ΤΕΙ ΣΕΡΡΩΝ

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Κληρονομικότητα

Ευάγγελος Γ. Ούτσιος
Θεόδωρος Γ. Λάντζος

Διάλεξη Νο6

Κληρονομικότητα

- Η δυνατότητα μοιράσματος ομοιοτήτων ανάμεσα σε κλάσεις ενώ δεσμεύονται οι διαφορές τους
- Κληρονομικότητα (Inheritance), Γενίκευση (Generalization), Ειδίκευση (Specialization) όροι ταυτόσημοι και αναφέρονται στην ίδια ιδιότητα
- Κληρονομικότητα είναι η σχέση ανάμεσα σε μια κλάση και σε μία ή περισσότερες εκδόσεις αυτής
- Δύο σχέσεις υπάρχουν στην κληρονομικότητα α. η υπέρ-κλάση (superclass) και β. η υπό-κλάση (subclass)
- Υπέρ-κλάση καλείται η κλάση η οποία ραφινάρετε και υπό κλάση καλείται η κάθε ειδίκευση της ραφιναρισμένης κλάσης
- Η κληρονομικότητα δεν είναι στατική σχέση
- Η κληρονομικότητα έχει μεταβατικότητα σε κάθε επίπεδο

Κληρονομικότητα

- Η κληρονομικότητα εισάγει δύο έννοιες του πρόγονου και του απογόνου οι οποίες αναφέρονται στην ειδίκευση της κλάσης σε πολλαπλά επίπεδα
- Μια στιγμή μιας υπό-κλάσης είναι ταυτόχρονα και μια στιγμή από όλες τις κλάσεις προγόνου
- Η κάθε στιγμή μιας κλάσης περιλαμβάνει τιμές για κάθε χαρακτηριστικό της κάθε κλάσης προγόνου
- Καθώς και κάθε λειτουργία της κλάσης προγόνου μπορεί να εφαρμοστεί από κάθε στιγμή της κλάσης. Ουσιαστικά, η δυνατότητα χρήσης των λειτουργιών από τις κλάσεις προγόνου.

Κληρονομικότητα

- Ταυτόσημη με την επαναχρησιμοποίηση κώδικα
- Ομαδοποίηση κοινών κλάσεων για επαναχρησιμοποίηση κοινών χαρακτηριστικών και λειτουργιών
- Το πιο σημαντικό πλεονέκτημα της κληρονομικότητας είναι η λογική απλούστευση η οποία μειώνει τον αριθμό των ανεξάρτητων χαρακτηριστικών μέσα σ' ένα σύστημα

Υλοποίηση κληρονομικότητας στη C

```
#include <iostream.h>

class Account
{
  protected:
      float balance;
  public:
      Account()
      {
        balance = 0;
      }
      Account(float balance1)
      {
        balance = balance1;
      }
      void withdraw(float money)
      {
        if (money <= balance)
          balance = balance – money;
        else
          cout << "Το ποσό ανάληψης υπερβαίνει το τρέχον!" << endl;
      }
      void deposit(float money)
      {
        balance += money;
      }
      float getBalance()
      {
        return balance;
      }
};
```

```
class AccInter : public Account
{
  public:
      void interest()
      {
        balance += balance*0.1;
      }
};
main()
{
  AccInter a1;
  cout << "Τρέχον ποσό λογαριασμού a1 = " << a1.getBalance();
  a1.deposit(100);
  cout << "Τρέχον ποσό λογαριασμού a1 = " << a1.getBalance();
  a1.interest();
  cout << "Τρέχον ποσό λογαριασμού a1 = " << a1.getBalance();
}
```

Κληρονομικότητα

Δημιουργία Αντικειμένου α1 με συνάρτηση Εγκατάστασης από την βασική κλάση

Κληρονομικότητα
- Αναζήτηση στην βασική κλάση
- Όχι ανάστροφα

Παράγωγες Κλάσεις και συναρτήσεις εγκατάστασης

Όταν θέλουμε να δώσουμε αρχικές τιμές σε παράγωγες κλάσεις θα πρέπει να γράψουμε συναρτήσεις εγκατάστασης για την παράγωγη κλάση οι οποίες θα Καλούν τις συναρτήσεις εγκατάστασης της βασικής κλάσης

Βλέπε παράδειγμα 7.2

Κληρονομικότητα και συναρτήσεις μελών

Στις παράγωγες κλάσεις μπορούμε να γράψουμε συναρτήσεις μέλη που έχουν το Ίδιο όνομα με κάποιες συναρτήσεις από της βασικής κλάσης

Στο παράδειγμα που ακολουθεί 7.3 θέλουμε να έχουμε συναρτήσεις ανάληψης Και κατάθεσης οι οποίες να εξετάζουν την περίπτωση αρνητικής κίνησης. Αυτό Υλοποιείται με υπερφόρτωση των ίδιων συναρτήσεων στην παράγωγη κλάση

Βλέπε παράδειγμα 7.3

Προσπελασιμότητα

Πίνακας προσπελασιμότητας

Καθοριστής πρόσβασης	Προσπελάσιμο από τη δική του κλάση	Προσπελάσιμο από την παράγωγη κλάση	Προσπελάσιμο από αντικείμενα έξω από την κλάση
public	ναι	ναι	ναι
protected	ναι	ναι	όχι
private	ναι	όχι	όχι

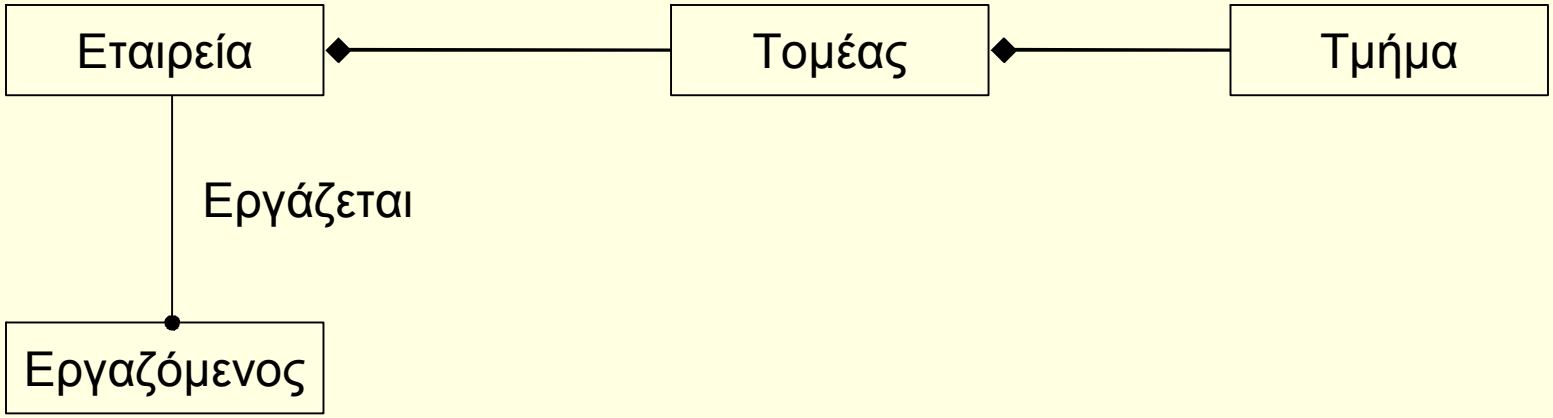
Συνδυασμοί προσπέλασης

- Αντικείμενα της παράγωγης κλάσης μπορούν να προσπελάσουν δημόσια δεδομένα της βασικής κλάσης, μόνον εφόσον η κλάση παράγεται δημόσια
- Αντικείμενα παράγωγης κλάσης που παράγεται ιδιωτικά δεν μπορούν να προσπελάσουν ούτε δημόσια δεδομένα της βασικής κλάσης
- Αν δεν δώσουμε κάποιο καθοριστή προσπέλασης όταν δημιουργούμε μια κλάση υποτίθεται ότι είναι ιδιωτικός (private)

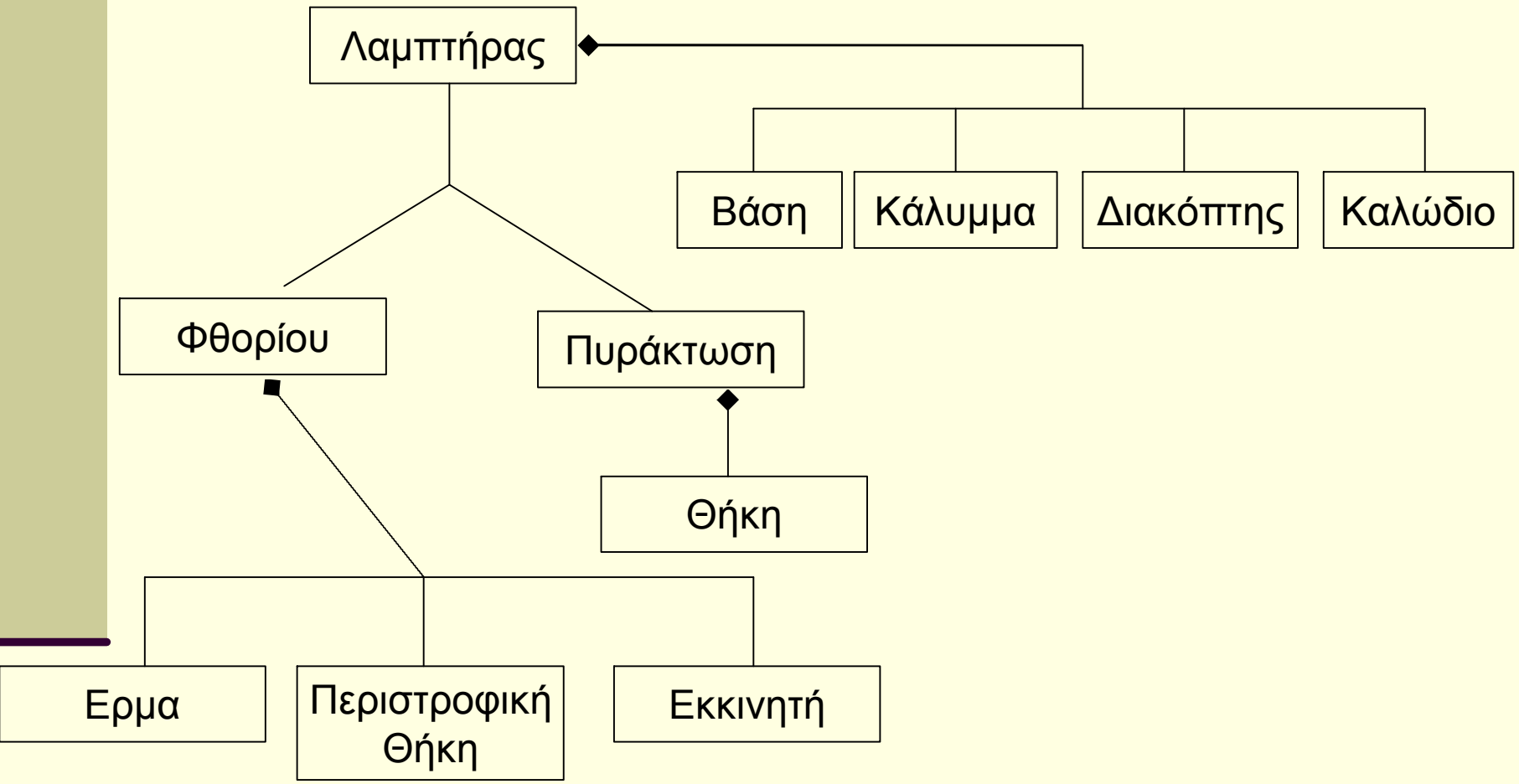
Περιεκτικότητα (Aggregation)

- Η περιεκτικότητα είναι μια μορφή σχέσης στον ΟΟ προγραμματισμό με βάση την οποία ένα αντικείμενο συν-αποτελείται από άλλα επιμέρους αντικείμενα (components)
- Εντοπίζετε στο σχεδιασμό με την χρήση των εκφράσεων ‘είναι μέρος από;’ ή ‘φτιάχνετε από’
- Τα αντικείμενα components μπορούν να αποτελούν μέρος πολλών σχέσεων περιεκτικότητας
- Η περιεκτικότητα είναι μεταβατική, δηλαδή ένα αντικείμενο αποτελείται από άλλα αντικείμενα components, τα οποία με την σειρά τους μπορούν να είναι και αυτά συν-αποτελούμενα από άλλα αντικείμενα

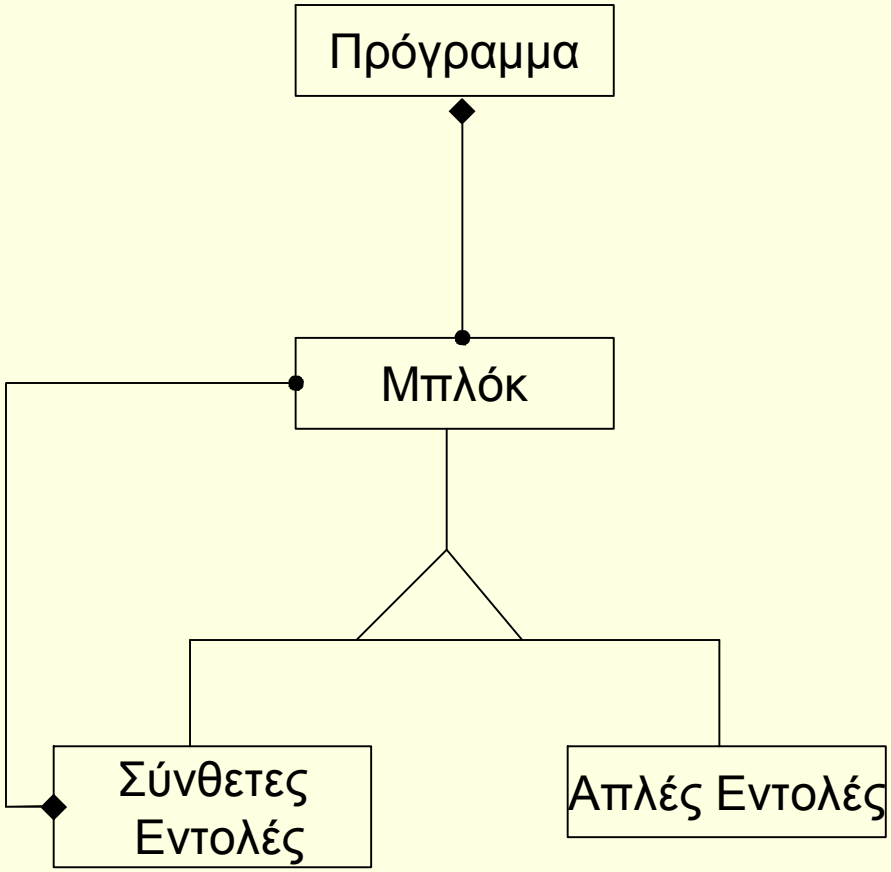
Περιεκτικότητα και Σχέσεις



Περιεκτικότητα και Κληρονομικότητα



Αναδρομική Περιεκτικότητα



```
class A
{
   B b;
};
class B
{};
```

```
#include <iostream.h>
#include <string.h>
class Account
{
  protected:
      float balance;
  public:
      Account()
      {
        balance = 0;
      }
      Account(float balance1)
      {
        balance = balance1;
      }
      void withdraw(float money)
      {
        if (money <= balance)
          balance = balance – money;
        else
          cout << "Το ποσό ανάληψης υπερβαίνει το τρέχον!" << endl;
      }
      void deposit(float money)
      {
          balance += money;
      }
      float getBalance()
      {
          return balance;
      }
};
class Customer
{
  private:
      int number;
      char name[20];
      Account a;
  public:
      Customer()
      {}
      Customer(int number1, char name1[], float bal):Account(bal)
      {
        number = number1;
        strcpy(name, name1);
      }
      void getMoney()
      {
        float money;
        cout << "Δώσε ποσό ανάληψης:";
        cin >> money;
        a.withdraw(money);
      }
      void putMoney()
      {
        float money;
        cout << "Δώσε ποσό κατάθεσης:";
        cin >> money;
        a.deposit(money);
      }
      void balanceReport()
      {
        cout << "Τρέχον ποσό λογαριασμού = " << a.getBalance();
      }
};
main()
{
  Customer c;

  c.balanceReport();
  c.putMoney();
  c.balanceReport();
  c.getMoney();
  c.balanceReport();
}
```