ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Σύνθεση αντικειμένων

Παράδειγμα: Τμήμα πανεπιστημίου

Μεγάλο παράδειγμα

- Θέλουμε να δημιουργήσουμε ένα λογισμικό για ένα τμήμα πανεπιστημίου. Το τμήμα έχει 4 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 2 μαθήματα. Το κάθε μάθημα έχει κωδικό και όνομα και κάποιες διδακτικές μονάδες. Το κάθε μάθημα ανατίθεται σε ένα καθηγητή. Οι φοιτητές γράφονται σε κάποιο μάθημα και αν περάσουν το μάθημα παίρνουν τις μονάδες. Θέλουμε να μπορούμε να τυπώσουμε τις πληροφορίες για το μάθημα: το όνομα, τον καθηγητή και τη λίστα των φοιτητών που παίρνουν το μάθημα.

Μεγάλο Παράδειγμα

- Θέλουμε να δημιουργήσουμε ένα λογισμικό για ένα τμήμα πανεπιστημίου.
- Το τμήμα έχει 4 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 2 μαθήματα. Το κάθε μάθημα έχει κωδικό και όνομα, και κάποιες διδακτικές μονάδες.
- Το κάθε μάθημα ανατίθεται σε ένα καθηγητή.
- Οι φοιτητές γράφονται σε κάποιο μάθημα και αν περάσουν θα πάρουν τις μονάδες.
- Θέλουμε να μπορούμε να τυπώσουμε τις πληροφορίες του μαθήματος: το όνομα, τον καθηγητή και τη λίστα των φοιτητών που παίρνουν το μάθημα.

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
  - Περνάω μάθημα
  - Παίρνω μονάδες
- 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
  - Περνάω μάθημα
  - Παίρνω μονάδες
- 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

Κλάση Professor

- Κρατάει το όνομα και το ΑΦΜ του καθηγητή
- Ενδεχομένως να κρατάει και τα μαθήματα που έχει αναλάβει

- Η μέθοδος για να αναλάβει ο καθηγητής ένα μάθημα θα πρέπει να είναι εδώ ή στην κλάση του μαθήματος?

Κλάση Student

- Κρατάει το όνομα του φοιτητή και τις μονάδες που έχει πάρει μέχρι τώρα.
- Ενδεχομένως να κρατάει και τα μαθήματα που παίρνει.
- Ενδεχομένως να κρατάει και τη λίστα με τα μαθήματα που έχει περάσει.
- Χρειαζόμαστε μέθοδο για να γραφτεί ο φοιτητής στο μάθημα, ή να το περάσει, ή καλύτερα να τις βάλουμε στην κλάση του μαθήματος?

Κλάση Course

- Κρατάει το όνομα του μαθήματος, τις μονάδες του μαθήματος, τον καθηγητή που κάνει το μάθημα, τους φοιτητές που παίρνουν το μάθημα
  - Τίποτα άλλο? Τι θα κάνουμε με τους βαθμούς και το ποιος πέρασε το μάθημα?
- Μέθοδοι
  - Ανάθεση καθηγητή
  - Εγγραφή φοιτητή στο μάθημα
  - Ανάθεση βαθμών στους φοιτητές.

Κλάση Department

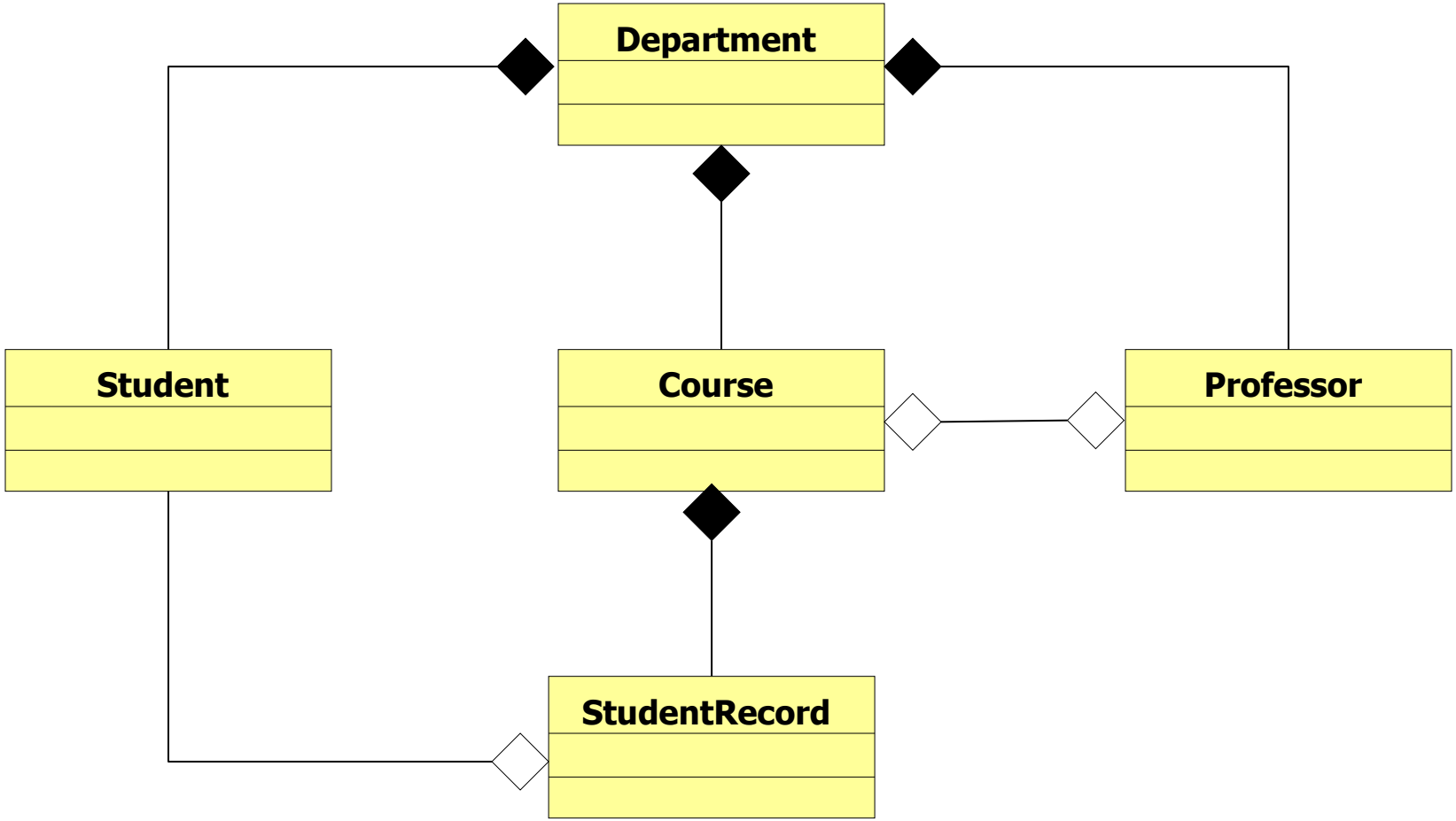
- Τα βάζει όλα μαζί, εδώ δημιουργούμε τους φοιτητές, καθηγητές, μαθήματα.
- Οι φοιτητές και οι καθηγητές ως άτομα θα μπορούσαν να υπάρχουν και εκτός του τμήματος.
- Εδώ δημιουργούμε την main.

- Χρειαζόμαστε άλλη κλάση?

Κλάση StudentRecord

- Χρειαζόμαστε να κρατάμε για κάθε φοιτητή τις πληροφορίες του (αυτά που έχουμε στο Student class) και το βαθμό του.
- Μας βολεύει να δημιουργήσουμε μια καινούρια κλάση που να βάζει μαζί αυτές τις πληροφορίες.

UML διάγραμμα



Αποθήκευση φοιτητών

- Η κλάση Course χρειάζεται να αποθηκεύσει τους φοιτητές που παίρνουν το μάθημα.
  - Δεν ξέρουμε εκ των προτέρων πόσοι φοιτητές θα πάρουν το μάθημα
- Θα χρησιμοποιήσουμε την κλάση ArrayList για να τους αποθηκεύσουμε.

ArrayList

- Μια βοηθητική κλάση είναι το ArrayList το οποίο είναι ένας δυναμικός πίνακας ο οποίος προσαρμόζει το μέγεθος του ανάλογα με τον αριθμό των στοιχείων που περιέχει
  - Το ArrayList μπορεί να κρατάει αντικείμενα οποιουδήποτε τύπου.
- Συντακτικό:
  - `import java.util.ArrayList;`
  - `ArrayList<Βασικος Τύπος> myList;`

- Ο βασικός τύπος είναι οποιοσδήποτε μια οποιαδήποτε κλάση.
  - Αυτός είναι ο τύπος των δεδομένων που αποθηκεύει ο πίνακας μας.
  - Για να αποθηκεύσουμε πρωταρχικούς τύπους (int, double, boolean) χρειαζόμαστε την wrapper class.

- Παραδείγματα:
  - `ArrayList<Integer> myList;` // λιστα από ακεραίους
  - `ArrayList<String> myList;` // λιστα από String
  - `ArrayList<Person> myList;` // λιστα από αντικείμενα Person

ArrayList

- Constructor
  - `ArrayList<T> myList = new ArrayList<T>();`
- Μέθοδοι
  - `add(T x)`: προσθέτει το στοιχειό `x` στο τέλος του πίνακα.
  - `add(int i, T x)`: προσθέτει το στοιχειό `x` στη θέση `i` και μετατοπίζει τα υπόλοιπα στοιχεία κατά μια θέση.
  - `remove(int i)`: αφαιρεί το στοιχείο στη θέση `i` και το επιστρέφει.
  - `remove(T x)`: αφαιρεί το στοιχείο
  - `set(int i, T x)`: θέτει στην θέση `i` την τιμή `x` αλλάζοντας την προηγούμενη
  - `get(int i)`: επιστρέφει το αντικείμενο τύπου `T` στη θέση `i`.
  - `size()`: ο αριθμός των στοιχείων του πίνακα.
- Διατρέχοντας τον πίνακα:
  - `ArrayList<T> myList = new ArrayList<T>();`
  - `for(T x: myList){…}`

ArrayList

- Διατρέχοντας τον πίνακα:

```
ArrayList<T> myList = new ArrayList<T>();
for(T x: myList){
   System.out.println(x);
}
```

- Εναλλακτικά

```
ArrayList<T> myList = new ArrayList<T>();
for(int i=0; i < myList.size(); i++){
   T x = myList.get(i);
   System.out.println(x);
}
```

Παράδειγμα ArrayList

```
class Player
{
   private String name;
   private int number;

   public Player(String name, int num){
          this.name = name;
          this.number = num;
   }

   public String toString(){
          return name+":"+number;
   }
}
```

```
import java.util.ArrayList;

class Team
{
   private ArrayList<Player> teamMembers
                  = new ArrayList<Player>();

   public void joinTeam(Player p){
     teamMembers.add(p);
   }

   public void leaveTeam(Player p){
      teamMembers.remove(p);
   }

   public void listMembers(){
     for (Player p: teamMembers){
         System.out.println(p);
      }
    }

   public static void main(String[] args){
      Team miami = new Team();
      Player lebron = new Player("Lebron", 6);
      miami.joinTeam(lebron);
      Player wade = new Player("Wade",3);
      miami.joinTeam(wade);
      Player bosh = new Player("Bosh",1);
      miami.joinTeam(bosh);
      miami.leaveTeam(lebron);
      miami.listMembers();
   }
}
```

```
public class Professor
{
	private String name;
	private int AFM;
	private Course lesson;

	public Professor(String name, int afm){
		this.name = name;
		this.AFM = afm;
	}

	public void setLesson(Course c){
		lesson = c;
	}

	public String toString(){
		return name + " " + AFM + " " + lesson;
	}
}
```

```
public class Student
{
	private String name;
	private int AM;
	private int units = 0;

	public Student(String name, int am){
		this.name = name;
		this.AM = am;
	}

	public int getAM(){
		return AM;
	}

	public void addUnits(int units){
		this.units += units;
	}

	public String toString(){
		return name +" AM:" + AM + " units:" + units;
	}
}
```

```
public class StudentRecord
{
        private Student student;
        private double grade;

        public StudentRecord(Student s){
                student = s;
        }

        public void setGrade(double grade){
                this.grade = grade;
        }

        public Student getStudent(){
                return student;
        }

        public String toString(){
                return student +" :" + grade;
        }

        public boolean passed(){
                if (grade >= 5){ return true;}
                return false;
        }
}
```

```
import java.util.ArrayList;
import java.util.Scanner;

public class Course
{
        private String name;
        private int code;
        private int units;
        private Professor prof;
        private ArrayList<StudentRecord> studentList
                                = new ArrayList<StudentRecord>();

        public Course(String name, int code, int units){
                this.name = name;
                this.code = code;
                this.units = units;
        }

        public void setProf(Professor p)
                prof = p;
                p.setLesson(this);
        }

        public void enroll(Student s){
                studentList.add(new StudentRecord(s));
        }
```

Χρησιμοποιούμε το this ως αναφορά στο παρόν αντικείμενο, ώστε να το προσθέσουμε στο αντικείμενο Professor

Δημιουργία του αντικειμένου StudentRecord και ταυτόχρονη προσθήκη στη λίστα
Λέγεται και «ανώνυμο αντικείμενο»

```
    public void assignGrades(){
            System.out.println("Give grades for course "+toString());
            Scanner input = new Scanner(System.in);
            for(StudentRecord record: studentList){
                System.out.println("Give grade for student "
                                   + record.getStudent().getAM() +":");
                double grade = input.nextDouble();
                record.setGrade(grade);
                if (record.passed()){
                    record.getStudent().addUnits(units);
                }
            }
        }

        public String toString(){
            return name + " " + code + "("+units + ")";
        }

        public void printInfo(){
           System.out.println("Course " + name
                              +" " + code + "("+units + ")");
           for (StudentRecord r: studentList){
              System.out.println(r);
           }
        }
}
```

Διασχίζουμε τη λίστα των φοιτητών

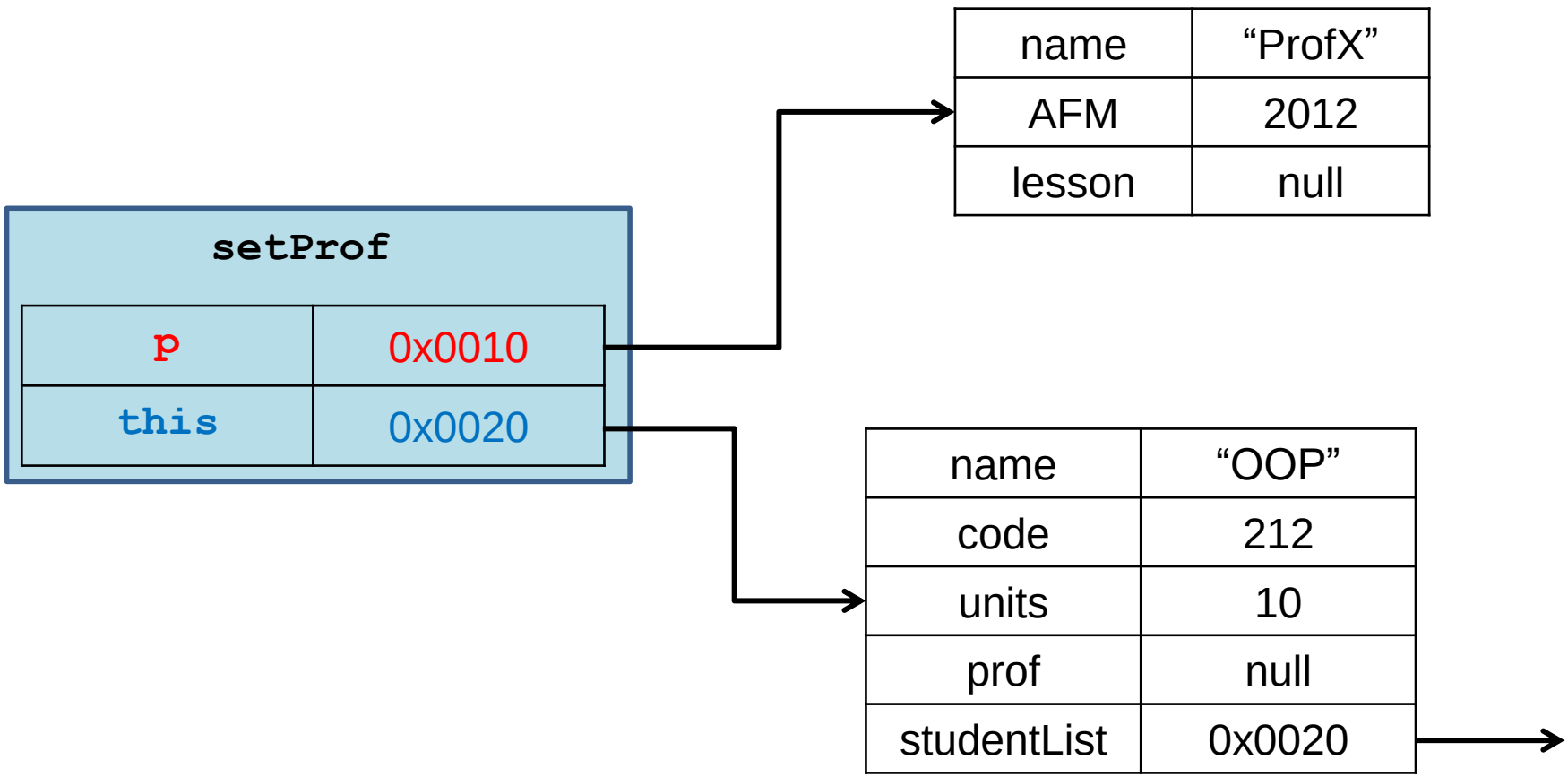
Αλυσιδωτές κλήσεις μεθόδων
Γίνεται εφόσον μια μέθοδος επιστρέφει αντικείμενο.

```
import java.util.Scanner;

class Department
{
  public static void main(String[] args)
  {
    int numOfStudents = Integer.parseInt(args[0]);

    Professor profX = new Professor("Prof X", 2012);
    Professor profY = new Professor("Prof Y", 2013);

    Course oop = new Course("oop", 212, 10);
    Course intro = new Course("intro", 101, 5);

    Student[] students = new Student[numOfStudents];
    Scanner input = new Scanner(System.in);
    for (int i = 0; i < numOfStudents; i ++){
          System.out.print("Give student name: ");
          String name = input.next();
          students[i] = new Student(name, i);
    }

    oop.setProf(profX);
    oop.enroll(students[0]);oop.enroll(students[1]);oop.enroll(students[3]);

    intro.setProf(profY);
    intro.enroll(students[2]); intro.enroll(students[3]);

    oop.assignGrades(); intro.assignGrades();

    System.out.println(profX); System.out.println(profY);
    oop.printInfo(); intro.printInfo();
  }
}
```

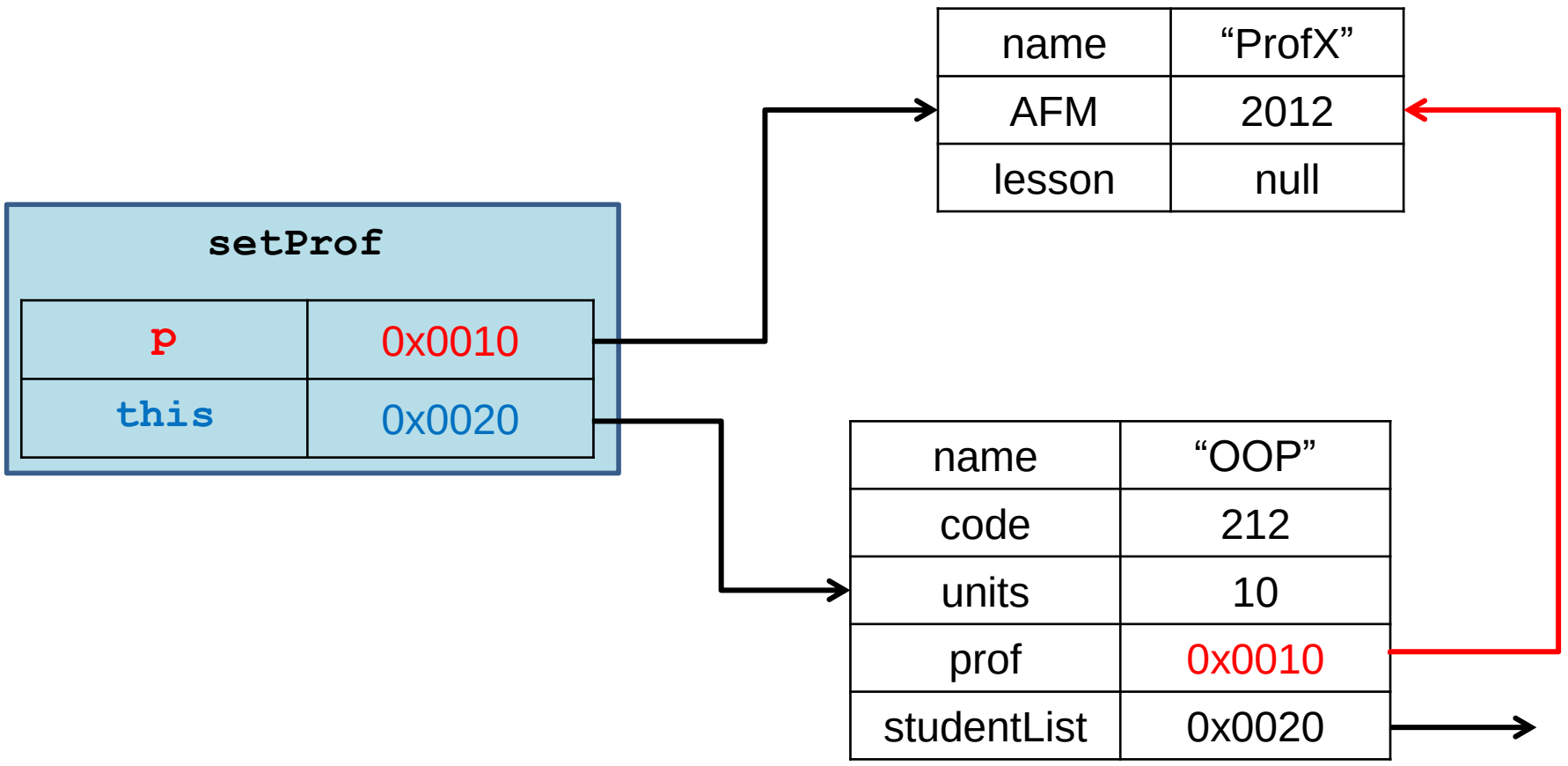
Χρησιμοποιούμε τις παραμέτρους εκτέλεσης (command line arguments) για να περάσουμε τον αριθμό των φοιτητών

Μετατρέπουμε το String σε ακέραιο με την μέθοδο Integer.parseInt

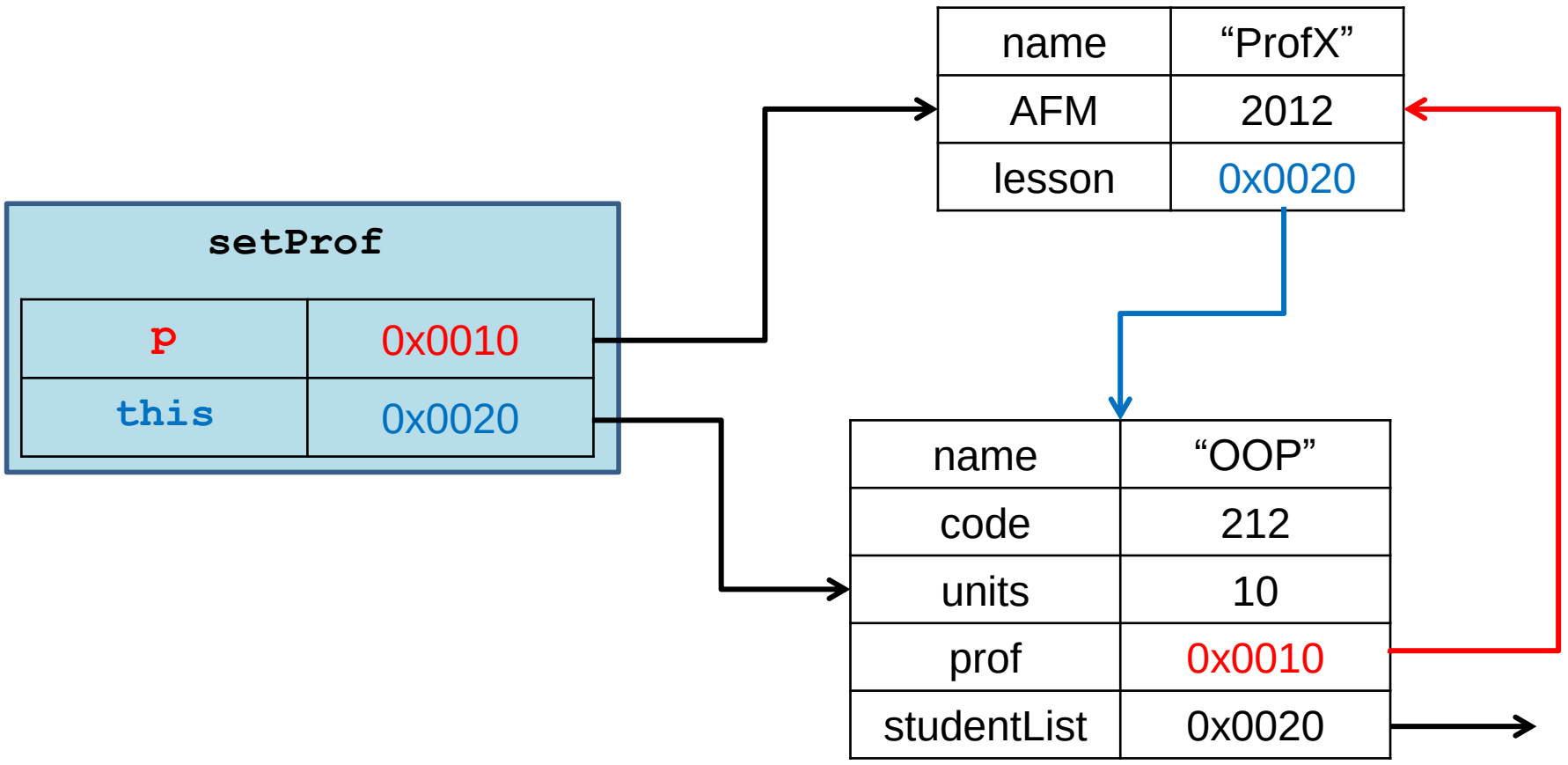
```
public void setProf(Professor p){
        prof = p;
        p.setLesson(this);
}
```



```
public void setProf(Professor p){
        prof = p;
        p.setLesson(this);
}
```



```
public void setProf(Professor p){
        prof = p;
        p.setLesson(this);
}
```



Παράδειγμα της χρήσης της μεταβλητής this

- Έχουμε την κλάση Person η οποία κρατάει για κάθε άνθρωπο το όνομα του, το ΑΦΜ του, και την ηλικία του.
- Μέσα στην κλάση Person θέλουμε να φτιάξουμε μια μέθοδο getOlderPerson η οποία παίρνει σαν όρισμα ένα άλλο Person και επιστρέφει το Person που είναι ηλικιακά μεγαλύτερο.

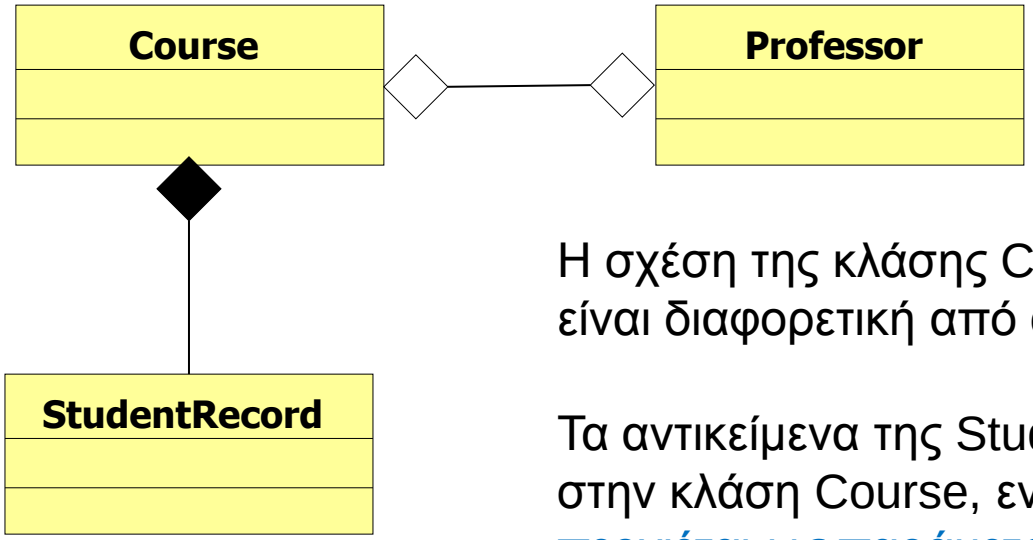
```
public class Person
{
	private String name;
	private int age;

	public Person(String name, int age){
		this.name = name;
		this.age = age;
	}

	public Person getOlderPerson(Person other){
		if(other.age > this.age){
			return other;
		}else{
			return this;
		}
	}
}
```

Η μέθοδος μας επιστρέφει μια αναφορά σε αντικείμενο Person

Αν η ηλικία του ατόμου που καλεί την μέθοδο είναι μεγαλύτερη αυτού που περνάμε σαν όρισμα επιστρέφουμε την αναφορά this

Αλλιώς επιστρέφουμε την αναφορά other.

Σχέσεις κλάσεων



Η σχέση της κλάσης Course με την StudentRecord είναι διαφορετική από αυτή με την Professor

Τα αντικείμενα της StudentRecord δημιουργούνται μέσα στην κλάση Course, ενώ το αντικείμενο Professor περνιέται ως παράμετρος στην setProf

Προσοχή: Σε πολλά βιβλία και οι δύο σχέσεις αναφέρονται ως σχέση σύνθεσης!
Υπάρχει ποιοτική διαφορά παρότι το όνομα μπορεί να μην διαφέρει

Κάποιες φορές, η πρώτη σχέση λέγεται σχέση σύνθεσης και η δεύτερη σχέση συνάθροισης

Η σχέση Course και Professor είναι αμφίδρομη μιας και κρατάμε το αντικείμενο Course μέσα στην Professor

Αν θέλουμε ο φοιτητής να κρατάει πληροφορία για το ποια μαθήματα παίρνει

```
public class Student
{
        private String name;
        private int AM;
        private int units = 0;
        ArrayList<Course> courses = new ArrayList<Course>();

        public Student(String name, int am){
                this.name = name;
                this.AM = am;
        }

        public String getName(){
                return name;
        }

        public void addUnits(int units){
                this.units += units;
        }

        public void addCourse(Course c){
                courses.add(c);
        }

        public String toString(){
                return name +" AM:" + AM + " units:" + units;
        }
}
```

```
import java.util.ArrayList;
import java.util.Scanner;

public class Course
{
        private String name;
        private int code;
        private int units;
        private Professor prof;
        private ArrayList<StudentRecord> studentList
                                = new ArrayList<StudentRecord>();

        public Course(String name, int code, int units){
                this.name = name;
                this.code = code;
                this.units = units;
        }

        public void setProf(Professor p){
                prof = p;
                p.setLesson(this);
        }

        public void enroll(Student s){
                studentList.add(new StudentRecord(s));
                s.addCourse(this);
        }
```

Αναζήτηση

- Τι γίνεται αν θέλουμε να μπορούμε να ζητήσουμε τον βαθμό ενός φοιτητή για ένα μάθημα?
  - Η κλάση Course θα πρέπει να μπορεί με το ΑΜ του φοιτητή να μας επιστρέφει τον βαθμό.
  - Για τέτοιου είδους αναζητήσεις βολεύει να χρησιμοποιούμε ένα λεξικό.
  - Η Java μας προσφέρει την κλάση HashMap που υλοποιεί ένα λεξικό.

HashMap (JavaDocs link)

- Αποθηκεύει ζευγάρια από κλειδιά και τιμές
- Constructors
  - `HashMap<K,V> myMap = new HashMap<K,V>();`
- Μέθοδοι
  - `put(K key, V value)`: προσθέτει το ζευγάρι (`key,value`) (δημιουργεί μία συσχέτιση)
  - `V get(K key)`: επιστρέφει την τιμή για το κλειδί `key`.
  - `remove(K key)`: αφαιρεί το ζευγάρι με κλειδί `key`.
  - `containsKey(K key)`: boolean αν το σύνολο περιέχει το κλειδί `key` ή όχι.
  - `containsValue(V value)`: boolean αν το σύνολο περιέχει την τιμή `value` ή όχι. (αργό)
  - `size()`: ο αριθμός των στοιχείων (κλειδιών) στο map.
  - `isEmpty()`: boolean αν έχει στοιχεία το map ή όχι.
  - `Set<K> keySet()`: επιστρέφει ένα Set με τα κλειδιά.
  - `Collection<V> values()`: επιστρέφει ένα Collection με τις τιμές

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Course
{
        private String name;
        private int code;
        private int units;
        private Professor prof;
        private ArrayList<StudentRecord> studentList
                                = new ArrayList<StudentRecord>();
        private HashMap<Integer,StudentRecord> studentMap
                        = new HashMap<Integer,StudentRecord>();

        public void enroll(Student s){
                studentRecord sRecord = new StudentRecord(s);
                studentList.add(sRecord);
                studentMap.put(s.getAM(),sRecord);
        }

        public double getGrade(int AM){
                if (studentMap.contains(AM)){
                        StudentRecord sRecord = studentMap.get(AM);
                        return sRecord.getGrade();
                }else{
                        System.out.println("Student "+ AM + " not enrolled");
                        return -1;
                }
        }
}
```

Έχοντας το λεξικό μπορούμε να κάνουμε διάφορες αναζητήσεις με το ΑΜ του φοιτητή

Η ίδια αναφορά αποθηκεύεται σε δύο σημεία