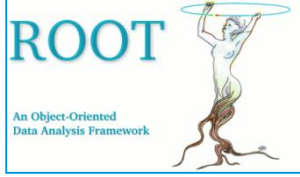
Σύνθεση και Κληρονομικότητα

- Σύνθεση (composition)
- Κληρονομικότητα (inheritance)
- Υπερφόρτωση κληρονομημένων μελών
- Εικονικές συναρτήσεις και Πολυμορφισμός
- Αφηρημένες (abstract) βασικές κλάσεις

Σύνθεση (composition)

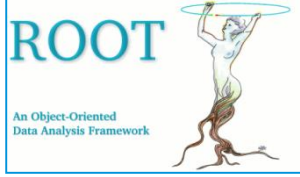


- Η σύνθεση (composition) κλάσεων αναφέρεται στη χρήση μιας ή περισσοτέρων κλάσεων στον ορισμό μιας άλλης κλάσης.
- Όταν ένα μέλος δεδομένων της νέας κλάσης είναι αντικείμενο άλλης κλάσης, λέμε ότι η νέα κλάση είναι ένα σύνθετο αντικείμενο (composite) άλλων αντικειμένων.
- Στο ακόλουθο παράδειγμα γίνεται η σύνθεση της κλάσης Date στην Person.

Αρχείο Date.h
(Περιγραφή της κλάσης Date)

```
// Date.h
#include <iostream>
using namespace std;
#include <string>

class Date
{
  friend istream& operator>>(istream&, Date&);
  friend ostream& operator<<(ostream&, const Date&);
public:
  Date(int d=0, int m=0, int y=0) : month(m), day(d), year(y) { }
  void setDate(int d, int m, int y) { day = d; month = m; year = y; }
private:
  int day, month, year;
};

istream& operator>>(istream& in, Date& x)
{
  in >> x.day >> x.month >> x.year;
  return in;
}

ostream& operator<<(ostream& out, const Date& x)
{
  static string monthName[13] = {"","January","February","March","April","May","June","July",
                                 "August","September","October","November","December"};
  out << x.day << ' ' << monthName[x.month] << ", " << x.year;
  return out;
}
```

Σύνθεση (composition)



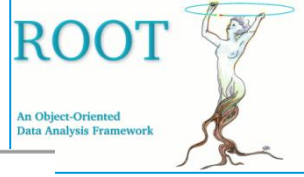
- Ελέγξτε το αρχείο Date.h
- cp Date.h testDate.cpp
- Προσθέστε την συνάρτηση main()

```
// testDate.cpp
#include <iostream>
using namespace std;
#include <string>

class Date
{
  friend istream& operator>>(istream&, Date&);
  friend ostream& operator<<(ostream&, const Date&);
public:
  Date(int d=0, int m=0, int y=0) : month(m), day(d), year(y) { }
  void setDate(int d, int m, int y) { day = d; month = m; year = y; }
private:
  int day, month, year;
};

istream& operator>>(istream& in, Date& x)
{
  in >> x.day >> x.month >> x.year;
  return in;
}

ostream& operator<<(ostream& out, const Date& x)
{
  static string monthName[13] = {"","January","February","March","April","May","June","July",
                                 "August","September","October","November","December"};
  out << x.day << ' ' << monthName[x.month] << ", " << x.year;
  return out;
}


int main (){

   Date a(15,11,2016);

   cout << a << endl;
}
```

C++ | Tab Width: 8 | Ln 37, Col 2 | INS

```
[panos@psepc26 Inheritance]$ c++ testDate.cpp
[panos@psepc26 Inheritance]$ ./a.out
15 November, 2016
```

Σύνθεση (composition)

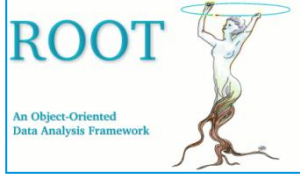



```
// Synthesi klasis Date stin Person
#include <iostream>
using namespace std;
#include "Date.h"

class Person
{
  friend ostream& operator<<(ostream&, const Person&);
  public:
    Person(string o="", string eth="", int f=0) : onoma(o), ethnikotita(eth), fylo(f) {}
    void set_im_Genisis(int d, int m, int y) {im_Genisis.setDate(d,m,y);}
    void set_im_Thanatou(int d, int m, int y) {im_Thanatou.setDate(d,m,y);}
    Date get_im_Genisis() { return im_Genisis;}
    Date get_im_Thanatou() {return im_Thanatou;}
  private:
    string onoma, ethnikotita;
    int fylo;                       // 0=Thylh, 1=Arren
    Date im_Genisis, im_Thanatou; // Imerominia Gennisis - Thanatou

};

ostream& operator<<(ostream& out, const Person& a)
{
   out << a.onoma << "   Ethnikotita:" << a.ethnikotita << "   Fylo : " << (a.fylo ? "Arren" : "Thylh");
   return out;
}

int main(){
    Person x("Ioannis Kapodistrias", "GR", 1);
    x.set_im_Genisis(11,2,1776);
    x.set_im_Thanatou(9,10,1831);

    cout << "O prwtos kyvernitis tis Elladas htan o:" << endl;
    cout << x << endl;
    cout << "Imerominia Genisis  : " << x.get_im_Genisis() << endl;
    cout << "Imerominia Thanatou : " << x.get_im_Thanatou() << endl;
}
```

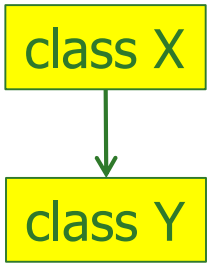
C++ | Tab Width: 8 | Ln 12, Col 1 | INS

```
[panos@psepc26 Inheritance]$ c++ sithesi.cpp
[panos@psepc26 Inheritance]$ ./a.out
O prwtos kyvernitis tis Elladas htan o:
Ioannis Kapodistrias   Ethnikotita:GR   Fylo : Arren
Imerominia Genisis  : 11 February, 1776
Imerominia Thanatou : 9 October, 1831
```

Κληρονομικότητα (inheritance)



- Ένας άλλος τρόπος επέκτασης λογισμικού είναι η δημιουργία νέου μέσω της κληρονομικότητας.
- Μέσω της κληρονομικότητας μπορούμε να δημιουργήσουμε μια νέα κλάση αντικειμένων προσδιορίζοντας μόνο τα σημεία εκείνα στα οποία αυτή διαφέρει από μια υπάρχουσα κλάση.



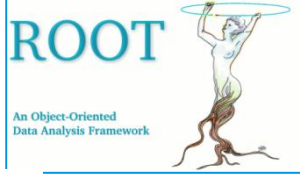

```
class Y : public X {
    // ......
};
```

- Η X ονομάζεται βασική κλάση (base class) ή υπερκλάση (superclass) και η Y ονομάζεται παράγωγη κλάση (derived class) ή δευτερεύουσα κλάση (subclass).
- Η λέξη-κλειδί public μετά την άνω-κάτω τελεία καθορίζει δημόσια κληρονομικότητα (public inheritance) που σημαίνει ότι τα δημόσια (public) μέλη της βασικής κλάσης γίνονται δημόσια μέλη της παράγωγης κλάσης.
- Στο ακόλουθο παράδειγμα η κλάση Student παράγεται από την κλάση Person.

Κληρονομικότητα (inheritance)

```
// Paragvgh thw klasis Student apo tin Person
#include <iostream>
using namespace std;
#include "Date.h"

class Person
{
  friend ostream& operator<<(ostream&, const Person&);
  public:
    Person(string o="", string eth="", int f=0) : onoma(o), ethnikotita(eth), fylo(f) {}
    void set_im_Genisis(int d, int m, int y) {im_Genisis.setDate(d,m,y);}
    void set_im_Thanatou(int d, int m, int y) {im_Thanatou.setDate(d,m,y);}
    Date get_im_Genisis() { return im_Genisis;}
    Date get_im_Thanatou() {return im_Thanatou;}
  protected:
    string onoma, ethnikotita;
    int fylo;                       // 0=Thylh, 1=Arren
    Date im_Genisis, im_Thanatou; // Imerominia Gennisis - Thanatou

};

ostream& operator<<(ostream& out, const Person& a)
{
   out << a.onoma << "   Ethnikotita:" << a.ethnikotita << "   Fylo : " << (a.fylo ? "Arren" : "Thylh");
   return out;
}


class Student : public Person
{
public:
Student(string o="", string eth="", int f=0, int v=0) : Person(o,eth,f), vathmos(v) {}
    void set_im_Engrafis(int d, int m, int y) {im_Engrafis.setDate(d,m,y);}
    Date get_im_Engrafis() { return im_Engrafis;}
    int get_vathmos() { return vathmos;}
protected:
    int AM;                     //Arithmos Mitrwou foititi
    Date im_Engrafis;           //Hmerominia Engrafis
    int vathmos;                //Vathmos mathimatos
    double mesi_vathmologia;    //Mesos oros vathmologias
};
```

Κληρονομικότητα (inheritance)

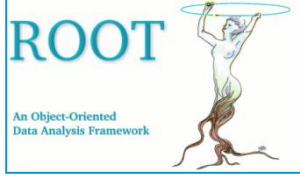



```
int main(){
    Student a("Nikolaos Papadopoulos", "GR", 1, 8);
    a.set_im_Genisis(11,2,1992);
    a.set_im_Engrafis(21,9,2010);

    cout << a << endl;
    cout << "Imerominia Genisis  : " << a.get_im_Genisis() << endl;
    cout << "Imerominia Engrafis : " << a.get_im_Engrafis() << endl;
    cout << "Vathmos sth C++     : " << a.get_vathmos() << endl;
}
```

```
[panos@psepc26 Inheritance]$ c++ klironomikotita.cpp
[panos@psepc26 Inheritance]$ ./a.out
Nikolaos Papadopoulos   Ethnikotita:GR   Fylo : Arren
Imerominia Genisis  : 11 February, 1992
Imerominia Engrafis : 21 September, 2010
Vathmos sth C++     : 8
```

- Στο παράδειγμα παρατηρούμε πως τα μέλη onoma, ethnikotita, im_Genisis, im_Thanatou και fylo της κλάσης Person έχουν δηλωθεί ως protected. Αυτό γίνεται ώστε τα μέλη αυτά να μπορούν να προσπελαστούν από την κλάση Student.
- Η κατηγορία πρόσβασης protected αποτελεί μια εξισορρόπηση μεταξύ των κατηγοριών private και public. Τα προστατευόμενα (protected) μέλη είναι προσπελάσιμα από το εσωτερικό της ίδιας της κλάσης και τις παράγωγες κλάσεις.
- Μια δευτερεύουσα κλάση κληρονομεί όλα τα δημόσια και προστατευόμενα μέλη της βασικής κλάσης.

Υπερφόρτωση κληρονομημένων μελών



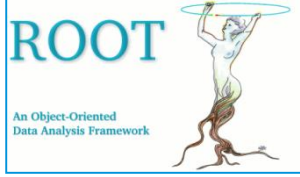
- Στο παρακάτω παράδειγμα η κλάση Y είναι δευτερεύουσα κλάση της X. Όπως παρατηρούμε στην κλάση Y ορίζονται η συνάρτηση-μέλος ff() και το μέλος a τα οποία υπερφορτώνουν (override) τα αντίστοιχα της κλάσης X.
- Στα σχήματα φαίνεται πως μπορούμε μέσω ενός αντικειμένου της κλάσης Y να προσπελάσουμε την ff() και το a τα οποία έχουν ορισθεί στην κλάση X.

```
class X
  { public:
      void ff() { cout << "X::ff()\n"; }
      int a;
  };
class Y : public X
  { public:
      void ff() { cout << "Y::ff()\n"; }
      int a;
  };
```

```
Y y;
y.ff();
y.a;
```

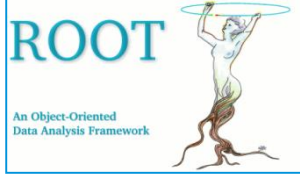
```
Y y;
y.X::ff();
y.X::a;
```

Άσκηση – Circle Shapes




```
#include<iostream>
#include<fstream>
#include<cmath>
#define PI 3.141593
using namespace std;

class Circle_based_shape
{
public:
  Circle_based_shape(double r=0) : aktina(r) {}
  ~Circle_based_shape() {}
  double emvado(){return PI*pow(aktina,2);}
  double perimetros(){return 2.*PI*aktina;}
protected:
  double aktina;
};


class Sphere : public Circle_based_shape
{
public:
  Sphere(double r=0) : Circle_based_shape(r) {}
  ~Sphere(){}
  double emvado(){return 4.*PI*pow(aktina,2);}
  double ogos(){return 4./3.*PI*pow(aktina,3);}
};


class Cylinder : public Circle_based_shape
{
public:
  Cylinder(double r=0, double h=0) : Circle_based_shape(r), ypsos(h) {}
  ~Cylinder(){}
  double emvado(){return 2.*PI*pow(aktina,2)+ 2.*PI*aktina*ypsos;}
  double ogos(){return PI*pow(aktina,2)*ypsos;}
protected:
  double ypsos;
};
```

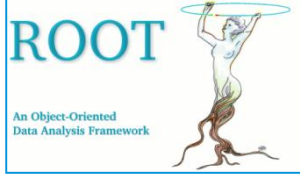
Άσκηση – Circle Shapes



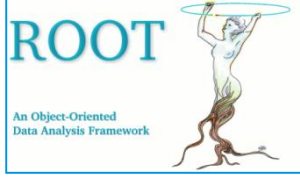

```
int main()
{
  Sphere a(10);
  Cylinder b(5,10);

  cout<<"Emvado Sfairas="<<a.emvado()<<endl;
  cout<<"Ogos Sfairas  ="<<a.ogos()<<endl;

  cout<<"Emvado Kylindrou="<<b.emvado()<<endl;
  cout<<"Ogos Kylindrou  ="<<b.ogos()<<endl;

}
```

```
[panos@psepc26 c++]$ ./a.out
Emvado Sfairas=1256.64
Ogos Sfairas  =4188.79
Emvado Kylindrou=471.239
Ogos Kylindrou  =785.398
```


THE
C++
PROGRAMMING
LANGUAGE
THIRD EDITION


ROOT
An Object-Oriented
Data Analysis Framework

Εικονικές συναρτήσεις και Πολυμορφισμός



- Ένα από τα πλέον ισχυρά χαρακτηριστικά της C++ είναι ότι επιτρέπει σε αντικείμενα διαφορετικού τύπου να ανταποκρίνονται διαφορετικά στην ίδια κλήση συνάρτησης. Αυτό ονομάζεται πολυμορφισμός (polymorphism) και επιτυγχάνεται μέσω των εικονικών συναρτήσεων (virtual functions).
- Ο πολυμορφισμός καθίσταται δυνατός από το γεγονός ότι ένας δείκτης προς ένα στιγμιότυπο βασικής κλάσης μπορεί επίσης να δείχνει και σε στιγμιότυπο οποιασδήποτε δευτερεύουσας κλάσης:

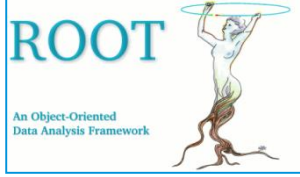
```
class X
   { // ……
   };
class Y : public X
   { // ……
   };
int main(){
   X* p;
   Y y;
   p = &y;
   ……
}
```

Η Y είναι δευτερεύουσα κλάση της X

p δείκτης προς αντικείμενα της βασικής κλάσης X

p μπορεί να δείχνει και σε αντικείμενα της δευτερεύουσας κλάσης Y

Εικονικές συναρτήσεις και Πολυμορφισμός



- Στα προγράμματα επίδειξης που ακολουθούν, ο p δηλώνεται ως δείκτης προς αντικείμενα της βασικής κλάσης X. Αριστερά γίνονται δύο κλήσεις της συνάρτησης p->f() οι οποίες καλούν την έκδοση της f() που έχει οριστεί στη βασική κλάση X.
- Δεξιά η δεύτερη κλήση p->f() καλεί την συνάρτηση Y::f() αντί για τη X::f(). Αυτό συμβαίνει γιατί μετασχηματίσαμε την X::f() σε εικονική (virtual) συνάρτηση.

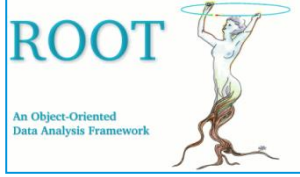
```
// Xrisi eikonikwn synarthsewn

#include <iostream>
using namespace std;

class X
{
public:
  void f() { cout << "X::f() executing\n"; }
};

class Y : public X
{
public:
  void f() { cout << "Y::f() executing\n"; }
};

int main()
{
  X x;
  Y y;
  X* p = &x;
  p->f();       // invokes X::f() because p has type X*
  p = &y;
  p->f();       // invokes X::f() because p has type X*
}
```

```
[panos@pc-247 Cpp]$ c++ vitrual_a.cpp
[panos@pc-247 Cpp]$ a.out
X::f() executing
X::f() executing
[panos@pc-247 Cpp]$
```

```
// Xrisi eikonikwn synartisewn

#include <iostream>
using namespace std;

class X
{
public:
  virtual void f() { cout << "X::f() executing\n"; }
};

class Y : public X
{
public:
  void f() { cout << "Y::f() executing\n"; }
};

int main()
{
  X x;
  Y y;
  X* p = &x;
  p->f();       // invokes X::f() because p has type X*
  p = &y;
  p->f();       // invokes X::f() because p has type X*
}
```

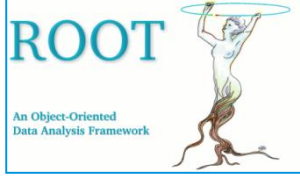
```
[panos@pc-247 Cpp]$ c++ vitrual_b.cpp
[panos@pc-247 Cpp]$ a.out
X::f() executing
Y::f() executing
[panos@pc-247 Cpp]$
```

Εικονικές συναρτήσεις και Πολυμορφισμός

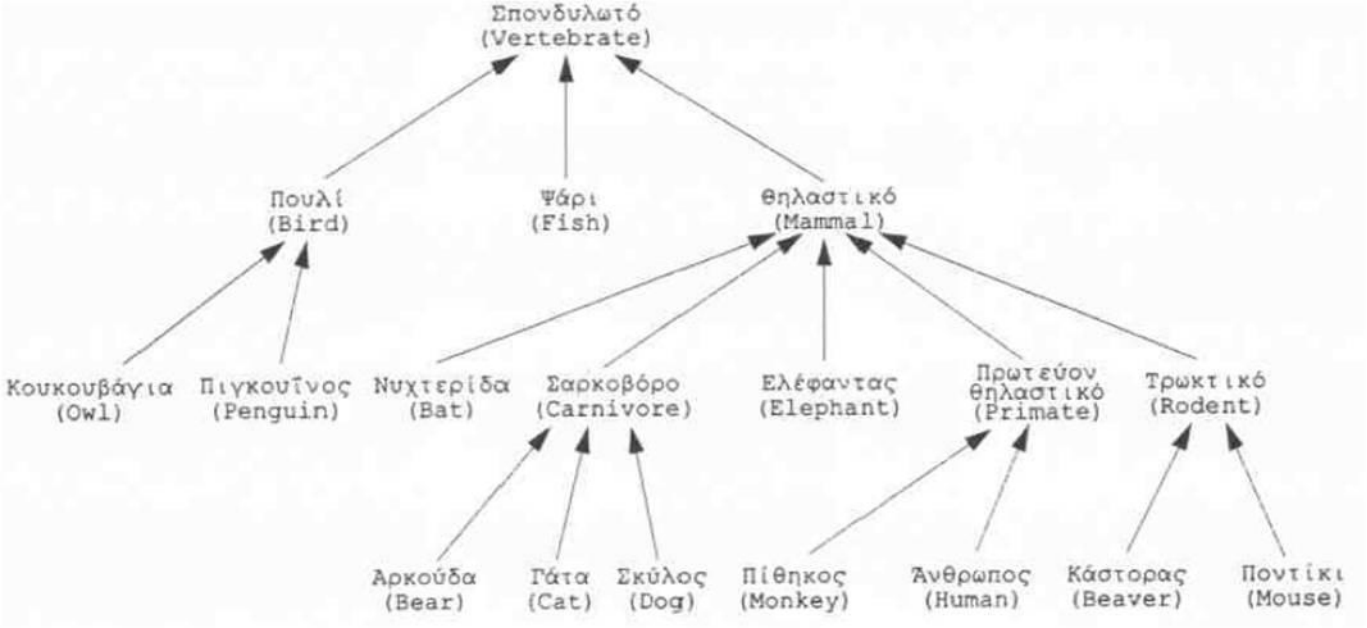


- Το προηγούμενο παράδειγμα παρουσιάζει τον πολυμορφισμό : η ίδια κλήση p->f() έχει αποτέλεσμα την κλήση διαφορετικών συναρτήσεων. Η συνάρτηση επιλέγεται ανάλογα με την κλάση στην οποία δείχνει ο δείκτης p.
- Αυτό ονομάζεται *δυναμική δέσμευση (dynamic binding)* επειδή η συσχέτιση (δηλαδή, η δέσμευση) της κλήσης με τον πραγματικό κώδικα που θα εκτελεστεί αναβάλλεται μέχρι τον χρόνο εκτέλεσης.
- Ο κανόνας που λέει ότι ο στατικά καθορισμένος τύπος του δείκτη προσδιορίζει ποια συνάρτηση-μέλος θα κληθεί, υποσκελίζεται με τη δήλωση της συνάρτησης μέλους ως εικονικής συνάρτησης (virtual function).
- Γενικά, μια συνάρτηση-μέλος πρέπει να δηλώνεται ως εικονική όταν αναμένουμε ότι τουλάχιστον μία από τις δευτερεύουσες κλάσεις της θα ορίσει τη δική της έκδοση της συνάρτησης.

Αφηρημένες (abstract) βασικές κλάσεις

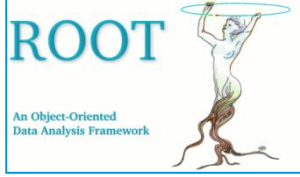


- Ένα καλά σχεδιασμένο αντικειμενοστραφές πρόγραμμα περιλαμβάνει μια ιεραρχία κλάσεων που οι σχέσεις μεταξύ τους μπορούν να περιγραφούν από ένα δενδρικό διάγραμμα όπως το επόμενο.



- Οι κλάσεις στα φύλλα αυτού του δένδρου (πχ. Owl, Fish Dog) περιλαμβάνουν ειδικές συναρτήσεις που υλοποιούν την συμπεριφορά των αντιστοίχων κλάσεων (πχ. Fish.swim(), Owl.fly(), Dog.dig()).
- Μερικές από αυτές τις συναρτήσεις μπορεί να είναι κοινές για όλες τις δευτερεύουσες κλάσεις μιας κλάσης (πχ. Vertebrate.eat(), Mammal.sucle() κτλ.)
- Τέτοιες συναρτήσεις είναι πιθανό να δηλωθούν ως εικονικές στις βασικές κλάσεις, και στη συνέχεια να υποσκελιστούν από τις δευτερεύουσες κλάσεις τους για ειδικές υλοποιήσεις.

Αφηρημένες (abstract) βασικές κλάσεις



- Αν μια εικονική συνάρτηση είναι βέβαιο ότι θα υποσκελιστεί σε όλες τις δευτερεύουσες κλάσεις της, τότε δεν είναι αναγκαίο να υλοποιηθεί στη βασική της κλάση.
- Σε αυτή την περίπτωση έχουμε μια *γνήσια εικονική συνάρτηση-μέλος* στην οποία αποδίδουμε την αρχική τιμή "=0;" στη θέση του σώματος της συνάρτησης, όπως εδώ:

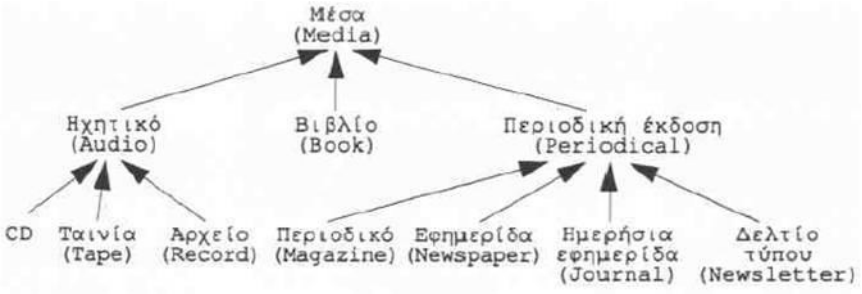
```
virtual int f() = 0;
```

- Για παράδειγμα, στην παραπάνω κλάση Vertebrate μπορεί να αποφασίσουμε ότι η συνάρτηση eat() θα υποσκελιστεί σε όλες τις δευτερεύουσες κλάσεις της, και έτσι τη δηλώνουμε ως γνήσια εικονική συνάρτηση στη βασική κλάση Vertebrate:

```
class Vertebrate
    { public:
            virtual void eat()=0;    // γνήσια εικονική συνάρτηση
    };
class Fish : public Vertebrate
    { public:
            void eat();    // υλοποιείται ειδικά για την κλάση Fish
    };
```

- Αφηρημένη (abstract) βασική κλάση είναι η κλάση που έχει μία η περισσότερες γνήσιες εικονικές συναρτήσεις-μέλη (πχ. η κλάση Vertebrate).
- Συμπαγής παράγωγη κλάση είναι η κλάση που δεν έχει γνήσιες εικονικές συναρτήσεις-μέλη (πχ. η κλάση Fish).
- Δεν μπορούν να δημιουργηθούν στιγμιότυπα των αφηρημένων βασικών κλάσεων.

Παράδειγμα: Μια ιεραρχία κλάσεων αποθηκευτικών μέσων




```cpp
// Ierarxia klasewn apothikeytikwn mesw

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

class Media
{
public:
  virtual void print() =0;
  virtual string id() =0;
protected:
  string title;
};

class Book : Media
{
public:
  Book(string a="", string t="", string p="", string i="")
    : author(a), publisher(p), isbn(i) { title = t; }
  void print() { cout << title << " by " << author << endl; }
  string id() { return isbn; }
private:
  string author, publisher, isbn;
};

class CD : Media
{
public:
  CD(string t="", string c="", string m="", string n="")
    : composer(c), make(m), number(n) { title = t; }
  void print() { cout << title << ", " << composer << endl; }
  string id() { return make + "" + number; }
private:
  string composer, make, number;
};
```

```cpp
class Magazine : Media
{
public:
  Magazine(string t="", string i="", int v=0,  int n=0)
    : issn(i), volume(v), number(n) { title = t; }
  void print()
  { cout << title << " Magazine, Vol. " << volume
         << ", No. " << number << endl;
  }
  string id() { return issn; }
private:
  string issn, publisher;
  int volume, number;
};

int main()
{
  Book book("Bjarne Stroustrup", "The C++ Programming Language",
            "Addison-Wesley", "0-201-53992-6");
  Magazine magazine("TIME", "0040-781X", 145, 23);
  CD cd("BACH CANTANAS", "Johann Sabastian Bach", "ARCHIV", "D120541");
  book.print();
  cout << "\tid: " << book.id() << endl;
  magazine.print();
  cout << "\tid: " << magazine.id() << endl;
  cd.print();
  cout << "\tid: " << cd.id() << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ ierarxia.cpp
[panos@pc-247 Cpp]$ a.out
The C++ Programming Language by Bjarne Stroustrup
        id: 0-201-53992-6
TIME Magazine, Vol. 145, No. 23
        id: 0040-781X
BACH CANTANAS, Johann Sabastian Bach
        id: ARCHIVD120541
[panos@pc-247 Cpp]$
```