Επίπεδα προστασίας και κατασκευαστές

#2

Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων (Άρτα)
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 μέλη**: Δεδομένα και συναρτήσεις των αντικειμένων που μπορούν να προσπελαστούν τόσο εκτός όσο και εντός των μελών συναρτήσεων της κλάσης.
  - **Ιδιωτικά 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 μηχανισμό με τον οποίο 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 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 στην διεπαφή της.

Κατασκευαστές (1/2)

- Ο κατασκευαστής 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.
- Παράδειγμα Circle:

  Circle C1;

```
class Circle {
public:
    Circle();          // αυτός είναι ένας κατασκευαστής
    Circle(double r); // και αυτός είναι ένας κατασκευαστής
    void setCenter(double x, double y);
    void SetRadius(double r);
    void Draw();
    double AreaOf();
private:
    double radius;
    double center_x;
    double center_y;
};
```

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

Κατασκευαστές (2/2)

- 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. C1.Circle();)
- 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 παραμέτρων σε έναν κατασκευαστή

- Γίνεται απλά προσθέτοντας (…) ως τμήμα της δημιουργίας του στιγμιοτύπου 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();         // αυτός είναι ένας κατασκευαστής
    Circle(double r); // και αυτός είναι ένας κατασκευαστής
    void setCenter(double x, double y);
    void SetRadius(double r);
    void Draw();
    double AreaOf();
private:
    double radius;
    double center_x;
    double center_y;
};
```

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

Αναφορές

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