Επίπεδα προστασίας και κατασκευαστές

#2

Τμήμα Πληροφορικής και Τηλεπικοινωνιών (Άρτα)

Πανεπιστήμιο Ιωαννίνων

Γκόγκος Χρήστος

Επίπεδα προστασίας

- Όταν σχεδιάζουμε μια κλάση μπορούμε να ελέγξουμε το πως θα προσπελαύνονται τα **μέλη δεδομένα** και τα **μέλη συναρτήσεις** της κλάσης.
- Σε κάθε μέλος της κλάσης ανατίθεται ένα επίπεδο προστασίας:
  - **Δημόσια (public) μέλη**: Δεδομένα και συναρτήσεις των αντικειμένων που μπορούν να προσπελαστούν τόσο εκτός όσο και εντός των μελών συναρτήσεων της κλάσης.
  - **Ιδιωτικά (private) μέλη**: Δεδομένα και συναρτήσεις των αντικειμένων που μπορούν να προσπελαστούν μόνο εντός των μελών συναρτήσεων της κλάσης.
    - Δηλαδή, ο κώδικας που υλοποιεί την κλάση έχει πρόσβαση στα ιδιωτικά δεδομένα και συναρτήσεις, ενώ ο κώδικας που χρησιμοποιεί την κλάση δεν έχει πρόσβαση στα ιδιωτικά δεδομένα και συναρτήσεις
- Ο κύριος στόχος των επιπέδων προστασίας είναι να επιτρέπει στους προγραμματιστές να παρέχουν μια δημόσια διεπαφή για την κλάση, ενώ παράλληλα να είναι σε θέση να αλλάζουν την υλοποίηση της κλάσης χωρίς να δημιουργούνται προβλήματα σε κώδικα που χρησιμοποιεί την κλάση.
- Πρόκειται για έναν μηχανισμό μέσω του οποίου υλοποιείται η **απόκρυψη πληροφορίας**.

sample1.cpp

```cpp
#include <iostream>
class Fraction {
public:
    void SetNumerator(int n);
    void SetDenominator(int d);
    double ToDecimal();
private:
    int numer;
    int denom;
};
void Fraction::SetNumerator(int n) {
    numer = n; // παρατηρήστε ότι προσπελαύνουμε το ιδιωτικό μέλος numer εδώ
}
void Fraction::SetDenominator(int d) {
    if (d != 0)
        denom = d;
    else
        denom = 1;
}
double Fraction::ToDecimal() {
    return (double)numer / denom;
}
int main() {
    Fraction F;
    // F.numer = 1;
    F.SetNumerator(1);
    F.SetDenominator(2);
    std::cout << "Decimal: " << F.ToDecimal() << std::endl;
    return 0;
}
```

- Έξοδος προγράμματος:

  `Decimal: 0.5`

- Τι θα συμβεί αν στη main αφαιρέσουμε τα σχόλια από την εντολή:

  `F.numer = 1;`

```
$ g++ sample1.cpp
sample1.cpp: In function 'int main()':
sample1.cpp:37:7: error: 'int Fraction::numer' is private within this context
   F.numer = 1;
     ^~~~~
sample1.cpp:11:9: note: declared private here
   int numer;
       ^~~~~
```

https://github.com/chgogos/oop/blob/master/various/COP3330/lect2/sample1.cpp

sample2.cpp

```cpp
#include <iostream>

class Fraction {
public:
    void SetNumerator(int n);
    void SetDenominator(int d);
    double ToDecimal();
private:
    void PrintHello();
    int numer;
    int denom;
};
void Fraction::SetNumerator(int n) {
    numer = n; // παρατηρήστε ότι προσπελαύνουμε το ιδιωτικό μέλος numer εδώ
}
void Fraction::SetDenominator(int d) {
    if (d != 0) denom = d; else denom = 1;
}
double Fraction::ToDecimal() {
    PrintHello(); // ok, διότι καλείται μέσα από την κλάση
    return (double)numer / denom;
}
void Fraction::PrintHello() {
    std::cout << "Hello!!!" << std::endl;
}
int main() {
    Fraction F;
    // F.PrintHello(); // δεν γίνεται διότι η PrintHello είναι ιδιωτικό μέλος της κλάσης
    F.SetNumerator(1);
    F.SetDenominator(2);
    std::cout << "Decimal: " << F.ToDecimal() << std::endl;
    return 0;
}
```

- Έξοδος προγράμματος:

```
Decimal: Hello!!!
0.5
```

- Τι θα συμβεί αν στη main αφαιρέσουμε τα σχόλια από την εντολή:

```
F.PrintHello();
```

```
$ g++ sample2.cpp
sample2.cpp: In function 'int main()':
sample2.cpp:44:18: error: 'void Fraction::PrintHello()' is private within this context
   F.PrintHello();
                 ^
sample2.cpp:35:6: note: declared private here
 void Fraction::PrintHello()
      ^~~~~~~~
```

https://github.com/chgogos/oop/blob/master/various/COP3330/lect2/sample2.cpp

Περισσότερα για τα επίπεδα προστασίας

- Λόγοι για την εφαρμογή της απόκρυψης πληροφορίας (ιδιωτικά μέλη):
  - Καθιστά τη διεπαφή απλούστερη για τον χρήστη.
  - Εφαρμογή της αρχής του ελάχιστου προνομίου (need to know).
  - Μεγαλύτερη ασφάλεια. Μικρότερη πιθανότητα κακής χρήσης (από λάθος ή κακόβουλα).
  - Ευκολία αλλαγής υλοποίησης της κλάσης χωρίς να επηρεάζονται άλλα τμήματα κώδικα που τη χρησιμοποιούν.
- Επιπλέον σημαντικές πληροφορίες:
  - Αν δεν ορίζεται ρητά επίπεδο προστασίας, τα μέλη είναι ιδιωτικά.
  - Κατά το σχεδιασμό της κλάσης θα πρέπει να ορίζονται ως ιδιωτικά τα μέλη που δεν ανήκουν στη διεπαφή της.

Κατασκευαστές (1/4)

- Ο κατασκευαστής (constructor) είναι μια ειδική συνάρτηση μέλος που συνήθως έχει ως σκοπό την αρχικοποίηση των μελών ενός αντικειμένου.
- Ο κατασκευαστής είναι εύκολο να αναγνωριστεί διότι:
  - Έχει το ίδιο όνομα με την κλάση.
  - Δεν έχει τύπο επιστροφής.

Κατασκευαστές (2/4) – Παράδειγμα κλάσης Circle (1-δήλωση, 2-ορισμός, 3-δημιουργία αντικειμένων)

1

```
// Δήλωση κλάσης Circle
class Circle {
public:
  Circle(); // Κατασκευαστής
  Circle(double r); // Κατασκευαστής
  void setCenter(double x, double y);
  void SetRadius(double r);
  void Draw();
  double AreaOf();

private:
  double radius;
  double center_x;
  double center_y;
};
```

2

```
// Ορισμοί συναρτήσεων μελών κλάσης Circle
Circle::Circle() : radius(5.0), center_x(0.0), center_y(0.0) {
  cout << "Default constructor Circle() called" << endl;
}
Circle::Circle(double r) : radius(r), center_x(0.0), center_y(0.0) {
  cout << "Constructor Circle(double) called" << endl;
}
void Circle::setCenter(double x, double y) {
  center_x = x;
  center_y = y;
}
void Circle::SetRadius(double r) { radius = r; }
void Circle::Draw() {
  cout << "A circle having center at " << center_x << ", " << center_y
       << " and radius " << radius << endl;
}
double Circle::AreaOf() { return 3.14 * radius * radius; }
```

3

```
// Πρόγραμμα οδηγός
int main() {
  Circle c1;  c1.Draw();
  cout << "Area: " << c1.AreaOf() << endl;
  c1.setCenter(7.1, 2.3);c1.SetRadius(16.6);
  cout << "Area: " << c1.AreaOf() << endl;
  Circle c2(13.5);  c2.Draw();
  cout << "Area: " << c2.AreaOf() << endl;
}
```

```
Default constructor Circle() called
A circle having center at 0, 0 and radius 5
Area: 78.5
Area: 865.258
Constructor Circle(double) called
A circle having center at 0, 0 and radius 13.5
Area: 572.265
```

https://github.com/chgogos/oop/blob/master/various/COP3330/lect2/circle.cpp

Κατασκευαστές (3/4) – εναλλακτικοί τρόποι συγγραφής κατασκευαστών για τη Circle

Δήλωση και ορισμός της κλάσης μαζί & κατασκευαστές με initializer lists

```
class Circle {
 public:
  Circle() : radius(5.0), center_x(0.0), center_y(0.0) {
    cout << "Default constructor Circle() called" << endl;
  }
  Circle(double r) : radius(r), center_x(0.0), center_y(0.0) {
    cout << "Constructor Circle(double) called" << endl;
  }
 ...
 private:
  double radius;
  double center_x;
  double center_y;
};
```

https://github.com/chgogos/oop/blob/master/various/COP3330/lect2/circle2.cpp

Δήλωση και ορισμός της κλάσης μαζί & κατασκευαστές χωρίς initializer lists

```
class Circle {
 public:
  Circle() {
    center_x = 0.0;
    center_y = 0.0;
    radius = 5.0;
    cout << "Default constructor Circle() called" << endl;
  }
  Circle(double r) {
    center_x = 0.0;
    center_y = 0.0;
    radius = r;
    cout << "Constructor Circle(double) called" << endl;
  }
 ...
 private:
  double radius;
  double center_x;
  double center_y;
};
```

https://github.com/chgogos/oop/blob/master/various/COP3330/lect2/circle3.cpp

Κατασκευαστές (4/4)

- Οι κατασκευαστές είναι συναρτήσεις, συνεπώς μπορούν να περιέχουν οποιοδήποτε κώδικα όπως και οι άλλες συναρτήσεις.
- Τι είναι ιδιαίτερο για τους κατασκευαστές;
  - Καλούνται αυτόματα όταν δημιουργείται ένα στιγμιότυπο ενός αντικειμένου (π.χ. Circle C1;)
  - Η συνάρτηση κατασκευαστής δεν μπορεί να κληθεί όπως οι άλλες συναρτήσεις χρησιμοποιώντας τον τελεστή . (π.χ. το C1.Circle(); δεν είναι έγκυρο)
- Ο όρος **προκαθορισμένος κατασκευαστής** (default constructor) αναφέρεται σε έναν κατασκευαστή χωρίς παραμέτρους.
- Κάθε κλάση **θα πρέπει** να έχει έναν κατασκευαστή. Αν δεν γραφεί από τον προγραμματιστή τότε ένας κενός προκαθορισμένος κατασκευαστής δημιουργείται από τη γλώσσα.

Πέρασμα ορισμάτων σε έναν κατασκευαστή

- Γίνεται απλά προσθέτοντας (…) ως τμήμα της δημιουργίας του αντικειμένου

  Circle C1(5); /* δηλώνει ένα νέο αντικείμενο Circle με το όρισμα 5 να περνά στο δεύτερο κατασκευαστή της Circle στη παράμετρο r */

```
class Circle {
 public:
  Circle() : radius(5.0), center_x(0.0), center_y(0.0) {
    cout << "Default constructor Circle() called" << endl;
  }
  Circle(double r) : radius(r), center_x(0.0), center_y(0.0) {
    cout << "Constructor Circle(double) called" << endl;
  }
 ...
 private:
  double radius;
  double center_x;
  double center_y;
};
```

```
int main() {
  Circle c1(5);
  c1.Draw();
  cout << "Area: " << c1.AreaOf() << endl;
}
```

```
Constructor Circle(double) called
A circle having center at 0, 0 and radius 5
Area: 78.5
```

Δείτε και το παράδειγμα με την κλάση Fraction:
https://github.com/chgogos/oop/blob/master/various/COP3330/lect2/sample3.cpp

Άσκηση

- Να υλοποιήσετε μια κλάση Book, η οποία θα περιγράφει ένα βιβλίο με ιδιωτικά μέλη δεδομένων τα title (string), isbn (string) και ratings (std::vector<int>) που είναι αρχικά κενό).
- Η κλάση θα έχει έναν default constructor που θα θέτει τιμές title="unknown", isbn="unknown".
- Η κλάση θα περιέχει έναν constructor που θα θέτει τιμές στον τίτλο και στο isbn του βιβλίου.
- Η κλάση θα έχει τις ακόλουθες δημόσιες συναρτήσεις μέλη:
  - τη συνάρτηση add_rating(score) που προσθέτει μια βαθμολογία από 1 έως 5 στα ratings.
  - τη συνάρτηση average_rating() που υπολογίζει και επιστρέφει το μέσο όρο των ratings (ή μήνυμα αν δεν υπάρχουν ratings).
  - τη συνάρτηση info() που επιστρέφει σε μορφή κειμένου τον τίτλο, το ISBN και τη μέση βαθμολογία (μέση τιμή των ratings) του βιβλίου.
- Στη main() να δημιουργηθούν δύο αντικείμενα της κλάσης Book, να προστεθούν ratings και να εμφανιστούν τα βιβλία στην οθόνη.

Χρήσιμες πληροφορίες για την άσκηση

- Ένα vector (διάνυσμα) είναι μια δομή δεδομένων της C++ που είναι δυναμική, δηλαδή μπορούν να προστίθενται και να αφαιρούνται από αυτή στοιχεία.
- Για να δηλωθεί ένα vector θα πρέπει να δηλωθεί ο τύπος των στοιχείων που περιέχει, π.χ. vector<int> για διάνυσμα ακεραίων, vector<string> για διάνυσμα αλφαριθμητικών κ.λπ.

```cpp
#include <iostream>
#include <print>  // διαθέσιμο από C++23
#include <string>
#include <vector>

int main() {
  std::vector<int> v;
  v.push_back(4);  v.push_back(2); v.push_back(4);

  for (int x : v) {
    std::print("{:d}\n", x);
  }

  std::print("# of items = {}, empty = {}\n", v.size(), v.empty());

  double avg = static_cast<double>(v[0] + v[1] + v[2]) / v.size();
  std::string s = std::format("Average value = {:.2f}", avg);
  std::cout << s << std::endl;

  return 0;
}
```

```
4
2
4
# of items = 3, empty = false
Average value = 3.33
```

https://github.com/chgogos/oop/blob/master/various/COP3330/lect2/exercise1_help.cpp

Λύση άσκησης

```cpp
#include <print>
#include <string>
#include <vector>
class Book {
 private:
  std::string title; std::string isbn;  std::vector<int> ratings;
 public:
  Book() : title("unknown"), isbn("unknown") {}
  Book(std::string t, std::string i) : title(t), isbn(i) {}
  void add_rating(int score) {
    if (score >= 1 && score <= 5)
      ratings.push_back(score);
    else
      std::print("Invalid rating: {}\n", score);
  }
  double average_rating_value() const {
    int sum = 0;
    for (int r : ratings) sum += r;
    return static_cast<double>(sum) / ratings.size();
  }
  std::string info() const {
    if (ratings.empty()) {
      return std::format("Title: {}, ISBN: {}, no ratings", title, isbn);
    } else {
      return std::format("Title: {}, ISBN: {}, Average rating: {:.2f}",
                         title, isbn, average_rating_value());
    }
  }
};
```

```cpp
int main() {
  Book b1("C++ Programming", "978-0-13-110362-7");
  Book b2;

  b1.add_rating(5);
  b1.add_rating(4);
  b1.add_rating(5);

  std::print("{}\n", b1.info());
  std::print("{}\n", b2.info());

  return 0;
}
```

```
Title: C++ Programming, ISBN: 978-0-13-110362-7, Average rating: 4.67
Title: unknown, ISBN: unknown, no ratings
```

https://github.com/chgogos/oop/blob/master/various/COP3330/lect2/exercise1.cpp

Σύνοψη (ερωτήσεις)

- Για ποια επίπεδα προστασίας μιλήσαμε;
- Γιατί χρησιμοποιούμε επίπεδα προστασίας;
- Ποιο τμήμα του προγράμματος μπορεί να προσπελάσει τα ιδιωτικά μέλη (δεδομένα ή συναρτήσεις);
- Ποιο τμήμα του προγράμματος δεν μπορεί να προσπελάσει τα ιδιωτικά μέλη;
- Τι συμβαίνει όταν ένα πρόγραμμα προσπαθεί να προσπελάσει με μη έγκυρο τρόπο ιδιωτικά μέλη της κλάσης;
- Τι είναι ένας κατασκευαστής;
- Πως καταλαβαίνουμε ότι μια συνάρτηση είναι κατασκευαστής μιας κλάσης;
- Τι είναι ο προκαθορισμένος κατασκευαστής;
- Πως περνάμε ορίσματα σε έναν κατασκευαστή;

Σύνοψη (ερωτήσεις/απαντήσεις)

- Για ποια επίπεδα προστασίας μιλήσαμε;
  - Ιδιωτικό και δημόσιο.
- Γιατί χρησιμοποιούμε επίπεδα προστασίας;
  - Για να επιτύχουμε απόκρυψη πληροφορίας.
- Ποιο τμήμα του προγράμματος μπορεί να προσπελάσει τα ιδιωτικά μέλη (δεδομένα ή συναρτήσεις);
  - Ο κώδικας που γράφεται μέσα στην ίδια την κλάση.
- Ποιο τμήμα του προγράμματος δεν μπορεί να προσπελάσει τα ιδιωτικά μέλη;
  - Κώδικας που βρίσκεται εκτός της κλάσης με τα ιδιωτικά μέλη, π.χ. η συνάρτηση main().
- Τι συμβαίνει όταν ένα πρόγραμμα προσπαθεί να προσπελάσει με μη έγκυρο τρόπο ιδιωτικά μέλη της κλάσης;
  - Σφάλμα μεταγλώττισης
- Τι είναι ένας κατασκευαστής;
  - Μα ειδική συνάρτηση που καλείται κατά τη δημιουργία ενός αντικειμένου.
- Πως καταλαβαίνουμε ότι μια συνάρτηση είναι κατασκευαστής μιας κλάσης;
  - Έχει το ίδιο όνομα με την κλάση, και δεν έχει τύπο επιστροφής τιμής.
- Τι είναι ο προκαθορισμένος κατασκευαστής;
  - Είναι ο κατασκευαστής που δεν έχει παραμέτρους
- Πως περνάμε ορίσματα σε έναν κατασκευαστή;
  - Γράφουμε το όνομα της κλάσης και μέσα σε παρενθέσεις περνάμε τα ορίσματα χωρισμένα με κόμματα

Αναφορές

- http://www.cs.fsu.edu/~xyuan/cop3330/