Διάλεξη 11:
Αντικειμενοστρεφής Σχεδιασμός ΙΙΙ

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

- Παράδειγμα Διαδικασίας Ανάπτυξης Λογισμικού: Βιβλίο Διευθύνσεων (συν.)
- Φάση 3: Υλοποίηση (αναλυτικά διαγράμματα κλάσης-detailed class diagrams, διαγράμματα ανάπτυξης-deployment diagrams), Φάση 4: Συντήρηση, Εξέλιξη
- UML, Παραδείγματα

Διδάσκων: Παναγιώτης Ανδρέου

Φάση 3: Υλοποίηση

Φάση της μετατροπής του αρχικού σχεδιασμού σε πρόγραμμα προς μετάφραση και εκτέλεση

3Α. Αναγνώριση Τμήματος Πυρήνα

- Πυρήνας: Στόχος εδώ είναι να εντοπίσετε το τμήμα που αποτελεί τον πυρήνα του προγράμματός σας, από τον οποίο θα προκύψει το τελικό σύστημα.
- Ο πυρήνας μπορεί να μην περιέχει τα πάντα, είναι όμως η βάση για το χτίσιμο των λειτουργιών του συστήματός σας.

Φάση 3: Υλοποίηση (συν.)

3Β. Πρόσθεση Λειτουργιών

- Μετά το κτίσιμο του πυρήνα, γίνεται η υλοποίηση των λειτουργιών του συστήματος.
- Κάθε ξεχωριστή λειτουργία αντιπροσωπεύεται στη φάση τού σχεδιασμού από μια αντίστοιχη περίπτωση χρήσης (use case).
- Η υλοποίηση των διαφορετικών περιπτώσεων χρήσης γίνεται διαδοχικά – σε διαφορετικές επαναλήψεις – και μέχρι να ολοκληρωθεί η υλοποίηση όλων των συστατικών του συστήματος.
- Πως δικαιολογείται η «επαναληπτική» προσέγγιση (iteration over the use cases);

Διαγράμματα Κλάσης (class diagrams)

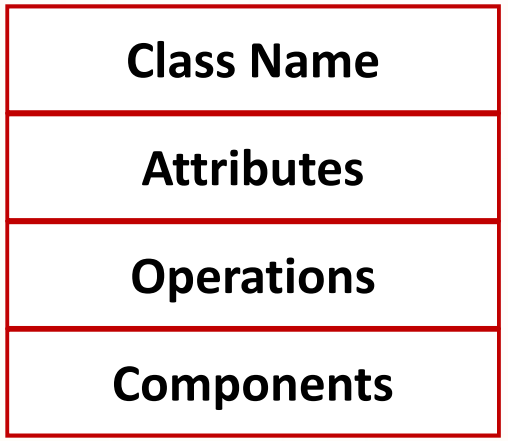
- Ένα από τα κυριότερα εργαλεία αντικειμενοστρεφής μοντελοποίησης
- **Περιγράφει** τη δομή του συστήματος παρουσιάζοντας:
  - τις **κλάσεις του συστήματος**
  - τις **σχέσεις μεταξύ κλάσεων**

- Δύο είδη διαγραμμάτων:
  - **Γενικό εννοιολογικό μοντέλο (ΓΜ):** χρησιμοποιείται στην ανάλυση και σχεδίαση
  - **Αναλυτικό μοντέλο (ΑΜ):** χρησιμοποιείται για μετάφραση των μοντέλων σε πραγματικό κώδικα

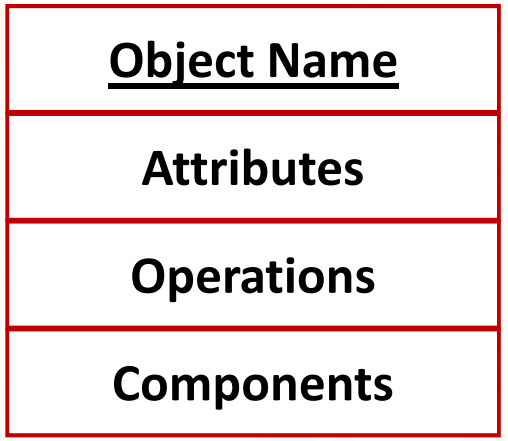
Διαγράμματα Κλάσης (class diagrams) (συν.)

Αναλυτικό μοντέλο

- **Διάγραμμα Κλάσης**
  Κάθε κλάση χωρίζεται σε 4 μέρη
  1. Το όνομα της κλάσης (class name)
  2. Τα χαρακτηριστικά (attributes) της κλάσης
  3. Τις λειτουργίες (operations) της κλάσης
  4. (Προαιρετικό) Επιπρόσθετα συστατικά (additional components) της κλάσης



- **Διάγραμμα αντικειμένου**
  - Όμοιο με Διάγραμμα Κλάσης
  - Όνομα αντικειμένου αντί Όνομα Κλάσης
  - Όνομα αντικειμένου υπογραμμισμένο



Προσδιοριστές για Διαγράμματα Κλάσης

- **Προσδιοριστές Ορατότητας (Visibility Modifiers)**
  - **(+) public :** Η πρόσβαση είναι **ανοικτή σε όλους!**
  - **(-) private:** Η πρόσβαση είναι **περιορισμένη μόνο στην κλάση**
  - **(~) package (friendly):** Η πρόσβαση είναι **περιορισμένη σε κλάσεις που βρίσκονται στο ίδιο πακέτο με την κλάση**
  - **(#) protected: Όπως το default** αλλά **και σε κλάσεις που κληρονομούν από την ίδια κλάση** (εντός πακέτου ή όχι)

Προσδιοριστές για Διαγράμματα Κλάσης

- **Άλλοι Προσδιοριστές**
  - **(underline) static :** Τα δεδομένα, μεθόδοι **ισχύουν για όλα τα αντικείμενα της κλάσης**
  - **(/) derived:** Τα δεδομένα, μεθόδοι **κληρονομούνται από άλλες κλάσεις**
  - **(italic) abstract: Οι μεθόδοι πρέπει να υλοποιηθούν**

Συμβολισμός	Επεξήγηση
+	Public
-	Private
#	Protected
~	Package
/	Derived
<underline>	Static
<italic>	Abstract

Βιβλίο Διευθύνσεων: Class Diagrams (αναλυτικά)

Person

- firstName: String
- lastName: String
- address: String
- city: String
- state: String
- zip: String
- phone: String

+ Person(String firstName, String lastName, String address, String city, String state, String zip, String phone)
+ getFirstName(): String
+ getLastName(): String
+ getAddress(): String
+ getCity(): String
+ getState(): String
+ getZip(): String
+ getPhone(): String

Βιβλίο Διευθύνσεων: Class Diagrams (αναλυτικά)

FileSystem
+ readFile(File file): AddressBook + saveFile(AddressBook addressBook, File file)

Βιβλίο Διευθύνσεων: Class Diagrams (αναλυτικά)

AddressBookApplication
- fileSystem: FileSystem - controller: AddressBookController
+ main() + quitApplication()

Βιβλίο Διευθύνσεων: Class Diagrams (αναλυτικά)

AddressBook

- collection: Person [ ] or Vector or ArrayList
- count: int
- file: File
- changedSinceLastSave: boolean

+ AddressBook()
+ getNumberOfPersons(): int
+ addPerson(String firstName, String lastName, …)
+ getFullNameOfPerson(int index): String
+ updatePerson(int index, String address, String city, String state, …)
+ removePerson(int index)
+ sortByName()
+ sortByZip()
+ printAll()
+ sortByName()
+ getFile(): File
+ getTitle(): String
…

Βιβλίο Διευθύνσεων: Class Diagrams (αναλυτικά)

AddressBookGUI
Εργασία για το σπίτι
Εργασία για το σπίτι

Βιβλίο Διευθύνσεων: Υλοποίηση

```
import java.io.*;
import java.util.*;

public class AddressBook {

    private Vector collection;
    private File file;
    private boolean changedSinceLastSave;

    public AddressBook() {
        collection = new Vector();
        file = null;
        changedSinceLastSave = false;
    }

    public int getNumberOfPersons() {
        return collection.size();
    }

    public void addPerson(String s, String s1,
String s2, String s3, String s4, String s5, String
s6)    {
        collection.addElement(new Person(s, s1,
s2, s3, s4, s5, s6));
        setChangedSinceLastSave(true);
    }
    public String getFullNameOfPerson(int i)    {
        Person person =
(Person)collection.elementAt(i);
        return person.getFirstName() + " " +
person.getLastName();
    }
public void updatePerson(int i, String s, String
s1, String s2, String s3, String s4)    {
        (Person)collection.elementAt(i)).update(s,
s1, s2, s3, s4);
        setChangedSinceLastSave(true);
    }
```

Βιβλίο Διευθύνσεων: Υλοποίηση

```
public class Person {

    private String firstName;
    private String lastName;
    private String address;
    private String city;
    private String state;
    private String zip;
    private String phone;

    public Person(String s, String s1, String s2,
String s3, String s4, String s5, String s6)
    {       firstName = s;      lastName = s1;
        address = s2; city = s3; state = s4;
        zip = s5; phone = s6;  }
```

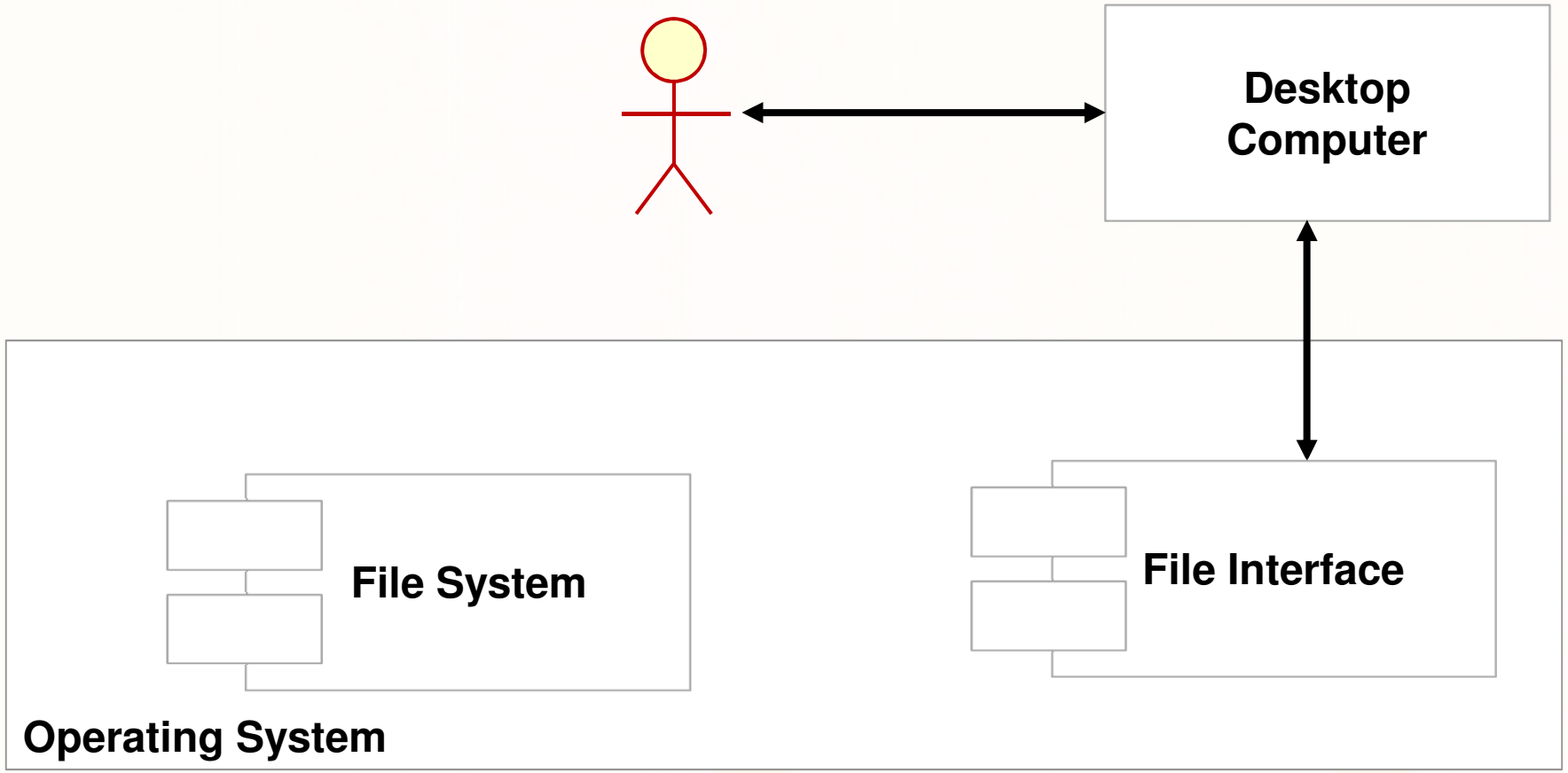
```
    public String getFirstName()    {
        return firstName;
    }
    public String getAddress()    {
        return address;
    }
    …
    public void update(String s, String s1, String
s2, String s3, String s4)
    {
        address = s;
        city = s1;
        state = s2;
        zip = s3;
        phone = s4;
    }

}
```

Διαγράμματα Ανάπτυξης (deployment diagrams)

- Παρουσιάζει πως θα αναπτυχθεί/εγκατασταθεί το λογισμικό στον πελάτη/οργανισμό.
- Παρουσιάζει και το υλικό (hardware), π.χ., εξυπηρετητές.
- Οι κόμβοι παρουσιάζονται σαν κουτιά
- Τα τεχνουργήματα (artifacts) που υπάρχουν σε κάθε κόμβο παρουσιάζονται σαν ορθογώνια μέσα στα κουτιά
- Οι κόμβοι μπορούν να έχουν υπο-κόμβους
- Δύο είδη κόμβων:
  - Κόμβος Συσκευής (Device Node)
  - Κόμβος Περιβάλλον Εκτέλεσης (Execution Environment Node)

Βιβλίο Διευθύνσεων: Deployment Diagram



Παράδειγμα: Βιβλίο Διευθύνσεων

Επόμενη Φάση 4: Συντήρηση

Φάση 4: Συντήρηση

- Η φάση της συντήρησης ή εξέλιξης του κώδικα:
  - Διόρθωση σφαλμάτων.
  - Αλλαγές στον κώδικα με βάση την εμπειρία χρήσης του.
  - Πρόσθεση νέων λειτουργιών.
  - Καλύτερη ικανοποίηση των απαιτήσεων.

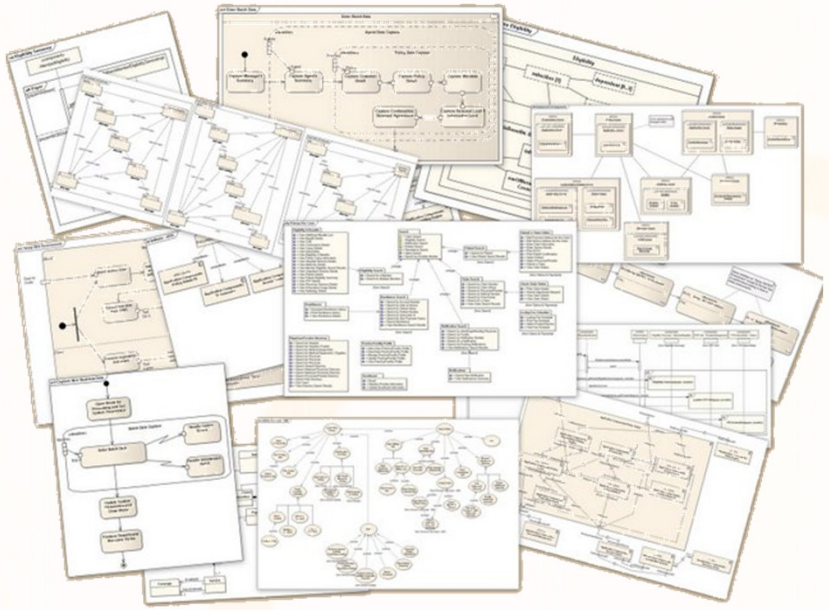
Βιβλίο Διευθύνσεων: Συντήρηση/Εξέλιξη

- Το use case Print Entries τυπώνει τα ονόματα στο System.out. Διαφορετικά θα μπορούσε να γράφει σε αρχείο.
- Το πρόγραμμα μπορεί να διαχειρίζεται πολλαπλά βιβλία διευθύνσεων. Αυτό συμπεριλαμβάνει τις εξής αλλαγές:
  - Το use case Create New Address Book και Open Existing Address Book Use Cases δεν θα κλείνουν το τρέχον address book. Αντίθετα θα δημιουργούν ένα καινούριο αντίγραφο του GUI και θα το διαχειρίζονται ξεχωριστά.
  - Θα δημιουργηθεί ένα καινούριο use case (Close Address Book) το οποίο θα εκτελείται όταν κλείνει ένα address book αντί του use case Quit Program.

UML: Unified Modelling Language

UML: Unified Modelling Language

- **Τυποποιημένη γλώσσα μοντελοποίησης γενικού σκοπού** στο πεδίου της αντικειμενοστρεφή ανάπτυξης λογισμικού (object-oriented software engineering).
- Δημιουργήθηκε από το Object Management Group το 1997
- Η UML περιλαμβάνει ένα σύνολο από διαγραμματικές τεχνικές αναπαράστασης για να δημιουργήσει visual models

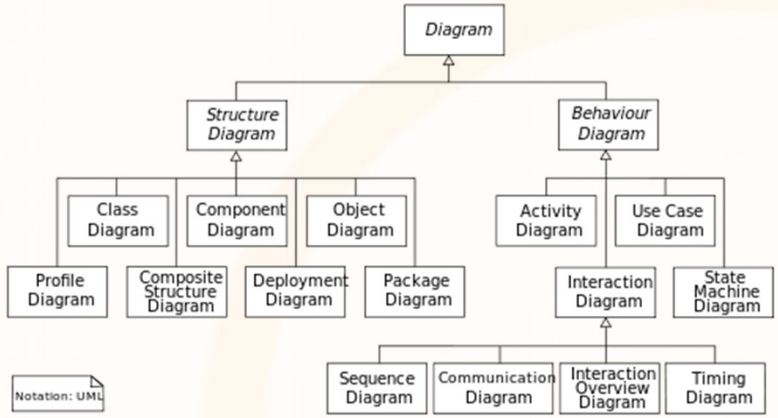


Κατηγορίες Διαγραμμάτων UML

UML 2.2: 14 διαγράμματα

Δύο βασικές κατηγορίες διαγραμμάτων:

- **Διαγράμματα Δομής (structure diagrams)**
  Δίνουν έμφαση στο τι πρέπει να είναι παρόν στο σύστημα που μοντελοποιείται, δηλ., την αρχιτεκτονική του.
- **Διαγράμματα Συμπεριφοράς (behavior diagrams)**
  Δίνουν έμφαση στο τι πρέπει να γίνει στο σύστημα που μοντελοποιείται, δηλ., την λειτουργικότητά του



Διαγράμματα Δομής (structure diagrams)

- **Class diagram:** περιγράφει τη δομή του συστήματος με κλάσεις, χαρακτηριστικά και σχέσεις.
- **Component diagram:** περιγράφει πως ένα σύστημα είναι διαχωρισμένο σε συστατικά (components) και τις εξαρτήσεις μεταξύ αυτών.
- **Composite structure diagram:** περιγράφει την εσωτερική δομή μίας κλάσης και τις συνεργασίες που επιτυγχάνει.
- **Deployment diagram:** περιγράφει το υλικό (hardware) που χρησιμοποιεί το σύστημα και τα περιβάλλοντα εκτέλεσης που αναπτύσσονται σε αυτό.
- **Object diagram:** περιγράφει μία πλήρης ή μερική όψη της δομής του συστήματος σε κάποιο συγκεκριμένο χρόνο.
- **Package diagram:** περιγράφει πως το σύστημα είναι διαχωρισμένο σε λογικές κατηγορίες και ποιες είναι οι εξαρτήσεις μεταξύ τους.
- **Profile diagram:** καθορίζει στερεότυπα τα οποία επιτρέπουν την προσαρμογή του μοντέλου σε διαφορετικές πλατφόρμες (π.χ., J2EE ή .NET) ή διαφορετικά πεδία domains (π.χ., real-time or business modeling).

Διαγράμματα Συμπεριφοράς (behavior diagrams)

- **Activity diagram:** περιγράφει ροές εργασίας (workflows) και ροές ελέγχου (flow of control) σε ένα σύστημα.
- **State machine diagram:** περιγράφει καταστάσεις και μεταβάσεις του συστήματος.
- **Use Case Diagram:** περιγράφει τη λειτουργικότητα του συστήματος με ρόλους/χρήστες και λειτουργίες.
- **Communication diagram:** περιγράφει την επικοινωνία μεταξύ αντικειμένων με μηνύματα. Είναι συνδυασμός Class, Sequence, and Use Case Diagrams.
- **Interaction overview diagram:** κάθε κόμβος είναι ένα communication diagram.
- **Sequence diagram:** Παρόμοιο με communication diagram με τη διαφορά ότι κάθε αντικείμενο παρουσιάζει και το χρόνο ζωής του.
- **Timing diagrams:** περιγράφει χρονικούς περιορισμούς.

UML: Τι έχουμε μάθει;

- Use Case diagram
- Class Analysis diagram
- State machine diagram
- Sequence diagram
- Class diagram
- Detailed Class diagram
- Deployment diagram

UML Παραδείγματα

- Κατανόηση των UML διαγραμμάτων
- Κατανόηση συσχετίσεων
- Παραδείγματα
  - Κατοικίδια ζώα και Ιδιοκτήτες
  - Οργάνωση σκληρών δίσκων
  - Τραπεζικό σύστημα
  - Σύστημα κεντρικής θέρμανσης
  - Σύστημα εκτύπωσης

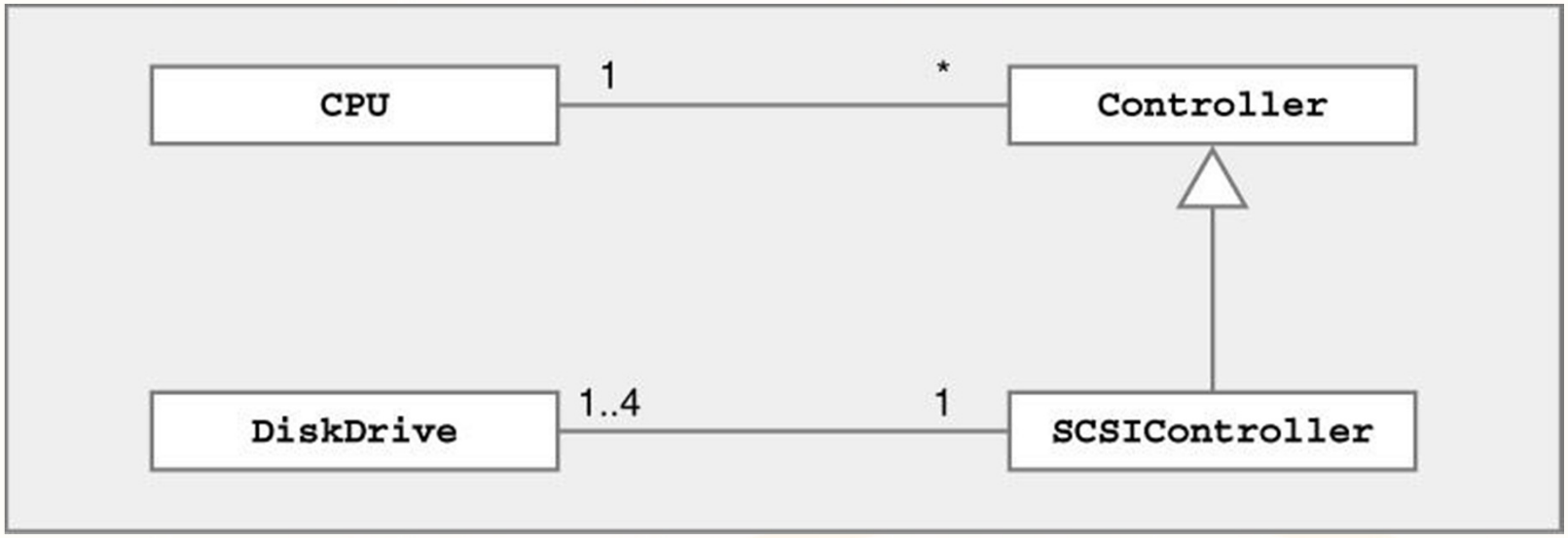
UML Example – Veterinary System

- Try to read & understand UML diagram

1 or more Pets associated with 1 PetOwner

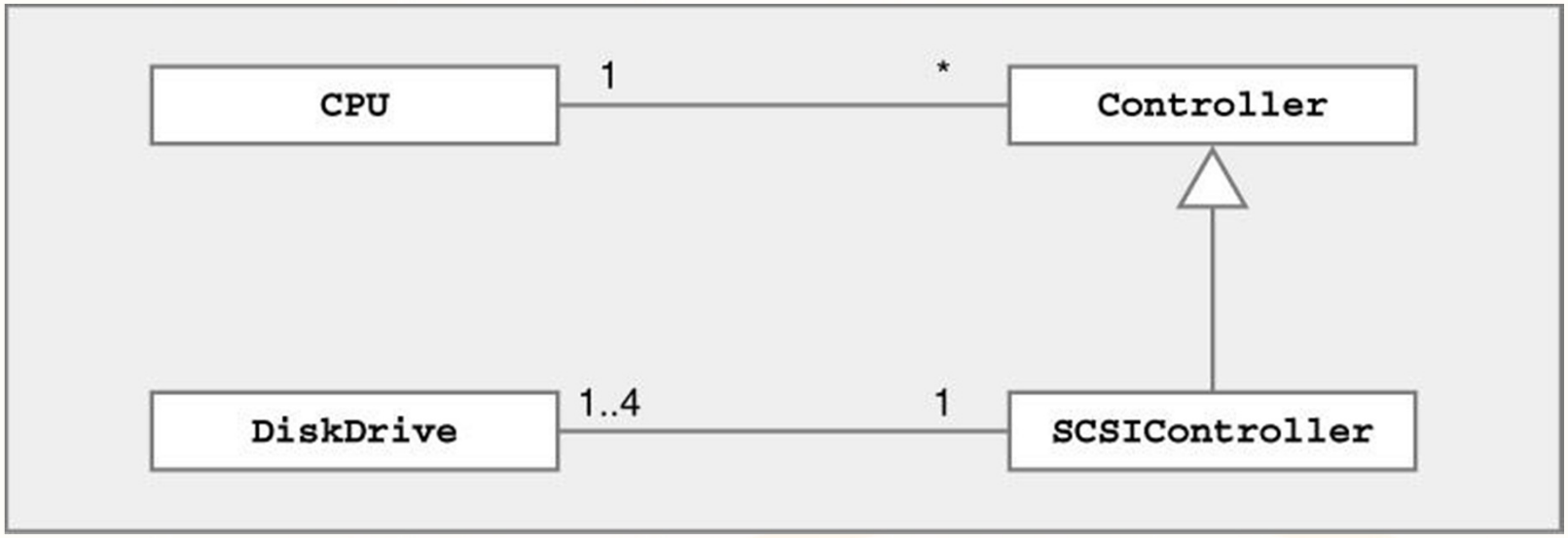
UML Example – Computer System

- Try to read & understand UML diagram



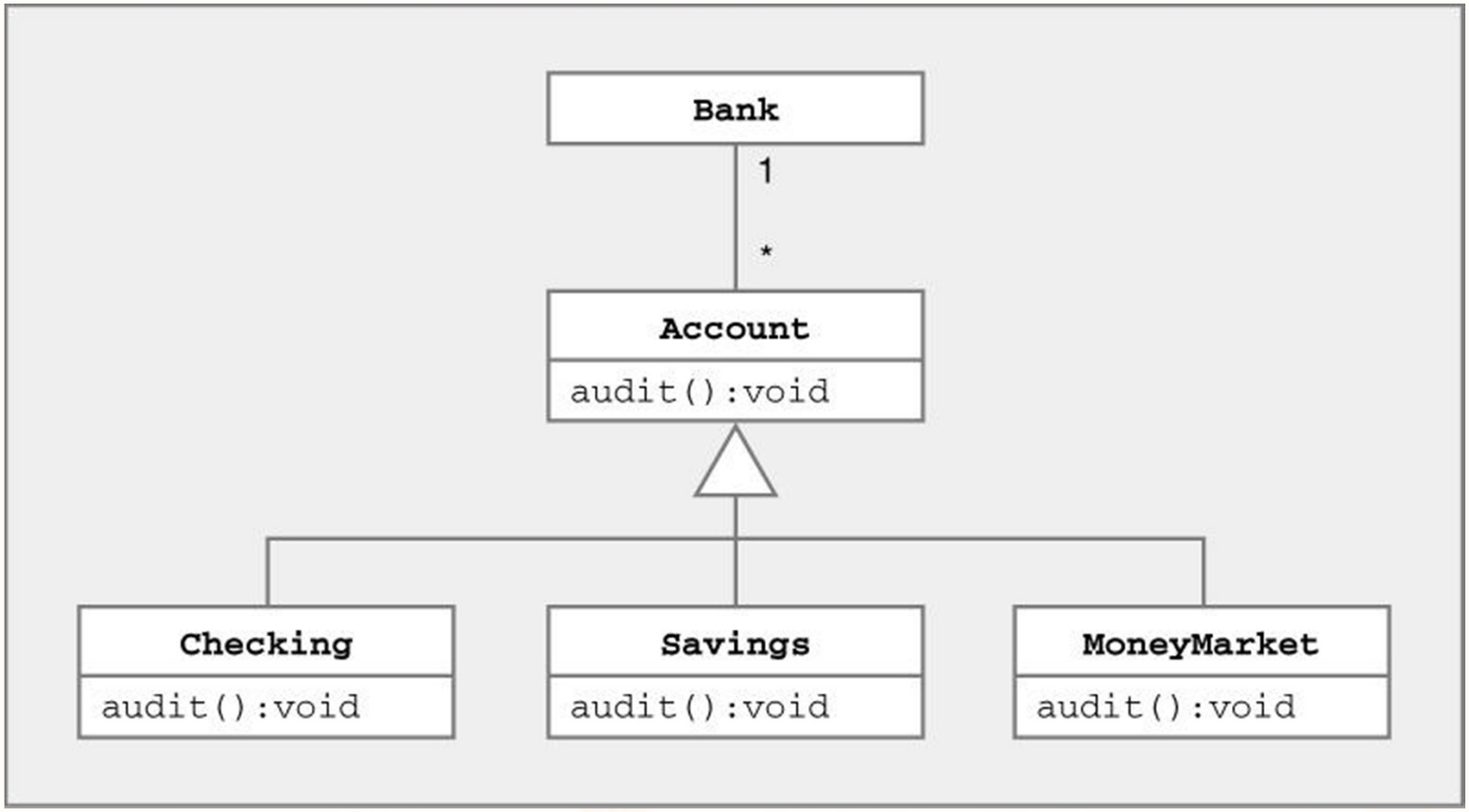
UML Example – Computer System

- Try to read & understand UML diagram



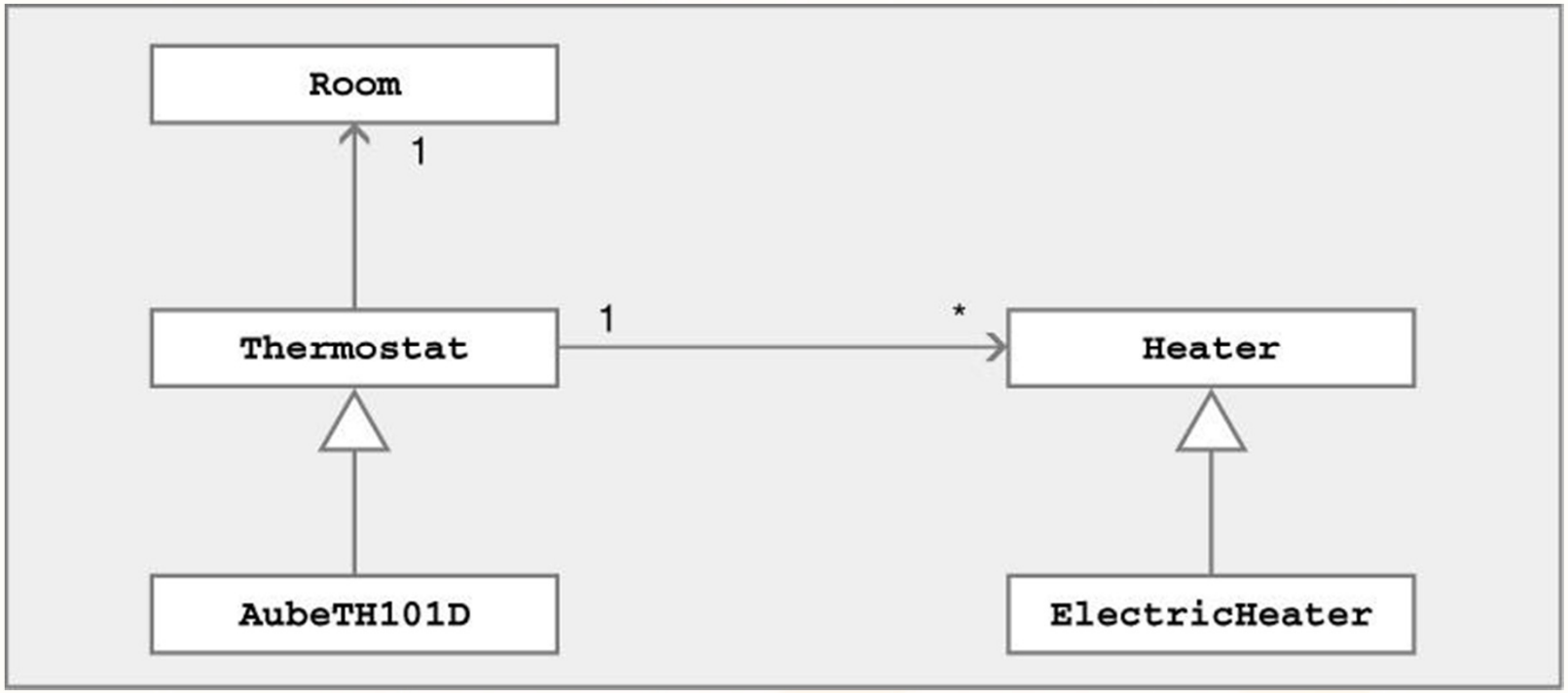
- **1 CPU associated with 0 or more Controllers**
- **1-4 DiskDrives associated with 1 SCSIController**
- **SCSIController is a (specialized) Controller**

UML Example – Banking System



- 1 Bank associated with 0 or more Accounts
- Checking, Savings, MoneyMarket are Accounts

UML Example – Home Heating System



- Each Thermostat associated with 1 Room
- Each Thermostat associated with 0 or more Heaters
- ElectricHeater is a specialized Heater
- AubeTH101D is a specialized Thermostat

UML Class Diagrams & Java

- Διαφορετική αναπαράσταση της ίδιας πληροφορίας
  - Name, state, behavior of class
  - Relationship(s) between classes
- Χρειάζεται εξάσκηση έτσι ώστε να μπορούμε να μετατρέπουμε το ένα στο άλλο και
  - να αναπαριστούμε με ακρίβεια τα μοντέλα

UML & Java : Veterinary System

- UML

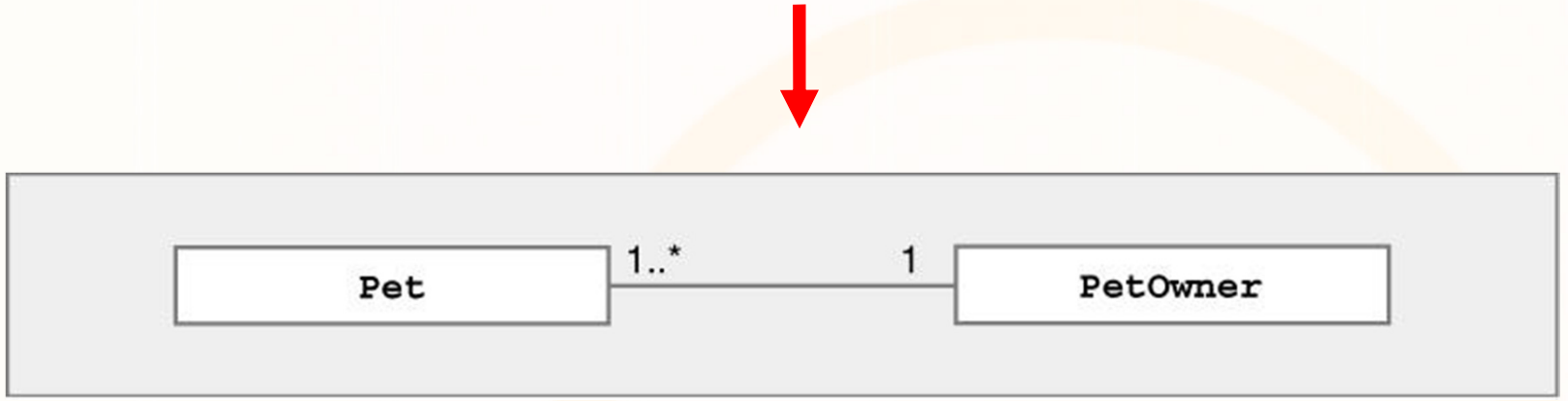
- Java

```
class Pet {
    PetOwner myOwner;        // 1 owner for each pet
}
class PetOwner {
    Pet [ ] myPets; // multiple pets for each owner
}
```

UML & Java : Veterinary System

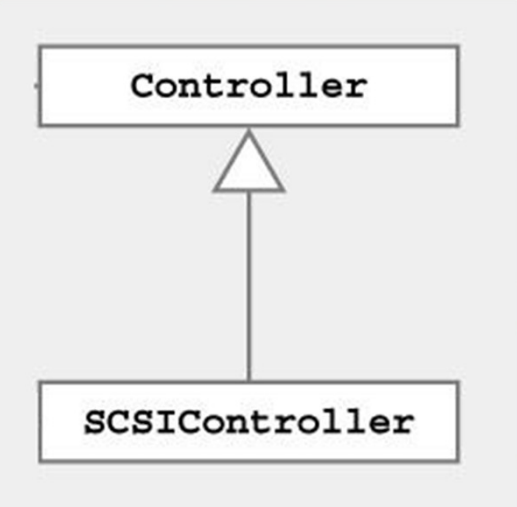
- Java
  - `class Pet {`
  - `    PetOwner myOwner; // 1 owner for each pet`
  - `}`
  - `class PetOwner {`
  - `    Pet [ ] myPets; // multiple pets for each owner`
  - `}`

- UML



UML & Java : Computer System

- UML



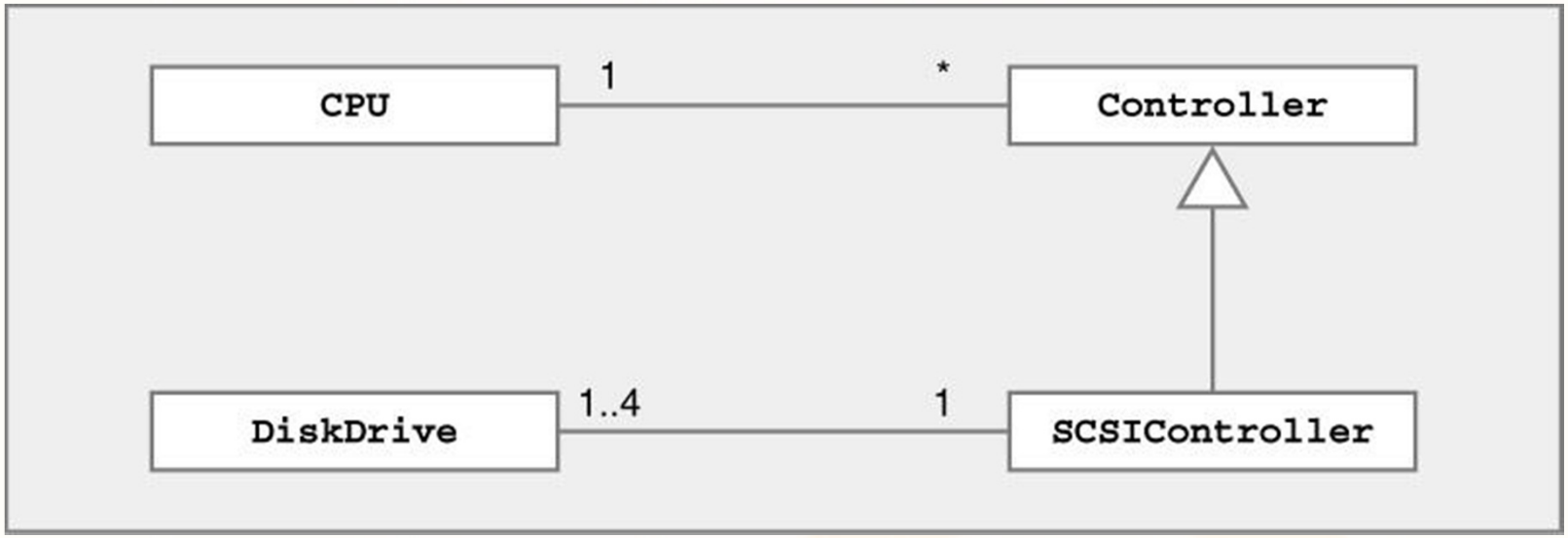
- Java
  - `class Controller {`
  - `}`
  - `class SCSIController extends Controller {`
  - `}`

κληρονόμος

Μητρική Κλάση

UML & Java : Computer System

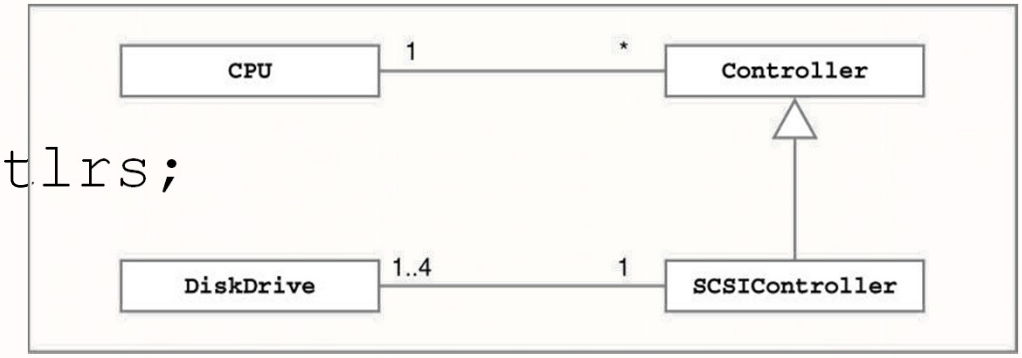
- UML



- Java
  - Design code using all available information in UML…

UML & Java : Computer System

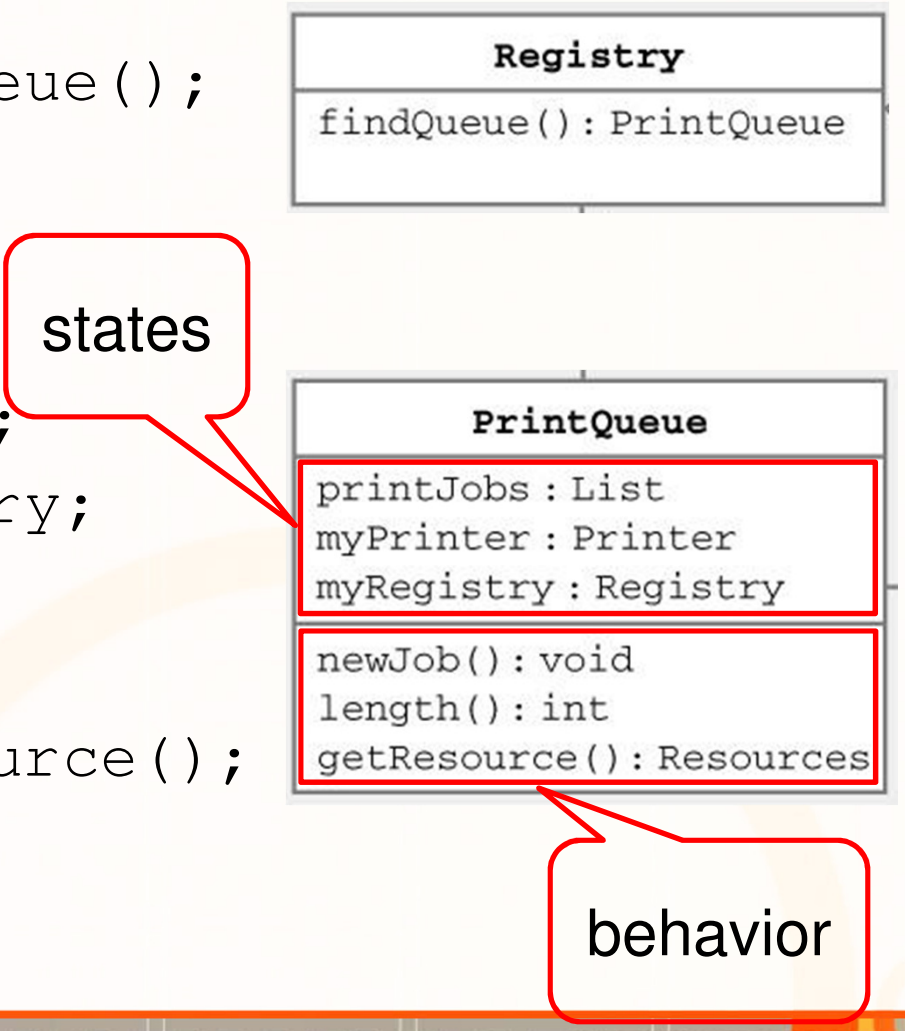
- Java




```
class CPU {
    Controller [ ] myCtlrs;
}
class Controller {
    CPU myCPU;
}
class SCSIController extends Controller {
    DiskDrive [ ] myDrives = new DiskDrive[4];
}
Class DiskDrive {
    SCSIController mySCSI;
}
```

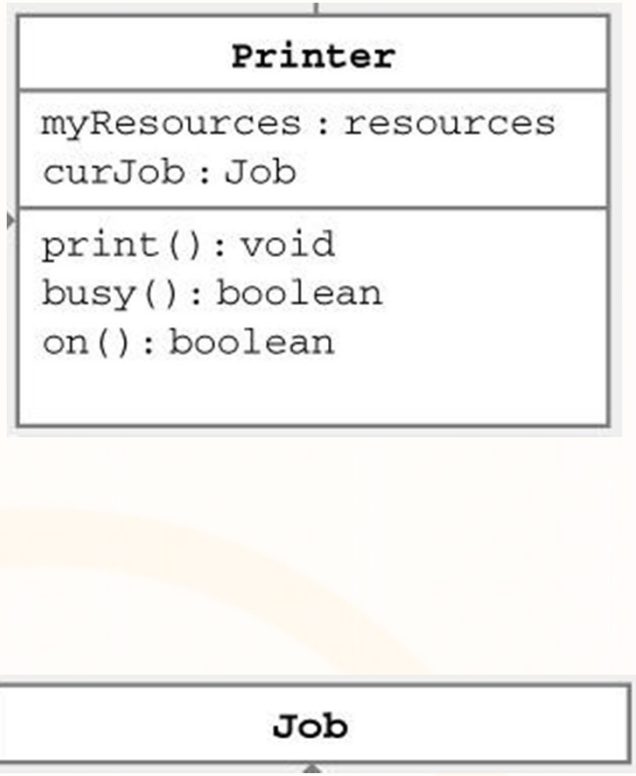
Java & UML : Printing System

- Java
  - class Registry {
  - PrintQueue findQueue();
  - }
  - class PrintQueue {
  - List printJobs;
  - Printer myPrinter;
  - Registry myRegistry;
  - void newJob();
  - int length();
  - Resources getResource();
  - }

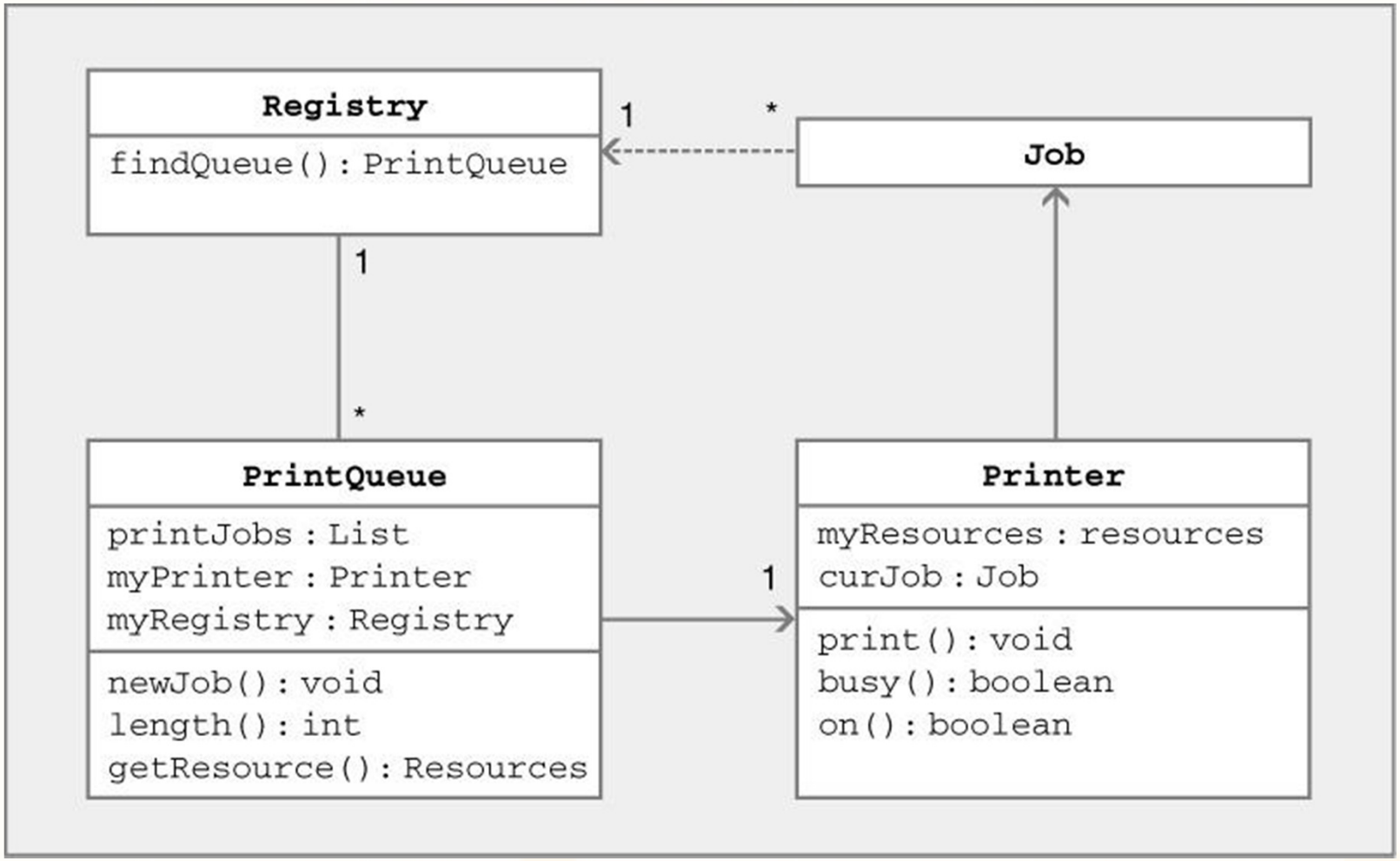


Java & UML : Printing System

- Java
  - Class Printer {
  - Resources myResources;
  - Job curJob;
  - void print();
  - boolean busy();
  - boolean on();
  - }
  - class Job {
  - Job(Registry r) {
  - …
  - }
  - }



Java & UML : Printing System



UML Σύνοψη

- Γλώσσα γραφικών μοντελοποιήσεων (Μόνο ??)
- Γραφική απεικόνιση των σχεδιαγραμμάτων του συστήματος
- **Εστιάσαμε περισσότερο στα διαγράμματα κλάσεων**
  - Περιεχόμενα της κλάσης
  - Συσχετίσεις μεταξύ των κλάσεων
- **Πρέπει να είστε σε θέση να:**
  - Γράφετε τον κώδικα Java από το UML διάγραμμα
  - Σχεδιάζετε τα UML διαγράμματα από τον κώδικα Java