει να επιστρέφουν πάντα true ή false, εφόσον δεν αλλάζουν οι τιμές στις οποίες στηρίζεται η σύγκριση των x και y.
- Σύγκριση με null.
  - Για x που δεν είναι null, η κλήση x.equals(null) θα πρέπει να επιστρέφει false.

Παράδειγμα (1/3)

```
public class Student {

    private int aem;
    private String name;

    public Student(int aem, String name) {
        this.aem = aem;
        this.name = name;
    }
}
```

Student.java

Παράδειγμα (2/3)

```
public boolean equals(Object obj) {
   if (this == obj)
      return true;
   if (!(obj instanceof Student))
      return false;
   Student student = (Student) obj;
   if (aem == student.aem &&
       name.equals(student.name))
      return true;
   else
      return false;
}
```

Γρήγορος έλεγχος αν δείχνουν στην ίδια θέση μνήμης

Έλεγχος αν είναι της ίδιας κλάσης (καλύπτει και null)

Σύγκριση των τιμών των ιδιοτήτων που θεωρείτε ότι χαρακτηρίζουν τη μοναδικότητα των αντικειμένων

Παράδειγμα (3/3)

```
Student s1 = new Student(123, "Τσουμάκας");
Student s2 = new Student(123, "Τσουμάκας");
if (s1 == s2) {
   System.out.println("true");
} else {
   System.out.println("false");
}
if (s1.equals(s2)) {
   System.out.println("true");
} else {
   System.out.println("false");
}
```

StudentEquality.java

Η Μέθοδος hashCode της Object

- Σκοπός.
  - Επιστροφή ακεραίου, βάσει του οποίου ένα αντικείμενο μπαίνει σε πίνακα κατακερματισμού.
  - Εξ' ορισμού επιστρέφει έναν αριθμό με βάση τη διεύθυνση μνήμης του αντικειμένου.
  - Όποτε υποσκελίζεται η *equals*, θα πρέπει να υποσκελίζεται και η *hashCode.*
- Αν δύο αντικείμενα
  - Είναι ίσα, θα πρέπει να έχουν ίδιο κωδικό.
  - Δεν είναι ίσα, θα πρέπει κατά προτίμηση (δεν είναι υποχρεωτικό) να έχουν διαφορετικό κωδικό.

Παράδειγμα (1/2)

```
public int hashCode() {
    int hash = 7;
    hash = 47 * hash + aem;
    hash = 47 * hash + name.hashCode();
    return hash;
}
```

Παράδειγμα (2/2)

```
Student s1 = new Student(123, "Τσουμάκας");
Student s2 = new Student(123, "Τσουμάκας");

HashSet<Student> students = new HashSet<>();
students.add(s1);
students.add(s2);

for (Student s : students) {
  System.out.println(s);
}
```

SetOfStudents.java

ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Τέλος Ενότητας

Επεξεργασία: Εμμανουήλ Ρήγας

Θεσσαλονίκη, Εαρινό Εξάμηνο 2013-2014

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης