ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ 2

ΥΠΕΡΦΟΡΤΩΣΗ ΜΕΘΟΔΟΥ - ΠΟΛΥΜΟΡΦΙΣΜΟΣ

Μπορούμε μέσα σε μία παραγόμενη κλάση να φτιάξουμε μεθόδους που έχουν το ίδιο όνομα με κάποιες μεθόδους της βασικής κλάσης. Με αυτόν τον τρόπο, γίνεται πιο εύκολη για εμάς η χρήση μεθόδων κλάσεων -που κάνουν παρόμοιες λειτουργίες- με το ίδιο όνομα, γιατί δεν αναγκαζόμαστε να μπαίνουμε στη διαδικασία να θυμόμαστε διαφορετικά ονόματα μεθόδων.
Επίσης μπορούμε να φτιάξουμε μεθόδους με το ίδιο όνομα στις παραγόμενες κλάσεις οι οποίες να περιέχουν κάποιον εξτρά έλεγχο, χωρίς να χρειάζεται να πειράζουμε τον κώδικα των μεθόδων της βασικής κλάσης για την προσθήκη σε αυτές του συγκεκριμένου ελέγχου.
Αυτή η διαδικασία ονομάζεται ***υπερφόρτωση μεθόδου***.
Είναι πολύ σημαντικό να γνωρίζουμε ότι μέσα σε μέθοδο της παραγόμενης κλάσης για να καλέσουμε την αντίστοιχη μέθοδο με το ίδιο όνομα της βασικής κλάσης, θα πρέπει να χρησιμοποιούμε τον ***τελεστή διάκρισης εμβέλειας*** **(::)**, γιατί αλλιώς η μέθοδος καλεί τον εαυτό της (αναδρομή) με όχι και τόσο επιθυμητά αποτελέσματα για το πρόγραμμα.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΥΠΕΡΦΟΡΤΩΣΗΣ ΜΕΘΟΔΟΥ

Παράδειγμα 1. Υπερφόρτωση μεθόδου στο παράδειγμα κληρονομικότητας κλάσεων δύο επιπέδων «Τραπεζικές συναλλαγές» της Άσκησης 8 με προσθήκη ελέγχου.

```
#include <iostream>
using namespace std;

//Δήλωση Παραγόμενης Κλάσης
class Interest:public Accounts{
public:
     //Υπερφορτωμένη μέθοδος ανάληψης
     void withdrawal(float amount){
         //Έλεγχος αρνητικής τιμής
          if(amount > 0) // Αν το ποσό είναι θετικός αριθμός……

            /*…… τότε κάλεσε την μέθοδο ανάληψης
            της βασικής κλάσης Accounts και στείλε το ποσό */
            Accounts :: withdrawal(amount);
          else
              cout << "Attention not valid amount " << endl;
     }

     //Υπερφορτωμένη μέθοδος κατάθεσης
     void deposit (float amount){
         //Έλεγχος αρνητικής τιμής
         if(amount > 0) // Αν το ποσό είναι θετικός αριθμός……

             /*…… τότε κάλεσε την μέθοδο κατάθεσης
             της βασικής κλάσης Accounts και στείλε το ποσό */
             Accounts :: deposit(amount);
          else
              cout << " Attention not valid amount " << endl;
     }

     //μέθοδος απόδοσης τόκου
     void interest_payment(){
          balance += balance * 0.1; //Απόδοση τόκου 10%
     }

}; //Τέλος Παραγόμενης Κλάσης
```

```
int main(){

	/* Αντικείμενο της παράγωγης κλάσης με υπόλοιπο 0 Euro
	από τον κατασκευαστή χωρίς όρισμα της βασικής κλάσης */
	Interest t1;

	t1.deposit(100); //Κατάθεση 100 Euro

	//Ενημέρωση υπολοίπου (Υπόλοιπο 100 Euro)
	cout << "New Balance is: " << t1.returnBalance() << endl;

	t1.deposit(-30); //Προσπάθεια κατάθεσης αρνητικού ποσού

	//Το υπόλοιπο δεν ενημερώνεται (Υπόλοιπο 100 Euro)
	cout << "New Balance is: " << t1.returnBalance () << endl;

	t1.withdrawal(50); //Ανάληψη 50 Euro

	//Ενημέρωση υπολοίπου (Υπόλοιπο 50 Euro)
	cout << "New Balance is: " << t1.returnBalance () << endl;

	t1.withdrawal(-45); //Προσπάθεια ανάληψης αρνητικού ποσού

	//Το υπόλοιπο δεν ενημερώνεται (Υπόλοιπο 50 Euro)
	cout << "New Balance is: " << t1.returnBalance () << endl;

	return 0;
} //Τέλος του προγράμματος
```

Σύντομη εξήγηση του προγράμματος.

Στο πιο πάνω πρόγραμμα έχουμε φτιάξει μεθόδους με το ίδιο όνομα και στην βασική `Accounts` και στην παραγόμενη κλάση `Interest`. Αυτό το κάνουμε για να προσθέσουμε ένα τμήμα ελέγχου μέσα στις μεθόδους `withdrawal` και `deposit` της παραγόμενης κλάσης, χωρίς να επηρεαστούν οι αντίστοιχες μέθοδοι της βασικής κλάσης.
Στην `main()` φτιάχνουμε ένα αντικείμενο `t1` της παραγόμενης κλάσης `Interest` το οποίο προσπαθεί να κάνει ανάληψη αλλά και κατάθεση, δίνοντας θετικό και αρνητικό ποσό.
Στην εντολή:

```
t1.deposit(100);
```

καλείται η μέθοδος `deposit()` της παραγόμενης κλάσης, όπου και γίνεται ο έλεγχος για το αν το όρισμα (ποσό) είναι θετικός αριθμός. Εάν το όρισμα είναι θετικό τότε καλείται η μέθοδος `deposit()` της βασικής κλάσης με την εντολή:

```
Accounts::deposit(amount);
```

στην οποία μεταβιβάζουμε το όρισμα (ποσό) που δέχθηκε η μέθοδος της παράγωγης κλάσης.
Το ίδιο ισχύει και για την μέθοδο `withdrawal()`.

Παράδειγμα 2. Υπερφόρτωση μεθόδου στο παράδειγμα κληρονομικότητας κλάσεων δύο επιπέδων «Τραπεζικές συναλλαγές» της Άσκησης 8 με επιλογή μεθόδου προς εκτέλεση από την κλάση.

```
#include <iostream>
using namespace std;

//Δήλωση Βασικής Κλάσης
class Accounts{
      public:
……
……
       void printData(){
             cout << " Object of class Accounts " << endl;
             cout << "The Balance is: " << balance << endl;
       }
}; //Τέλος Βασικής Κλάσης

//Δήλωση Παραγόμενης Κλάσης
class Interest:public Accounts{
      public:
……
……
       void printData(){
             cout << " Object of class Interest " << endl;
             cout << "The Balance is: " << balance << endl;
       }
}; //Τέλος Παραγόμενης Κλάσης

int main(){
       Accounts Acc1; //Αντικείμενο της βασικής κλάσης
       Interest AI1; //Αντικείμενο της παράγωγης κλάσης

       /* Χρήση της μεθόδου της βασικής κλάσης για την
      εμφάνιση του υπολοίπου του αντικειμένου Acc1 */
       Acc1.printData();

      /* Χρήση της μεθόδου της παράγωγης κλάσης για την
      εμφάνιση του υπολοίπου του αντικειμένου AI1 */
       AI1.printData();

      return 0;
}
```

Σύντομη εξήγηση του προγράμματος.

Στο πιο πάνω πρόγραμμα έχουμε φτιάξει μία μέθοδο `printData()` με το ίδιο όνομα και στην βασική `Accounts` και στην παραγόμενη κλάση `Interest`.

Στην προκειμένη περίπτωση η λειτουργία των μεθόδων και στις δύο κλάσεις είναι η ίδια, εμφάνιση δηλαδή του υπόλοιπου του αντικειμένου που την καλεί, με την διαφορά όμως ότι κάθε φορά καλείται η μέθοδος που αντιστοιχεί στην κλάση από την οποία προέρχεται το αντικείμενο που την καλεί.

ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ ΚΑΙ ΑΡΧΕΙΑ ΕΠΙΚΕΦΑΛΙΔΑΣ

Αρχεία επικεφαλίδας έχουμε τη δυνατότητα να φτιάξουμε και με τις κλάσεις που προέρχονται από σχήματα κληρονομικότητας. Σε αυτή την περίπτωση τις κλάσεις τις χωρίζουμε από το υπόλοιπο πρόγραμμα και τις ονομάζουμε με επέκταση .hpp όπως έχουμε μάθει.
Η διαφορά εδώ είναι ότι μέσα στο αρχείο της παράγωγης κλάσης συμπεριλαμβάνουμε την βασική κλάση και στο κυρίως πρόγραμμα συμπεριλαμβάνουμε όποια κλάση εμείς επιθυμούμε να χρησιμοποιήσουμε. Αντιμετωπίζουμε δηλαδή και την βασική αλλά και τις παραγόμενες κλάσεις ως ανεξάρτητες βιβλιοθήκες συναρτήσεων.

Παράδειγμα 3. Αρχεία επικεφαλίδας στο παράδειγμα κληρονομικότητας κλάσεων δύο επιπέδων «Τραπεζικές συναλλαγές» της Άσκησης 8.

Αρχείο Accounts.hpp

```
//Δήλωση βασικής κλάσης Accounts

#include <iostream>
using namespace std;

//Δήλωση Βασικής Κλάσης
class Accounts{
……
……
}; //Τέλος Βασικής Κλάσης
```

Αρχείο Interest.hpp

```
//Δήλωση παραγόμενης κλάσης Interest

#include <iostream>
#include "Accounts.hpp" //Συμπεριλαμβάνεται η βασική κλάση

using namespace std;

//Δήλωση Παραγόμενης Κλάσης
class Interest:public Accounts{
……
……
}; //Τέλος Παραγόμενης Κλάσης
```

Κυρίως πρόγραμμα

```
//Κυρίως πρόγραμμα

#include <iostream>
#include "Interest.hpp" //Συμπεριλαμβάνεται η επιθυμητή κλάση

using namespace std;

int main(){

       Interest AI1; //Αντικείμενο της παράγωγης κλάσης

      /* Χρήση της μεθόδου της παράγωγης κλάσης για την
      εμφάνιση του υπολοίπου του αντικειμένου AI1 */
       AI1.printData();

      return 0;
}
```