Πρώτη γνωριμία με τον αντικειμενοστραφή προγραμματισμό

Η μετάβαση από τον δομημένο προγραμματισμό στον αντικειμενοστραφή

Δομημένος Προγραμματισμός

Ο θεμέλιος λίθος του ***δομημένου προγραμματισμού*** είναι ο τεμαχισμός μεγάλων προγραμμάτων σε επιμέρους κομμάτια (*μονάδες λογισμικού*), όπου το καθένα έχει συγκεκριμένη αποστολή, αλλά συνδέεται και επικοινωνεί με τα άλλα για να δημιουργήσουν ένα ολοκληρωμένο πρόγραμμα.

Μειονεκτήματα Δομημένου Προγραμματισμού

- ✓ Η πολυπλοκότητα.
- ✓ Τα δεδομένα έχουν υποεκτιμηθεί.
- ✓ Η τοπικότητα των μεταβλητών
  - ➢ Υπάρχουν οι καθολικές μεταβλητές αλλά τότε αυξάνει η πολυπλοκότητα του προγράμματος.
- ✓ Αυξημένη πιθανότητα λάθους.
- ✓ Αυξημένη πολυπλοκότητα και στη διαχείριση των δεδομένων.
- ✓ Η δημιουργία νέων τύπων δεδομένων είναι δύσκολη.

Αντικειμενοστραφής Προγραμματισμός

Στον Αντικειμενοστραφή Προγραμματισμό, δίνεται έμφαση στη δημιουργία μονάδων οι οποίες θα περιλαμβάνουν τα δεδομένα αλλά και τις εντολές που τα διαχειρίζονται.

Ορισμοί (1)

- ✓ Ο κόσμος αποτελείται από αντικείμενα απλά ή σύνθετα, που αλληλεπιδρούν μεταξύ τους.
- ✓ Για κάθε αντικείμενο του φυσικού κόσμου μπορούμε να ορίσουμε μια αφηρημένη έννοια που περιγράφει:
  - ➢ τον τρόπο με τον οποίο το αντικείμενο επικοινωνεί με το περιβάλλον του
  - ➢ το πώς αυτό είναι οργανωμένο εσωτερικά
- ✓ Ο αντικειμενοστραφής προγραμματισμός ονομάζει στιγμιότυπα (instances) τα πραγματικά αντικείμενα και κλάσεις (classes) τις αφηρημένες έννοιες που τα περιγράφουν.

Ορισμοί (2)

- ***Κλάση (Class):*** Είναι μια συλλογή από αντικείμενα, που μοιράζονται τα ίδια χαρακτηριστικά και επιδρούν με το σύστημα με τον ίδιο τρόπο. Τα χαρακτηριστικά και η επίδραση ορίζονται για τις κλάσεις.

- ***Αντικείμενο (Object):*** Είναι μέλος μιας κλάσης. Αν και ο ορισμός γίνεται σε επίπεδο κλάσης, η πραγματική επίδραση συμβαίνει με ανεξάρτητα αντικείμενα.

Ορισμοί (3)

- ✓ Ο όρος αντικείμενο χρησιμοποιείται και για την αφηρημένη έννοια (κλάση) αλλά και για τα στιγμιότυπά της.
- ✓ Ο αντικειμενοστραφής προγραμματισμός θεωρεί όλες τις οντότητες πραγματικές και αφηρημένα τα αντικείμενα(objects).
- ✓ Εδώ θα ονομάζουμε την αφηρημένη έννοια κλάση και τα στιγμιότυπα της αντικείμενα.
- ✓ Τα αντικείμενα είναι σαν ***μεταβλητές*** και οι κλάσεις σαν ***τύποι***.

Κλάσεις

Καθηγητής
Αρ. Ταυτότητας Όνομα Επώνυμο Διεύθυνση Τηλέφωνο
Προσθήκη Καθηγητή() Διαγραφή Καθηγητή() Μεταβολή στοιχείων Καθηγητή()

Μάθημα
Κωδικός μαθήματος Θεματική ενότητα Τίτλος Διδάσκων
Προσθήκη Μαθήματος() Διαγραφή Μαθήματος() Μεταβολή στοιχείων Μαθήματος() Ανάθεση Μαθήματος()

Αντικείμενα ή Στιγμιότυπα

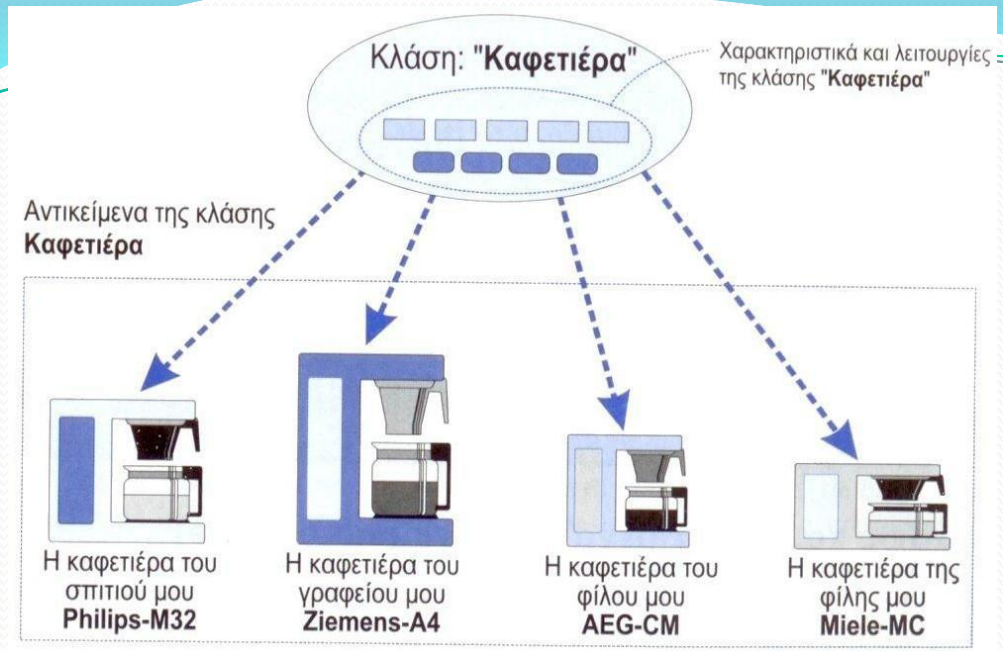
Καθηγητής 01
Αρ. Ταυτότητας: **Α123456** Όνομα: **Βασίλειος** Επώνυμο: **Βασιλείου** Διεύθυνση: **Αγ. Βασιλείου 1** Τηλέφωνο: **9876543**
Προσθήκη Καθηγητή() Διαγραφή Καθηγητή() Μεταβολή στοιχείων Καθηγητή()

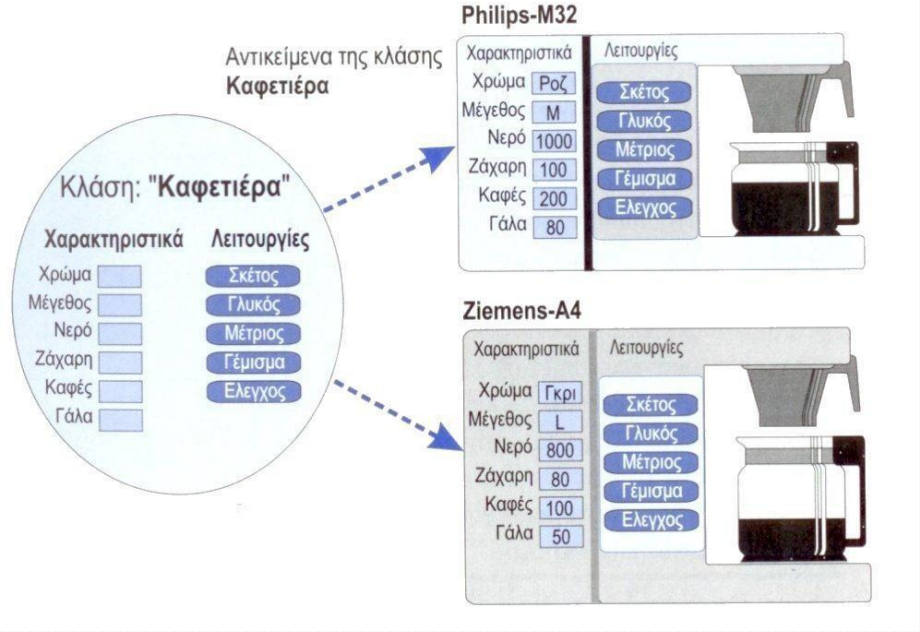
Καθηγητής 02
Αρ. Ταυτότητας: **Β987654** Όνομα: **Γεώργιος** Επώνυμο: **Γεωργίου** Διεύθυνση: **Αγ. Γεωργίου 1** Τηλέφωνο: **7654321**
Προσθήκη Καθηγητή() Διαγραφή Καθηγητή() Μεταβολή στοιχείων Καθηγητή()

Μάθημα 01
Κωδικός μαθήματος: **ΠΛ-034** Θεματική ενότητα: **Πληροφορική** Τίτλος: **Προγραμματισμός Η/Υ** Διδάσκων: **Β987654**
Προσθήκη Μαθήματος() Διαγραφή Μαθήματος() Μεταβολή στοιχείων Μαθήματος() Ανάθεση Μαθήματος()

Κλάση/Αντικείμενο

- ✓ Στον ΑΠ δεν ασχολούμαστε με το πώς θα χωρίσουμε το πρόβλημα σε συναρτήσεις αλλά σε κλάσεις και αντικείμενα.
- ✓ Μία κλάση μπορεί να έχει ένα ή περισσότερα αντικείμενα.
- ✓ Κάθε αντικείμενο μπορεί να περιέχει μία ή περισσότερες συναρτήσεις & συναρτήσεις-μέλη.
- ✓ Η κλάση λειτουργεί ως πρότυπο: δηλώνουμε σε αυτή δεδομένα και συναρτήσεις που θα έχουν τα αντικείμενα της.
- ✓ ‘ Όταν δημιουργήσουμε αντικείμενα θα έχουν ό,τι έχει η κλάση που ανήκουν.


Κλάση: "Καφετιέρα"
Χαρακτηριστικά και λειτουργίες της κλάσης "Καφετιέρα"
Αντικείμενα της κλάσης
Καφετιέρα
Η καφετιέρα του σπιτιού μου
Philips-M32
Η καφετιέρα του γραφείου μου
Ziemens-A4
Η καφετιέρα του φίλου μου
AEG-CM
Η καφετιέρα της φίλης μου
Miele-MC


Αντικείμενα της κλάσης
Καφετιέρα
Κλάση: "Καφετιέρα"
Χαρακτηριστικά
Λειτουργίες
Χρώμα
Μέγεθος
Νερό
Ζάχαρη
Καφές
Γάλα
Σκέτος
Γλυκός
Μέτριος
Γέμισμα
Ελεγχος
Philips-M32
Χαρακτηριστικά
Λειτουργίες
Χρώμα Ροζ
Μέγεθος M
Νερό 1000
Ζάχαρη 100
Καφές 200
Γάλα 80
Σκέτος
Γλυκός
Μέτριος
Γέμισμα
Ελεγχος
Ziemens-A4
Χαρακτηριστικά
Λειτουργίες
Χρώμα Γκρι
Μέγεθος L
Νερό 800
Ζάχαρη 80
Καφές 100
Γάλα 50
Σκέτος
Γλυκός
Μέτριος
Γέμισμα
Ελεγχος

Συναρτήσεις

- Στον προγραμματισμό ο κώδικας για μια συγκεκριμένη ενέργεια ονομάζεται συνάρτηση.
- Στον ΑΠ την αλληλεπίδραση μεταξύ αντικειμένων και του έξω κόσμου την χειρίζονται οι συναρτήσεις. Αυτό ονομάζεται συμπεριφορά.
- Μια αντικειμενοστραφής συνάρτηση είναι παρόμοια με μια συνηθισμένη συνάρτηση αλλά γενικά οι αντικειμενοστραφείς συναρτήσεις είναι μικρότερες και απλούστερες.

Βασικά χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού

Ενθυλάκωση

- Ορίζει τον βαθμό προσπελασιμότητας που μια κλάση επιτρέπει σε άλλες κλάσεις. Δίνει τη δυνατότητα σε γλώσσες αντικειμενοστραφούς προγραμματισμού να ομαδοποιούν και να αποκρύπτουν δεδομένα και διαδικασίες των αντικειμένων.
- Κάποιες πληροφορίες είναι προσπελάσιμες μόνο μέσα σε κάποια κλάση.
- Κάποιες πληροφορίες είναι προσπελάσιμες γενικά.
- Κάποιες πληροφορίες είναι προσπελάσιμες έξω από την κλάση αλλά μόνο σε συγγενείς κλάσεις.
- Στόχος του ΑΠ είναι να κρατάει τις πληροφορίες όσο γίνεται πιο ιδιωτικές.

Philips-M32

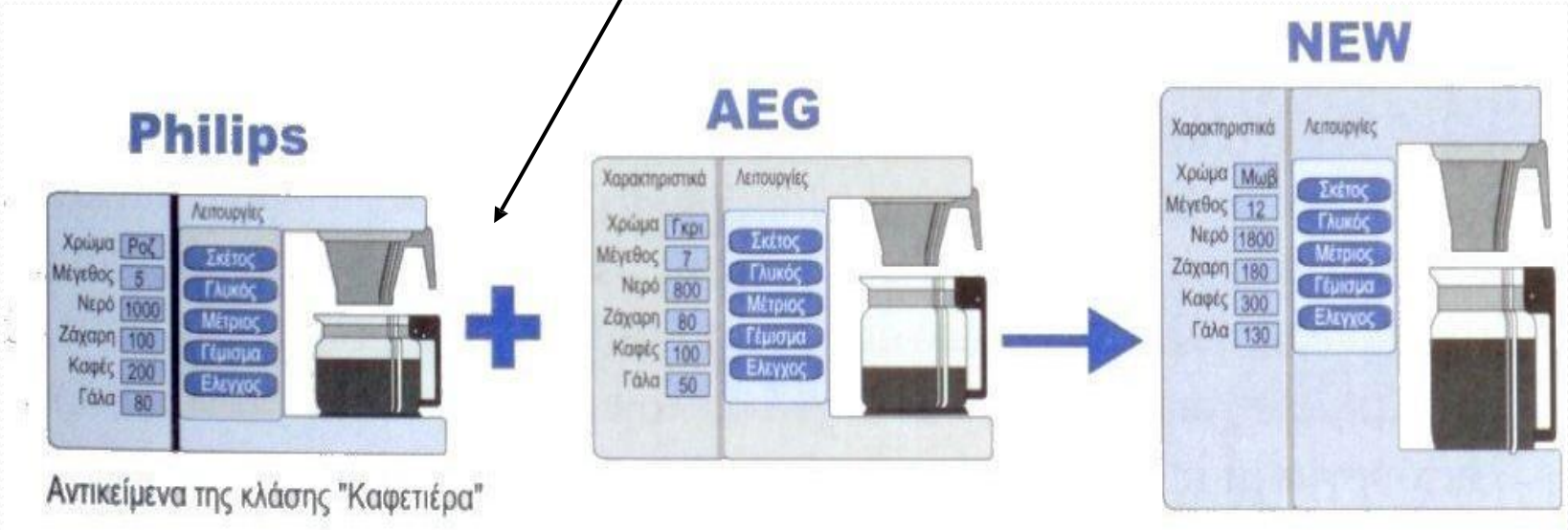
Υπάρχει πρόσβαση σε όλα τα χαρακτηριστικά και τις λειτουργίες του αντικειμένου

Philips-M32

Πολυμορφισμός

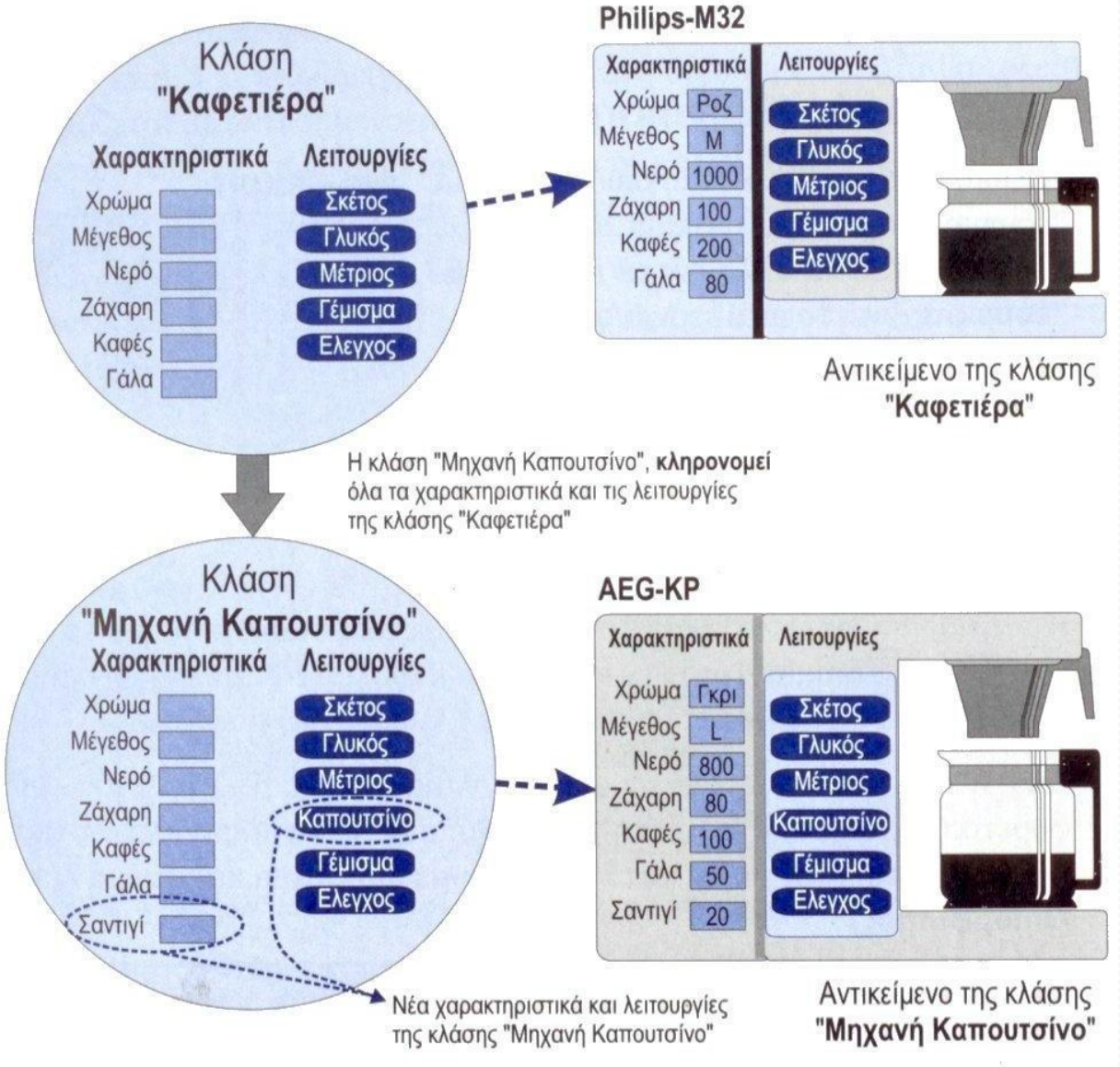
- Αναφέρεται στην αλληλεπίδραση μεταξύ αντικειμένων. Τα αντικείμενα σχετίζονται μέσα στο σύστημα και με τον έξω κόσμο μέσω των ενεργειών.
- Η ίδια ενέργεια μπορεί να αναπαρασταθεί με μια σειρά από διαφορετικούς τρόπους.
- Είναι χαρακτηριστικό των ενεργειών και όχι των αντικειμένων.
- Παρέχεται από τις αντικειμενοστραφείς γλώσσες στα αντικείμενα ώστε να συμπεριφέρονται διαφορετικά ανάλογα με τον τρόπο με τον οποίο χρησιμοποιούνται.

Υπερφόρτωση του τελεστή +: Χαρακτηριστικό του πολυμορφισμού.



Κληρονομικότητα

- Αντίστοιχο με το πως κάποια χαρακτηριστικά περνούν από τους γονείς στα παιδιά.
- Στον ΑΠ, οι κλάσεις μπορούν να ομαδοποιηθούν σε ιεραρχίες, όπου τα κατώτερα επίπεδα της ιεραρχίας (τα παιδιά) μοιράζονται τα ίδια χαρακτηριστικά με τα ανώτερα επίπεδα (γονείς).
- Είναι η δυνατότητα παραγωγής μιας νέας κλάσης από μια υπάρχουσα (βασική). Η νέα κλάση κληρονομεί όλα τα χαρακτηριστικά και τις λειτουργίες της βασικής κλάσης, αλλά ταυτόχρονα μπορεί να ορίσει τα δικά της επιπλέον χαρακτηριστικά και λειτουργίες.


Κλάση
"Καφετιέρα"
Χαρακτηριστικά
Λειτουργίες
Χρώμα
Μέγεθος
Νερό
Ζάχαρη
Καφές
Γάλα
Σκέτος
Γλυκός
Μέτριος
Γέμισμα
Ελεγχος
Philips-M32
Χαρακτηριστικά
Λειτουργίες
Χρώμα Ροζ
Μέγεθος M
Νερό 1000
Ζάχαρη 100
Καφές 200
Γάλα 80
Σκέτος
Γλυκός
Μέτριος
Γέμισμα
Ελεγχος
Αντικείμενο της κλάσης
"Καφετιέρα"
Η κλάση "Μηχανή Καπουτσίνο", κληρονομεί
όλα τα χαρακτηριστικά και τις λειτουργίες
της κλάσης "Καφετιέρα"
Κλάση
"Μηχανή Καπουτσίνο"
Χαρακτηριστικά
Λειτουργίες
Χρώμα
Μέγεθος
Νερό
Ζάχαρη
Καφές
Γάλα
Σαντιγί
Σκέτος
Γλυκός
Μέτριος
Καπουτσίνο
Γέμισμα
Ελεγχος
AEG-KP
Χαρακτηριστικά
Λειτουργίες
Χρώμα Γκρι
Μέγεθος L
Νερό 800
Ζάχαρη 80
Καφές 100
Γάλα 50
Σαντιγί 20
Σκέτος
Γλυκός
Μέτριος
Καπουτσίνο
Γέμισμα
Ελεγχος
Νέα χαρακτηριστικά και λειτουργίες
της κλάσης "Μηχανή Καπουτσίνο"
Αντικείμενο της κλάσης
"Μηχανή Καπουτσίνο"

Δημιουργία αντικειμενοστραφούς κώδικα στη C++

Δήλωση κλάσης

(παράδειγμα: Ratio.h)

C++ / ROOT

```
1  // αρχείο Ratio.h
2  #pragma once
3
4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
7      Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int  getDenominator();  //   πρόσβαση
12     void  setNumerator(int);  //  πρόσβαση
13     void setDenominator(int); // πρόσβαση
14     double toDouble();      // πραγματική τιμή
15     void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Μία κλάση δηλώνεται στη C++ με τη λέξη `class` και το όνομά της (π.χ. στη γραμμή 4: Ratio). Συνηθίζεται το όνομα μίας κλάσης να ξεκινά με κεφαλαίο γράμμα.

Δήλωση κλάσης

(παράδειγμα: Ratio.h)

C++ / ROOT

```
1  // αρχείο Ratio.h
2  #pragma once
3
4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
7      Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int  getDenominator();   //   πρόσβαση
12     void  setNumerator(int);  //  πρόσβαση
13     void setDenominator(int); // πρόσβαση
       double toDouble();      // πραγματική τιμή
15     void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Μία κλάση δηλώνεται στη C++ με τη λέξη `class` και το όνομά της (π.χ. στη γραμμή 4: Ratio). Συνηθίζεται το όνομα μίας κλάσης να ξεκινά με κεφαλαίο γράμμα.
- Ακολουθούν οι δηλώσεις των πεδίων που αφορούν τη δομή της κλάσης (γραμμές 5-19): Περιλαμβάνονται μέθοδοι (συναρτήσεις) κατασκευής και καταστροφής, μέθοδοι πρόσβασης, άλλες μέθοδοι και δεδομένα (απλών τύπων ή και άλλων αντικειμένων).

Δήλωση κλάσης

(παράδειγμα: Ratio.h)

C++ / ROOT

```
1  // αρχείο Ratio.h
2  #pragma once
3
4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
7      Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int  getDenominator();  //   πρόσβαση
12     void  setNumerator(int);  //  πρόσβαση
13     void setDenominator(int); // πρόσβαση
       double toDouble();      // πραγματική τιμή
15     void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Μία κλάση δηλώνεται στη C++ με τη λέξη `class` και το όνομά της (π.χ. στη γραμμή 4: Ratio). Συνηθίζεται το όνομα μίας κλάσης να ξεκινά με κεφαλαίο γράμμα.
- Ακολουθούν οι δηλώσεις των πεδίων που αφορούν τη δομή της κλάσης (γραμμές 5-19): Περιλαμβάνονται μέθοδοι (συναρτήσεις) κατασκευής και καταστροφής, μέθοδοι πρόσβασης, άλλες μέθοδοι και δεδομένα (απλών τύπων ή και άλλων αντικειμένων).
- Τα πεδία της κλάσης δηλώνονται εντός `{ };`

Προσοχή στο τελικό ερωτηματικό…

Δήλωση κλάσης

(παράδειγμα: Ratio.h)

C++ / ROOT

```
1  // αρχείο Ratio.h
2  #pragma once
3
4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
       Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int  getDenominator();  //   πρόσβαση
12     void  setNumerator(int);  //  πρόσβαση
13     void setDenominator(int); // πρόσβαση
       double toDouble();      // πραγματική τιμή
15     void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Μία κλάση δηλώνεται στη C++ με τη λέξη `class` και το όνομά της (π.χ. στη γραμμή 4: Ratio). Συνηθίζεται το όνομα μίας κλάσης να ξεκινά με κεφαλαίο γράμμα.
- Ακολουθούν οι δηλώσεις των πεδίων που αφορούν τη δομή της κλάσης (γραμμές 5-19): Περιλαμβάνονται μέθοδοι (συναρτήσεις) κατασκευής και καταστροφής, μέθοδοι πρόσβασης, άλλες μέθοδοι και δεδομένα (απλών τύπων ή και άλλων αντικειμένων).
- Τα πεδία της κλάσης δηλώνονται εντός `{ };`

  Προσοχή στο τελικό ερωτηματικό…
- Η οδηγία `#pragma once` (γραμμή 2) προς τον προ-μεταγλωττιστή, εξασφαλίζει ότι το περιεχόμενο του αρχείου θα διαβαστεί μόνο μία φορά κατά τη μεταγλώττιση, ακόμη κι αν αυτό κληθεί περισσότερες φορές με εντολές `#include "Ratio.h"`.

Δήλωση κλάσης

(παράδειγμα: Ratio.h)

C++ / ROOT

```
1  // αρχείο Ratio.h

4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
       Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int  getDenominator();    //  πρόσβαση
12     void  setNumerator(int);  //  πρόσβαση
13     void setDenominator(int); // πρόσβαση
       double toDouble();      // πραγματική τιμή
15     void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Οι δεσμευμένες λέξεις/ετικέτες *public* και *private* χρησιμοποιούνται για τον καθορισμό της πρόσβασης των διαφόρων πεδίων, από τους χρήστες της κλάσης.

Δήλωση κλάσης

(παράδειγμα: Ratio.h)

C++ / ROOT

```
1  // αρχείο Ratio.h

4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
       Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int  getDenominator();     //  πρόσβαση
12     void  setNumerator(int);   //  πρόσβαση
13     void setDenominator(int); // πρόσβαση
       double toDouble();      // πραγματική τιμή
15     void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Οι δεσμευμένες λέξεις/ετικέτες *public* και *private* χρησιμοποιούνται για τον καθορισμό της κληρονομικότητας των διαφόρων πεδίων, από τους χρήστες της κλάσης.
- public:
  Ό,τι δηλώνεται ως κοινόχρηστο (public), είναι ορατό/προσβάσιμο από τον χρήστη της κλάσης.
  Το τμήμα αυτό αποτελεί τη διεπαφή (interface) της κλάσης.

Δήλωση κλάσης

(παράδειγμα: Ratio.h)

C++ / ROOT

```
1  // αρχείο Ratio.h
4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
       Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int  getDenominator();     //  πρόσβαση
12     void  setNumerator(int);   //  πρόσβαση
13     void setDenominator(int); // πρόσβαση
       double toDouble();      // πραγματική τιμή
15     void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Οι δεσμευμένες λέξεις/ετικέτες *public* και *private* χρησιμοποιούνται για τον καθορισμό της κληρονομικότητας των διαφόρων πεδίων, από τους χρήστες της κλάσης.
- public:
  - Ό,τι δηλώνεται ως κοινόχρηστο (public), είναι ορατό/προσβάσιμο από τον χρήστη της κλάσης.
  - Το τμήμα αυτό αποτελεί τη διεπαφή
  - (interface) της κλάσης.
- private:
  - Ό,τι δηλώνεται ως ιδιωτικό (private), δεν είναι προσβάσιμο από τον χρήστη της κλάσης.
  - Στο διπλανό παράδειγμα, τα εσωτερικά
  - δεδομένα της κλάσης μας (ο αριθμητής και ο παρανομαστής) είναι ιδιωτικά.

Δήλωση κλάσης

(παράδειγμα: Ratio.h)

C++ / ROOT

```
1  // αρχείο Ratio.h

4  class Ratio {
5    public:
6      Ratio();                 // κατασκευή
       Ratio(int);              // κατασκευή
8      Ratio(int,int);          // κατασκευή
9      ~Ratio();                // καταστροφή
10     int getNumerator();          // πρόσβαση
11     int  getDenominator();     //  πρόσβαση
12     void  setNumerator(int);   //  πρόσβαση
13     void setDenominator(int); // πρόσβαση
       double toDouble();       // πραγματική τιμή
15     void invert();           // αντιστροφή
16     void print();            // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

Κανόνας:
Όλα τα δεδομένα μίας κλάσης πρέπει να είναι ιδιωτικά (private).

- Οι δεσμευμένες λέξεις/ετικέτες *public* και *private* χρησιμοποιούνται για τον καθορισμό της κληρονομικότητας των διαφόρων πεδίων, από τους χρήστες της κλάσης.
- public:
  - Ό,τι δηλώνεται ως κοινόχρηστο (public), είναι ορατό/προσβάσιμο από τον χρήστη της κλάσης.
  - Το τμήμα αυτό αποτελεί τη διεπαφή
  - (interface) της κλάσης.
- private:
  - Ό,τι δηλώνεται ως ιδιωτικό (private), δεν είναι προσβάσιμο από τον χρήστη της κλάσης.
  - Στο διπλανό παράδειγμα, τα εσωτερικά
  - δεδομένα της κλάσης μας (ο αριθμητής και ο παρανομαστής) είναι ιδιωτικά.
- Αν χρειάζεται, για την πρόσβαση (ανάγνωση/get, μεταβολή/set) στα ιδιωτικά δεδομένα, υλοποιούνται κατάλληλες μέθοδοι (π.χ. *getNumerator()* και *setNumerator()*).

Μέθοδοι κατασκευής αντικειμένων (constructors)

C++/ROOT

```
1  // αρχείο Ratio.h
2  #pragma once

4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
       Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int getDenominator();      // πρόσβαση
12     void setNumerator(int);    // πρόσβαση
13     void setDenominator(int); // πρόσβαση
14     double toDouble();      // πραγματική τιμή
15     void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Η μέθοδος κατασκευής εκτελείται κατά τη δημιουργία αντικειμένων (στιγμιοτύπων) μίας κλάσης.

- Κλάση: πρότυπο/μήτρα για την κατασκευή αντικειμένων
- Αντικείμενο:
  Δημιουργείται καλώντας μία μέθοδο κατασκευής (constructor) της κλάσης.
  Καταστρέφεται όταν το αντικείμενο καλέσει τη μέθοδο κατάστροφής του (destructor).

Μέθοδοι κατασκευής αντικειμένων (constructors)

C++ / ROOT

```
1  // αρχείο Ratio.h
2  #pragma once

4  class Ratio {
5    public:
6      Ratio();                  // κατασκευή
       Ratio(int);               // κατασκευή
8      Ratio(int,int);           // κατασκευή
9      ~Ratio();                 // καταστροφή
10     int getNumerator();         // πρόσβαση
11     int getDenominator();       // πρόσβαση
12     void setNumerator(int);     // πρόσβαση
13     void setDenominator(int); // πρόσβαση
14     double toDouble();        // πραγματική τιμή
15     void invert();            // αντιστροφή
16     void print();             // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Κλάση: πρότυπο/μήτρα για την κατασκευή αντικειμένων
- Αντικείμενο:
  Δημιουργείται καλώντας μία μέθοδο κατασκευής (constructor) της κλάσης.
  Καταστρέφεται όταν το αντικείμενο καλέσει τη μέθοδο κατάστροφής του (destructor).

- Η μέθοδος κατασκευής εκτελείται κατά τη δημιουργία αντικειμένων (στιγμιοτύπων) μίας κλάσης.
- Η εξ ορισμού μέθοδος κατασκευής (default constructor) υπάρχει για κάθε κλάση, και δεν έχει κανένα όρισμα. Το όνομά της είναι ίδιο με το όνομα της κλάσης, π.χ. *Ratio()* .

Μέθοδοι κατασκευής αντικειμένων (constructors)

C++/ROOT

```
1  // αρχείο Ratio.h
2  #pragma once
4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
       Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int getDenominator();      // πρόσβαση
12     void setNumerator(int);    // πρόσβαση
13     void setDenominator(int); // πρόσβαση
14     double toDouble();      // πραγματική τιμή
       void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Κλάση: πρότυπο/μήτρα για την κατασκευή αντικειμένων
- Αντικείμενο:
  Δημιουργείται καλώντας μία μέθοδο κατασκευής (constructor) της κλάσης.
  Καταστρέφεται όταν το αντικείμενο καλέσει τη μέθοδο κατάστροφής του (destructor).

- Η μέθοδος κατασκευής εκτελείται κατά τη δημιουργία αντικειμένων (στιγμιοτύπων) μίας κλάσης.
- Η εξ ορισμού μέθοδος κατασκευής (default constructor) υπάρχει για κάθε κλάση, και δεν έχει κανένα όρισμα. Το όνομά της είναι ίδιο με το όνομα της κλάσης, π.χ. *Ratio()* .
- Ανάλογα με τις ανάγκες που προκύπτουν από τον σχεδιασμό μίας κλάσης, μπορούν να υπάρχουν περισσότερες από μία μέθοδοι κατασκευής (constructors), με διαφορετική υπογραφή η κάθε μία (constructor overloading).

Μέθοδοι κατασκευής αντικειμένων (constructors)

C++/ROOT

```
1  // αρχείο Ratio.h
2  #pragma once
4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
       Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int getDenominator();      // πρόσβαση
12     void setNumerator(int);    // πρόσβαση
13     void setDenominator(int); // πρόσβαση
14     double toDouble();      // πραγματική τιμή
       void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Κλάση: πρότυπο/μήτρα για την κατασκευή αντικειμένων
- Αντικείμενο:
  Δημιουργείται καλώντας μία μέθοδο κατασκευής (constructor) της κλάσης.
  Καταστρέφεται όταν το αντικείμενο καλέσει τη μέθοδο κατάστροφής του (destructor).

- Η μέθοδος κατασκευής εκτελείται κατά τη δημιουργία αντικειμένων (στιγμιοτύπων) μίας κλάσης.
- Η εξ ορισμού μέθοδος κατασκευής (default constructor) υπάρχει για κάθε κλάση, και δεν έχει κανένα όρισμα. Το όνομά της είναι ίδιο με το όνομα της κλάσης, π.χ. *Ratio()* .
- Ανάλογα με τις ανάγκες που προκύπτουν από τον σχεδιασμό μίας κλάσης, μπορούν να υπάρχουν περισσότερες από μία μέθοδοι κατασκευής (constructors), με διαφορετική υπογραφή η κάθε μία (constructor overloading).
- Οι μέθοδοι κατασκευής δεν έχουν τύπο και δεν επιστρέφουν τίποτε.

Μέθοδοι καταστροφής αντικειμένων (destructors)

C++ / ROOT

```
1  // αρχείο Ratio.h
2  #pragma once

4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
       Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκεΥή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int getDenominator();      // πρόσβαση
12     void setNumerator(int);    // πρόσβαση
13     void setDenominator(int); // πρόσβαση
14     double toDouble();      // πραγματική τιμή
15     void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Κλάση: πρότυπο/μήτρα για την κατασκευή αντικειμένων
- Αντικείμενο:
  Δημιουργείται καλώντας μία μέθοδο κατασκευής (constructor) της κλάσης.
  Καταστρέφεται όταν το αντικείμενο καλέσει τη μέθοδο κατάστροφής του (destructor).

- Η μέθοδος καταστροφής εκτελείται όταν ζητείται η διαγραφή ενός αντικειμένου (με την εντολή delete ή όταν το αντικείμενο βρεθεί εκτός εμβέλειας από την περιοχή ορισμού του). Συνήθως είναι άδεια. Έχει έννοια όταν πρέπει να ελευθερωθεί μνήμη που δεσμεύτηκε δυναμικά κατά τη διάρκεια ζωής του στιγμιοτύπου.

Μέθοδοι καταστροφής αντικειμένων (destructors)

C++/ROOT

```
1  // αρχείο Ratio.h
2  #pragma once
4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
       Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int getDenominator();      // πρόσβαση
12     void setNumerator(int);    // πρόσβαση
13     void setDenominator(int); // πρόσβαση
14     double toDouble();      // πραγματική τιμή
       void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Κλάση: πρότυπο/μήτρα για την κατασκευή αντικειμένων
- Αντικείμενο:
  Δημιουργείται καλώντας μία μέθοδο κατασκευής (constructor) της κλάσης.
  Καταστρέφεται όταν το αντικείμενο καλέσει τη μέθοδο κατάστροφής του (destructor).

- Η μέθοδος καταστροφής εκτελείται όταν ζητείται η διαγραφή ενός αντικειμένου (με την εντολή delete ή όταν το αντικείμενο βρεθεί εκτός εμβέλειας από την περιοχή ορισμού του). Συνήθως είναι άδεια. Έχει έννοια όταν πρέπει να ελευθερωθεί μνήμη που δεσμεύτηκε δυναμικά κατά τη διάρκεια ζωής του στιγμιοτύπου.
- Η μέθοδος καταστροφής, έχει το ίδιο όνομα με την κλάση με την πρόθεση μίας περισπωμένης, π.χ. *~Ratio()*, και δεν έχει ορίσματα, οπότε υπάρχει μόνο μία (δεν μπορεί να υπερφορτωθεί).

Μέθοδοι καταστροφής αντικειμένων (destructors)

C++/ROOT

```
1  // αρχείο Ratio.h
2  #pragma once
4  class Ratio {
5    public:
6      Ratio();                // κατασκευή
       Ratio(int);             // κατασκευή
8      Ratio(int,int);         // κατασκευή
9      ~Ratio();               // καταστροφή
10     int getNumerator();        // πρόσβαση
11     int getDenominator();      // πρόσβαση
12     void setNumerator(int);    // πρόσβαση
13     void setDenominator(int); // πρόσβαση
14     double toDouble();      // πραγματική τιμή
       void invert();          // αντιστροφή
16     void print();           // εκτύπωση
17   private:
18     int num;       // αριθμητής
19     int den;       // παρανομαστής
20 };
```

- Κλάση: πρότυπο/μήτρα για την κατασκευή αντικειμένων
- Αντικείμενο:
  Δημιουργείται καλώντας μία μέθοδο κατασκευής (constructor) της κλάσης.
  Καταστρέφεται όταν το αντικείμενο καλέσει τη μέθοδο κατάστροφής του (destructor).

- Η μέθοδος καταστροφής εκτελείται όταν ζητείται η διαγραφή ενός αντικειμένου (με την εντολή delete ή όταν το αντικείμενο βρεθεί εκτός εμβέλειας από την περιοχή ορισμού του). Συνήθως είναι άδεια. Έχει έννοια όταν πρέπει να ελευθερωθεί μνήμη που δεσμεύτηκε δυναμικά κατά τη διάρκεια ζωής του στιγμιοτύπου.
- Η μέθοδος καταστροφής, έχει το ίδιο όνομα με την κλάση με την πρόθεση μίας περισπωμένης, π.χ. *~Ratio()*, και δεν έχει ορίσματα, οπότε υπάρχει μόνο μία (δεν μπορεί να υπερφορτωθεί).
- Η μέθοδος καταστροφής δεν έχει τύπο και δεν επιστρέφει τίποτε.

Υλοποίηση των μεθόδων της κλάσης (παράδειγμα: Ratio.cpp)


```cpp
// αρχείο Ratio.cpp
#include <iostream>
#include "Ratio.h"

Ratio::Ratio()             { num=0; den=1; } // κατασκευή
Ratio::Ratio(int i)        { num=i; den=1; } // κατασκευή
Ratio::Ratio(int i,int j)  { num=i; den=j; } // κατασκευή
Ratio::~Ratio() { }     // άδεια μέθοδος καταστροφής

int Ratio::getNumerator() {          // πρόσβαση
  return num;
}
int Ratio::getDenominator() {        // πρόσβαση
  return den;
}
void Ratio::setNumerator(int i) {    // πρόσβαση
  num=i;
}
void Ratio::setDenominator(int i) { // πρόσβαση
  den=i;
}
double Ratio::toDouble() {     // μετατροπή
  return (double) num / (double) den;
}
void Ratio::invert() {         // αντιστροφή
   int tmp=num;
   num=den;
   den=tmp;
}
void Ratio::print() {          // εκτύπωση
  std::cout << num << "/" << den;
}
```

- Όλες οι μέθοδοι που δηλώθηκαν στο αρχείο Ratio.h, υλοποιούνται εντός του αρχείου Ratio.cpp.

Υλοποίηση των μεθόδων της κλάσης (παράδειγμα: Ratio.cpp)


```
1  // αρχείο Ratio.cpp
2  #include <iostream>
3  #include "Ratio.h"
5  Ratio::Ratio()             { num=0; den=1; } // κατασκευή
6  Ratio::Ratio(int i)        { num=i; den=1; } // κατασκευή
7  Ratio::Ratio(int i,int j)  { num=i; den=j; } // κατασκευή
8  Ratio::~Ratio() { }     // άδεια μέθοδος καταστροφής
9
10 int Ratio::getNumerator() {          // πρόσβαση
11   return num;
12 }
13 int Ratio::getDenominator() {        // πρόσβαση
14   return den;
15 }
16 void Ratio::setNumerator(int i) {    // πρόσβαση
17   num=i;
18 }
19 void Ratio::setDenominator(int i) { // πρόσβαση
20   den=i;
21 }
22 double Ratio::toDouble() {     // μετατροπή
23   return (double) num / (double) den;
24 }
25 void Ratio::invert() {         // αντιστροφή
26    int tmp=num;
27    num=den;
28    den=tmp;
29 }
30 void Ratio::print() {          // εκτύπωση
31   std::cout << num << "/" << den;
32 }
```

- Όλες οι μέθοδοι που δηλώθηκαν στο αρχείο Ratio.h, υλοποιούνται εντός του αρχείου Ratio.cpp.
- Στο παράδειγμά μας υλοποιείται η κλάση Ratio με την οποία θέλουμε να περιγράψουμε και να χειριστούμε ρητούς αριθμούς.

Υλοποίηση των μεθόδων της κλάσης (παράδειγμα: Ratio.cpp)


```cpp
// αρχείο Ratio.cpp
#include <iostream>
#include "Ratio.h"

Ratio::Ratio()             { num=0; den=1; } // κατασκευή
Ratio::Ratio(int i)        { num=i; den=1; } // κατασκευή
Ratio::Ratio(int i,int j)  { num=i; den=j; } // κατασκευή
Ratio::~Ratio() { }     // άδεια μέθοδος καταστροφής

int Ratio::getNumerator() {          // πρόσβαση
  return num;
}
int Ratio::getDenominator() {        // πρόσβαση
  return den;
}
void Ratio::setNumerator(int i) {    // πρόσβαση
  num=i;
}
void Ratio::setDenominator(int i) { // πρόσβαση
  den=i;
}
double Ratio::toDouble() {     // μετατροπή
  return (double) num / (double) den;
}
void Ratio::invert() {         // αντιστροφή
   int tmp=num;
   num=den;
   den=tmp;
}
void Ratio::print() {          // εκτύπωση
  std::cout << num << "/" << den;
}
```

- Όλες οι μέθοδοι που δηλώθηκαν στο αρχείο Ratio.h, υλοποιούνται εντός του αρχείου Ratio.cpp.
- Στο παράδειγμά μας υλοποιείται η κλάση Ratio με την οποία θέλουμε να περιγράψουμε και να χειριστούμε ρητούς αριθμούς.
- Με την ολοκλήρωση της υλοποίησης (implementation) των μεθόδων της κλάσης, είναι έτοιμη να χρησιμοποιηθεί, αφού μεταγλωττιστεί:

```
g++ -std=c++11 -c Ratio.cpp
```

Υλοποίηση των μεθόδων της κλάσης (παράδειγμα: Ratio.cpp)


```cpp
// αρχείο Ratio.cpp
#include <iostream>
#include "Ratio.h"

Ratio::Ratio()             { num=0; den=1; } // κατασκευή
Ratio::Ratio(int i)        { num=i; den=1; } // κατασκευή
Ratio::Ratio(int i,int j)  { num=i; den=j; } // κατασκευή
Ratio::~Ratio() { }     // άδεια μέθοδος καταστροφής

int Ratio::getNumerator() {          // πρόσβαση
  return num;
}
int Ratio::getDenominator() {        // πρόσβαση
  return den;
}
void Ratio::setNumerator(int i) {    // πρόσβαση
  num=i;
}
void Ratio::setDenominator(int i) { // πρόσβαση
  den=i;
}
double Ratio::toDouble() {     // μετατροπή
  return (double) num / (double) den;
}
void Ratio::invert() {         // αντιστροφή
   int tmp=num;
   num=den;
   den=tmp;
}
void Ratio::print() {          // εκτύπωση
  std::cout << num << "/" << den;
}
```

- Όλες οι μέθοδοι που δηλώθηκαν στο αρχείο Ratio.h, υλοποιούνται εντός του αρχείου Ratio.cpp.
- Στο παράδειγμά μας υλοποιείται η κλάση Ratio με την οποία θέλουμε να περιγράψουμε και να χειριστούμε ρητούς αριθμούς.
- Με την ολοκλήρωση της υλοποίησης (implementation) των μεθόδων της κλάσης, είναι έτοιμη να χρησιμοποιηθεί, αφού μεταγλωττιστεί:

```
g++ -std=c++11 -c Ratio.cpp
```

- Από τη μεταγλώττιση προκύπτει το αρχείο Ratio.o αντικειμενικού κώδικα (object file), το οποίο μπορεί να χρησιμοποιηθεί αργότερα για την παραγωγή ενός τελικού εκτελέσιμου αρχείου.

Πρόγραμμα που χρησιμοποιεί την κλάση Ratio (testRatio.cpp)

```
1  // αρχείο testRatio.cpp
2  #include <iostream>
3  #include "Ratio.h"
4  using namespace std;
5
6  int main() {
     int i=37;
     int j=11;
     Ratio r1;       // εξ'ορισμού μέθοδος κατασκευής: 0/1
     Ratio r2(15);   // κατασκευή με ένα όρισμα: 15/1
     Ratio r3(3,4); // κατασκευή με δύο ορίσματα: 3/4

     cout << "r1="; r1.print();  cout << endl;
     cout << "r2="; r2.print();  cout << endl;
     cout << "r3="; r3.print(); cout << endl;

     r1.setNumerator(i);
     cout << "r1=";   r1.print();   cout << endl;
     r1.setDenominator(j);
     cout << "r1=";   r1.print();   cout << endl;

     return 0;
  }
```

- Τα r1, r2 και r3 δηλώνονται ως αντικείμενα της κλάσης Ratio. Ανάλογα με τα ορίσματα που δίνονται, ο μεταγλωτιστής αποφασίζει κάθε φορά ποια μέθοδος κατασκευής θα χρησιμοποιηθεί.

Πρόγραμμα που χρησιμοποιεί την κλάση Ratio (testRatio.cpp)

```
1  // αρχείο testRatio.cpp
2  #include <iostream>
3  #include "Ratio.h"
4  using namespace std;
5
6  int main() {
     int i=37;
     int j=11;
     Ratio r1;       // εξ'ορισμού μέθοδος κατασκευής: 0/1
     Ratio r2(15);   // κατασκευή με ένα όρισμα: 15/1
     Ratio r3(3,4); // κατασκευή με δύο ορίσματα: 3/4

     cout << "r1="; r1.print();  cout << endl;
     cout << "r2="; r2.print();  cout << endl;
     cout << "r3="; r3.print(); cout << endl;

     r1.setNumerator(i);
     cout << "r1=";   r1.print();   cout << endl;
     r1.setDenominator(j);
     cout << "r1=";   r1.print();   cout << endl;

     return 0;
   }
```

- Τα r1, r2 και r3 δηλώνονται ως αντικείμενα της κλάσης Ratio. Ανάλογα με τα ορίσματα που δίνονται, ο μεταγλωτιστής αποφασίζει κάθε φορά ποια μέθοδος κατασκευής θα χρησιμοποιηθεί.
- Για την κλήση μεθόδων ενός αντικειμένου, χρησιμοποιείται ο τελεστής επιλογής `'.'` (τελεία).

Πρόγραμμα που χρησιμοποιεί την κλάση Ratio (testRatio.cpp)

```
1  // αρχείο testRatio.cpp
2  #include <iostream>

4  using namespace std;
5
6  int main() {
     int i=37;
     int j=11;
     Ratio r1;       // εξ'ορισμού μέθοδος κατασκευής: 0/1
     Ratio r2(15);   // κατασκευή με ένα όρισμα: 15/1
     Ratio r3(3,4); // κατασκευή με δύο ορίσματα: 3/4

     cout << "r1="; r1.print();  cout << endl;
     cout << "r2="; r2.print();  cout << endl;
     cout << "r3="; r3.print(); cout << endl;

     r1.setNumerator(i);
     cout << "r1=";   r1.print();   cout << endl;
     r1.setDenominator(j);
     cout << "r1=";   r1.print();   cout << endl;

     return 0;
   }
```

- Τα r1, r2 και r3 δηλώνονται ως αντικείμενα της κλάσης Ratio. Ανάλογα με τα ορίσματα που δίνονται, ο μεταγλωτιστής αποφασίζει κάθε φορά ποια μέθοδος κατασκευής θα χρησιμοποιηθεί.
- Για την κλήση μεθόδων ενός αντικειμένου, χρησιμοποιείται ο τελεστής επιλογής `'.'` (τελεία).
- Χρησιμοποιώντας τις μεθόδους πρόσβασης, μπορούμε να λάβουμε ή να αλλάξουμε τις ιδιωτικές μεταβλητές ενός αντικειμένου. Στο παράδειγμα αλλάζουμε τον αριθμητή και τον παρανομαστή του r1.

Πρόγραμμα που χρησιμοποιεί την κλάση Ratio (testRatio.cpp)

```
1  // αρχείο testRatio.cpp
2  #include <iostream>

4  using namespace std;
5
6  int main() {
       int i=37;
       int j=11;
       Ratio r1;       // εξ'ορισμού μέθοδος κατασκευής: 0/1
       Ratio r2(15);   // κατασκευή με ένα όρισμα: 15/1
       Ratio r3(3,4); // κατασκευή με δύο ορίσματα: 3/4

       cout << "r1="; r1.print();   cout << endl;
       cout << "r2="; r2.print();   cout << endl;
       cout << "r3="; r3.print(); cout << endl;

       r1.setNumerator(i);
       cout << "r1=";    r1.print();    cout << endl;
       r1.setDenominator(j);
       cout << "r1=";    r1.print();    cout << endl;

       return 0;
   }
```

- Τα r1, r2 και r3 δηλώνονται ως αντικείμενα της κλάσης Ratio. Ανάλογα με τα ορίσματα που δίνονται, ο μεταγλωτιστής αποφασίζει κάθε φορά ποια μέθοδος κατασκευής θα χρησιμοποιηθεί.
- Για την κλήση μεθόδων ενός αντικειμένου, χρησιμοποιείται ο τελεστής επιλογής `'.'` (τελεία).
- Χρησιμοποιώντας τις μεθόδους πρόσβασης, μπορούμε να λάβουμε ή να αλλάξουμε τις ιδιωτικές μεταβλητές ενός αντικειμένου. Στο παράδειγμα αλλάζουμε τον αριθμητή και τον παρανομαστή του r1.
- Με την ολοκλήρωση της ανάπτυξης του προγράμματος, μπορούμε να το μεταγλωττίσουμε:

```
g++ -std=c++11 -c testRatio.cpp
```

Από τη μεταγλώττιση προκύπτει το αρχείο testRatio.o αντικειμενικού κώδικα (object file).

Πρόγραμμα που χρησιμοποιεί την κλάση Ratio (testRatio.cpp)

```
1  // αρχείο testRatio.cpp
2  #include <iostream>

4  using namespace std;
5
6  int main() {
     int i=37;
     int j=11;
     Ratio r1;       // εξ'ορισμού μέθοδος κατασκευής: 0/1
     Ratio r2(15);   // κατασκευή με ένα όρισμα: 15/1
     Ratio r3(3,4); // κατασκευή με δύο ορίσματα: 3/4

     cout << "r1="; r1.print();   cout << endl;
     cout << "r2="; r2.print();   cout << endl;
     cout << "r3="; r3.print(); cout << endl;

     r1.setNumerator(i);
     cout << "r1=";   r1.print();   cout << endl;
     r1.setDenominator(j);
     cout << "r1=";   r1.print();   cout << endl;

     return 0;
   }
```

Για να δημιουργήσουμε το τελικό εκτελέσιμο αρχείο, συνδυάζουμε τα object files που πλέον έχουμε:

```
g++ -std=c++11 Ratio.o testRatio.o -o testRatio.exe
```

- Τα r1, r2 και r3 δηλώνονται ως αντικείμενα της κλάσης Ratio. Ανάλογα με τα ορίσματα που δίνονται, ο μεταγλωτιστής αποφασίζει κάθε φορά ποια μέθοδος κατασκευής θα χρησιμοποιηθεί.
- Για την κλήση μεθόδων ενός αντικειμένου, χρησιμοποιείται ο τελεστής επιλογής `'.'` (τελεία).
- Χρησιμοποιώντας τις μεθόδους πρόσβασης, μπορούμε να λάβουμε ή να αλλάξουμε τις ιδιωτικές μεταβλητές ενός αντικειμένου. Στο παράδειγμα αλλάζουμε τον αριθμητή και τον παρανομαστή του r1.
- Με την ολοκλήρωση της ανάπτυξης του προγράμματος, μπορούμε να το μεταγλωττίσουμε:

```
g++ -std=c++11 -c testRatio.cpp
```

Από τη μεταγλώττιση προκύπτει το αρχείο testRatio.o αντικειμενικού κώδικα (object file).