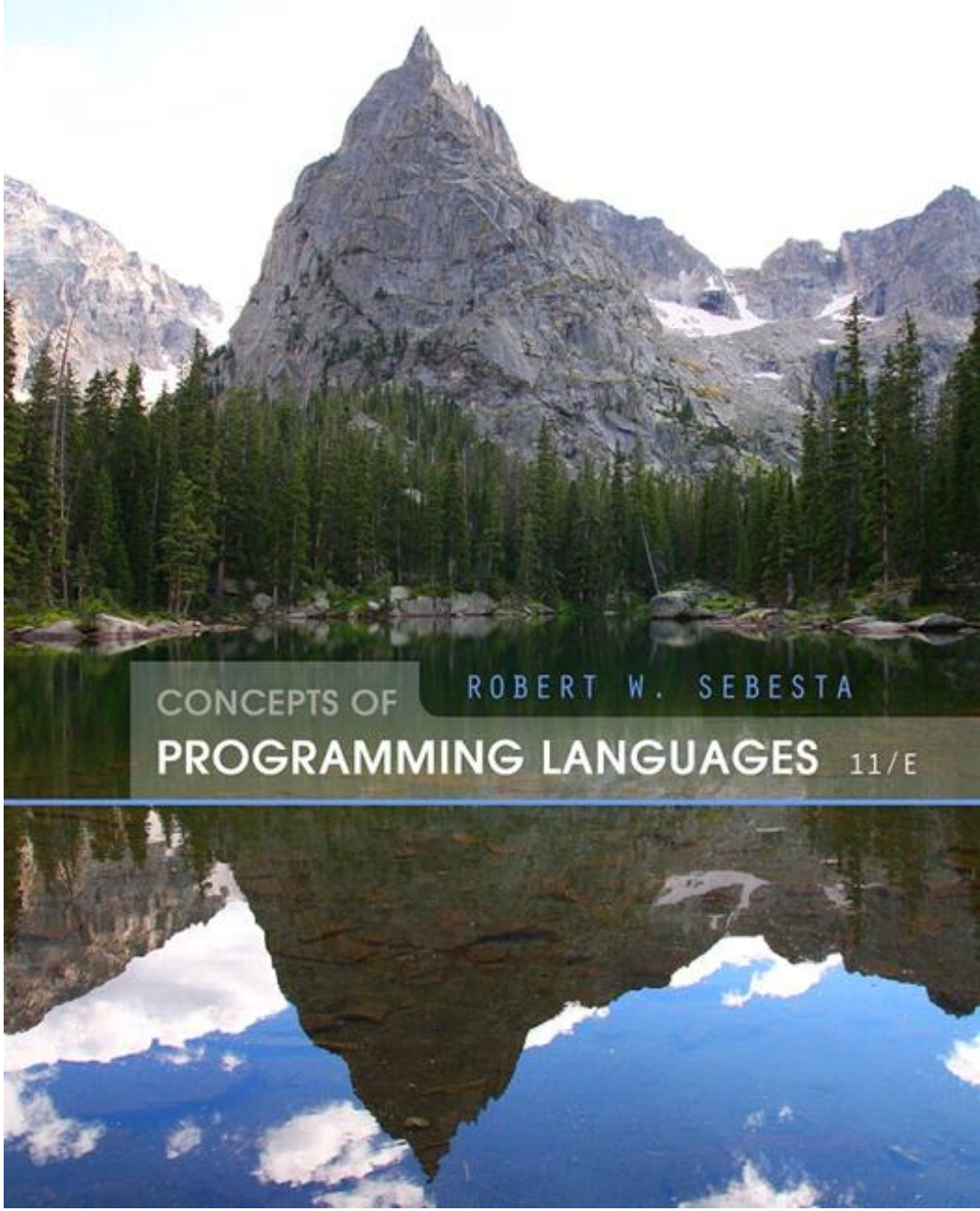
Έκδοση:
10/4/2024

Κεφάλαιο 12

Υποστήριξη για Αντικειμενοστρεφή Προγραμματισμό



ISBN 0-321-49362-1

Περιεχόμενα κεφαλαίου 12

- Εισαγωγή
- Αντικειμενοστραφής Προγραμματισμός (Object Oriented Programming)
- Θέματα Σχεδιασμού για Αντικειμενοστραφείς Γλώσσες
- Υποστήριξη Αντικειμενοστραφούς Προγραμματισμού στη Smalltalk
- Υποστήριξη Αντικειμενοστραφούς Προγραμματισμού στη C++
- Υποστήριξη Αντικειμενοστραφούς Προγραμματισμού στη Objective-C
- Υποστήριξη Αντικειμενοστραφούς Προγραμματισμού στη Java
- Υποστήριξη Αντικειμενοστραφούς Προγραμματισμού στη C#
- Υποστήριξη Αντικειμενοστραφούς Προγραμματισμού στη Ruby
- Υλοποίηση Αντικειμενοστραφών Κατασκευών
- Ανάκλαση (reflection)

Εισαγωγή

- Υπάρχουν πολλές αντικειμενοστραφής γλώσσες προγραμματισμού (OOP=Object Oriented Programming)
  - Ορισμένες γλώσσες υποστηρίζουν διαδικασιακό προγραμματισμό και προγραμματισμό προσανατολισμένο στα δεδομένα (data-oriented), (π.χ., C++)
  - Ορισμένες υποστηρίζουν τον συναρτησιακό προγραμματισμό (π.χ., CLOS)
  - Ορισμένες γλώσσες προγραμματισμού δεν υποστηρίζουν άλλα προγραμματιστικά παραδείγματα πέρα του αντικειμενοστραφούς (π.χ., Java και C#)
  - Ορισμένες γλώσσες είναι καθαρές OOP (π.χ., Smalltalk & Ruby)
  - Ορισμένες συναρτησιακές γλώσσες υποστηρίζουν τον OOP

Αντικειμενοστραφής Προγραμματισμός

- Τα 3 κύρια χαρακτηριστικά των αντικειμενοστραφών γλωσσών:
  - Αφηρημένοι Τύποι Δεδομένων (ADT = Abstract Data Types)
  - Κληρονομικότητα
    - Η κληρονομικότητα είναι ένα βασικό θέμα για τον OOP και τις γλώσσες που τον υποστηρίζουν
  - Πολυμορφισμός

Κληρονομικότητα

- Η αύξηση της παραγωγικότητας επιτυγχάνεται μέσω της επαναχρησιμοποίησης
  - Τα ADTs επαναχρησιμοποιούνται δύσκολα (καθώς συνεχώς χρειάζονται αλλαγές)
  - Όλα τα ADTs είναι ανεξάρτητα και βρίσκονται στο ίδιο επίπεδο
- Η κληρονομικότητα επιτρέπει να αντιμετωπιστούν και τα δύο παραπάνω θέματα – επαναχρησιμοποίηση ADTs μετά από μικρές αλλαγές και ιεραρχίες κλάσεων
- Η κληρονομικότητα επιτρέπει σε νέες κλάσεις να οριστούν με όρους υπαρχόντων κλάσεων, π.χ., επιτρέποντας να κληρονομηθούν κοινά τμήματα

Έννοιες Αντικειμενοστραφούς Προγραμματισμού

- Οι ADTs συνήθως αναφέρονται ως **κλάσεις**
- Τα στιγμιότυπα των κλάσεων ονομάζονται **αντικείμενα**
- Μια κλάση που κληρονομεί είναι μια **παραγόμενη κλάση** (*derived class*) ή μια **υποκλάση** (*subclass*)
- Η κλάση από την οποία κληρονομεί μια άλλη κλάση ονομάζεται **γονική κλάση** ή **υπερκλάση** (*superclass)*
- Τα υποπρογράμματα που ορίζουν λειτουργίες σε αντικείμενα ονομάζονται **μέθοδοι** *(methods)*

Έννοιες Αντικειμενοστραφούς Προγραμματισμού (συνέχεια)

- Οι κλήσεις μεθόδων ονομάζονται μηνύματα
- Η πλήρης συλλογή των μεθόδων ενός αντικειμένου αποτελεί το πρωτόκολλο μηνυμάτων του ή αλλιώς τη διεπαφή (interface) μηνυμάτων
- Τα μηνύματα έχουν δύο μέρη – το όνομα του μηνύματος και το αντικείμενο που αποδέχεται το μήνυμα
- Στην απλούστερη περίπτωση, μια κλάση κληρονομεί όλα τα μέλη του γονέα της

Έννοιες Αντικειμενοστραφούς Προγραμματισμού (συνέχεια)

- Η κληρονομικότητα μπορεί να έχει πολύπλοκους κανόνες που επιβάλλονται με access controls:
  - Μια κλάση μπορεί να κρύβει μέλη της από τις υποκλάσεις της
  - Μια κλάση μπορεί να κρύβει μέλη της από τους πελάτες της
  - Μια κλάση μπορεί να κρύβει μέλη της από τους πελάτες της ενώ να επιτρέπει σε υποκλάσεις της την πρόσβαση
- Πέρα από το να κληρονομούνται οι μέθοδοι ως έχουν, μια κλάση μπορεί να τροποποιεί τη μέθοδο που κληρονομεί.
  - Η νέα μέθοδος παρακάμπτει (*overrides)* την κληρονομούμενη μέθοδο
  - Η μέθοδος του γονέα παρακάμπτεται *(overriden)*

Έννοιες Αντικειμενοστραφούς Προγραμματισμού (συνέχεια)

- Μια κλάση μπορεί να διαφέρει με 3 τρόπους από τον γονέα της:
  1. Η υποκλάση μπορεί να προσθέτει μεταβλητές ή/και μεθόδους σε αυτές που κληρονομούνται από τον γονέα
  2. Η υποκλάση μπορεί να τροποποιεί τη συμπεριφορά μιας ή περισσότερων από τις μεθόδους που κληρονομεί
  3. Η γονική κλάση μπορεί να ορίζει ορισμένες από τις μεταβλητές ή τις μεθόδους της να έχουν ιδιωτική πρόσβαση (private access), που σημαίνει ότι δεν θα είναι ορατές στην υποκλάση

Έννοιες Αντικειμενοστραφούς Προγραμματισμού (συνέχεια)

- Υπάρχουν δύο είδη μεταβλητών σε μια κλάση:
  - *Μεταβλητές κλάσης (class variables)* – μία ανά κλάση
  - *Μεταβλητές στιγμιοτύπου (instance variables)* – μία ανά αντικείμενο
- Υπάρχουν δύο είδη μεθόδων σε μια κλάση:
  - *Μέθοδοι κλάσης (class methods)* – δέχονται μηνύματα με αποδέκτη την κλάση
  - *Μέθοδοι στιγμιοτύπου (instance methods)* – δέχονται μηνύματα με αποδέκτες τα αντικείμενα της κλάσης
- Απλή vs. Πολλαπλή Κληρονομικότητα
- Ένα μειονέκτημα της κληρονομικότητας σχετικά με την επαναχρησιμοποίηση:
  - Δημιουργεί αλληλοεξαρτήσεις μεταξύ των κλάσεων που κάνουν πολύπλοκη τη συντήρησή τους

Δυναμική Πρόσδεση

- Μια πολυμορφική μεταβλητή (*polymorphic variable)* μπορεί να οριστεί για μια κλάση έτσι ώστε να είναι αναφορά (ή να δείχνει σε) αντικείμενα της κλάσης ή αντικείμενα οποιουδήποτε από τους απογόνους της κλάσης
- Όταν μια ιεραρχία κλάσεων περιέχει κλάσεις που παρακάμπτουν μεθόδους και οι μέθοδοι αυτοί καλούνται μέσω της πολυμορφικής μεταβλητής, η σύνδεση της ορθής μεθόδου μπορεί να είναι δυναμική
- Η δυναμική πρόσδεση επιτρέπει την ευκολότερη επέκταση του λογισμικού τόσο κατά την αρχική του ανάπτυξη όσο και κατά τη συντήρησή του

Έννοιες δυναμικής πρόσδεσης

- Μια αφηρημένη μέθοδος είναι μια μέθοδος που δεν περιλαμβάνει κάποιον ορισμό (ορίζει μόνο ένα πρωτόκολλο)
- Μια αφηρημένη κλάση *(abstract class)* είναι μια κλάση που περιλαμβάνει τουλάχιστον μια ιδεατή μέθοδο (virtual method)
- Δεν μπορούν να δημιουργηθούν αντικείμενα από μια αφηρημένη κλάση

Θέματα Σχεδιασμού σε OOP Γλώσσες

- Είναι όλα αντικείμενα;
- Είναι οι υποκλάσεις, υποτύποι;
- Απλή και πολλαπλή κληρονομικότητα
- Δέσμευση μνήμης για κατανομή αντικειμένων και αποδέσμευση μνήμης για διαγραφή αντικειμένων
- Δυναμική και στατική πρόσδεση
- Εμφωλευμένες κλάσεις
- Αρχικοποίηση αντικειμένων

Η αποκλειστικότητα των αντικειμένων

- Αν όλα είναι αντικείμενα:
  - Πλεονέκτημα – Κομψότητα και καθαρότητα
  - Μειονέκτημα – Αργές πράξεις σε απλά αντικείμενα
- Προσθήκη της έννοιας των αντικειμένων σε ένα πλήρες σύστημα τύπων:
  - Πλεονέκτημα – γρήγορες πράξεις σε απλά αντικείμενα
  - Μειονέκτημα – οδηγεί σε «μπερδεμένο» σύστημα τύπων, καθώς υπάρχουν δύο ειδών οντότητες
- Συμπερίληψη ενός imperative-style συστήματος τύπων για τις πρωτογενείς τιμές, αλλά όλα τα άλλα είναι αντικείμενα:
  - Πλεονέκτημα – γρήγορες πράξεις σε απλά αντικείμενα και ένα σχετικά μικρό σύστημα τύπων
  - Μειονέκτημα –δημιουργεί κάποια σύγχυση καθώς υπάρχουν δύο ειδών συστήματα τύπων

Είναι οι υποκλάσεις υποτύποι;

- Υπάρχει σχέση “is-a” μεταξύ αντικειμένων της γονικής κλάσης και αντικειμένων της υποκλάσης;
  - Αν η παραγόμενη κλάση έχει σχέση “is-a” με τη γονική κλάση, τότε τα αντικείμενα της παραγόμενης κλάσης θα πρέπει να συμπεριφέρονται όμοια με τα αντικείμενα της γονικής κλάσης
- Μια παραγόμενη κλάση είναι υποτύπος αν έχει σχέση “is-a” με τη γονική κλάση
  - Η υποκλάση μπορεί μόνο να προσθέτει μεταβλητές και μεθόδους και να παρακάμπτει μεθόδους με «συμβατούς» τρόπους
  - Οι υποκλάσεις κληρονομούν την υλοποίηση, ενώ οι υποτύποι κληρονομούν διεπαφή και συμπεριφορά

Απλή και πολλαπλή κληρονομικότητα

- Η πολλαπλή κληρονομικότητα επιτρέπει σε νέες κλάσεις να κληρονομούν από δύο ή περισσότερες κλάσεις
- Μειονεκτήματα πολλαπλής κληρονομικότητας:
  - Πολυπλοκότητα γλώσσας και της υλοποίησής της (λόγω συγκρούσεων ονομάτων)
  - Πιθανή υποβάθμιση απόδοσης – η δυναμική πρόσδεση οδηγεί σε υψηλότερο (αλλά όχι κατά πολύ) κόστος από την πολλαπλή κληρονομικότητα
- Πλεονέκτημα:
  - Ορισμένες φορές η πολλαπλή κληρονομικότητα είναι αρκετά βολική και έχει αξία

Δέσμευση και αποδέσμευση αντικειμένων

- Από που γίνεται δέσμευση μνήμης για τα αντικείμενα;
  - Αν συμπεριφέρονται όπως τα ADTs, η δέσμευση μνήμης μπορεί να γίνει από οπουδήποτε
    - Δέσμευση μνήμης από τη στοίβα
    - Σαφής (explicit) δημιουργία στο σωρό (μέσω `new`)
  - Αν όλα είναι δυναμικά στο σωρό, οι αναφορές μπορούν να γίνονται ομοιόμορφα μέσω ενός δείκτη ή μιας μεταβλητής αναφοράς
    - Απλοποιεί την ανάθεση – η αποαναφορά μπορεί να είναι έμμεση (implicit)
  - Αν τα αντικείμενα είναι δυναμικά στη στοίβα, δημιουργείται πρόβλημα σχετικά με τους υποτύπους – object slicing
- Είναι η αποδέσμευση της μνήμης που καταλαμβάνουν τα αντικείμενα σαφής ή υπονοούμενη;

Δυναμική και στατική πρόσδεση

- Πρέπει όλες οι συνδέσεις των μηνυμάτων σε μεθόδους να είναι δυναμικές;
  - Αν δεν είναι καμία, τότε χάνονται τα πλεονεκτήματα της δυναμικής πρόσδεσης
  - Αν είναι όλες, δεν είναι αποδοτικό
- Ίσως η σχεδίαση της γλώσσας θα πρέπει να επιτρέπει στο χρήστη να το αποφασίσει

Εμφωλευμένες (nested) κλάσεις

- Αν μια νέα κλάση απαιτείται από μια μόνο κλάση, τότε δεν υπάρχει λόγος να ορίζεται έτσι ώστε να είναι ορατή από τις άλλες κλάσεις
  - Μπορεί μια νέα κλάση να είναι εμφωλευμένη μέσα στην κλάση που την χρησιμοποιεί;
  - Σε ορισμένες περιπτώσεις, η νέα κλάση είναι εμφωλευμένη σε ένα υποπρόγραμμα αντί να είναι απευθείας εμφωλευμένη σε μια άλλη κλάση
- Άλλα θέματα:
  - Ποια μέλη της εμφωλευμένης κλάσης πρέπει να είναι ορατά στην κλάση που την περιέχει και αντίστροφα;

Θέματα αρχικοποίησης αντικειμένων

- Κατά τη δημιουργία των αντικειμένων ποια αρχικοποίηση τιμών γίνεται;
  - Υπονοούμενη ή σαφής αρχικοποίηση
- Πως αρχικοποιούνται τα μέλη της γονικής κλάσης όταν δημιουργείται ένα αντικείμενο της υποκλάσης;

Υποστήριξη για OOP στη Smalltalk

- Η Smalltalk είναι μια «καθαρή» γλώσσα αντικειμενοστραφούς προγραμματισμού
  - Όλα είναι αντικείμενα
  - Όλα τα αντικείμενα έχουν τοπική μνήμη
  - Όλοι οι υπολογισμοί γίνονται μέσω αντικειμένων που στέλνουν μηνύματα σε αντικείμενα
  - Δεν έχει παρόμοια μορφή με καμία προστακτική γλώσσα προγραμματισμού
  - Όλα τα αντικείμενα δεσμεύονται από το σωρό
  - Όλες οι αποδεσμεύσεις μνήμης είναι έμμεσες
  - Οι κλάσεις στην Smalltalk δεν μπορούν να είναι εμφωλευμένες σε άλλες κλάσεις

Υποστήριξη για OOP στη Smalltalk (συνέχεια)

- Κληρονομικότητα
  - Μια υποκλάση στη Smalltalk κληρονομεί όλες τις μεταβλητές στιγμιοτύπων, τις μεθόδους στιγμιοτύπων, και τις μεθόδους κλάσης από την υπερκλάση της
  - Όλες οι υποκλάσεις είναι υποτύποι (δεν κρύβεται τίποτα)
  - All inheritance is implementation inheritance
  - Δεν υποστηρίζει πολλαπλή κληρονομικότητα

Υποστήριξη για OOP στη Smalltalk (συνέχεια)

- Δυναμική πρόσδεση
  - Όλες οι προσδέσεις των μηνυμάτων σε μεθόδους είναι δυναμικές
    - Η διαδικασία είναι ότι αναζητείται το αντικείμενο στο οποίο στέλνεται το μήνυμα για τη μέθοδο, αν δεν βρεθεί τότε αναζητείται στην υπερκλάση κ.ο.κ. μέχρι την κλάση του συστήματος που δεν έχει υπερκλάση
  - Ο μοναδικός έλεγχος τύπων είναι δυναμικός και ο μοναδικός τύπος σφάλματος συμβαίνει όταν το μήνυμα στέλνεται σε ένα αντικείμενο για το οποίο δεν υπάρχει μέθοδος που ταιριάζει στο μήνυμα

Υποστήριξη για OOP στη Smalltalk (συνέχεια)

- Αποτίμηση της Smalltalk
  - Το συντακτικό της γλώσσας είναι απλό και παρουσιάζει κανονικότητα
  - Αποτελεί καλό παράδειγμα της ισχύος που προκύπτει από μια μικρή γλώσσα
  - Είναι αργή σε σύγκριση με τις παραδοσιακές μεταγλωττιζόμενες προστακτικές γλώσσες
  - Η δυναμική πρόσδεση προκαλεί λάθη τύπων να μην ανιχνεύονται παρά μόνο κατά το χρόνο εκτέλεσης
  - Εισήγαγε ένα γραφικό περιβάλλον διεπαφής (graphical user interface)
  - Η μεγαλύτερη επίδραση της Smalltalk ήταν ότι ενίσχυσε την εικόνα του OOP

Υποστήριξη για OOP στην C++

- Γενικά χαρακτηριστικά:
  - Εξελίχθηκε από τη C και τη SIMULA 67
  - Είναι ανάμεσα στις OOP γλώσσες που χρησιμοποιείται περισσότερο
  - Μεικτό σύστημα τύπων
  - Διαθέτει κατασκευαστές (constructors) και καταστροφείς (destructors)
  - Διαθέτει προχωρημένο σύστημα ελέγχου πρόσβασης στα μέλη των κλάσεων

Υποστήριξη για OOP στην C++ (συνέχεια)

- Κληρονομικότητα
  - Μια κλάση δεν είναι απαραίτητο να είναι υποκλάση μιας άλλης κλάσης
  - Τα επίπεδα πρόσβασης στα μέλη των κλάσεων είναι:
    - Ιδιωτικό (private), ορατό μόνο στην κλάση στους «φίλους», απαγορεύει στις υποκλάσεις να είναι υποτύποι
    - Δημόσιο (public), ορατό στις υποκλάσεις και σε κώδικα που χρησιμοποιεί την κλάση
    - Προστατευμένος (protected), ορατό στην κλάση και στις υποκλάσεις της, αλλά όχι σε κώδικα που χρησιμοποιεί την κλάση

Υποστήριξη για OOP στην C++ (συνέχεια)

- Επιπλέον, η δημιουργία υποκλάσεων μπορεί να δηλωθεί private ή public, που επηρεάζει την πρόσβαση από την υποκλάση στα μέλη της κλάσης
  - Ιδιωτική παραγωγή υποκλάσης – τα κληρονομούμενα δημόσια και προστατευμένα μέλη είναι ιδιωτικά στις υποκλάσεις
  - Δημόσια παραγωγή κλάσης – τα κληρονομούμενα δημόσια και προστατευμένα μέλη είναι επίσης δημόσια και προστατευμένα στις υποκλάσεις

Παράδειγμα κληρονομικότητας στη C++

```
class base_class {
  private:
    int a;
    float x;
  protected:
    int b;
    float y;
  public:
    int c;
    float z;
};

class subclass_1 : public base_class { ... };
//      In this one, b and y are protected and
//      c and z are public

class subclass_2 : private base_class { ... };
//     In this one, b, y, c, and z are private,
//     and no derived class has access to any
//     member of base_class
```

Reexportation στην C++

- Ένα μέλος που δεν είναι προσβάσιμο σε μια υποκλάση (λόγω ιδιωτικής παραγωγής) μπορεί να γίνει ορατό χρησιμοποιώντας τον τελεστή ανάλυσης εμβέλειας (scope resolution operator) (: :), π.χ.,

```
class subclass_3 : private base_class {
        base_class :: c;
        …
}
```

Reexportation (συνέχεια)

- Ένα σκεπτικό περί της χρησιμότητας της ιδιωτικής παραγωγής:
  - Μια κλάση διαθέτει μέλη που πρέπει να είναι ορατά, συνεπώς τα ορίζει ως δημόσια, μια υποκλάση της προσθέτει ορισμένα νέα μέλη, αλλά δεν θέλει οι πελάτες της να βλέπουν τα μέλη της γονικής κλάσης, παρά το ότι έπρεπε να είναι δημόσια στον ορισμό της πατρικής κλάσης

Υποστήριξη για OOP στην C++ (συνέχεια)

- Υποστηρίζει την πολλαπλή κληρονομικότητα
  - Αν υπάρχουν δύο κληρονομούμενα μέλη με το ίδιο όνομα , τότε και τα δύο μπορούν να αναφερθούν ξεχωριστά με το τελεστή ανάλυσης αναφοράς (::)

```
class Thread { ... }
class Drawing { ... }
class DrawThread : public Thread, public Drawing
  { … }
```

Υποστήριξη για OOP στην C++ (συνέχεια)

- Δυναμική πρόσδεση
  - Μια μέθοδος μπορεί να οριστεί ως `virtual`, που σημαίνει ότι μπορεί από πολυμορφικές μεταβλητές και να πραγματοποιηθεί δυναμική πρόσδεση σε μηνύματα
  - Μια καθαρή ιδεατή συνάρτηση (pure virtual function) δεν έχει κώδικα που να την ορίζει
  - Μια κλάση που έχει τουλάχιστον μια καθαρή ιδεατή συνάρτηση είναι μια αφηρημένη κλάση

Υποστήριξη για OOP στην C++ (συνέχεια)

```
class Shape {
  public:
    virtual void draw() = 0;
    ...
};
class Circle : public Shape {
  public:
    void draw() { ... }
  ...
};
class Rectangle : public Shape {
  public:
    void draw() { ... }
  ...
};
class Square : public Rectangle {
  public:
    void draw() { ... }
  ...
};
```

```
Square* sq = new Square;
Rectangle* rect = new Rectangle;
Shape* ptr_shape;
ptr_shape = sq;   // δείχνει σε ένα  Square
ptr_shape ->draw();  // Δυναμική πρόσδεση
                     // του draw στο Square
rect->draw();        // Στατική πρόσδεση του
                     // draw στο Rectangle
```

Υποστήριξη για OOP στην C++ (συνέχεια)

- Αν τα αντικείμενα δεσμεύουν μνήμη από τη στοίβα (stack), τότε είναι διαφορετικά:

```
Square sq;              // Δεσμεύει ένα αντικείμενο Square από τη
                        // στοίβα
Rectangle rect;         // Δεσμεύει ένα αντικείμενο Rectangle από
                        // τη στοίβα
rect = sq;              // Αντιγράφει τις τιμές των μελών
                        // δεδομένων από το αντικείμενο sq
rect.draw();            // Καλεί την draw του Rectangle
```

Υποστήριξη για OOP στην C++ (συνέχεια)

- Αποτίμηση
  - Η C++ παρέχει προχωρημένους μηχανισμούς ελέγχου πρόσβασης (σε αντίθεση με την Smalltalk)
  - Η C++ υποστηρίζει την πολλαπλή κληρονομικότητα
  - Στη C++, ο προγραμματιστής πρέπει να αποφασίσει κατά το σχεδιασμό της εφαρμογής ποιες μέθοδοι θα πρέπει να προσδένονται στατικά και ποιες μέθοδοι θα πρέπει να προσδένονται δυναμικά
  - Η στατική πρόσδεση είναι ταχύτερη!
  - Οι έλεγχοι τύπων της Smalltalk type είναι δυναμικοί (ευελιξία, αλλά κάπως ανασφαλές)
  - Λόγω της μεταγλώττισης και της δυναμικής πρόσδεσης η Smalltalk είναι περίπου 10 φορές αργότερη από τη C++

Υποστήριξη για OOP στην Objective-C

- Παρόμοια με τη C++, η Objective-C προσθέτει υποστήριξη για OOP στη C
- Σχεδιάστηκε περίπου στην ίδια εποχή με την C++
- Σημαντικότερη διαφορά στο συντακτικό: ο τρόπος που γίνονται οι κλήσεις μεθόδων
- Στο τμήμα διεπαφής (interface section) μιας κλάσης δηλώνονται οι μεταβλητές στιγμιοτύπων και οι μέθοδοι
- Το τμήμα υλοποίησης της κλάσης (implementation section) ορίζει τις μεθόδους
- Οι κλάσεις δεν μπορούν να είναι εμφωλευμένες

Υποστήριξη για OOP στην Objective-C (συνέχεια)

- Κληρονομικότητα
  - Μόνο απλή κληρονομικότητα
  - Κάθε κλάση πρέπει να έχει γονική κλάση
  - Η NSObject είναι η βασική κλάση

```
@interface myNewClass: NSObject { … }
   …
@end
```

  - Καθώς όλα τα δημόσια μέλη μιας βασικής κλάσης είναι επίσης δημόσια και στην παραγόμενη κλάση όλες οι υποκλάσεις είναι υποτύποι
  - Οποιαδήποτε μέθοδος που έχει το ίδιο όνομα, τον ίδιο τύπο επιστροφής, το ίδιο πλήθος και τύπο παραμέτρων με μια κληρονομούμενη μέθοδο παρακάμπτει την κληρονομούμενη μέθοδο
  - Μια μέθοδος που έχει παρακαμφθεί μπορεί να κληθεί με το **`super`**
  - Η κληρονομικότητα είναι πάντα δημόσια (σε αντίθεση με τη C++)

Υποστήριξη για OOP στην Objective-C (συνέχεια)

- Κληρονομικότητα (συνέχεια)
- Η Objective-C έχει δύο προσεγγίσεις για την επέκταση των κλάσεων πέρα από τη χρήση υποκλάσεων
  - Μια κατηγορία (*category)* είναι ένα δευτερεύον interface μιας κλάσης που περιέχει δηλώσεις μεθόδων (αλλά όχι μεταβλητές στιγμιοτύπων)

```
#import "Stack.h"
@interface Stack (StackExtend)
  -(int) secondFromTop;
  -(void) full;
@end
```

  - Μια κατηγορία είναι ένα *mixin* – οι μέθοδοί της προστίθενται στη γονική κλάση
  - Η υλοποίηση της κατηγορίας γίνεται ξεχωριστά:

```
@implementation Stack (StackExtend)
```

Υποστήριξη για OOP στην Objective-C (συνέχεια)

- Κληρονομικότητα (συνέχεια)
  - Εναλλακτικός τρόπος για την επέκταση μιας κλάσης: πρωτόκολλα (protocols)
  - Ένα πρωτόκολλο είναι μια λίστα από δηλώσεις μεθόδων

```
@protocol MatrixOps
  -(Matrix *) add: (Matrix *) mat;
  -(Matrix *) subtract: (Matrix *) mat;
@optional
  -(Matrix *) multiply: (Matrix *) mat;
@end
```

  - Το `MatrixOps` είναι το όνομα του πρωτοκόλλου
  - Οι μέθοδοι `add` και `subtract` πρέπει να υλοποιηθούν στην κλάση που χρησιμοποιεί το πρωτόκολλο
  - Η κλάση που «υιοθετεί» ένα πρωτόκολλο θα πρέπει να το προσδιορίζει:

```
@interface MyClass: NSObject <YourProtocol>
```

Υποστήριξη για OOP στην Objective-C (συνέχεια)

- Δυναμική πρόσδεση
  - Διαφορετική προσέγγιση σε σχέση με άλλες OOP γλώσσες – μια πολυμορφική μεταβλητή είναι τύπου `id`
  - Μια μεταβλητή τύπου `id` μπορεί να αποτελεί αναφορά σε οποιοδήποτε αντικείμενο
  - Το σύστημα χρόνου εκτέλεσης παρακολουθεί τον τύπο του αντικειμένου στο οποίο αναφέρεται κάθε μια μεταβλητή τύπου `id`
  - Αν μια κλήση σε μια μέθοδο γίνεται μέσω μιας μεταβλητής τύπου `id`, η πρόσδεση με τη μέθοδο είναι δυναμική

Υποστήριξη για OOP στην Objective-C (συνέχεια)

- Αποτίμηση
  - Η υποστήριξη για OOP είναι επαρκής, με τις ακόλουθες ελλείψεις:
    - Δεν υπάρχει τρόπος να αποτρέψει κανείς την παράκαμψη μιας μεθόδου που κληρονομείται
    - Η χρήση μεταβλητών τύπου `id` για δυναμική πρόσδεση είναι ίσως υπερβολική – οι μεταβλητές τύπου `id` ενδεχόμενα χρησιμοποιούνται για λάθος λόγους
  - Οι κατηγορίες και τα πρωτόκολλα είναι χρήσιμες προσθήκες στη γλώσσα

Υποστήριξη OOP στην Java

- Λόγω της στενής σχέσης με την C++, στη συνέχεια δίνεται έμφαση στις διαφορές της Java με τη C++
- Γενικά χαρακτηριστικά
  - Όλα τα δεδομένα είναι αντικείμενα εκτός από τους πρωτογενείς τύπους
  - Όλοι οι πρωτογενείς τύποι έχουν wrapper κλάσεις που αποθηκεύουν μια τιμή δεδομένων
  - Όλα τα αντικείμενα είναι δυναμικά–σωρού, γίνεται αναφορά σε αυτά μέσω μεταβλητών αναφοράς, και για τα περισσότερα η δέσμευση μνήμης γίνεται με το **`new`**
  - Η μέθοδος **`finalize`** καλείται έμμεσα όταν ο συλλογέας απορριμμάτων (garbage collector) πρόκειται να επανα–διεκδικήσει το χώρο αποθήκευσης που έχει καταλάβει το αντικείμενο

Υποστήριξη OOP στην Java (συνέχεια)

- Κληρονομικότητα
  - Υποστηρίζεται μόνο η απλή κληρονομικότητα, αλλά υπάρχει μια κατηγορία αφηρημένων κλάσεων (οι διεπαφές – interfaces) που προσφέρουν κάποια από τα πλεονεκτήματα της πολλαπλής κληρονομικότητας
  - Ένα interface μπορεί να περιέχει μόνο δηλώσεις μεθόδων και σταθερές με ονόματα, π.χ.,

```
public interface Comparable <T> {
        public int comparedTo (T b);
}
```

  - Οι μέθοδοι μπορούν να είναι **`final`** (δεν μπορούν να παρακαμφθούν)
  - Όλες οι υποκλάσεις είναι υποτύποι

Υποστήριξη OOP στην Java (συνέχεια)

- Δυναμική πρόσδεση
  - Στην Java, όλα τα μηνύματα προσδένονται δυναμικά με μεθόδους, εκτός αν μια μέθοδος είναι `final` (δλδ, δεν μπορεί να παρακαμφθεί, συνεπώς η δυναμική πρόσδεση δεν έχει νόημα)
  - Η στατική πρόσδεση χρησιμοποιείται σε μεθόδους που είναι `static` ή `private` καθώς και τα δύο απαγορεύουν την παράκαμψη μεθόδων

Υποστήριξη OOP στην Java (συνέχεια)

- Εμφωλευμένες κλάσεις
  - Όλα είναι κρυμμένα εκτός από την κλάση που περιέχει την εμφωλευμένη κλάση
  - Οι μη στατικές κλάσεις που είναι απευθείας εμφωλευμένες ονομάζονται *innerclasses*
    - Μια innerclass μπορεί να προσπελάσει μέλη από την κλάση που την περιέχει
    - Μια στατική εμφωλευμένη κλάση δεν μπορεί να προσπελάσει μέλη από την κλάση που την περιέχει
  - Οι εμφωλευμένες κλάσεις δεν μπορεί να είναι ανώνυμες
  - Μια τοπικά εμφωλευμένη κλάση ορίζεται σε μια μέθοδο της κλάση που την περιέχει
    - Δεν χρησιμοποιούνται προσδιοριστές πρόσβασης

Υποστήριξη OOP στην Java (συνέχεια)

- Αποτίμηση
  - Οι σχεδιαστικές αποφάσεις που έχουν ληφθεί προκειμένου να υποστηριχθεί ο OOP είναι παρόμοιες με αυτές που έχουν ληφθεί για τη C++
  - Δεν υποστηρίζει τον διαδικασιακό προγραμματισμό
  - Δεν υπάρχουν κλάσεις χωρίς γονέα
  - Η δυναμική πρόσδεση είναι ο «κανονικός» τρόπος πρόσδεσης κλήσεων μεθόδων σε ορισμούς μεθόδων
  - Χρησιμοποιεί διεπαφές (interfaces) για να υποστηρίξει μια απλή μορφή πολλαπλής κληρονομικότητας

Υποστήριξη για OOP στην C#

- Γενικά χαρακτηριστικά:
  - Η υποστήριξη για OOP είναι παρόμοια με την Java
  - Περιέχει τόσο κλάσεις όσο και δομές (`struct`)
  - Οι κλάσεις είναι παρόμοιες με τις κλάσεις της Java
  - Τα `structs` είναι σε σχέση με τις κλάσεις λιγότερα ισχυρές stack-dynamic κατασκευές (π.χ., δεν υποστηρίζουν την κληρονομικότητα)

Υποστήριξη για OOP στην C# (συνέχεια)

- Κληρονομικότητα
  - Χρησιμοποιεί τη σύνταξη της C++ για τον ορισμό κλάσεων
  - Μια μέθοδος που κληρονομείται από μια γονική κλάση μπορεί να αντικατασταθεί από την την παραγόμενη κλάση σημειώνοντας τον ορισμό με το **`new`**
  - Η έκδοση της γονική κλάσης εξακολουθεί να μπορεί να κληθεί σαφώς (explicitly) με το prefix **`base:`**

```
base.Draw()
```

  - Οι υποκλάσεδις είναι υποτύποι αν κανένα μέλος της γονικής κλάσης δεν είναι ιδιωτικό
  - Υποστηρίζει μόνο την απλή κληρονομικότητα

Υποστήριξη για OOP στην C# (συνέχεια)

- Δυναμική πρόσδεση
  - Για να επιτραπεί η δυναμική πρόσδεση των κλήσεων μεθόδων με τις μεθόδους:
    - Η βασική κλάση σημειώνεται ως **`virtual`**
    - Οι αντίστοιχες μέθοδοι στις παραγόμενες κλάσεις σημειώνονται ως **`override`**
  - Οι αφηρημένες μέθοδοι σημειώνονται ως **`abstract`** και πρέπει να υλοποιηθούν σε όλες τις υποκλάσεις
  - Όλες οι κλάσεις στην C# παράγονται τελικά από μια κλάση ρίζα την `Object`

Υποστήριξη για OOP στην C# (συνέχεια)

- Εμφωλευμένες κλάσεις
  - Μια C# κλάση που είναι απευθείας εμφωλευμένη σε μια κλάση, συμπεριφέρεται όπως οι στατικά εμφωλευμένες κλάσεις της Java
  - Η C# δεν υποστηρίζει εμφωλευμένες κλάσεις με συμπεριφορά όπως οι μη-στατικές κλάσεις της Java

Υποστήριξη για OOP στην C# (συνέχεια)

- Αποτίμηση
  - Η C# είναι μια σχετικά πρόσφατη αντικειμενοστραφής γλώσσα που βασίζεται στην C
  - Οι διαφορές μεταξύ της C# και της Java σχετικά με την υποστήριξη για OOP είναι σχετικά μικρές

Υποστήριξη για OOP στην Ruby

- Γενικά χαρακτηριστικά
  - Όλα είναι αντικείμενα
  - Όλοι οι υπολογισμοί γίνονται μέσω περάσματος μηνυμάτων
  - Οι ορισμοί κλάσεων είναι εκτελέσιμοι, επιτρέποντας σε δευτερεύοντες ορισμούς να προσθέσουν μέλη σε υπάρχοντες ορισμούς
  - Οι ορισμοί των μεθόδων είναι επίσης εκτελέσιμοι
  - Όλες οι μεταβλητές είναι αναφορές χωρίς τύπο σε αντικείμενα
  - Ο έλεγχος πρόσβασης είναι διαφορετικός για τα δεδομένα και τις μεθόδους
    - Είναι ιδιωτικός για όλα τα δεδομένα και αυτό δεν μπορεί να αλλάξει
    - Οι μέθοδοι μπορούν να είναι είτε ιδιωτικές είτε δημόσιες είτε προστατευμένες
    - Η πρόσβαση σε μεθόδους ελέγχεται κατά το χρόνο εκτέλεσης
  - Οι getters και οι setters μπορούν να οριστούν με συνοπτικό τρόπο

Υποστήριξη για OOP στην Ruby (συνέχεια)

- Κληρονομικότητα
  - Ο έλεγχος πρόσβασης σε μεθόδους που κληρονομούνται μπορεί να είναι διαφορετικός από τη γονική κλάση
  - Οι υποκλάσεις δεν είναι απαραίτητα υποτύποι
- Δυναμική πρόσδεση
  - Όλες οι μεταβλητές είναι χωρίς τύπο (typeless) και πολυμορφικές
- Αποτίμηση
  - Δεν υποστηρίζει αφηρημένες κλάσεις
  - Δεν υποστηρίζει πλήρως πολλαπλή κληρονομικότητα
  - Ο έλεγχος πρόσβασης είναι ασθενέστερος από άλλες γλώσσες που υποστηρίζουν OOP

Υλοποίηση ΟΟ κατασκευών

- Δύο ενδιαφέροντα θέματα:
  - Οι δομές αποθήκευσης για τις μεταβλητές στιγμιοτύπων
  - Δυναμική πρόσδεση μηνυμάτων σε μεθόδους

Αποθήκευση δεδομένων στιγμιοτύπων

- Οι εγγραφές στιγμιοτύπων κλάσεων (CIRs=Class Instance Records) αποθηκεύουν την κατάσταση (state) ενός αντικειμένου
  - Στατικά (δημιουργείται κατά το χρόνο μεταγλώττισης)
- Αν μια κλάση έχει γονέα, οι μεταβλητές στιγμιοτύπων της υποκλάσης προστίθενται στο γονικό CIR
- Λόγω του ότι το CIR είναι στατικό, η πρόσβαση σε όλες τις μεταβλητές των στιγμιοτύπων γίνεται όπως και στις εγγραφές
  - Αποδοτικό

Δυναμική πρόσδεση των κλήσεων μεθόδων

- Οι μέθοδοι μιας κλάσης που είναι στατικά προσδεδεμένες δεν εμπλέκονται με το CIR, ενώ οι μέθοδοι που θα προσδένονται δυναμικά θα πρέπει να έχουν εγγραφές στο CIR
  - Οι κλήσεις σε δυναμικά προσδεδεμένες μεθόδους μπορούν να συνδέονται με τον αντίστοιχους κώδικες μέσω δεικτών στο CIR
  - Η δομή αποθήκευσης συχνά ονομάζεται *virtual method tables* (vtable)
  - Οι μέθοδοι μπορούν να αναπαρασταθούν ως μετατοπίσεις (offsets) από την αρχή του vtable

Ανάκλαση (reflection)

- Μια γλώσσα προγραμματισμού που υποστηρίζει ανάκλαση επιτρέπει στα προγράμματά της να έχουν πρόσβαση κατά το χρόνο εκτέλεσης στους τύπους τους και στη δομή τους και να μπορούν δυναμικά να τροποποιούν την συμπεριφορά τους
- Οι τύποι και η δομή ενός προγράμματος ονομάζονται μεταδεδομένα *(metadata)*
- Η διαδικασία εξέτασης των μεταδεδομένων ενός προγράμματος ονομάζεται ενδοσκόπηση *(introspection)*
- Η μεσολάβηση στην εκτέλεση ενός προγράμματος λέγεται *intercession*

Ανάκλαση (συνέχεια)

- Χρήσεις της ανάκλασης από εργαλεία λογισμικού:
  - Οι class browsers στα IDEs χρειάζεται να παρουσιάζουν όλες τις κλάσεις ενός προγράμματος
  - Τα IDEs χρησιμοποιούν πληροφορίες σχετικά με τους τύπους για να βοηθήσουν τους προγραμματιστές να γράφουν σωστό κώδικα
  - Οι αποσφαλματωτές (debuggers) χρειάζεται να εξετάζουν ιδιωτικά πεδία και μεθόδους κλάσεων
  - Τα συστήματα ελέγχου χρειάζεται να γνωρίζουν όλες τις μεθόδους μιας κλάσης

Ανάκλαση στη Java

- Περιορισμένη υποστήριξη από το `java.lang.Class`
- Το Java runtime αρχικοποιεί ένα στιγμιότυπο της `Class` για κάθε αντικείμενο του προγράμματος
- Η μέθοδος `getClass` της `Class` επιστρέφει το αντικείμενο `Class` ενός αντικειμένου

```
float[] totals = new float[100];
Class fltlist = totals.getClass();
Class stg = "hello".getClass();
```

- Αν δεν υπάρχει αντικείμενο, γίνεται χρήση του πεδίου `class`

```
Class stg = String.class;
```

Ανάκλαση στη Java (συνέχεια)

- Η `Class` διαθέτει 4 χρήσιμες μεθόδους:
  - `getMethod` αναζητά για μια συγκεκριμένη δημόσια μέθοδο μιας κλάσης
  - `getMethods` επιστρέφει έναν πίνακα με όλες τις δημόσιες μεθόδους μιας κλάσης
  - `getDeclaredMethod` αναζητά μια συγκεκριμένη μέθοδο μιας κλάσης
  - `getDeclaredMethods` επιστρέφει έναν πίνακα με όλες τις μεθόδους μιας κλάσης

Ανάκλαση στη Java (συνέχεια)

- Η κλάση `Method` ορίζει τη μέθοδο `invoke`, που χρησιμοποιείται για να εκτελέσει τις μεθόδους που εντοπίζονται με τη `getMethod`

Ανάκλαση στη C#

- Στις γλώσσες .NET ο μεταγλωττιστής τοποθετεί τον ενδιάμεσο κώδικά που παράγει μαζί με τα metadata για το πρόγραμμα
- Το `System.Type` είναι το namespace για την ανάκλαση
- Χρησιμοποιείται το `getType` στη θέση του `getClass` της Java
- Χρησιμοποιείται ο τελεστής `typeof` αντί για το πεδίο `.class` που χρησιμοποιείται στη Java
- Το namespace `System.Reflection.Emit` παρέχει τη δυνατότητα δημιουργίας ενδιάμεσου κώδικα και τοποθέτησής σε assembly (Η Java δεν έχει αυτή τη δυνατότητα)

Μειονεκτήματα της ανάκλασης

- Κόστη απόδοσης
- Εκθέτει τα ιδιωτικά πεδία και μεθόδους
- Voids the advantages of early type checking
- Ορισμένος κώδικας με ανάκλαση μπορεί μα μην εκτελείται υπό τον security manager, καθιστώντας τον κώδικα μη φορητό

Σύνοψη

- Ο αντικειμενοστραφής προγραμματισμός εμπεριέχει 3 θεμελιώδεις έννοιες: ADTs, κληρονομικότητα, δυναμική πρόσδεση
- Κύρια θέματα σχεδίασης: αποκλειστικότητα αντικειμένων, υποκλάσεις και υποτύποι, έλεγχος τύπων και πολυμορφισμός, απλή και πολλαπλή κληρονομικότητα, δυναμική πρόσδεση, σαφής και υπονοούμενη δέσμευση και αποδεσμευση αντικειμένων και εμφωλευμένες κλάσεις
- Η Smalltalk είναι μια «καθαρή» OOP γλώσσα
- Η C++ έχει δύο διακριτά συστήματα τύπων (υβριδική)
- Η Java δεν είναι υβριδική γλώσσα όπως η C++, υποστηρίζει μόνο OOP
- Η C# στηρίζεται στη C++ και στη Java
- Η Ruby είναι μια σχετικά πρόσφατη «καθαρή» OOP γλώσσα, παρέχει ορισμένες νέες ιδέες για τη υποστήριξη του OOP
- Η υλοποίηση του OOP επιτυγχάνεται με ορισμένες νέες δομές δεδομένων
- Η ανάκλαση (reflection) υποστηρίζεται στην Java και στην C#, όπως και στις περισσότερες γλώσσες δυναμικών τύπων