ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Σύνθεση αντικειμένων

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.

ArrayList

- Μια βοηθητική δομή είναι το ArrayList το οποίο είναι ένας δυναμικός πίνακας ακριβώς όπως αυτός που υλοποιήσατε στην άσκηση
  - Το ArrayList μπορεί να κρατάει αντικείμενα οποιουδήποτε τύπου.
- Συντακτικό:
  - `import java.util.ArrayList;`
  - `ArrayList<Βασικος Τύπος> myList;`

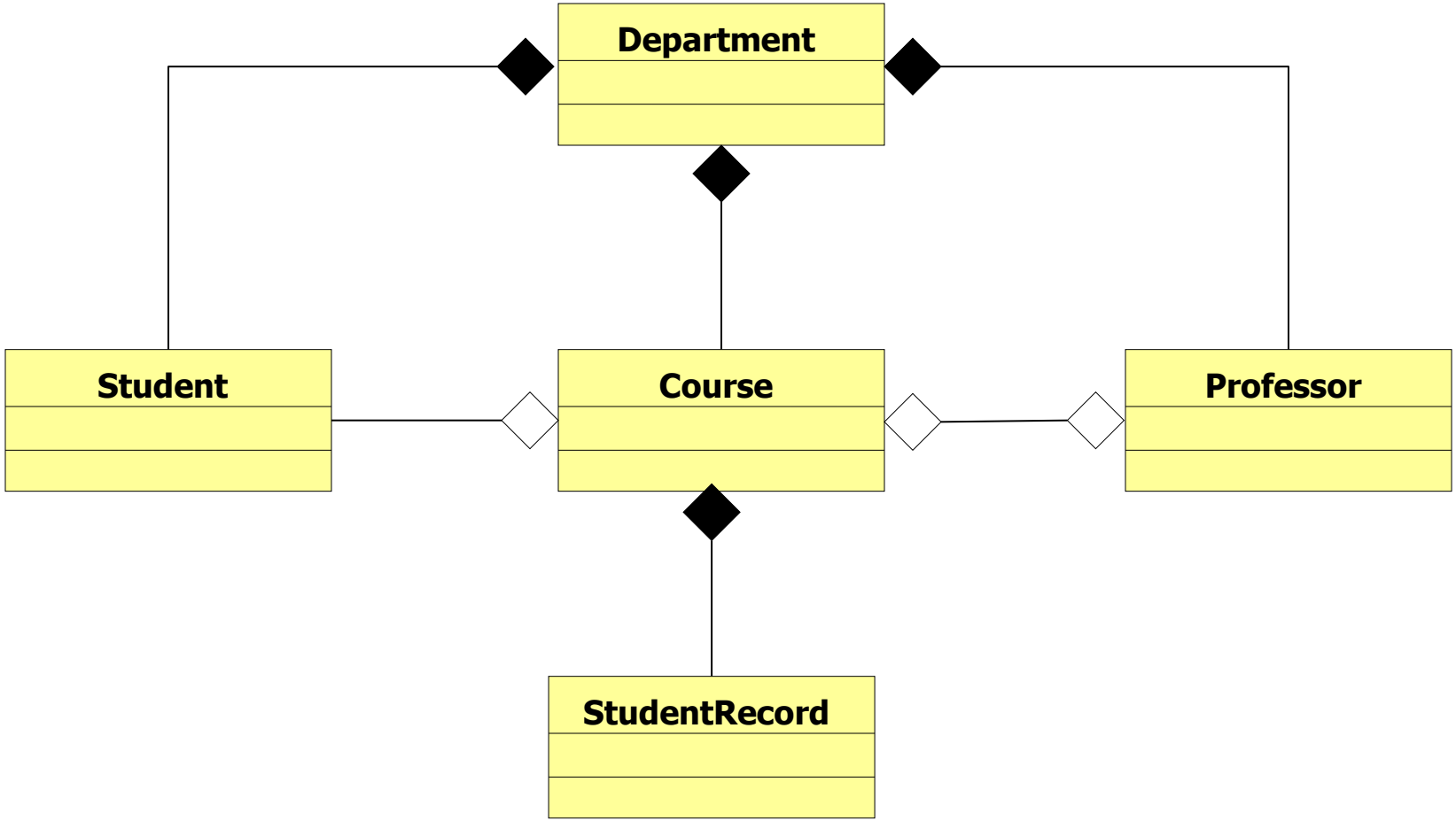
- Ο βασικός τύπος είναι οποιοσδήποτε μια οποιαδήποτε κλάση.
  - Αυτός είναι ο τύπος των δεδομένων που αποθηκεύει ο πίνακας μας.
  - Για να αποθηκεύσουμε βασικούς τύπους χρειαζόμαστε την wrapper class.

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

ArrayList

- Constructors
  - `ArrayList<T> myList = new ArrayList<T>();`
  - `ArrayList<T> myList = new ArrayList<T>(10);`
    `//λιστα με χωρητικότητα 10`
- 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`
  - `set(int i, T x)`: αλλάζει την τιμή της θέσης `i` με την τιμή `x`
  - `get(int i)`: επιστρέφει την τιμή στη θέση `i`.
  - `size()`: ο αριθμός των στοιχείων του πίνακα.
- Διατρέχοντας τον πίνακα:
  - `ArrayList<T> myList = new ArrayList<T>();`
  - `for(T x: myList){…}`

UML διάγραμμα



```
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 String getName(){
		return name;
	}

	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 "
                                    + sr.getStudent().getName() +":");
                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);
           }
        }
}
```

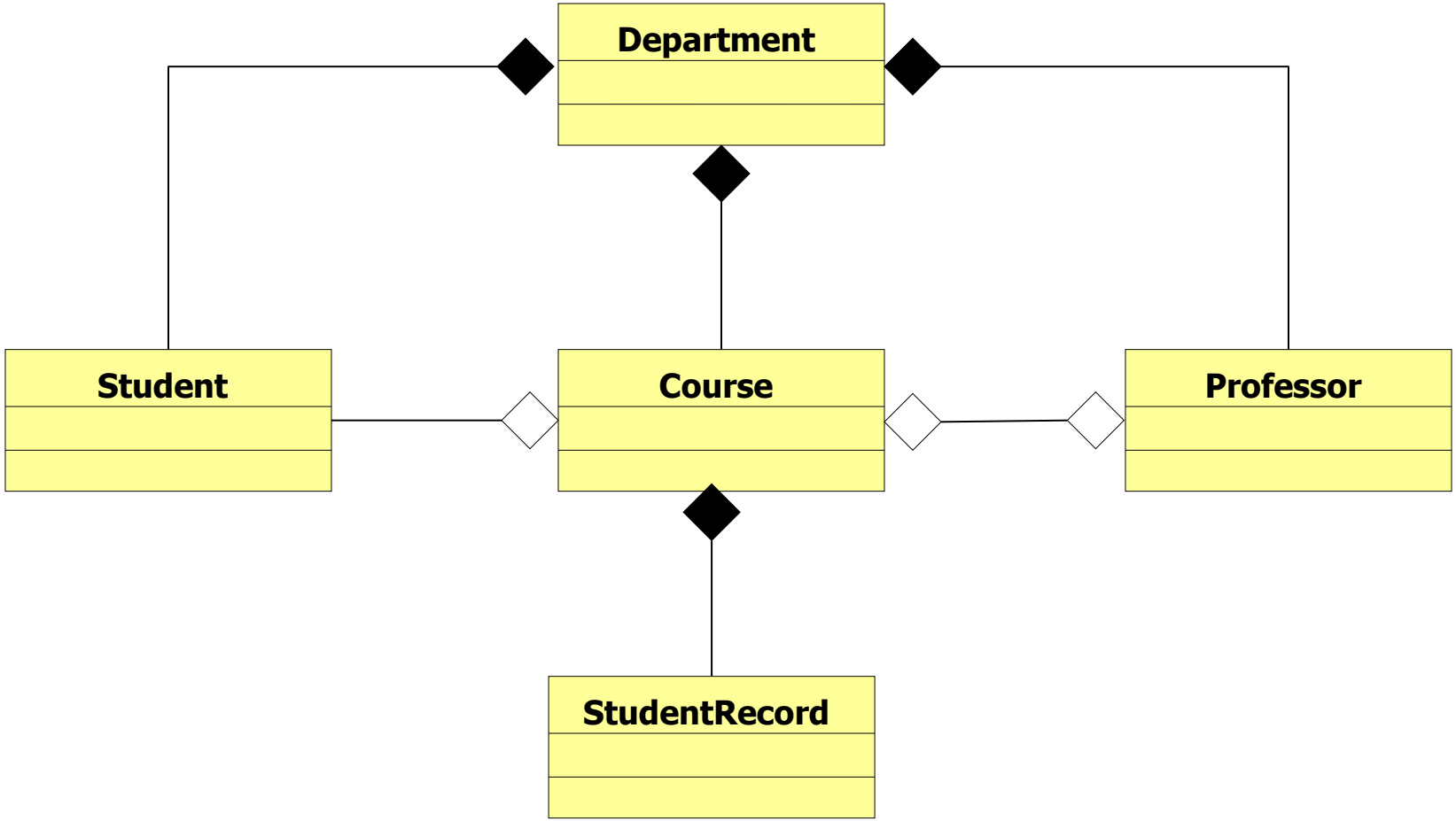
Αλυσιδωτές κλήσεις μεθόδων
Γίνεται εφόσον μια μέθοδος επιστρέφει αντικείμενο.

```
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();
  }
}
```

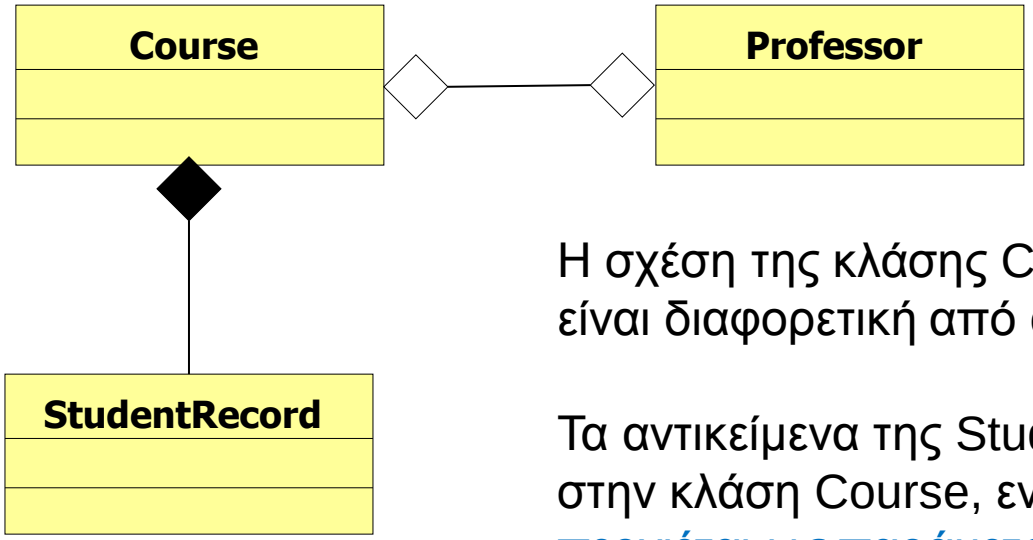
Χρησιμοποιούμε τις παραμέτρους εκτέλεσης (command line arguments) για να περάσουμε τον αριθμό των φοιτητών

Μετατρέπουμε το String σε ακέραιο με την μέθοδο Integer.parseInt

UML διάγραμμα



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

Παράδειγμα

```
class Examination
{
  String examinerName;
  Date examDate;

  public Examination(String name, int day, int month, int year)
  {
    examinerName = name;
    examDate = new Date(day, month, year);
  }

  …

}
```

Στην περίπτωση αυτή έχουμε μια σχέση σύνθεσης μεταξύ της κλάσης Examiner και Date. Η ημερομηνία δημιουργείται στον constructor της Examiner

Παράδειγμα

```
class Examination
{
  String examinerName;
  Date examDate;

  public Examination(String name, Date examDate)
  {
    examinerName = name;
    this.examDate = examDate;
  }
  …

}

Class Exam
{
  public void static main(String[] args){
        Date today = new Date(8,4,2103);
        Examination exam = new Examination("Alice", today);
        …
  }
}
```

Στην περίπτωση αυτή έχουμε μια σχέση συνάθροισης μεταξύ της κλάσης Examiner και Date. Η ημερομηνία δημιουργείται εκτός της Examiner και περνιέται σαν όρισμα

Παράδειγμα

```
class Examination
{
  String examinerName;
  Date examDate;

  public Examination(String name, Date examDate)
  {
    examinerName = name;
    this.examDate = examDate;
  }
  …

}

Class Exam
{
  public void static main(String[] args){
        Examination exam = new Examination("Alice", new Date(8,4,2103););
        …
  }
}
```

Αυτή περίπτωση είναι κάπου μεταξύ των δύο, εξαρτάται από το πρόγραμμα μας.