POINTERS, AGGREGATION, COMPOSITION

POINTERS TO OBJECTS

Η κλάση myString

```
class myString
{
private:
  char *s;
  int size;
public:
  myString(const char *);
  ~myString();
  char *GetString();
  int GetSize();
};
```

```
myString::myString(const char * x)
{
  size = strlen(x);
  s = new char[size +1];
  strcpy(s,x);
}

myString::~myString()
{
  delete [] s;
}
```

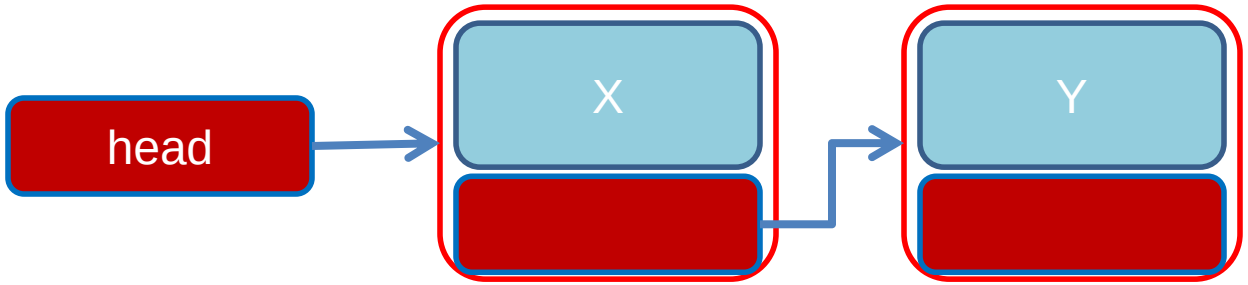
```
char * myString::GetString()
{
  return s;
}

int myString::GetSize()
{
  return size;
}
```

Άσκηση

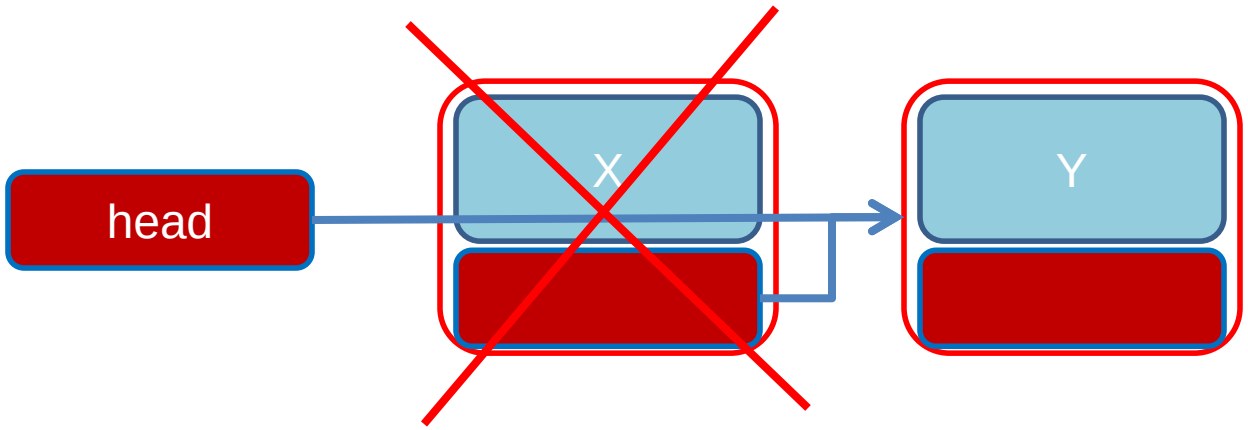
- Διάβασε από το αρχείο “input.txt” ένα άγνωστο αριθμό από ονόματα και τύπωσε αυτά που έχουν μέχρι 5 χαρακτήρες σε αντίθετη σειρά από αυτή που τα διάβασες.
  - Αρχείο: θα χρησιμοποιήσουμε τις συναρτήσεις διαβάσματος που είδαμε στο προηγούμενο μάθημα.
  - Ονόματα: θα τα αποθηκεύσουμε σε myString.
  - Εκτύπωση ονομάτων: πρέπει να τα αποθηκεύσουμε σε μια δυναμική δομή.
    - Stack: αποθηκεύει σε last-in-first-out σειρά οπότε είναι ιδανική για τις ανάγκες μας.

Στοίβα



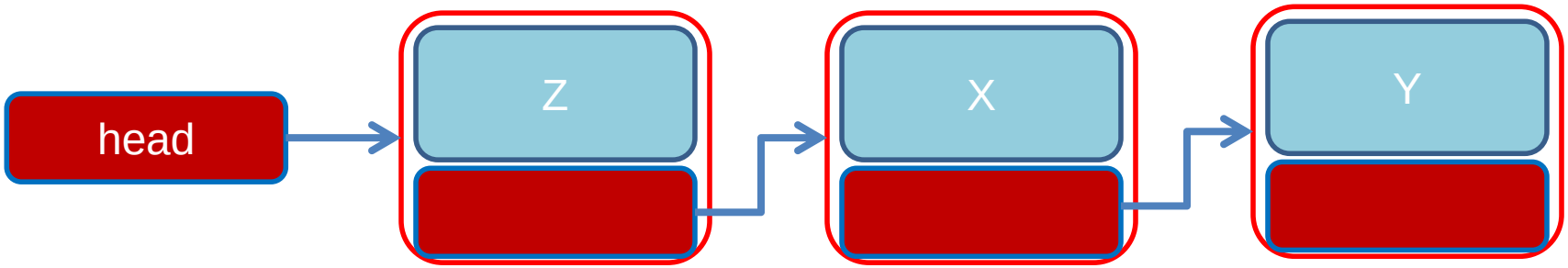
Top(): Επιστρέφει την τιμή στην κορυφή της στοίβας
(X στο παράδειγμα μας)

Στοίβα



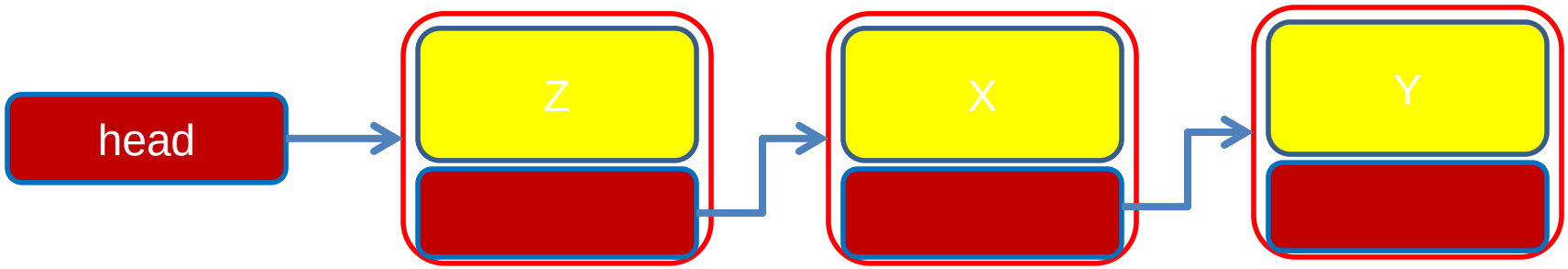
Pop(): Αφαιρεί το στοιχείο στην κορυφή της στοίβας και επιστρέφει την τιμή του (X στο παράδειγμα μας)

Στοίβα



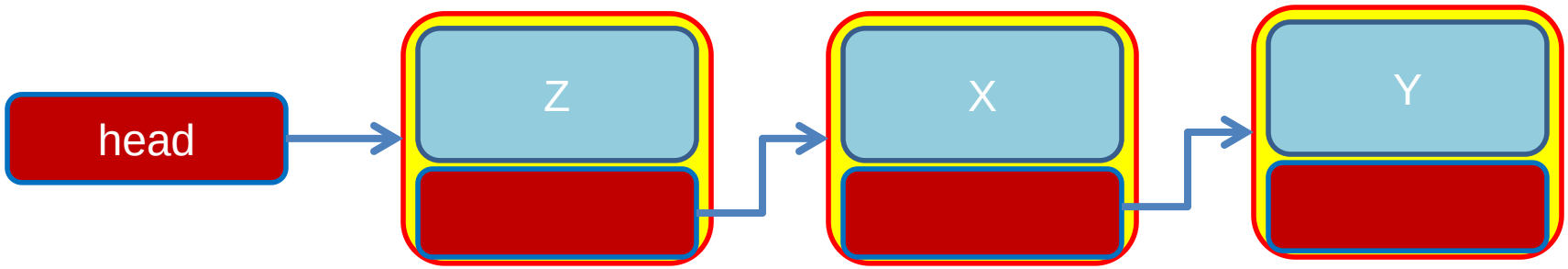
Push(Z): Προσθέτει την τιμή Z στην κορυφή της στοίβας

Στοίβα - Υλοποιηση



Τα X,Y,Z μπορεί να είναι δεδομένα οποιουδήποτε τύπου ή κλασης. Στην περίπτωση μας δείκτες σε myString

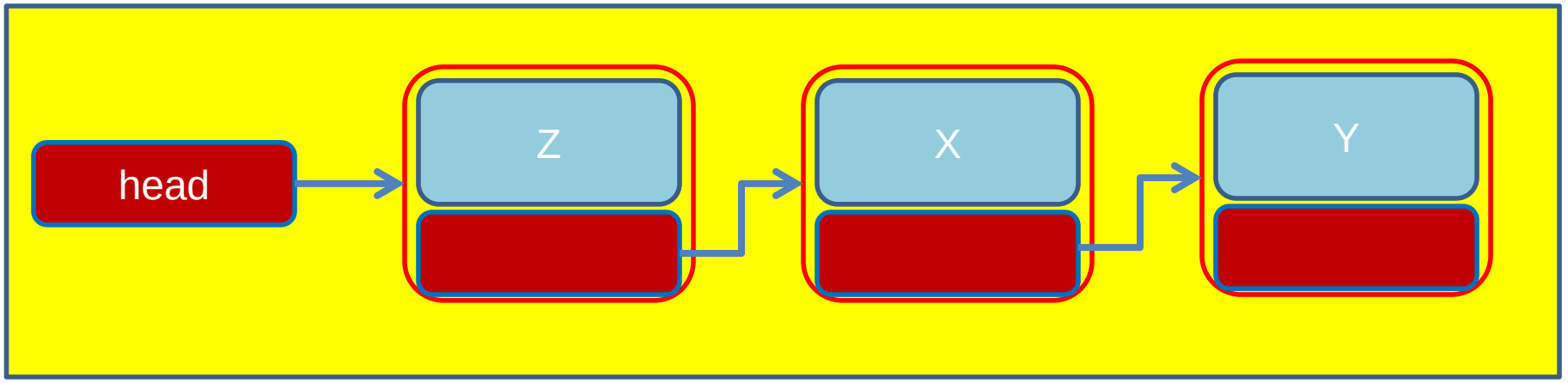
Στοίβα - Υλοποιηση



Τα X,Y,Z μπορεί να είναι δεδομένα οποιουδήποτε τύπου ή κλασης. Στην περίπτωση μας δείκτες σε myString

Θα ορισουμε StackElement μια κλάση που κρατάει το κάθε στοιχείο της στοιβας.

Στοίβα - Υλοποιηση



- Τα X,Y,Z μπορεί να είναι δεδομένα οποιουδήποτε τύπου ή κλασης. Στην περίπτωση μας δείκτες σε myString
- Θα ορισουμε StackElement μια κλάση που κρατάει το κάθε στοιχείο της στοιβας.
- Και μια κλάση Stack που υλοποιεί την στοίβα και όλες τις λειτουργίες της

```
class StackElement
{
private:
  myString *data;
  StackElement *next;
public:
  StackElement(myString *, StackElement *);
  StackElement *Next();
  myString *Data();
};

StackElement::StackElement(myString *s, StackElement *p)
{
  data = s;
  next = p;
}

StackElement *StackElement::Next()
{
  return next;
}

myString *StackElement::Data()
{
  return data;
}
```

```
class Stack
{
private:
  StackElement *head;
  int size;
public:
  Stack();
  ~Stack();
  myString *Pop();
  myString *Top();
  void Push(myString *);
  int GetSize();
  void Print();
  bool IsEmpty();
};

Stack::Stack():head(NULL),size(0)
{}

myString *Stack::Top()
{
  return head->Data();
}

myString *Stack::Pop()
{
  if (head == NULL){
    return NULL;
  }
  myString *s = head->Data();
  StackElement *temp = head;
  head = head->Next();
  delete temp;
  return s;
}
```

```
void Stack::Push(myString *s)
{
  StackElement *newElement = new StackElement(s,head);
  head = newElement;
}

int Stack::GetSize()
{
  return size;
}

bool Stack::IsEmpty(){
  if (head == NULL){
    return true;
  }
  return false;
}

void Stack::Print(){
  StackElement *el = head;
  while (el != NULL){
    cout << el->Data()->GetString() << endl;
    el = el->Next();
  }
}

Stack::~Stack()
{
  StackElement *temp = head;
  while (head != NULL){
    head = head->Next();
    delete temp;
    temp = head;
  }
}
```

```
int main(){
  ifstream fin("input.txt");
  Stack *nameStack = new Stack();
  char s[100];
  while (fin.getline(s,100)){
    cout << s << endl;
    myString *sp = new myString(s);
    if (sp->GetSize() <=5){
      nameStack->Push(sp);
    }
  }
  nameStack->Print();
  myString *first = nameStack->Top(); // if we deleted the strings
         // in the destructor first could end up pointing nowhere
  //delete nameStack; // Not a good idea since
                      //we will not delete the strings
  while (!nameStack->IsEmpty()){
    myString *sp = nameStack->Pop();
    delete sp;
  }
  fin.close();
}
```

Ο pointer **this**

- Ας υποθέσουμε ότι (για κάποιο λόγο) θέλουμε στο τελευταίο στοιχείο ο δείκτης next να δείχνει στον εαυτό του αντί για NULL.
  - Μπορούμε να το κάνουμε αυτό χρησιμοποιώντας τον δείκτη this, που επιστρέφει ένα δείκτη στο αντικείμενο που τον χρησιμοποιεί.

```
StackElement::StackElement(myString *s, StackElement *p)
{
  data = s;
  next = p;
  if (next == NULL) { next = this}
}
```

Ο pointer **this**

- Ο δείκτης this μπορεί να χρησιμοποιηθεί και σε άλλα σημεία για να κάνει τον κώδικα πιο ευανάγνωστο.
  - Ο παρακάτω κώδικας είναι ισοδύναμος με τον προηγούμενο.

```
StackElement::StackElement(myString *s, StackElement *p)
{
   this->data = s;
   this->next = p;
   if (next == NULL) { this->next = this}
}
```

AGGREGATION AND COMPOSITION

Σχέσεις μεταξύ κλάσεων

- Στο παράδειγμα με τη στοίβα έχουμε τρείς διαφορετικές κλάσεις (myString, Stack, StackElement) οι οποίες συσχετίζονται μεταξύ τους με διαφορετικούς τρόπους.
- Υπάρχουν πολλές διαφορετικές σχέσεις μεταξύ κλάσεων.
  - Στην περίπτωση μας, η μία κλάση ορίζεται χρησιμοποιώντας αντικείμενα της άλλης
- Αυτού του είδους τις σχέσεις τις χωρίζουμε σε δύο κατηγορίες: σχέσεις συνάθροισης (aggregation) σχέσεις σύνθεσης (composition)

Σχέση σύνθεσης – Composition

- Η κλάση X έχει σχέση σύνθεσης με την κλάση Y, αν το αντικείμενο της κλάσης X αποτελείται από αντικείμενο/α της κλάσης Y.
  - Τα αντικείμενα της κλάσης Y δεν υπάρχουν εκτός της κλάσης X.
  - Η κλάση X δημιουργεί και καταστρέφει τα αντικείμενα της κλάσης Y.
  - Τα αντικείμενα της κλάσης Y μπορούν να υπάρχουν ως μεταβλητές μέσα στον ορισμό της X.
- Παραδείγματα:
  - Ένας άνθρωπος αποτελείται από μέρη του σώματος: κεφάλι, πόδια, χέρια κλπ.
  - Ένα κτήριο αποτελείται από τοίχους, δωμάτια, πόρτες, κλπ.
- Στην περίπτωση μας η κλάση Stack έχει σχέση σύνθεσης με την κλάση StackElement.

UML διάγραμμα

- Η UML (Unified Modeling Language) είναι μια γλώσσα για να περιγράφουμε και να καταλαβαίνουμε τον κώδικα μας.
- Τα UML διαγράμματα παρέχουν μια οπτικοποίηση των σχέσεων μεταξύ των κλάσεων.

- Οι σχέσεις σύνθεσης συμβολίζονται όπως φαίνεται.

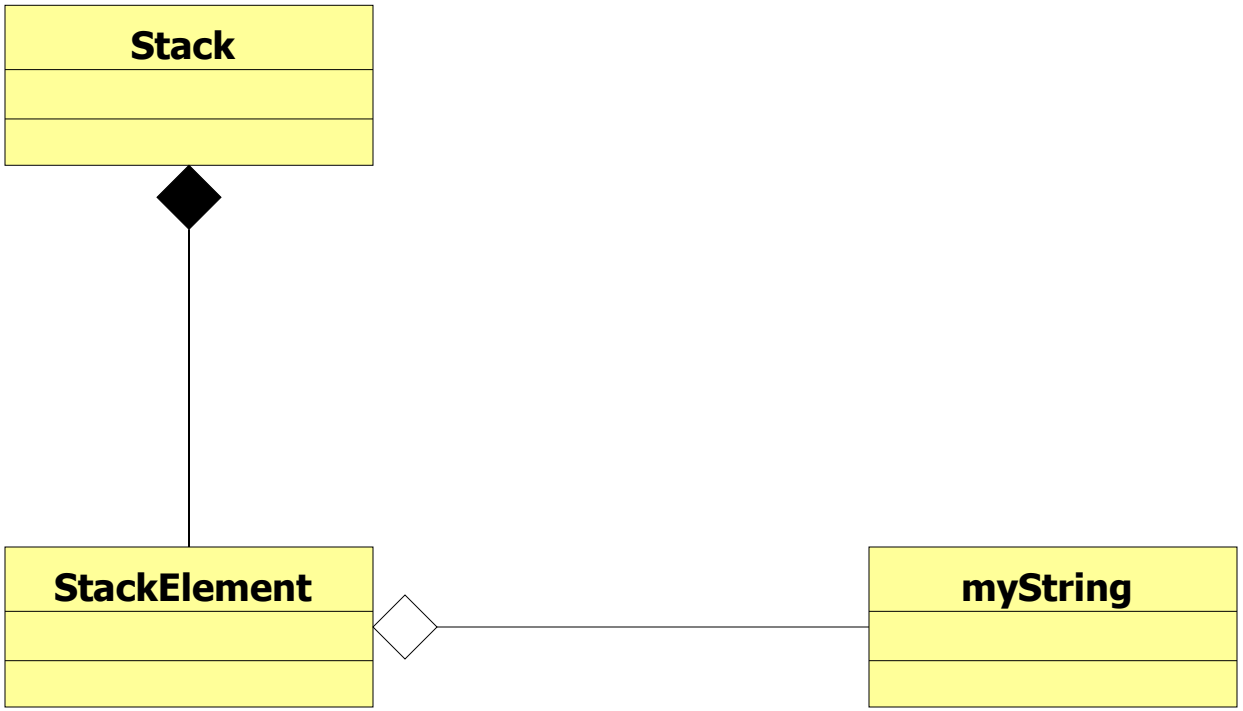
Σχέση συνάθροισης – Aggregation

- Η κλάση X έχει σχέση συνάθροισης με την κλάση Y, αν αντικείμενο/α της κλάσης Y ανήκουν στο αντικείμενο της κλάσης X.
  - Τα αντικείμενα της κλάσης Y έχουν υπόσταση και εκτός της κλάσης X.
  - Όταν καταστρέφεται ένα αντικείμενο της κλάσης X δεν καταστρέφονται και τα αντικείμενα της κλάσης Y.
  - Το αντικείμενο της κλάσης X έχει δείκτες στα αντικείμενα της κλάσης Y.
- Παραδείγματα:
  - Σε έναν άνθρωπο μπορεί να ανήκει ένα αυτοκίνητο, ρούχα, κλπ.
  - Ένα κτήριο μπορεί να έχει μέσα ανθρώπους, έπιπλα, κλπ.
- Στην περίπτωση μας η κλάση StackElement έχει σχέση συνάθροισης με την κλάση myString.

UML διάγραμμα

- Οι σχέσεις συνάθροισης συμβολίζονται όπως φαίνεται.

Όλο το UML διάγραμμα



Aggregation and Composition

- Το αν θα είναι μια σχέση, σχέση συναθροισης ή σύνθεσης εξαρτάται κατά πού και από την υλοποίηση μας και τον σχεδιασμό.
  - Π.χ., σε ένα διαφορετικό πρόγραμμα μπορεί να επαναχρησιμοποιούμε το StackElement.
  - Π.χ., σε μία διαφορετική εφαρμογή, τα ανθρώπινα όργανα υπάρχουν και χωρίς τον άνθρωπο.

΄ΕΝΑ ΜΕΓΑΛΟ ΠΑΡΑΔΕΙΓΜΑ

Άσκηση

- Θέλουμε να δημιουργήσουμε ένα λογισμικό για ένα τμήμα πανεπιστημίου. Το τμήμα έχει 10 φοιτητές οπού ο καθένας έχει ένα όνομα και ένα αριθμό μητρώου (ΑΜ), και 2 καθηγητές που ο καθένας έχει ένα όνομα και ένα ΑΦΜ. Το τμήμα δίνει 3 μαθήματα. Το κάθε μάθημα έχει κωδικό και όνομα. Το κάθε μάθημα ανατίθεται σε ένα καθηγητή. Οι φοιτητές μπορούν να γραφτούν σε κάποιο μάθημα και παίρνουν βαθμό. Θέλουμε να μπορούμε να τυπώσουμε τη λίστα των φοιτητών που παίρνουν το μάθημα και τους βαθμούς τους.

Άσκηση

- Θέλουμε να δημιουργήσουμε ένα λογισμικό για ένα τμήμα πανεπιστημίου.
- Το τμήμα έχει 5 φοιτητές οπού ο καθένας έχει ένα όνομα και ένα αριθμό μητρώου (ΑΜ).
- Το τμήμα έχει 2 καθηγητές που ο καθένας έχει ένα όνομα και ένα ΑΦΜ.
- Το τμήμα δίνει 3 μαθήματα. Το κάθε μάθημα έχει κωδικό και όνομα.
- Το κάθε μάθημα ανατίθεται σε ένα καθηγητή.
- Οι φοιτητές μπορούν να γραφτούν σε κάποιο μάθημα και παίρνουν βαθμό.
- Θέλουμε να μπορούμε να τυπώσουμε τις πληροφορίες του μαθήματος: το όνομα, τον καθηγητή και τη λίστα των φοιτητών που παίρνουν το μάθημα και τους βαθμούς τους.

Κλάσεις μέθοδοι και πεδία

- Ουσιαστικά:
  - Τμήμα
  - Φοιτητές
  - Καθηγητές
  - Μαθήματα
  - Όνομα
  - ΑΜ, ΑΦΜ, κωδικός
  - Βαθμός
  - Λίστα φοιτητών
- Τα ουσιαστικά είναι υποψήφια για κλάσεις ή πεδία

- Ρήματα:
  - Ανατίθεται
  - Εγγράφεται
  - Τυπώνει
  - Παίρνω βαθμό
- Τα ρήματα είναι υποψήφια για να γίνουν μέθοδοι και μηνύματα μεταξύ αντικειμένων.

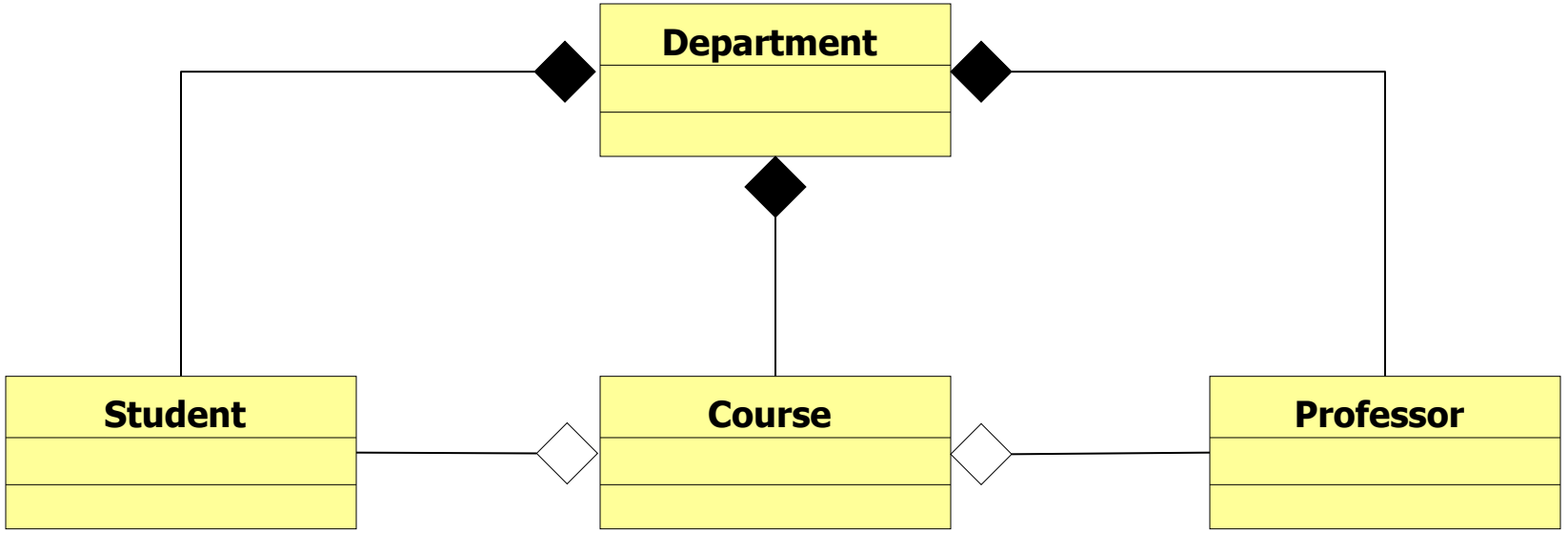
Κλάσεις μέθοδοι και πεδία

- Ουσιαστικά:
  - Τμήμα
  - Φοιτητές
  - Καθηγητές
  - Μαθήματα
  - Όνομα
  - ΑΜ, ΑΦΜ, κωδικός
  - Βαθμός
  - Λίστα φοιτητών
- Τα ουσιαστικά είναι υποψήφια για κλάσεις ή πεδία

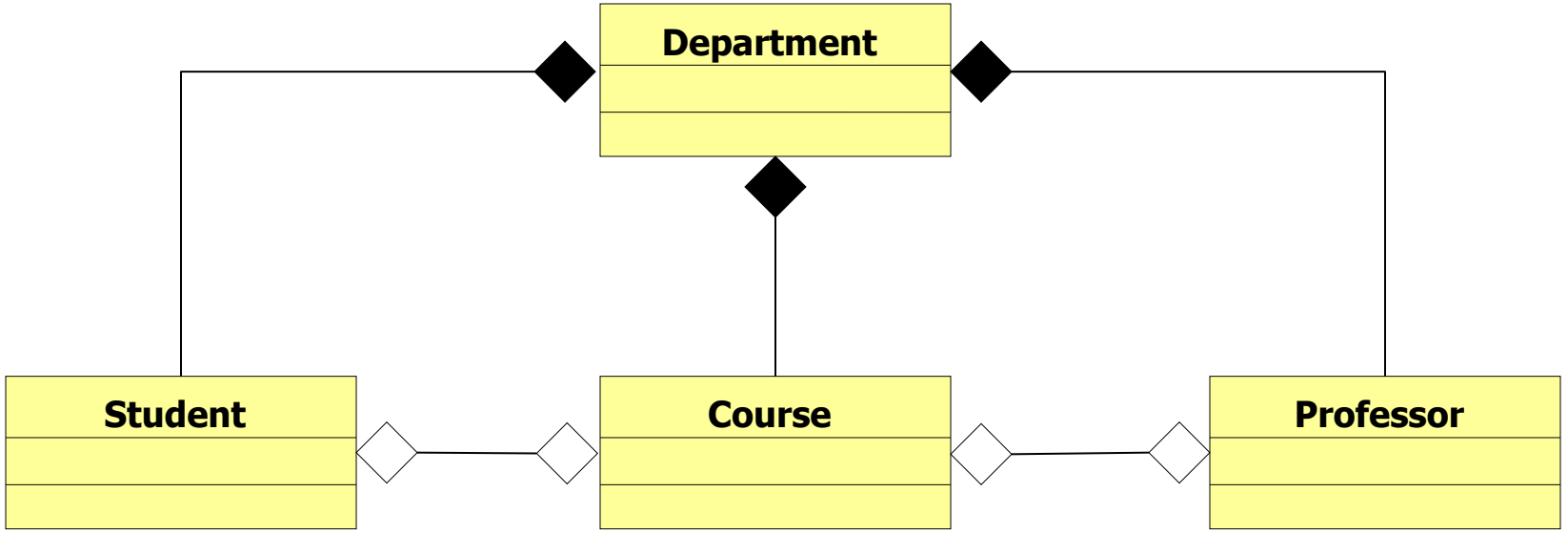
- Ρήματα:
  - Ανατίθεται
  - Εγγράφεται
  - Τυπώνει
  - Παίρνω βαθμό
- Τα ρήματα είναι υποψήφια για να γίνουν μέθοδοι και μηνύματα μεταξύ αντικειμένων.

Όλα τα ουσιαστικά μπορούν να γίνουν κλάσεις αλλά συνήθως διαλέγουμε αυτά για τα οποία υπάρχει αρκετή πολυπλοκότητα

UML διάγραμμα.



Μια παραλλαγή

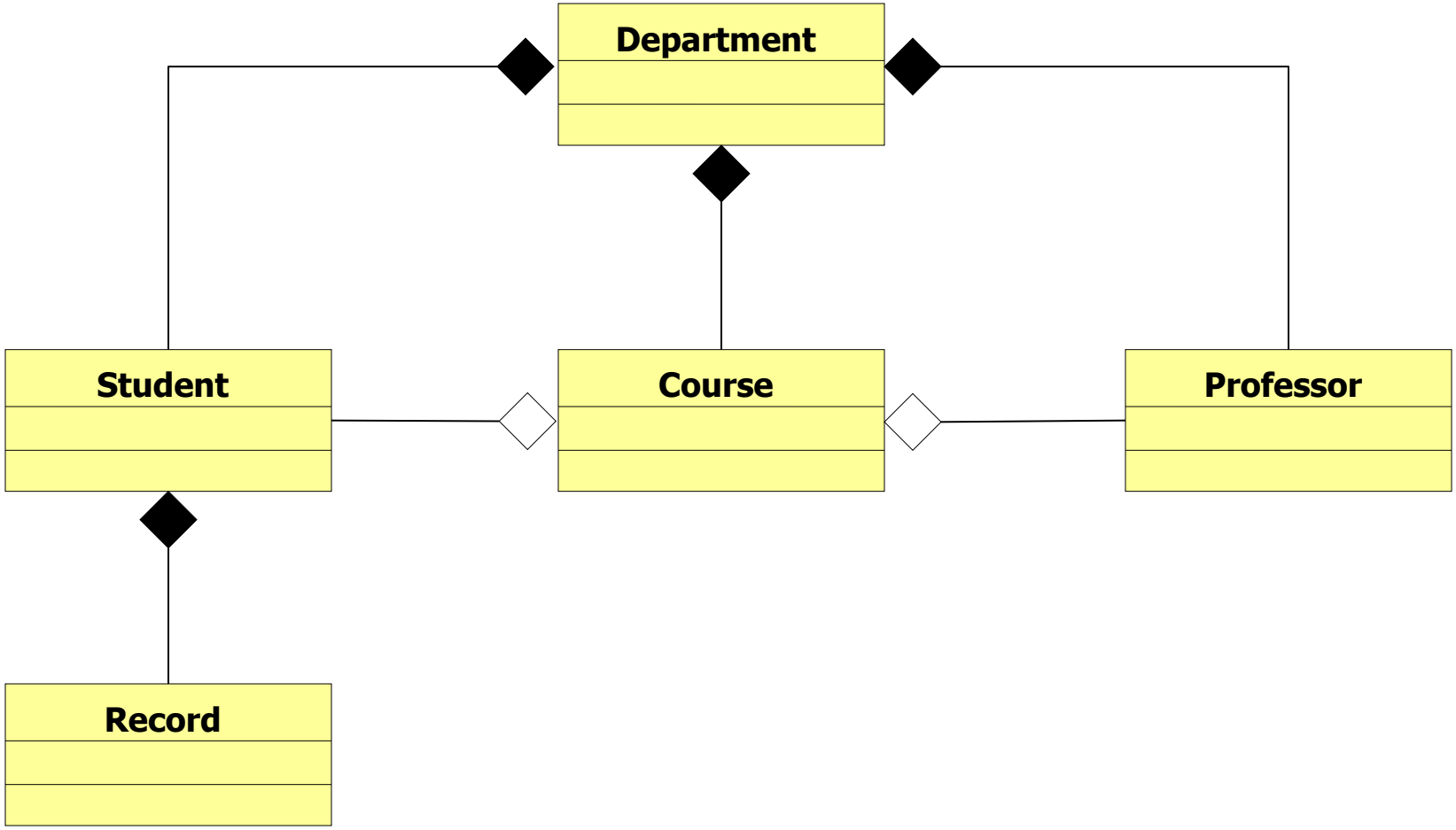


Για τους φοιτητές κρατάμε μια λίστα με τα μαθήματα που έχουν πάρει, και για τους καθηγητές μια λίστα με τα μαθήματα που διδάσκουν.

Άσκηση

- Θέλουμε να δημιουργήσουμε ένα λογισμικό για ένα τμήμα πανεπιστημίου.
- Το τμήμα έχει 5 φοιτητές οπού ο καθένας έχει ένα όνομα και ένα αριθμό μητρώου (ΑΜ).
- Το τμήμα έχει 2 καθηγητές που ο καθένας έχει ένα όνομα και ένα ΑΦΜ.
- Το τμήμα δίνει 3 μαθήματα. Το κάθε μάθημα έχει κωδικό και όνομα.
- Το κάθε μάθημα ανατίθεται σε ένα καθηγητή.
- Οι φοιτητές μπορούν να γραφτούν σε κάποιο μάθημα και παίρνουν βαθμό.
- Ο κάθε φοιτητής έχει ένα αρχείο με τους βαθμούς των μαθημάτων που έχει περάσει.
- Θέλουμε να μπορούμε να τυπώσουμε τις πληροφορίες του μαθήματος: το όνομα, τον καθηγητή και τη λίστα των φοιτητών που παίρνουν το μάθημα και τους βαθμούς τους.

UML διάγραμμα.



ΥΛΟΠΟΙΗΣΗ