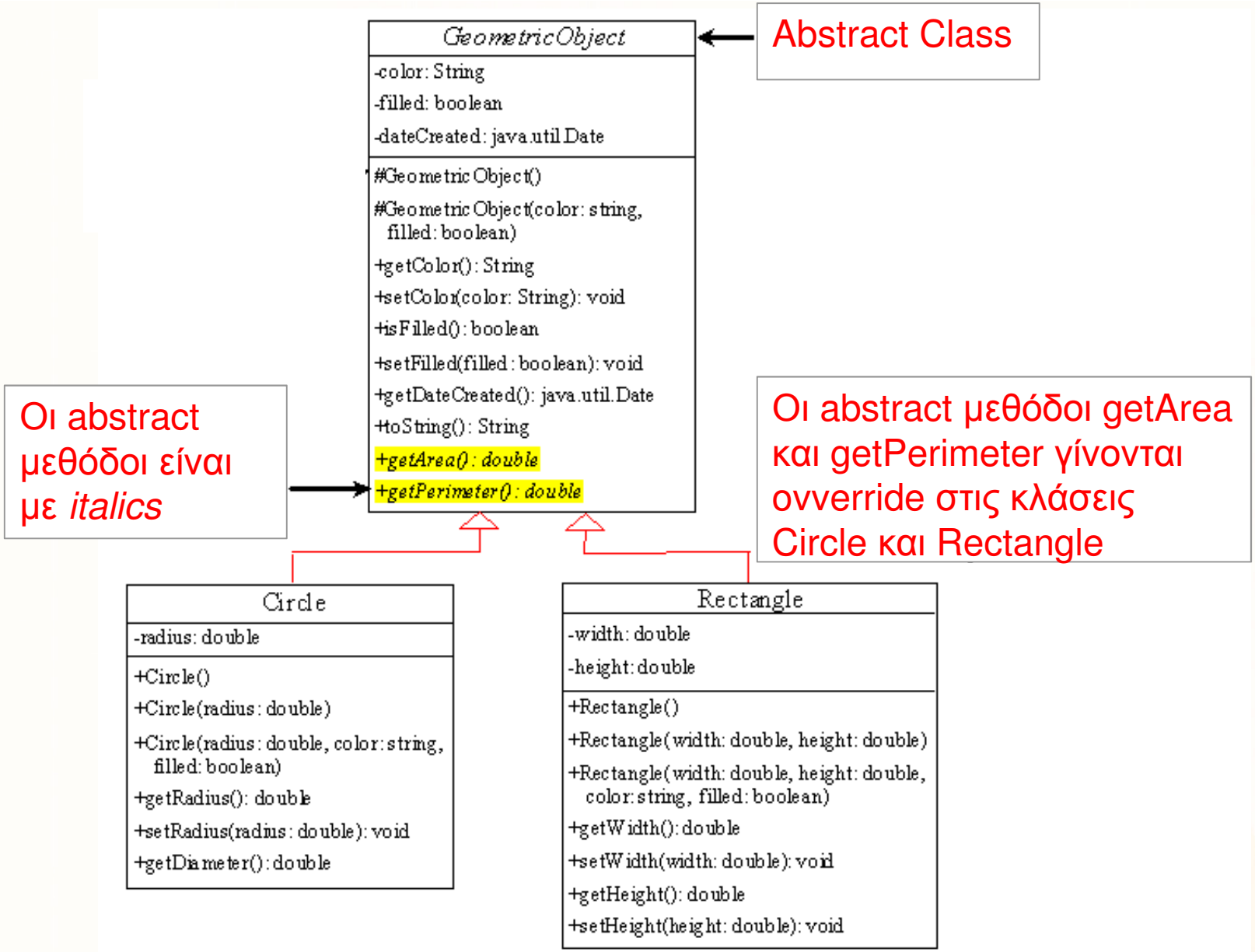
Διάλεξη 15: Αφαιρετικές Κλάσεις (Abstract Classes) & Διαπροσωπείες (Interfaces)

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

- Αφαιρετικές Κλάσεις, Αφαιρετικές Μεθόδοι
- Διαπροσωπείες: Ορισμός, Δημιουργία, Υλοποίηση
- Χρήσιμες Διαπροσωπείες
- Σύγκριση Αφαιρετικών Κλάσεων με Διαπροσωπείες

Διδάσκων: Παναγιώτης Ανδρέου

Αφαιρετικές Κλάσεις και Μέθοδοι



Ο τροποποιητής abstract

- **Μία αφαιρετική κλάση - abstract class**
  - Δεν μπορεί να αρχικοποιηθεί αντικείμενο της κλάσης
  - Πρέπει να επεκταθεί (κληρονομικότητα) και να υλοποιηθεί από τις υποκλάσεις

- **Μία αφαιρετική μέθοδος - abstract method**
  - Η υπογραφή μίας μεθόδου χωρίς υλοποίηση

Αφαιρετικές Μεθόδοι

- Η Java μας επιτρέπει να δηλώνουμε **ρητά** (explicit) ορισμένες μεθόδους μιας κλάσης σαν αφαιρετικές ώστε:
  - Να έχουμε τη δυνατότητα να προσδιορίζουμε την διεπαφή της κλάσης που τις περιέχει.
  - Να αποτρέπεται η εκ παραδρομής κλήση τους από άλλες μεθόδους.

- Ο προσδιορισμός μιας μεθόδου ως αφαιρετικής γίνεται ως εξής:

abstract void X();

- **Τα abstract methods μπορούν να υπάρχουν μόνο σε abstract classes**

Αφαιρετικές κλάσεις

- Μια κλάση η οποία περιλαμβάνει έστω και μια «αφαιρετική» μέθοδο, καθίσταται επίσης αφαιρετική **και πρέπει να δηλωθεί ως αφαιρετική**.
- Όταν μία κλάση κληρονομεί από μια αφαιρετική κλάση θα πρέπει να υλοποιήσουμε **όλες τις αφαιρετικές μεθόδους της υπερκλάσης**. Διαφορετικά ο μεταφραστής επιβάλλει να προσδιορίσουμε την κλάση μας σαν αφαιρετική.
- Μπορούμε τέλος να δηλώσουμε μια κλάση σαν αφαιρετική, χωρίς ωστόσο η κλάση αυτή να περικλείει αφαιρετικές μεθόδους. Γιατί να θέλουμε κάτι τέτοιο; **Για να αποκλείσουμε τη δημιουργία αντικειμένων αυτής της κλάσης**.

Παρατηρήσεις

- Μία κλάση μπορεί να είναι αφαιρετική χωρίς να περιλαμβάνει αφαιρετικές μεθόδους
- Μία κλάση που κληρονομεί από μία αφαιρετική κλάση πρέπει να υλοποιήσει όλες τις αφαιρετικές μεθόδους της υπερκλάσης ανεξάρτητα αν θα τις χρησιμοποιήσει ή όχι
- Σε μία αφαιρετική κλάση δηλώνουμε κατασκευαστές αλλά δεν μπορούμε να δημιουργήσουμε αντικείμενο

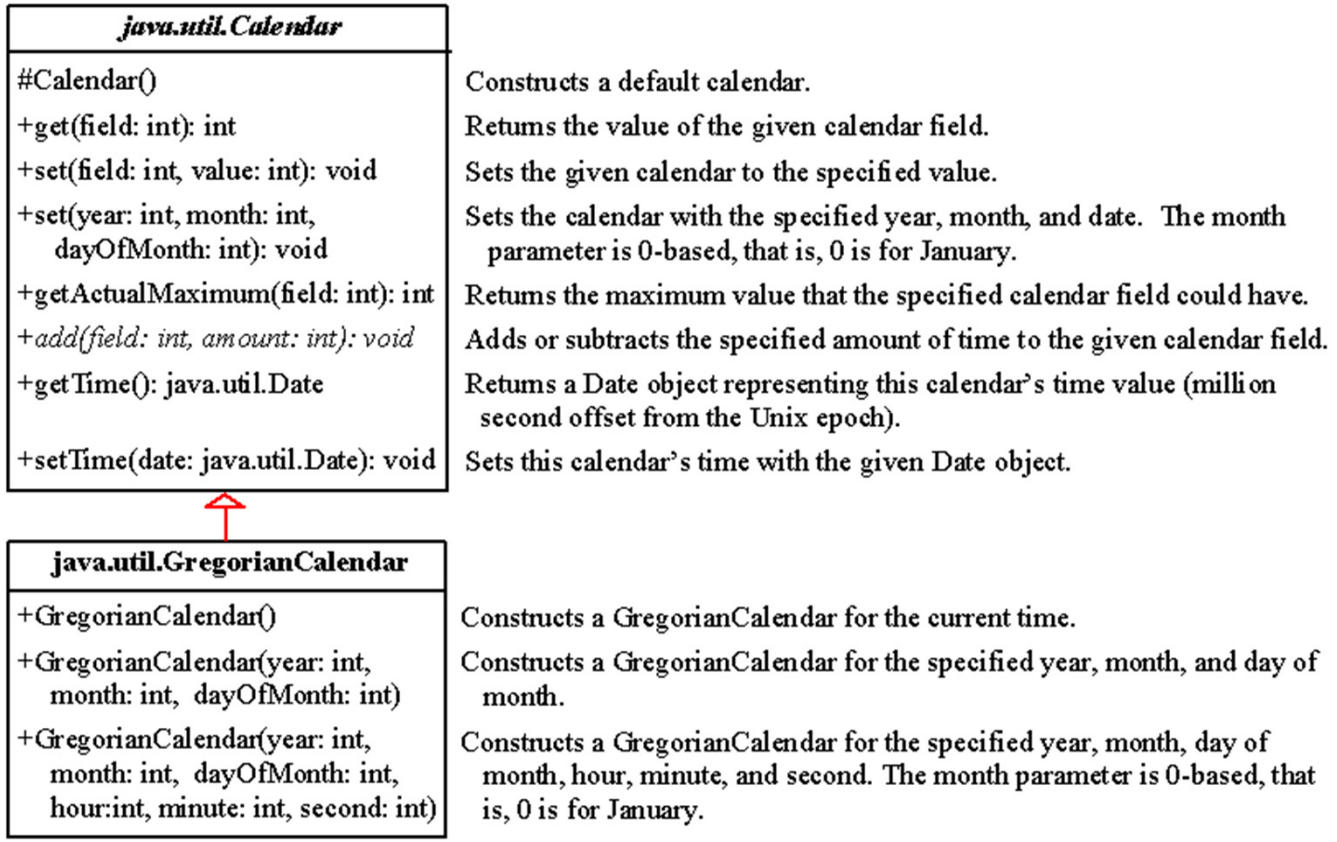
Παρατηρήσεις (συν.)

- Μία κλάση μπορεί να είναι abstract έστω και αν κληρονομεί από μία συμπαγή κλάση
- Μία αφαιρετική κλάση μπορεί να υπερκαλύψει μία μέθοδο μίας συμπαγής υπερκλάσης και να τη δηλώσει σαν abstract
- Μία αφαιρετική κλάση μπορεί να χρησιμοποιηθεί σαν τύπος, π.χ., GeometricObject array = new GeometricObject[10]; (μέσω από πολυμορφισμό)

Παράδειγμα: Αφαιρετική Κλάση και Μέθοδος

Η αφαιρετική κλάση java.util.Calendar χρησιμοποιείτε για να επιστρέφει χρήσιμες πληροφορίες για από αντικείμενα τύπου date.

Οι υποκλάσεις τις Calendar μπορούν να υλοποιήσουν συγκεκριμένα ημερολόγια (π.χ., GregorianCalendar)



Διαπροσωπείες (Interfaces)

- **Μία διαπροσωπεία (interface)** είναι μία δομή (παρόμοια με την κλάση) ή οποία **περιέχει:**
  - **Δηλώσεις μεθόδων** και **όχι καθορισμό τους (εμμέσως είναι abstract)**
  - **Σταθερές (constant) μεταβλητές**

- Καθορίζει την μορφή που πρέπει να υπάρχει σε μία κλάση, ένα συμβόλαιο το οποίο δηλώνει πως δουλεύει μία κλάση

- Δηλώνεται με την λέξη **`interface`**

- Οι Διαπροσωπείες είναι **παρόμοιες με τις Αφαιρετικές κλάσεις, αλλά όχι οι ίδιες**

Δήλωση Διαπροσωπείας (Interface Declaration)

- Σύνταξη δήλωσης διαπροσωπείας:

```
public interface InterfaceName {
  //constant declarations
  //method signatures
}
```

- Παράδειγμα

```
public interface Edible{
  // Describe how to eat
  public String howToEat();
}
```

Σταθερές Μεταβλητές (Constant Variables)

- Παράδειγμα δήλωσης Σταθερών Μεταβλητών

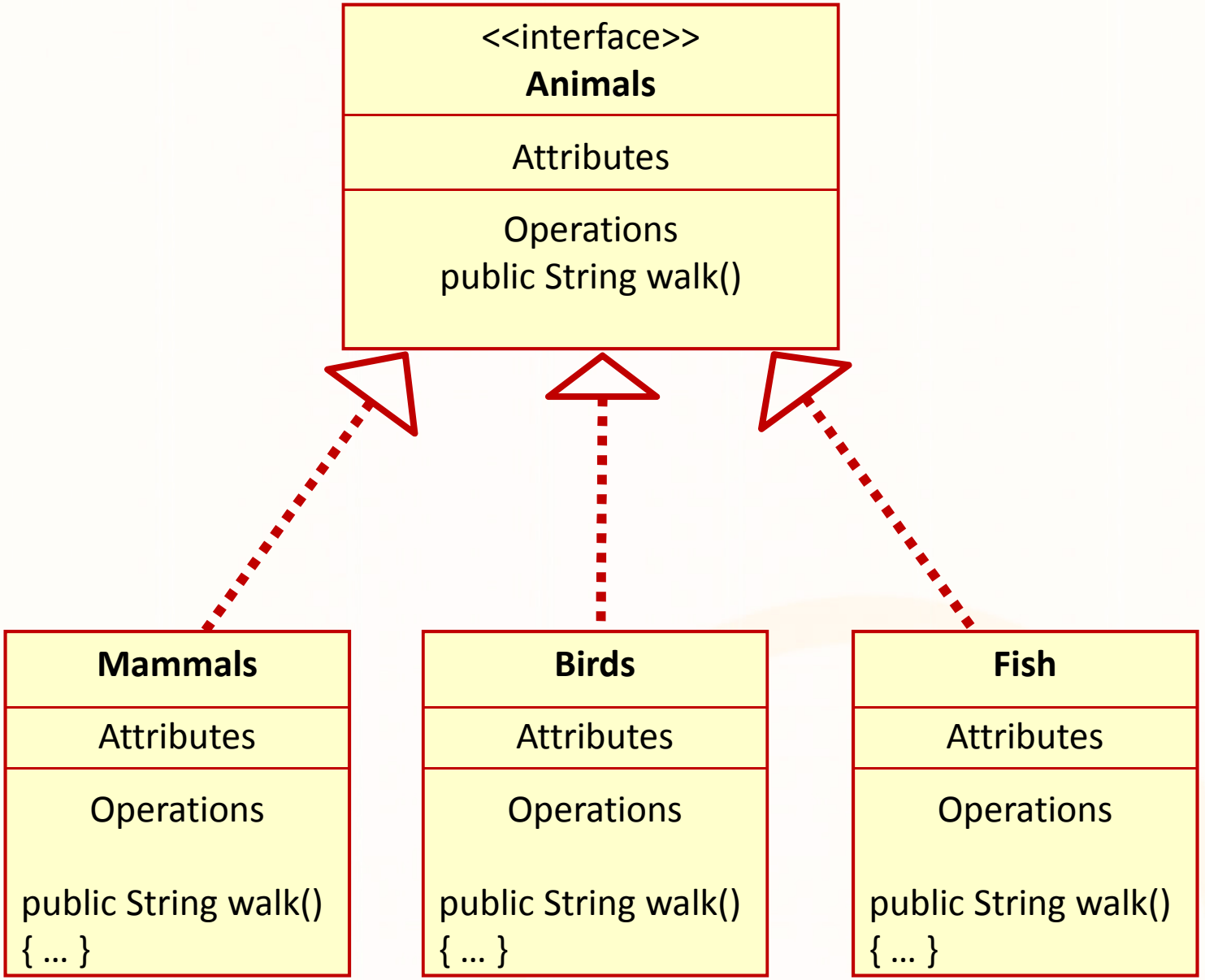
```
public interface InterfaceName {
  //constant declarations
  int x=1;
  double PI=3.1415;
}
```

- Το πιο πάνω είναι ισοδύναμο με το πιο κάτω

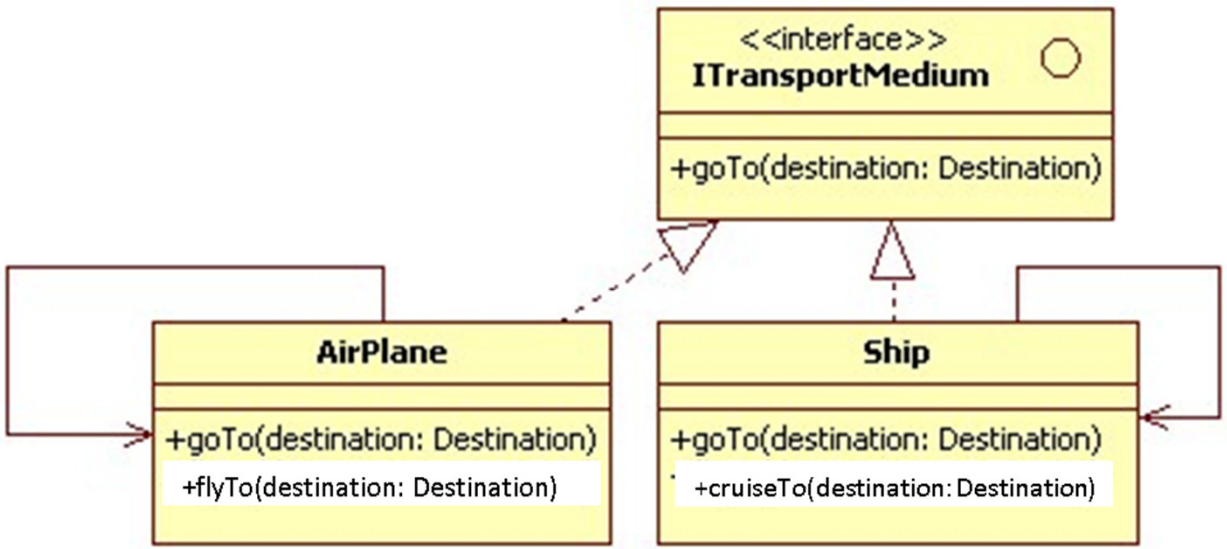
```
public interface InterfaceName {
  //constant declarations
  public static final int x=1;
  public static final double PI=3.1415;
}
```

- Πρόσβαση με <διαπροσωπεία>.<μεταβλητή>
  π.χ., **InterfaceName.x**

Παράδειγμα Διαπροσωπείας 1



Παράδειγμα Διαπροσωπείας 2



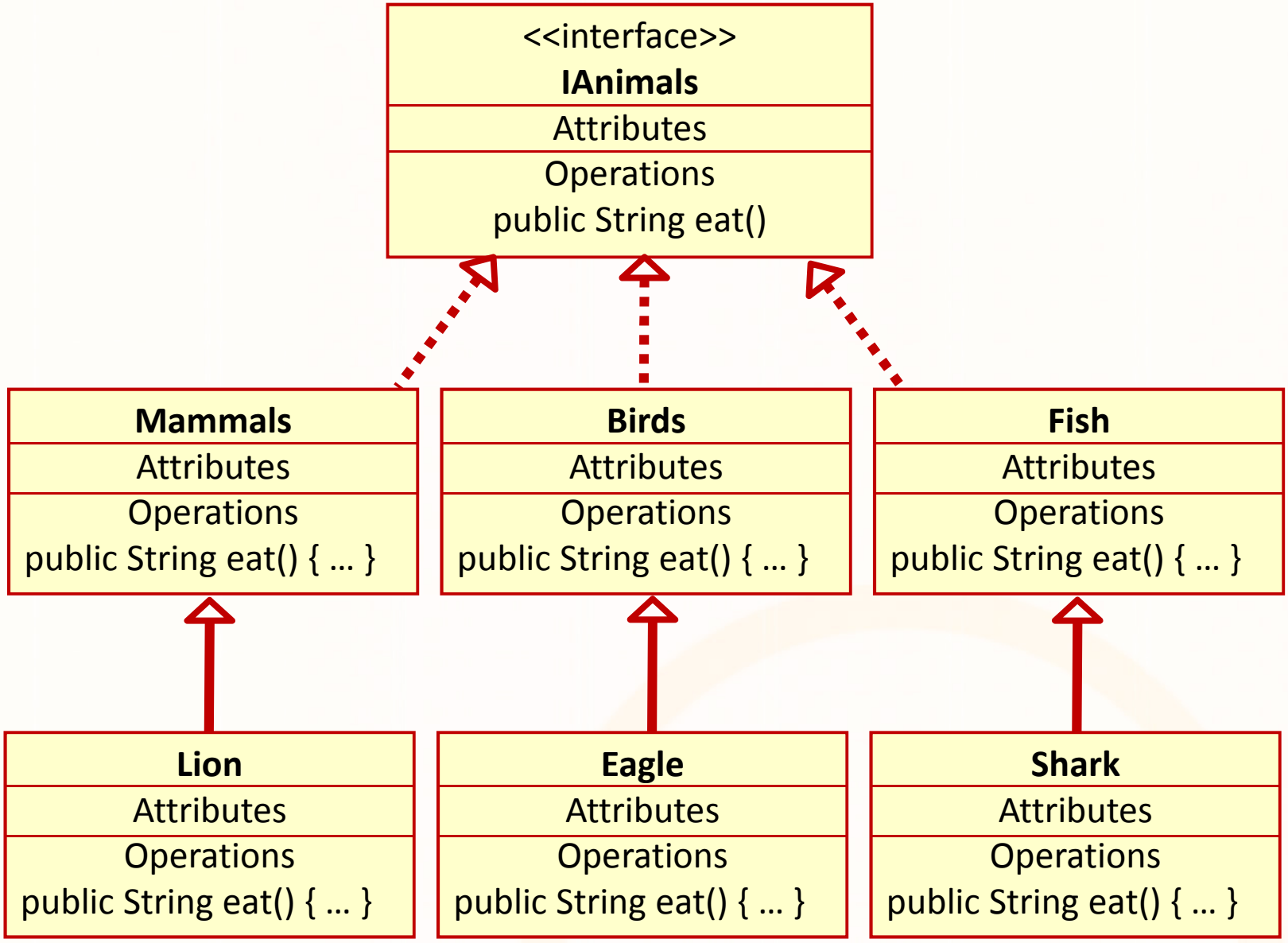

```
class Airplane implements ITransportMedium{
   public goTo(Destination destination){
      flyTo(destination);
   }

 public flyTo(Destination destination){
       /*code to implement the special case*/
    }
}
```

```
class Ship implements ITransportMedium{
   public goTo(Destination destination){
      cruiseTo(destination);
   }

 public cruiseTo (Destination destination){
       /*code to implement the special case*/
    }
}
```

Παράδειγμα Διαπροσωπείας και Κληρονομικότητας



Παράδειγμα Διαπροσωπείας και Κληρονομικότητας

```
public interface IAnimals {
abstract void eat();
String getName();
}

class Mammals implements IAnimals {
  public void eat() { }
  public String getName() {}
}

class Birds implements IAnimals {
  public void eat() { }
  public String getName() {}
}

class Fish implements IAnimals {
  public void eat() { }
  public String getName() {}
}
```

```
class Lion extends Mammals {
  public void eat() { }
  public String getName() {}
}

class Eagle extends Birds {
  public void eat() { }
  public String getName() {}
}

class Shark extends Fish {
  public void eat() { }
 public String getName() {}
}
```

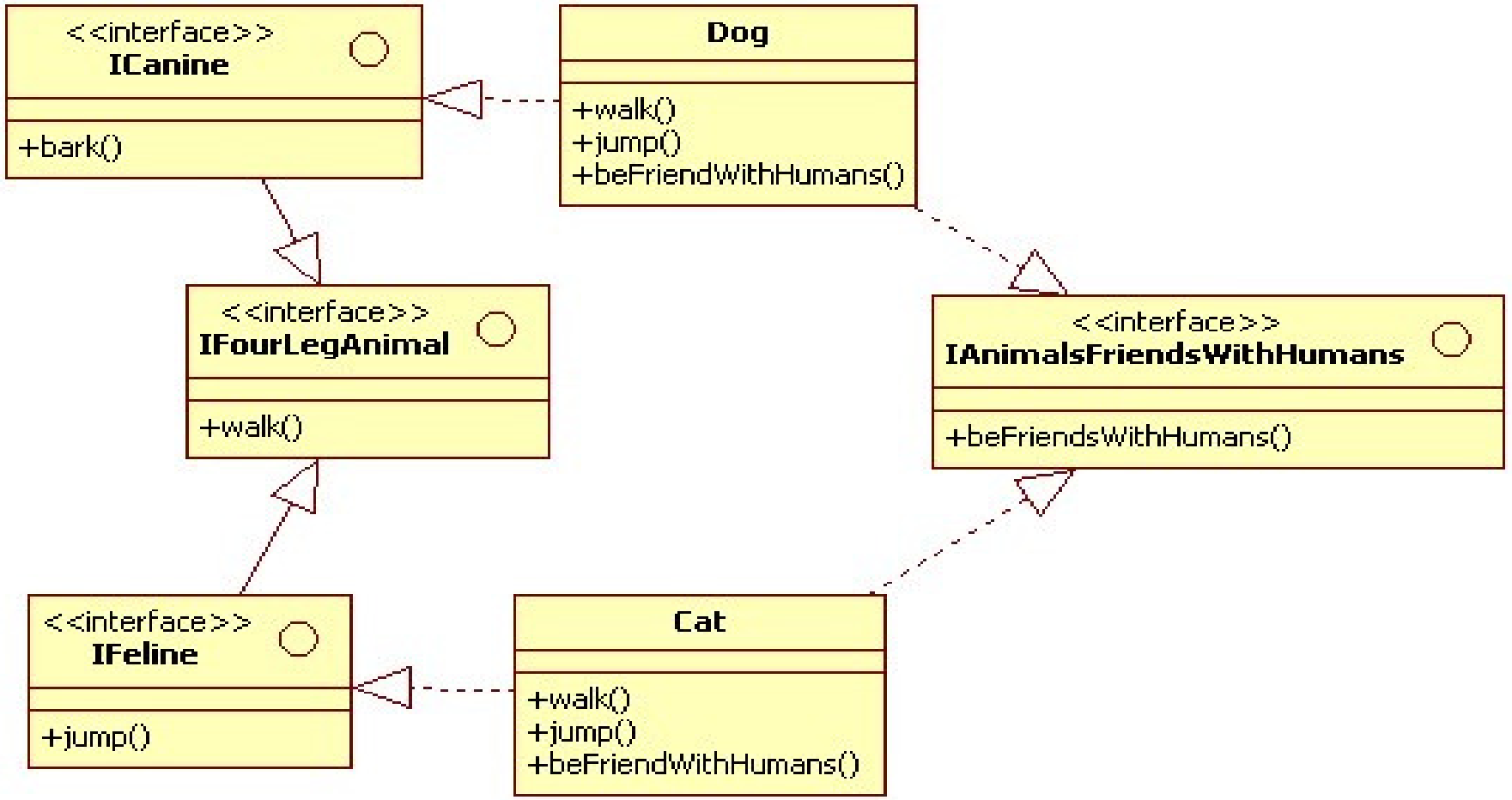
Πολλαπλή Κληρονομικότητα με Διαπροσωπείες

- Στην JAVA κάθε κλάση κληρονομεί από ακριβώς μία κλάση
- Μια **Διαπροσωπεία** (interface), **δεν έχει καθόλου υλοποίηση** και **δεν δεσμεύει μνήμη**.

- **Μία κλάση** μπορεί να υλοποιήσει **πολλές διαπροσωπείες**
- **Σύνταξη:** class ClassA **implements** interface1**,** interface2**,** …
- Είναι μία **μορφή πολλαπλής κληρονομικότητας** (multiple inheritance).

- **Μία διαπροσωπεία μπορεί να επεκταθεί με την χρήση κληρονομικότητας**

Παράδειγμα


<<interface>>
ICanine
+bark()
Dog
+walk()
+jump()
+beFriendWithHumans()
<<interface>>
IFourLegAnimal
+walk()
<<interface>>
IAnimalsFriendsWithHumans
+beFriendsWithHumans()
<<interface>>
IFeline
+jump()
Cat
+walk()
+jump()
+beFriendWithHumans()

Αρχικοποίηση Διαπροσωπειών;;;

- Δείκτες σε αντικείμενα μπορούν να έχουν τύπο διαπροσωπείας (μέσω από πολυμορφισμό)

- Για παράδειγμα:
  ```
  IAnimals a = new Lion();
  ```

- Το Animals είναι διαπροσωπεία
- Το Lion είναι κλάση.

- Τι έχουμε πετύχει με την πιο πάνω δήλωση;

- **Μόνο μεθόδοι που ανήκουν στην διαπροσωπεία Animals μπορούν να κληθούν!**

Σημαντικές διαπροσωπείες: Comparable

- Η διαπροσωπεία Comparable: χρησιμοποιείται για καθορισμό του πως γίνεται σύγκριση αντικειμένων (π.χ., String, Date)

  java.lang Interface Comparable

  ```
  int compareTo(T o)
  ```

- Καθορίζει μία φυσική σειρά για τα αντικείμενα όπως ισχύει για αριθμητικά δεδομένα, π.χ., 1<2, “ab”<”ac”
- Περιλαμβάνει την μέθοδο `compareTo`: Συγκρίνει το αντικείμενο this με το αντικείμενο που δίνεται σαν παράμετρος
- Ακολούθως συγκρίνουμε αντικείμενα σαν να είναι αριθμοί, π.χ.,

Παράδειγμα Χρήσης Comparable

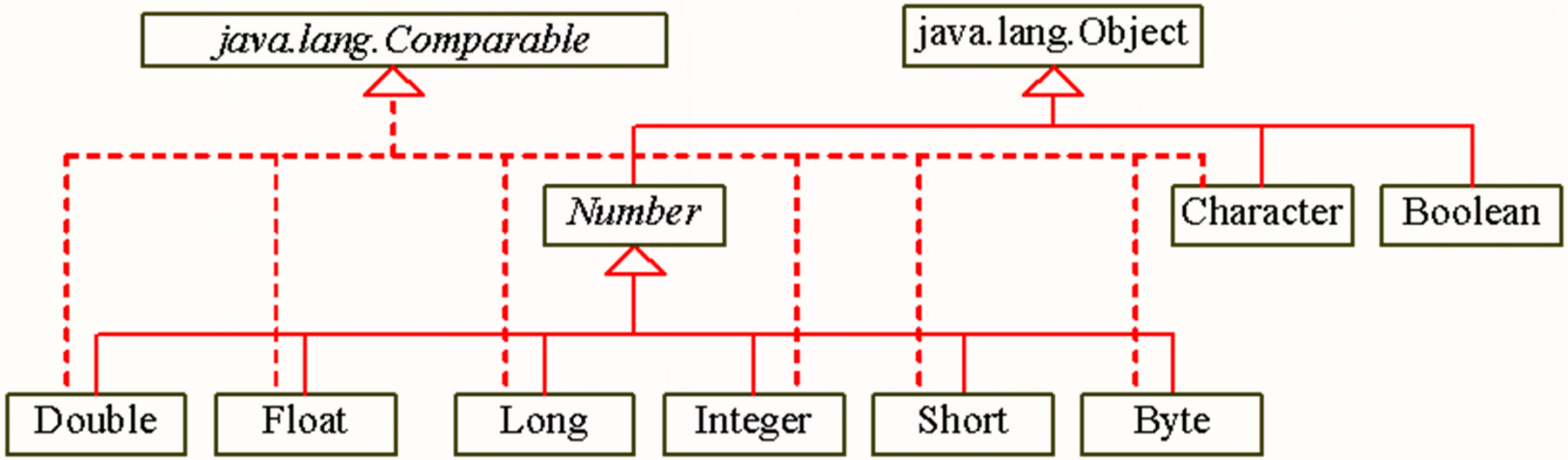
```
class ComparableCircle
      extends Circle
        implements Comparable {

  public ComparableCircle(double radius) {
    super(radius);
  }

  /** Implement the compareTo method defined in Comparable */
  public int compareTo(Object o) {
    if (getRadius() > ((ComparableCircle) o).getRadius())
      return 1;
    else if (getRadius() < ((ComparableCircle) o).getRadius())
      return -1;
    else
      return 0;
  }
}
```

Comparable και Wrapper Classes

- Όλες οι wrapper classes υλοποιούν την διαπροσωπεία comparable



- Οι wrapper classes δεν έχουν no-arg constructors
- Είναι immutable – δεν αλλάζουν τιμές μετά την αρχικοποίηση
- Υπερκαλύπτουν (override) τις μεθόδους toString(), equals
- Υλοποιούν την compareTo της διαπροσωπείας Comparable

Η abstract class Number

public abstract class Number

- Κάθε numeric wrapper class κληρονομεί από την abtrsact class Number
- Η Number περιέχει μεθόδους για doubleValue, floatValue, intValue, longValue, shortValue, και byteValue.
- Αυτές οι μεθόδοι μετατρέπουν τα αντικείμενα στους αρχέγονους τους τύπους, δηλ. τιμές
- Οι μεθόδοι byteValue και shortValue δεν είναι abtract και επιστρέφουν (byte)intValue() και (short)intValue()

Σημαντικές διαπροσωπείες: List

- Η διαπροσωπεία List<E>: αντιπροσωπεύει τον ΑΤΔ Λίστα συνοδευόμενο από τις βασικές του πράξεις

  java.util Interface List<E>

- Κλάσεις που υλοποιούν την διαπροσωπεία List: AbstractList, AbstractSequentialList, ArrayList, AttributeList, CopyOnWriteArrayList, LinkedList, RoleList, RoleUnresolvedList, Stack, Vector

- Κάποιες μεθόδοι της διαπροσωπείας: add, remove, get, set, size, …

Σημαντικές διαπροσωπείες: Cloneable

java.util Interface Clonable <άδεια διαπροσωπεία>

- Μία κλάση που υλοποιηεί τη διαπροσωπεία Cloneable δείχνει στην μέθοδο Object.clone() ότι είναι επιτρεπτό να προχωρήσει με αντιγραφή field-for-field για τα στιγμιότυπα της κλάσης.
  - Οι κλάσεις που υλοποιούν την διαπροσωπεία αυτή θα πρέπει να υπερκαλύψουν (override) την protected Object.clone() με public τροποποιητή
  - Όταν καλεστεί η Object.clone() χωρίς να είναι υλοποιημένη η διαπροσωπεία Cloneable θα γίνει throw CloneNotSupportedException
  - ΠΡΟΣΟΧΗ: η διαπροσωπεία δεν περιέχει την clone(). Ακόμα κι αν την καλέσουμε δεν υπάρχει καμία εξασφάλιση ότι θα πετύχει.

Σημαντικές διαπροσωπείες: Runnable

- java.lang. Interface Runnable

- Η διαπροσωπεία Runnable πρέπει να υλοποιείται από κάθε κλάση που προορίζεται να εκτελεστεί σαν ένα νήμα (thread)

- Η κλάση θα πρέπει να ορίσει μία μέθοδο public void **run**()

Αρχικοποιήσεις Μελών Διαπροσωπείας

- Μπορούμε να αρχικοποιήσουμε τα μέλη μίας διαπροσωπείας κατά τη διάρκεια φόρτωσης της διαπροσωπείας στη μνήμη
- Παράδειγμα:

```
interface itest{

  Random rand = new Random();

  int randomInt = rand.nextInt(10);

  long randomLong = rand.nextLong();

  double randomDouble = rand.nextDouble();
}
```

Παρατηρήσεις

- Μία κλάση πρέπει να υλοποιεί όλες τις μεθόδους μίας διαπροσωπείας εκτός και αν είναι δηλωμένη σαν abstract
- Μία διαπροσωπεία μπορεί να χρησιμοποιηθεί σαν τύπος που μπορεί να είναι δείκτης σε οποιαδήποτε κλάση υλοποιεί την διαπροσωπεία
- Υπάρχουν κενές διαπροσωπείες
- Μπορούν να υπάρχουν συγκρούσεις ονομάτων μεταξύ διαφορετικών διαπροσωπειών, π.χ., δύο μεταβλητές με το ίδιο όνομα

Σύγκριση Abstract Classes με Interfaces

Χαρακτηριστικά	interface	abstract class
πολλαπλή κληρονομικότητα	Υλοποίηση πολλαπλών διεπαφών από μία κλάση	Κληρονόμος μόνο μίας κλάσης
υλοποίηση	Καμία	Κανένας περιορισμός
καταστάσεις	**Static final** σταθερές μόνο. Δε μπορεί να ορίσει καταστάσεις	Κανένας περιορισμός
third party convenience	Η υλοποίηση της διεπαφής μπορεί να προστεθεί σε οποιανδήποτε βιβλιοθήκη εξωτερικής προέλευσης	Μία κλάση εξωτερικής προέλευσης πρέπει να οριστεί από την αρχή έτσι ώστε να κληρονομεί μόνο από την αφαιρετική κλάση
«είναι» vs «συμπεριφέρεται ως»	Η κλάση **συμπεριφέρεται σαν την διεπαφή της οποίας υλοποιεί τις δηλώσεις της**	Οι πελάτες των αφαιρετικών κλάσεων **«είναι»** τύπου κληροδότη

Σύγκριση Abstract Classes με Interfaces

Χαρακτηριστικά	Interface	abstract class
ομοιογένεια	Αν οι κλάσεις πελάτες είναι ετερογενείς τότε η χρήση του **interface** είναι η βέλτιστη.	Αν οι κλάσεις πελάτες είναι ομοιογενείς τότε η χρήση του **abstract** είναι η βέλτιστη. Ομοιογενείς φυσικά ως προς τις **καταστάσεις** και **συμπεριφορές**.
διαχείριση	Εύκολη αλλαγή της υλοποίησης χωρίς μεταβολή της διεπαφής. Ο πελάτης (λογισμικό ή χρήστης) δε γνωρίζει την αλλαγή.	Εύκολη αλλαγή της υλοποίησης χωρίς μεταβολή της διεπαφής. Ο πελάτης (λογισμικό ή χρήστης) δε γνωρίζει την αλλαγή.
ταχύτητα εκτέλεσης	Αργή, χρειάζεται επιπλέον ανακατεύθυνση για την ανίχνευση της μεθόδου της κλάσης. Οι σύγχρονες JVM προσπαθούν να μειώσουν τον χρόνο ανίχνευσης.	Υψηλή ταχύτητα

Σύγκριση Abstract Classes με Interfaces

Χαρακτηριστικά	Interface	abstract class
σαφήνια	Οι διεπαφές δε διαθέτουν καταστάσεις. Όλες οι μεταβλητές θεωρούνται **static final** δηλαδή σταθερές.	Μοιράζονται «κοινό» κώδικα ο οποίος μπορεί να χρησιμοποιηθεί στην αρχικοποίηση κοινών καταστάσεων.
πρόσθεση λειτουργικότητας	Κάθε πρόσθεση μεθόδου σε μία διεπαφή έχει σημαντικό αντίκτυπο καθώς πρέπει να δοθεί σε όλες τις υλοποιήσεις της διεπαφής κατάλληλη υλοποίηση της καινούριας μεθόδου.	Αν προστεθεί νέα μέθοδο τότε υπάρχει η επιλογή να δοθεί και μία τυπική υλοποίηση της έτσι ώστε να όλοι οι κληρονόμοι να συνεχίσουν να λειτουργούν χωρίς καμιά αλλαγή.
ενσωμάτωση	Εύκολο στην ενσωμάτωση και αλλαγή καθώς δεν περιέχει κώδικα ή μεταβλητές πέραν κάποιων σταθερών.	Πρέπει να διατηρηθεί η αφαιρετική κλάση ως έχει. Η κλάση και τα περιεχόμενα της είναι επιβεβλημένα στον κληρονόμο.