Informatics 43

LECTURE 10

“HOW DO WE STRUCTURE THE SOFTWARE IN DETAIL? (PART 2)”

Homework 2

- Homework 2 requires that you complete your Requirements Specification from Homework 1 with the following:
- The Functional Requirements Section
  - Functional Requirements
  - Two use cases (textual)
  - One Use Case Diagram

Last Lecture

- Design phase of software engineering
  - The “how” to the “what” of requirements
  - Architecture, functional decomposition, relational database design, OO design/UML, UI design, sketching
- Designs are used iteratively to think, talk, and prescribe
- Software engineering is all about constructing and elaborating abstractions/models

Today’s Lecture - **How do we structure the software in detail?**

- Design: recap
- Design notations / diagrams
  - UML class diagrams
  - Other diagrams
- Design principles

Today’s Lecture - **How do we structure the software in detail?**

- Design: recap
- Design notations / diagrams
  - UML class diagrams
  - Other diagrams
- Design principles

Software Design Recap

- All creative decisions, includes high-level and low-level
- Different notations and models allow designers to focus on a perspective, while freed from thinking of others
- Designs used to
  - Think
  - Talk
  - Prescribe

Today's Lecture - **How do we structure the software in detail?**

- Design: recap
- Design notations / diagrams
  - UML class diagrams
  - Other diagrams
- Design principles

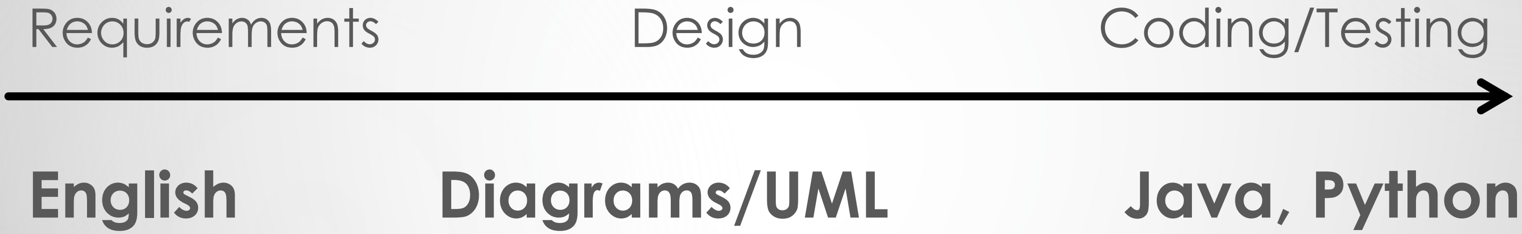
Design notations

"By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race."

-A.N. Whitehead (1911)

Software Development Languages

Different languages are used at different stages:

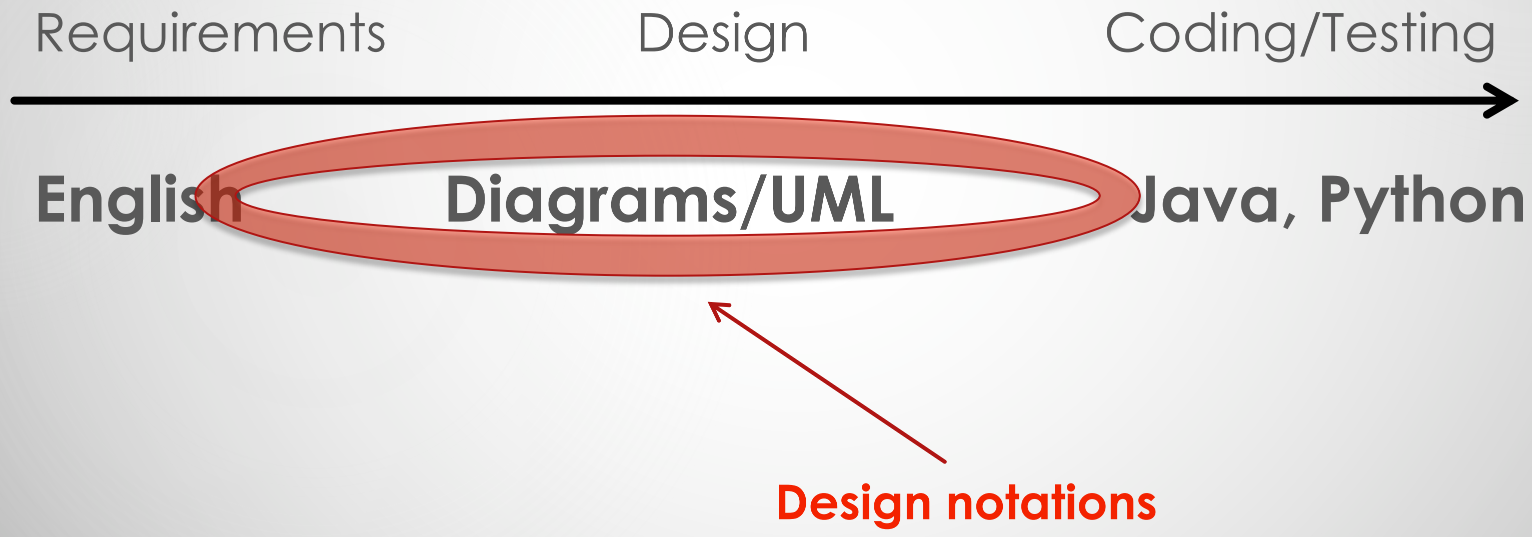


Requirements — Design — Coding/Testing

English — **Diagrams/UML** — **Java, Python**

Software Development Languages

Different languages are used at different stages:



Requirements Design Coding/Testing

English **Diagrams/UML** **Java, Python**

Design notations

Today's Lecture - **How do we structure the software in detail?**

- Design: recap
- Design notations / diagrams
  - UML class diagrams
  - Other diagrams
- Design principles

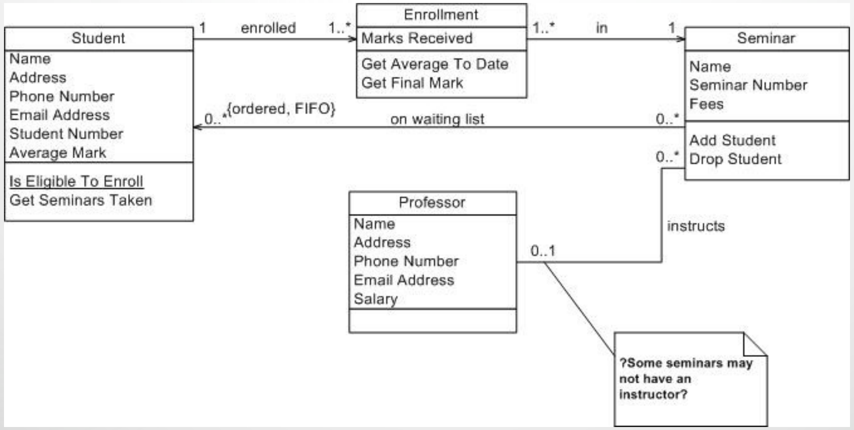
UML (Unified Modeling Language)

- Industry standard for software design/modeling
- Different types of UML diagrams are used to represent different aspects (structure, behavior, interactions) of a system
  - Class diagrams
  - Activity diagrams
  - Sequence diagrams
  - Use case diagrams
  - …

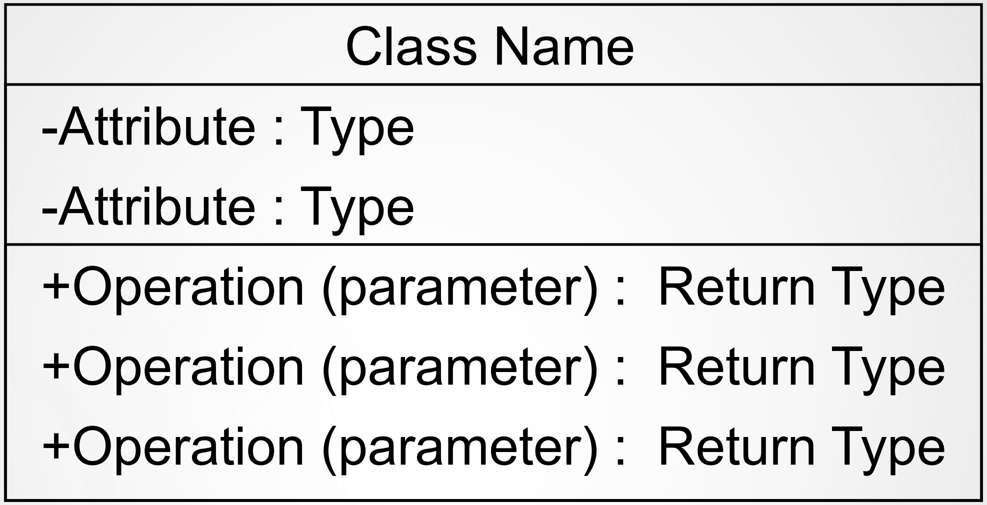
Some following slides from
www.cs.drexel.edu/~spiros/teaching/CS575/slides/uml.ppt

UML Class Diagrams

- Used in decomposing a system into modules known as classes
- Typically used to
  - model domain concepts
  - create a detailed, object-oriented design of the code



UML Class Diagrams



Class Name

-Attribute : Type

-Attribute : Type

+Operation (parameter) : Return Type

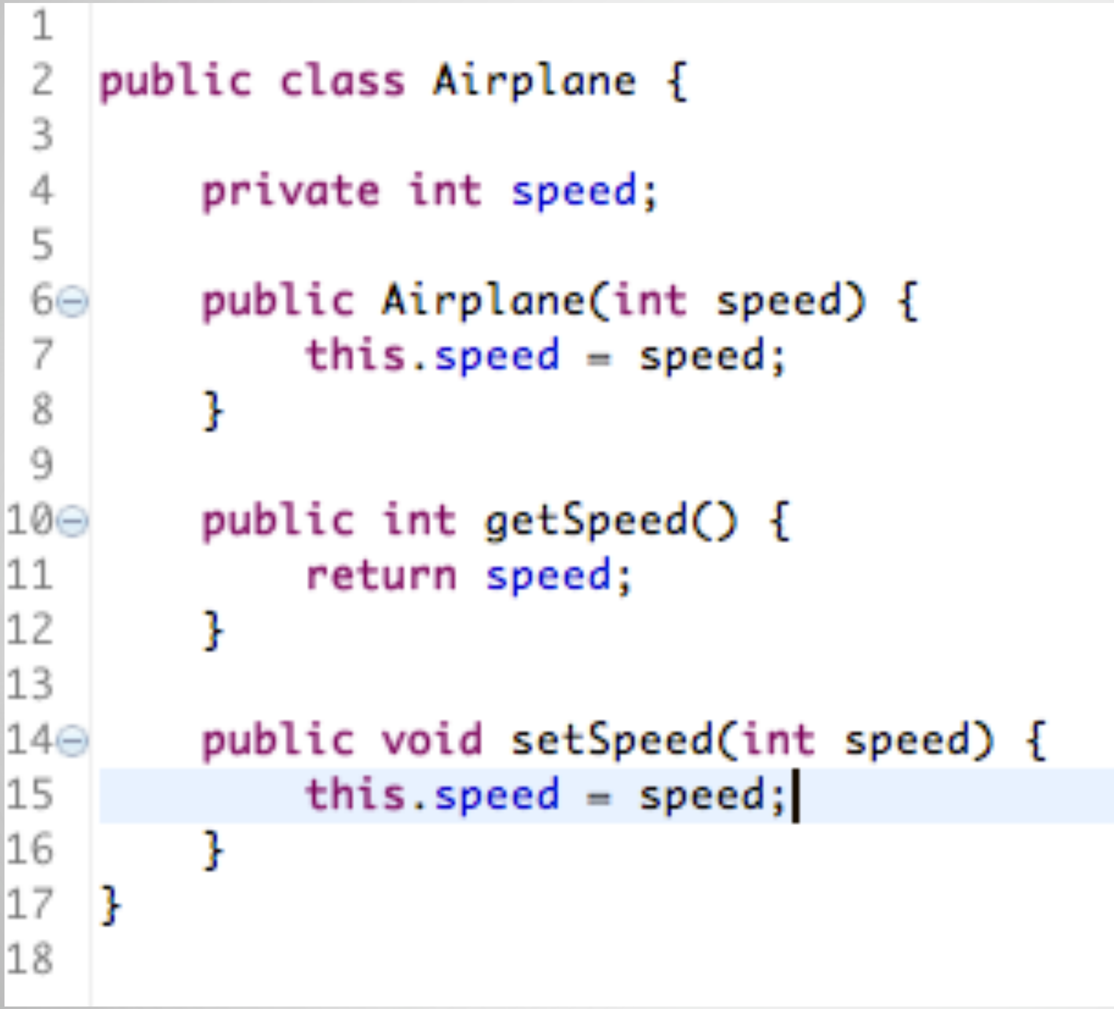
+Operation (parameter) : Return Type

+Operation (parameter) : Return Type

‘+’ means public visibility

‘-’ means private visibility

Translation to Code



```java

public class Airplane {

    private int speed;

    public Airplane(int speed) {
        this.speed = speed;
    }

    public int getSpeed() {
        return speed;
    }

    public void setSpeed(int speed) {
        this.speed = speed;
    }
}

```

Airplane

- speed: int

+ getSpeed() : int
+ setSpeed(int): void

Using Airplane:

Airplane a = new Airplane(5);

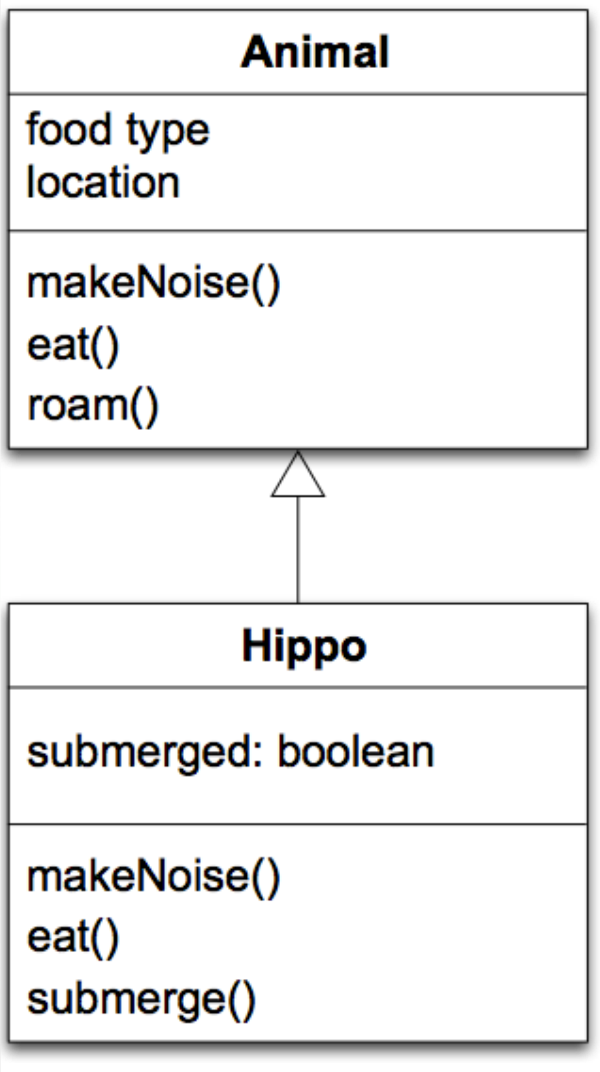
a.setSpeed(10);

System.out.println("" +
        a.getSpeed());

Relationships Between Classes

- Inheritance
- Association
  - Multiplicity
- Whole-Part (Aggregation and Composition)
- …

Relationships: Inheritance



Animal

food type
location

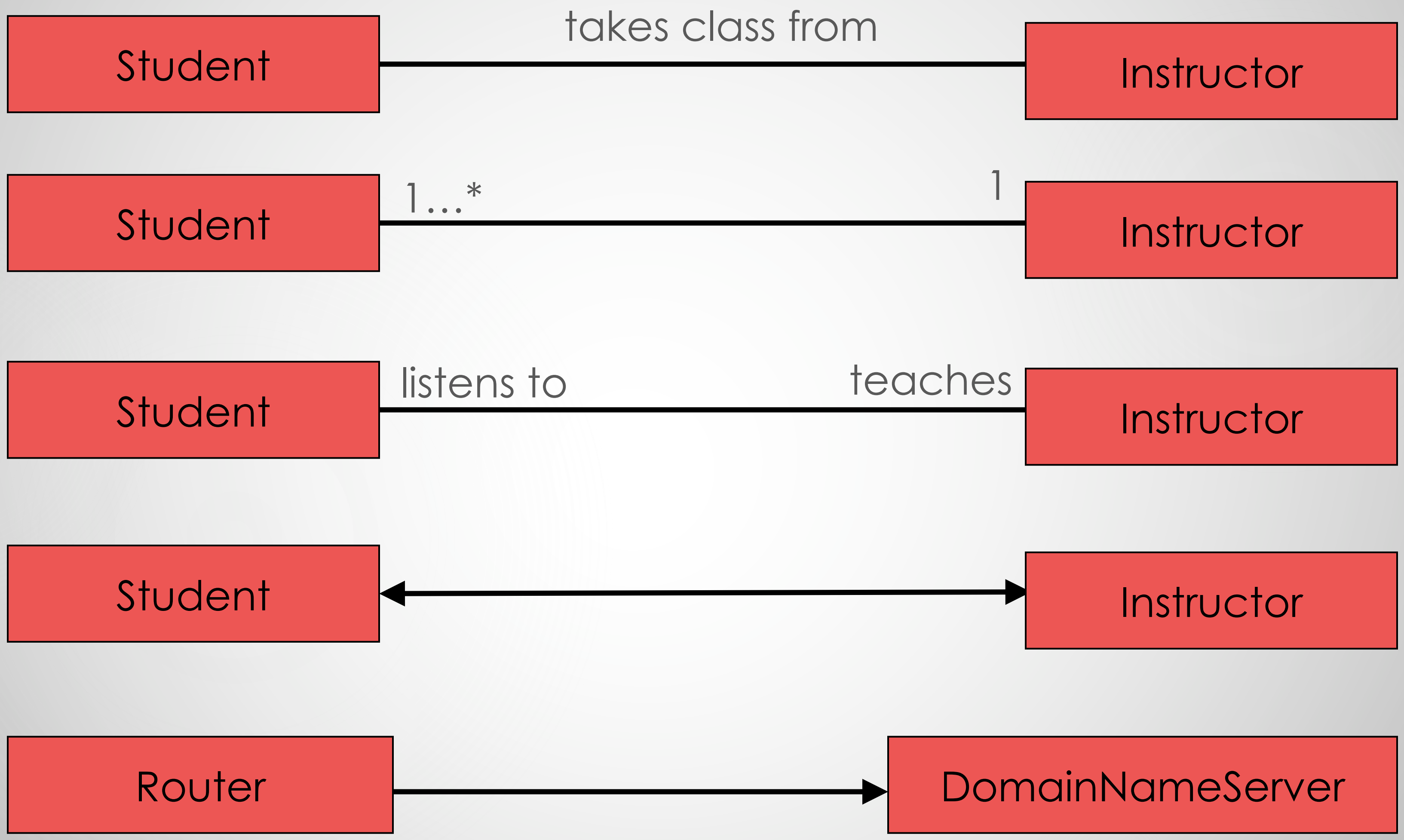
makeNoise()
eat()
roam()

Hippo

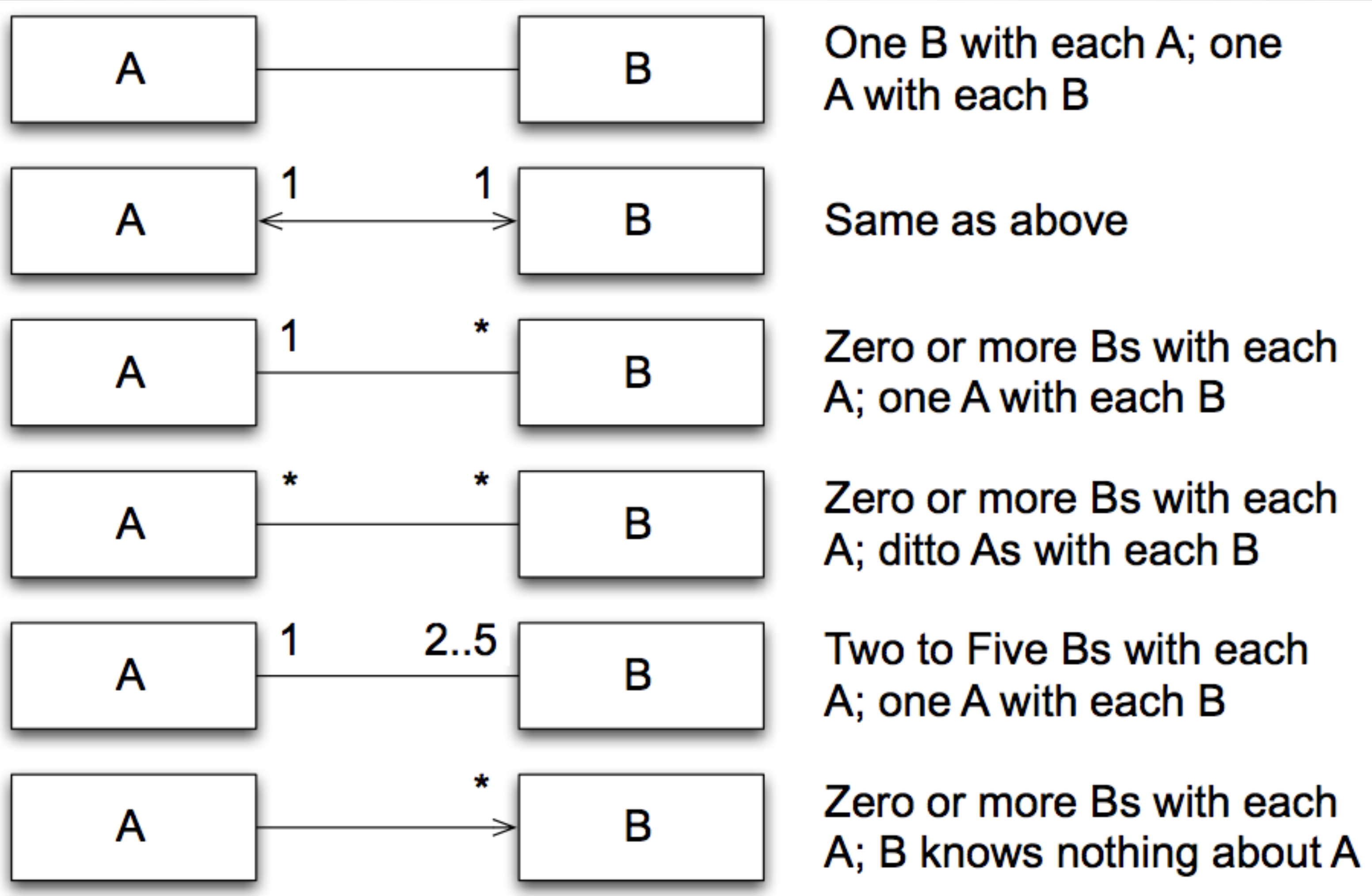
submerged: boolean

makeNoise()
eat()
submerge()

Association Relationships



Multiplicity Examples



One B with each A; one A with each B

Same as above

Zero or more Bs with each A; one A with each B

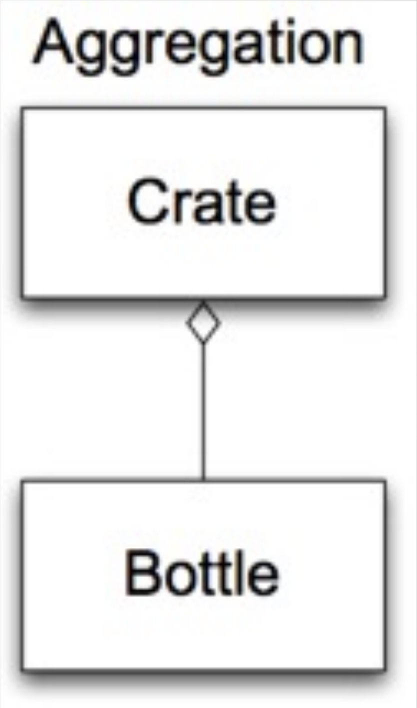
Zero or more Bs with each A; ditto As with each B

Two to Five Bs with each A; one A with each B

Zero or more Bs with each A; B knows nothing about A

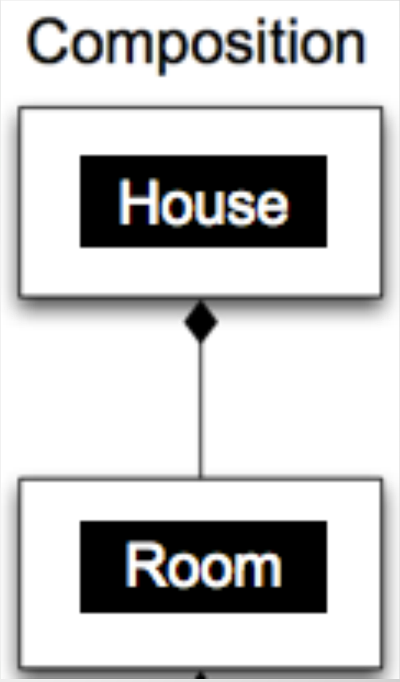
Relationships: Aggregation

- One object contains (or is composed of) a set of other objects
- Aggregation relationships are **transitive and assymetric**

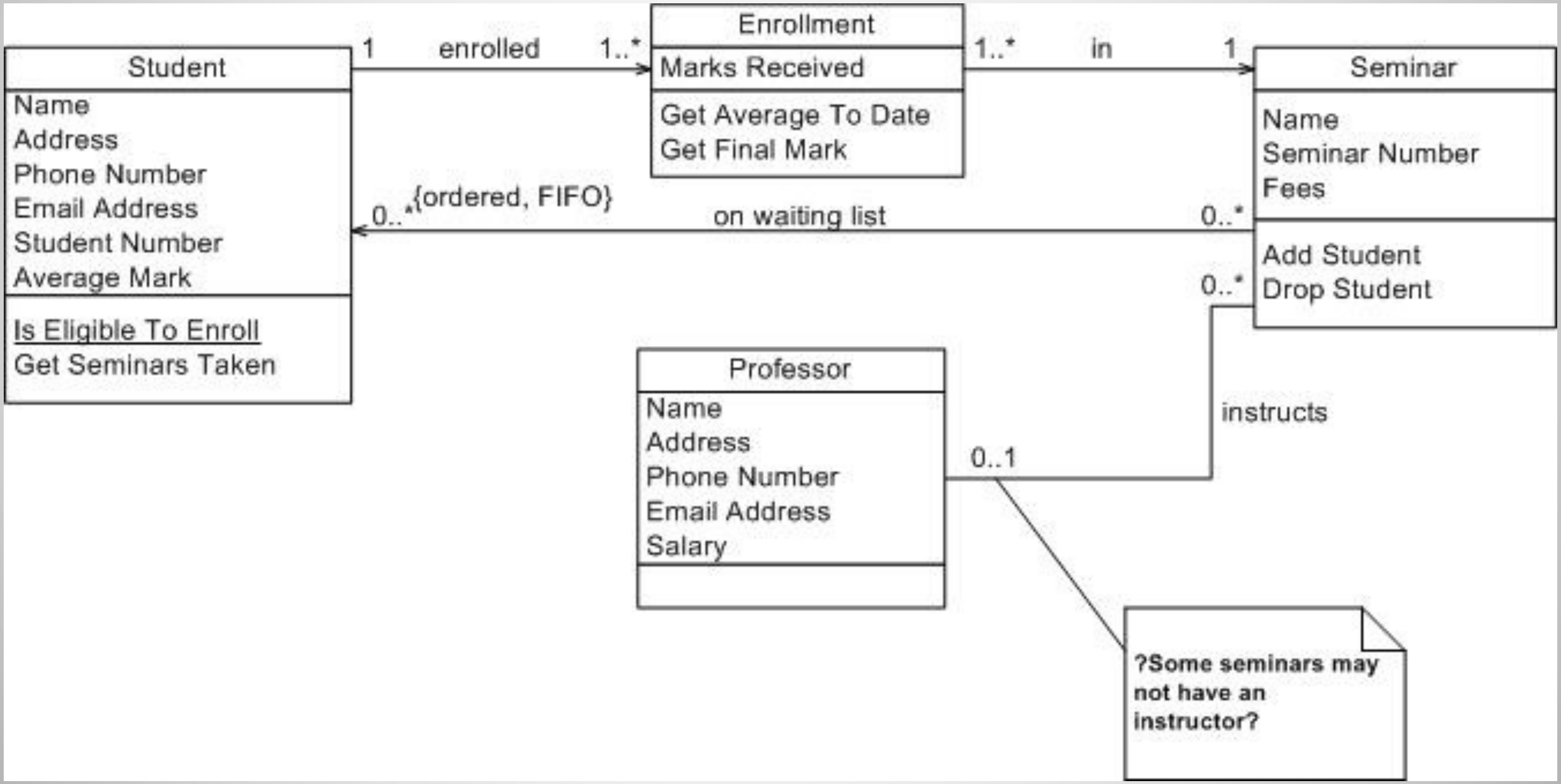


Relationships: Composition

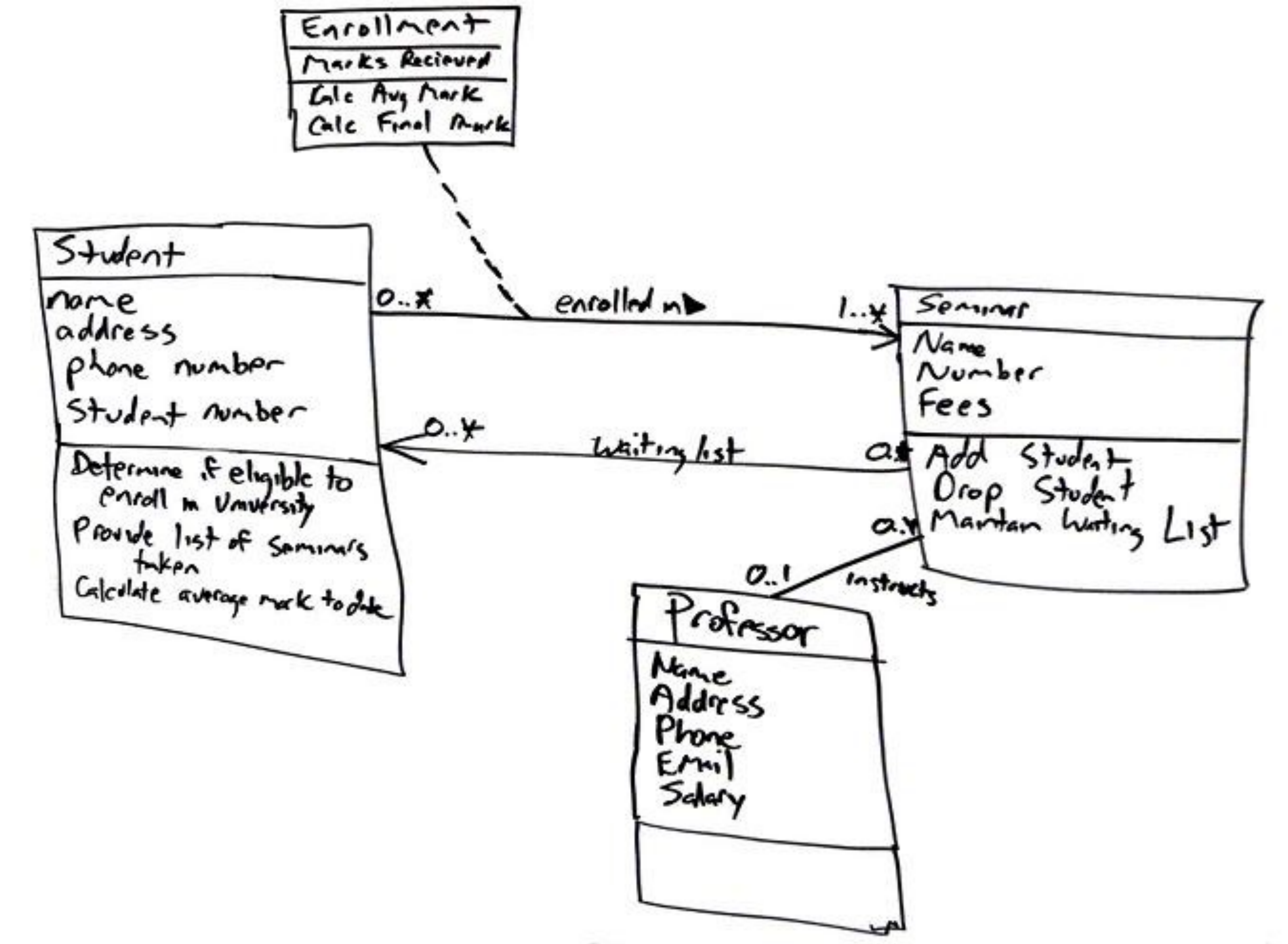
- A variant of aggregation which adds the property of **existence dependency**



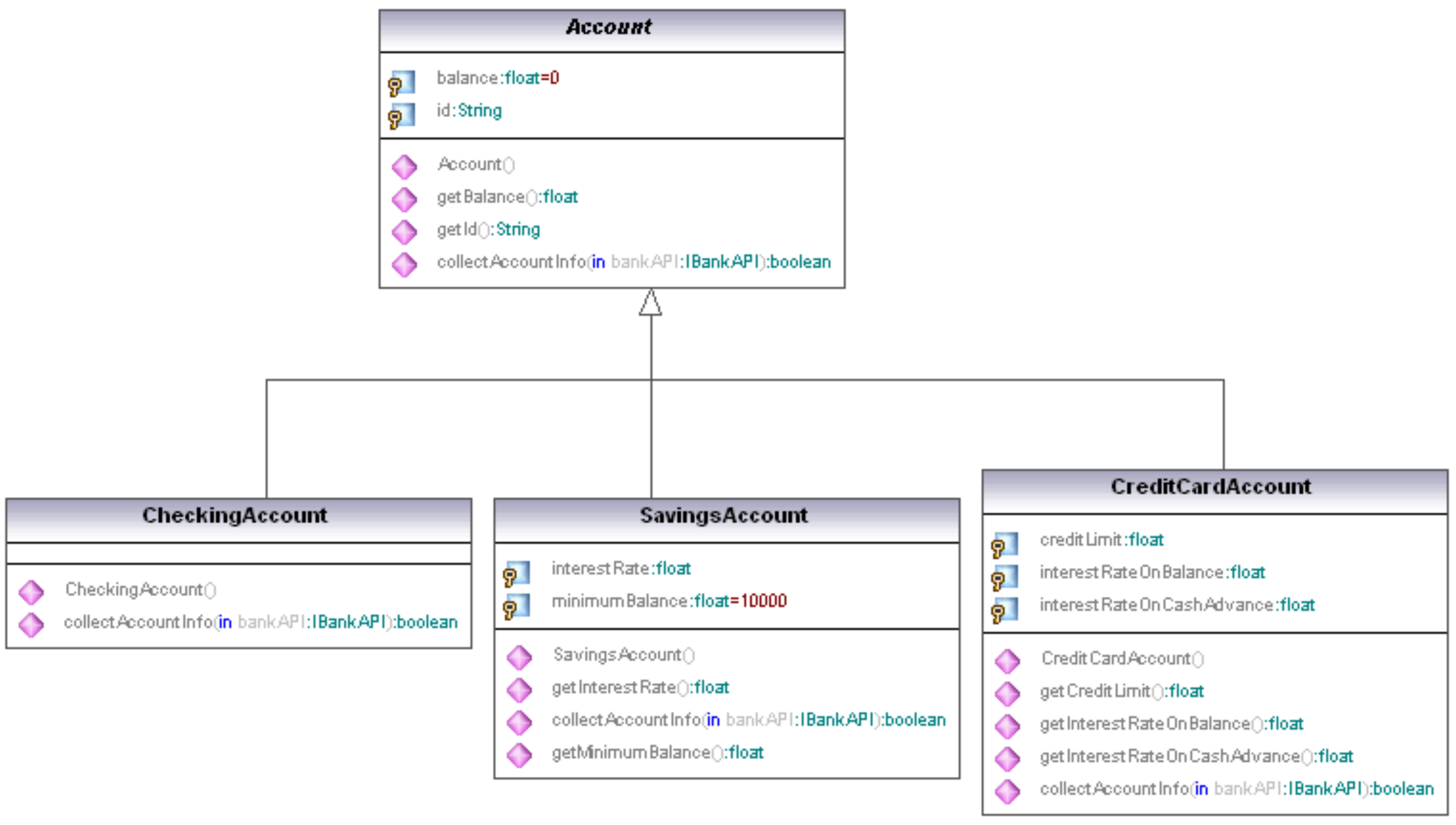
Examples – UML Class Diagrams



Examples – UML Class Diagrams



Examples – UML Class Diagrams



Account

balance:float=0
id:String

Account()
getBalance():float
getId():String
collectAccountInfo(in bankAPI:IBankAPI):boolean

CheckingAccount

CheckingAccount()
collectAccountInfo(in bankAPI:IBankAPI):boolean

SavingsAccount

interestRate:float
minimumBalance:float=10000

SavingsAccount()
getInterestRate():float
collectAccountInfo(in bankAPI:IBankAPI):boolean
getMinimumBalance():float

CreditCardAccount

creditLimit:float
interestRateOnBalance:float
interestRateOnCashAdvance:float

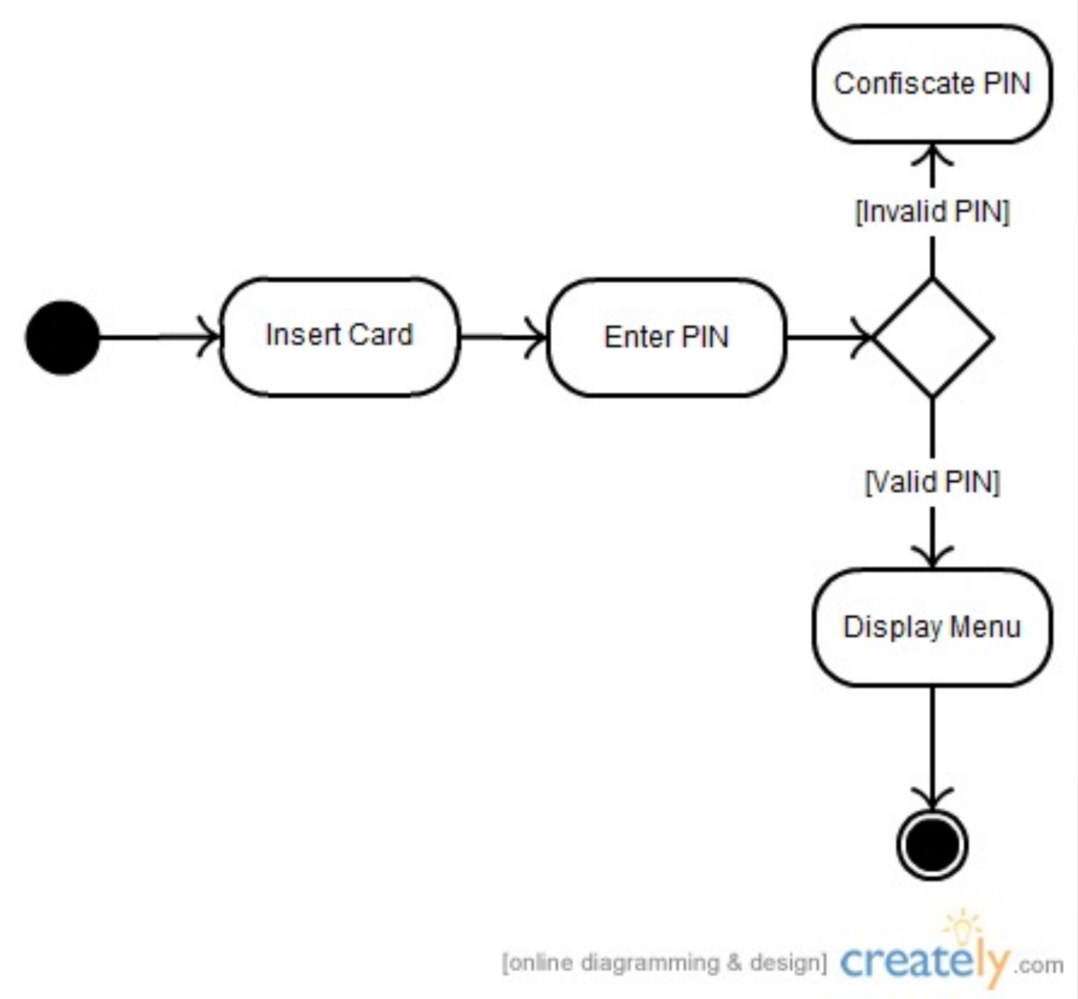
CreditCardAccount()
getCreditLimit():float
getInterestRateOnBalance():float
getInterestRateOnCashAdvance():float
collectAccountInfo(in bankAPI:IBankAPI):boolean

Attendance Quiz

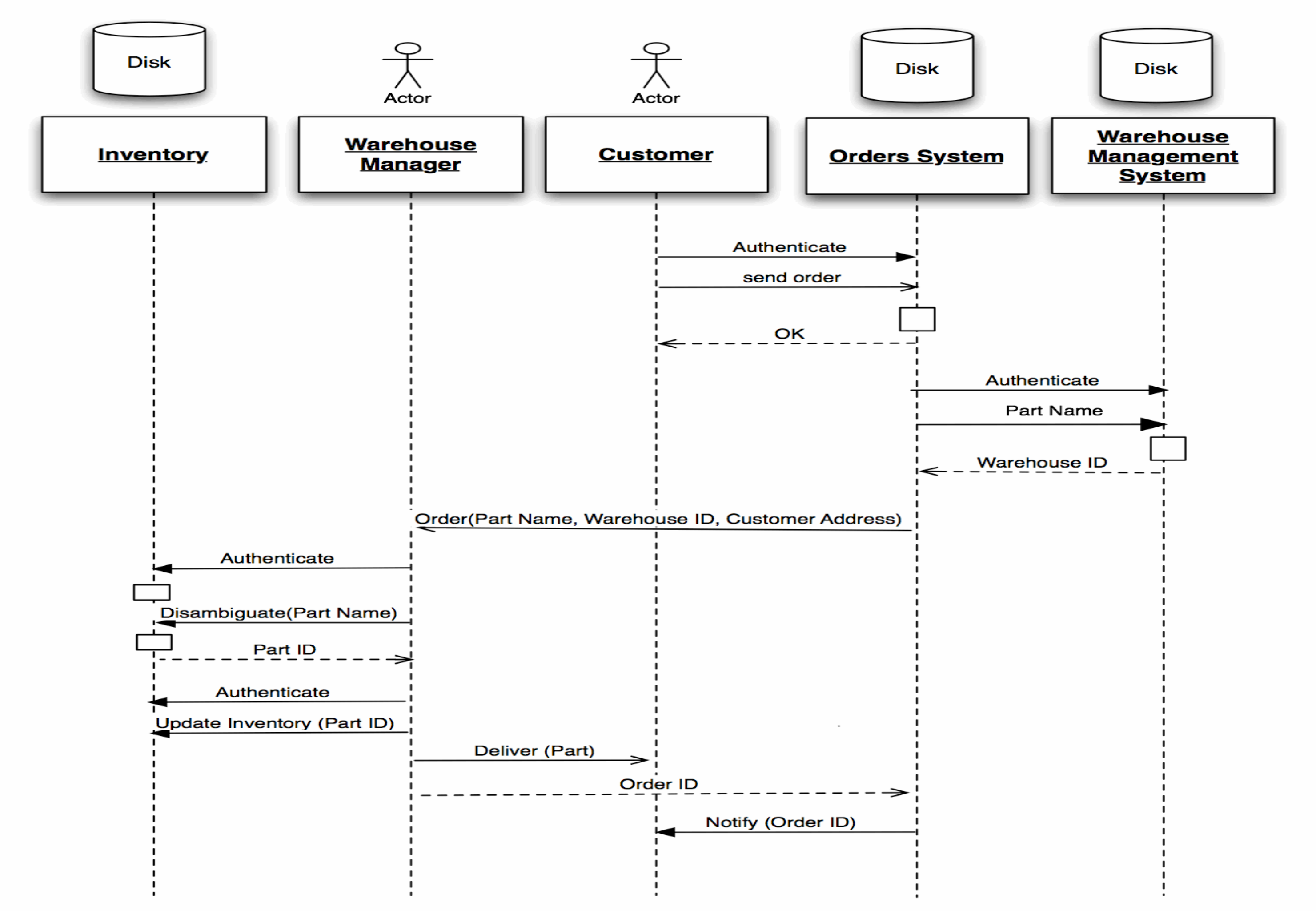
Other UML Diagrams (besides the class diagram)

- Activity Diagrams
- Sequence Diagrams
- Use Case Diagrams

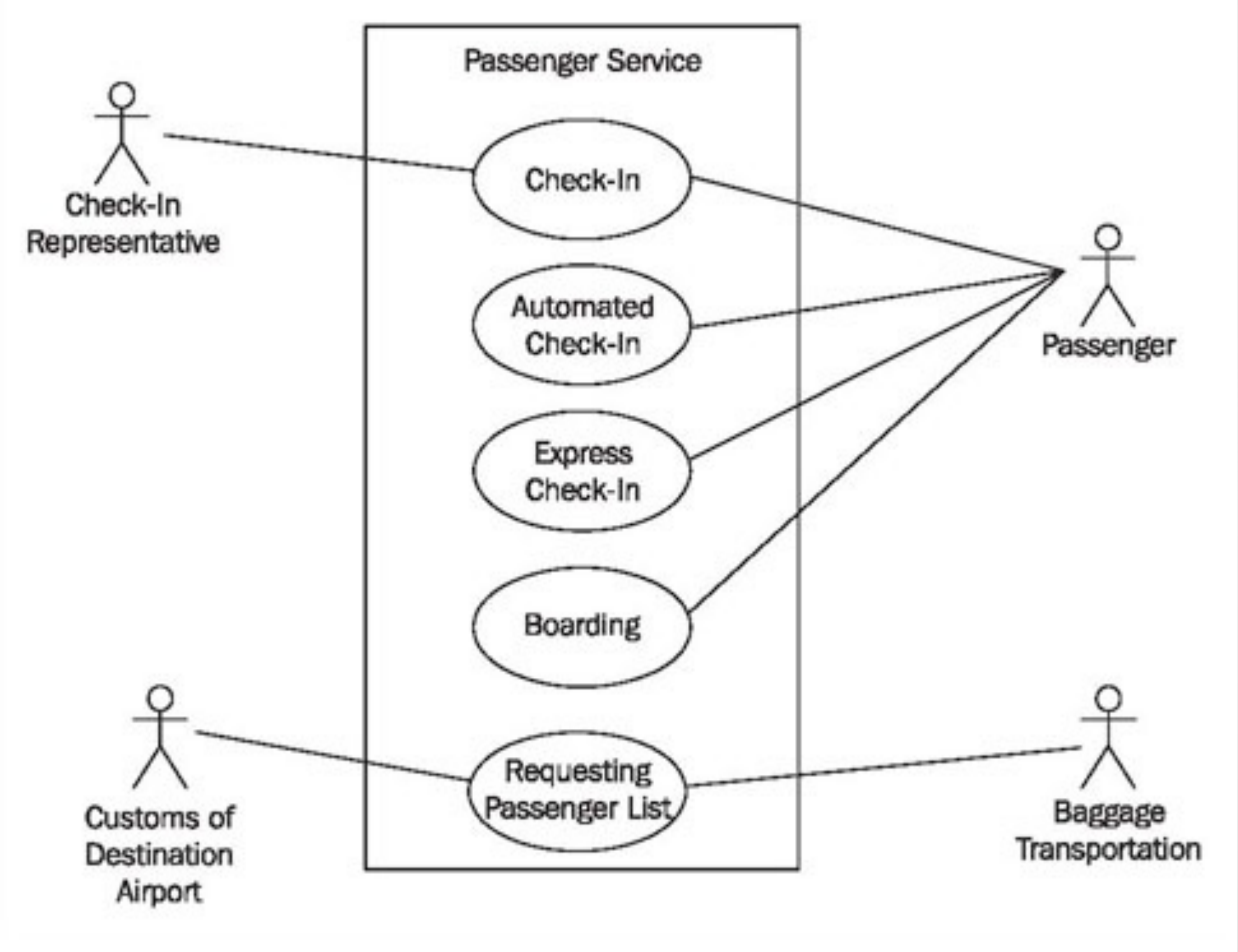
Examples – UML Activity Diagrams



Examples – UML Sequence Diagram



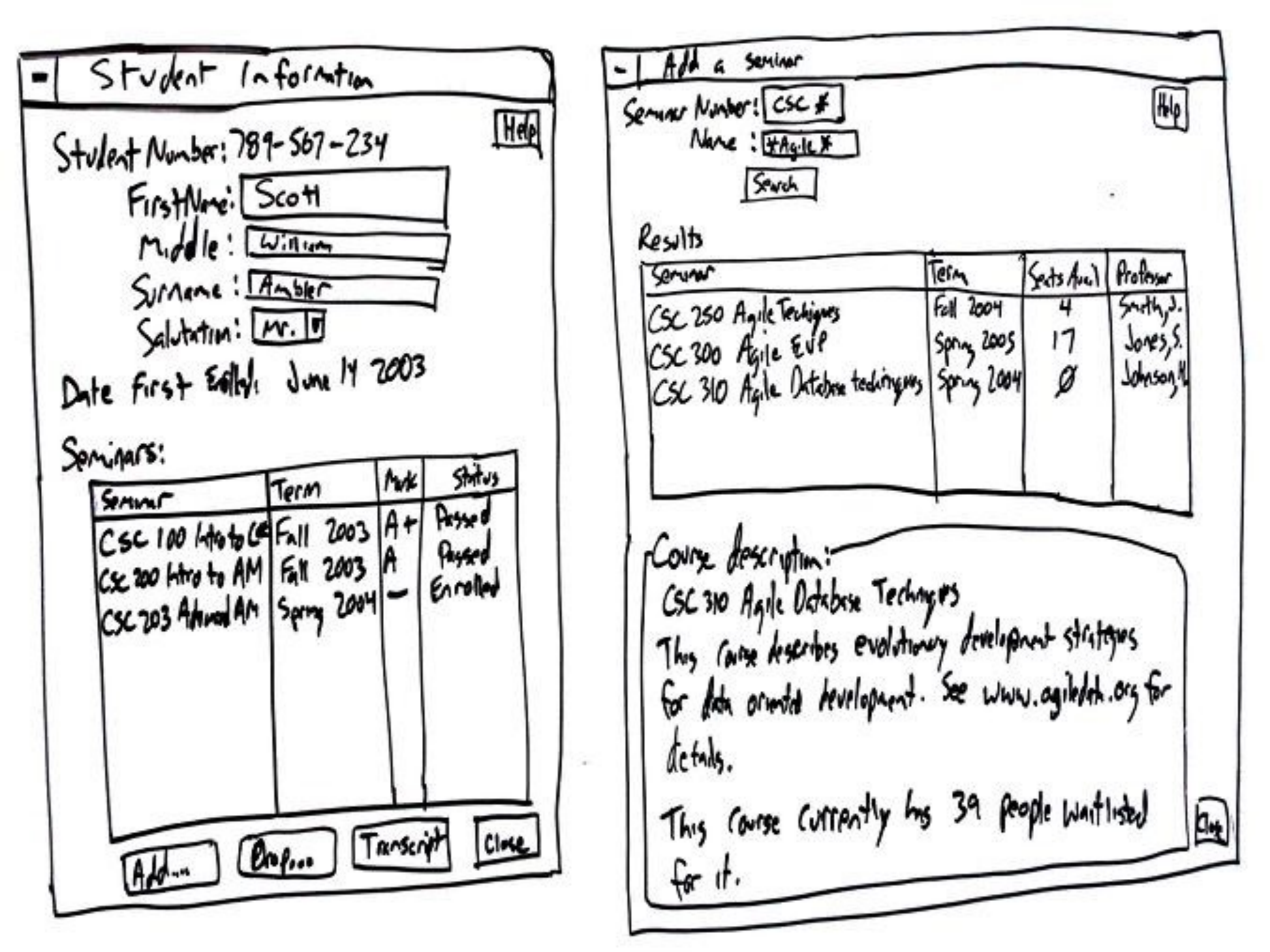
Examples - UML Use Case Diagrams



Today's Lecture - **How do we structure the software in detail?**

- Design: recap
- Design notations / diagrams
  - UML diagrams
  - Other diagrams
- Design principles

Other Diagrams - User Interface Mockups



Student Number: 789-567-234

FirstName: Scott

Middle: William

Surname: Ambler

Salutation: Mr.

Date First Enrl'd: June 14 2003

Seminars:

Seminar	Term	Mark	Status
CSC 100 Intro to C#	Fall 2003	A+	Passed
CSC 200 Intro to AM	Fall 2003	A	Passed
CSC 203 Advanced AM	Spring 2004	—	Enrolled

Seminar Number: CSC *

Name: *Agile*

Results

Seminar	Term	Seats Avail	Professor
CSC 250 Agile Techiques	Fall 2004	4	Smith, J.
CSC 300 Agile EUP	Spring 2005	17	Jones, S.
CSC 310 Agile Database techingues	Spring 2004	Ø	Johnson, M.

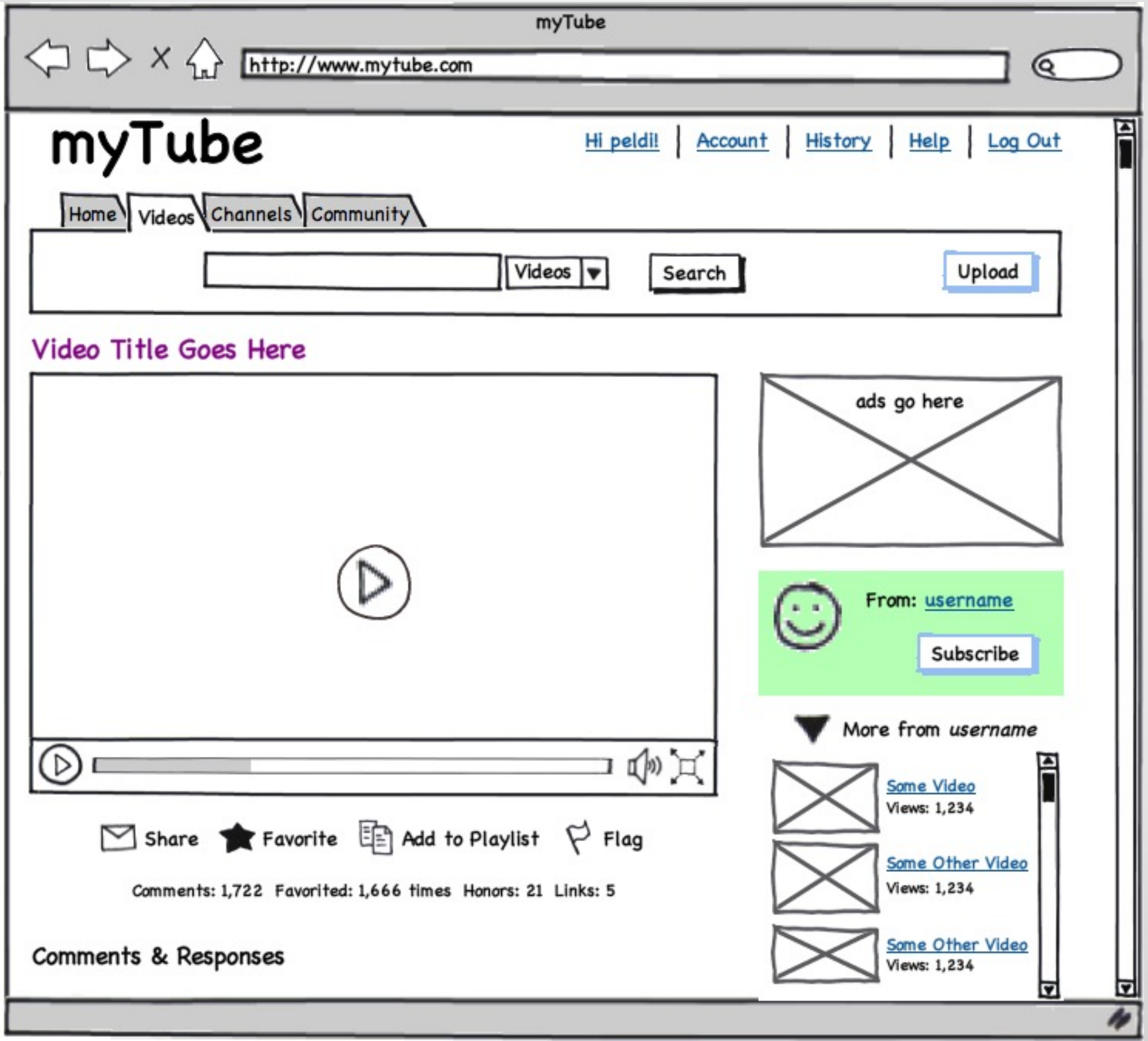
Course description:

CSC 310 Agile Database Techniques

This course describes evolutionary development strategies for data oriented development. See www.agiledata.org for details.

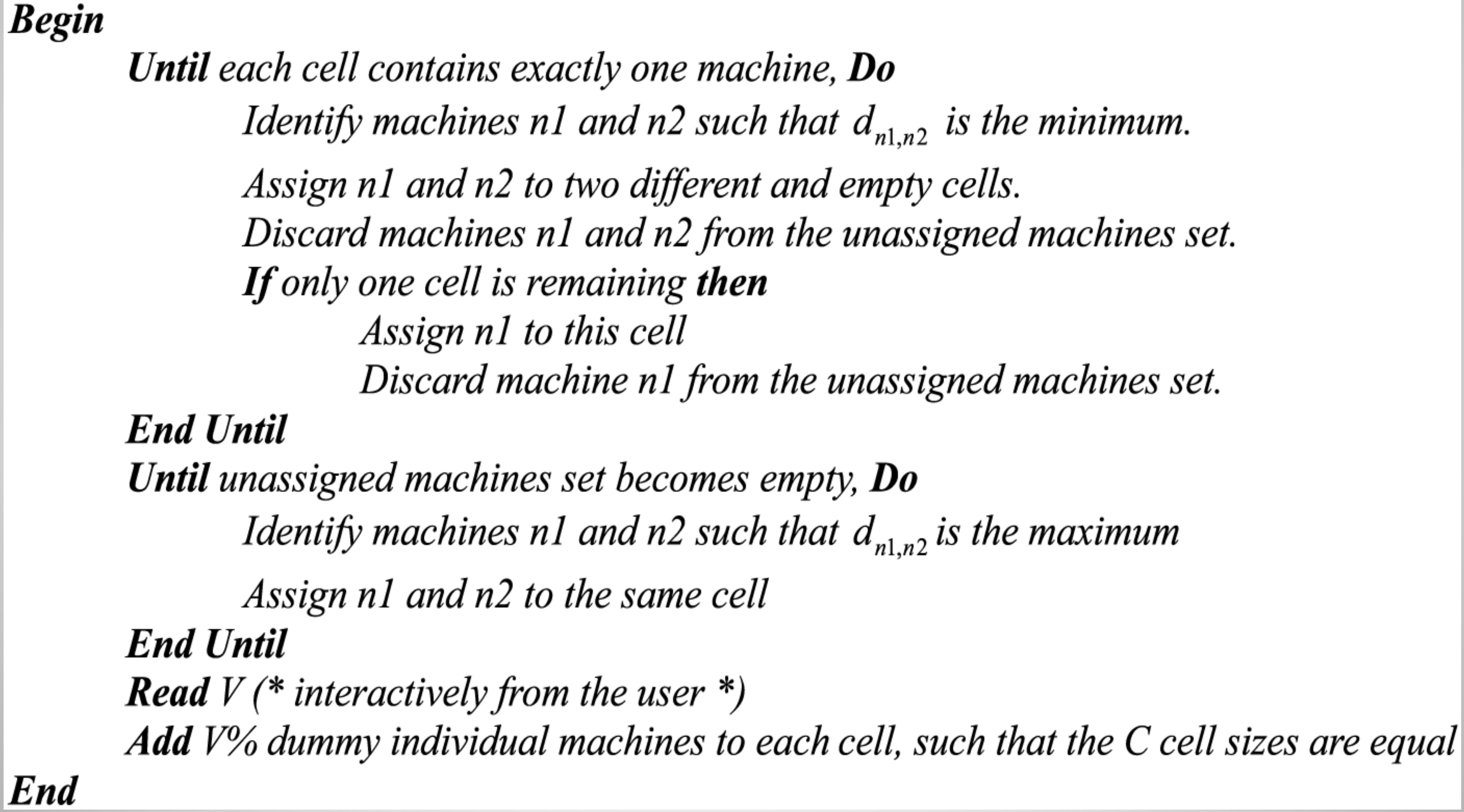
This course currently has 39 people waitlisted for it.

Other Diagrams - User Interface Mockups



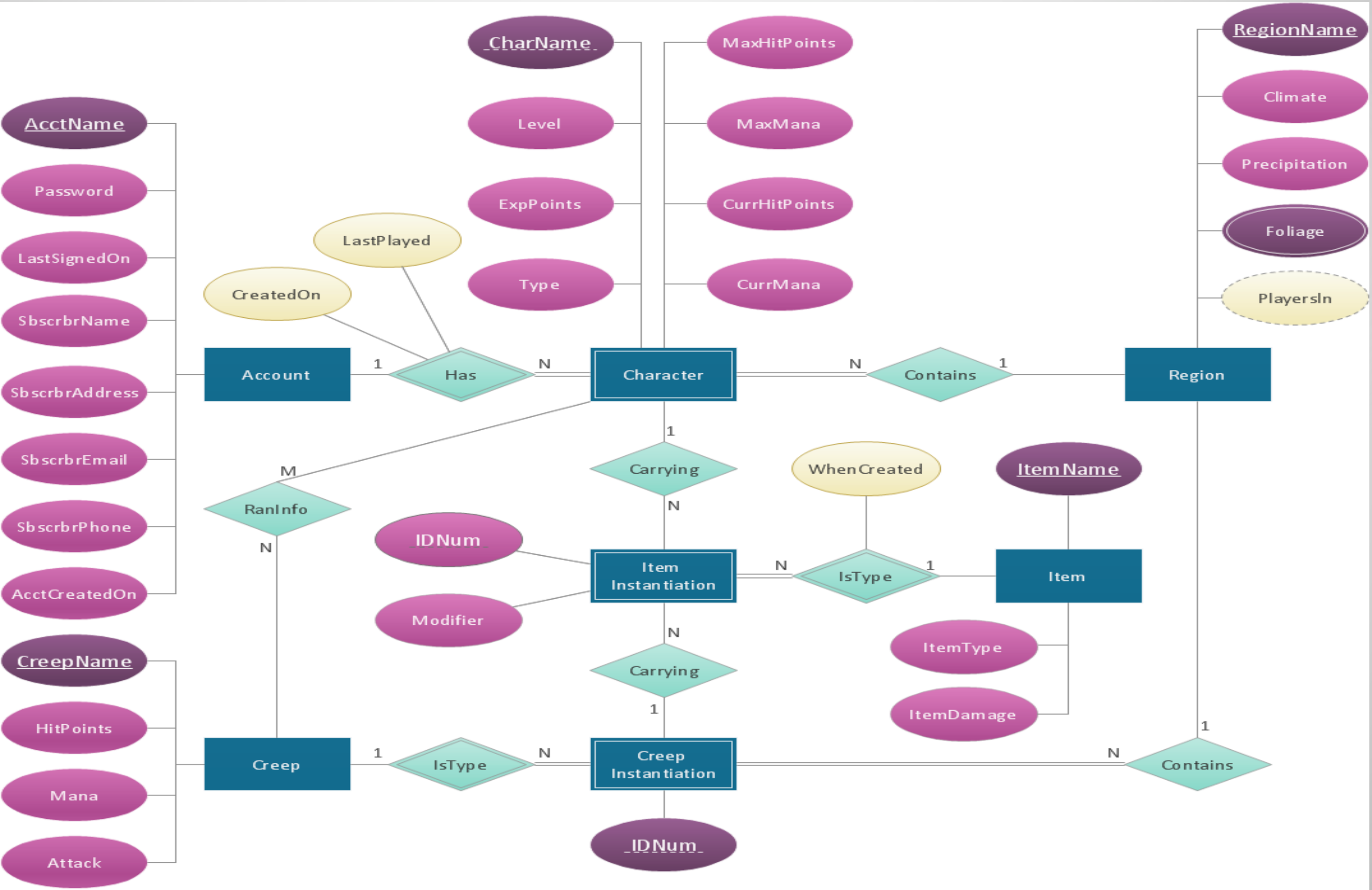
[balsamiq]

Other Diagrams – Pseudo Code

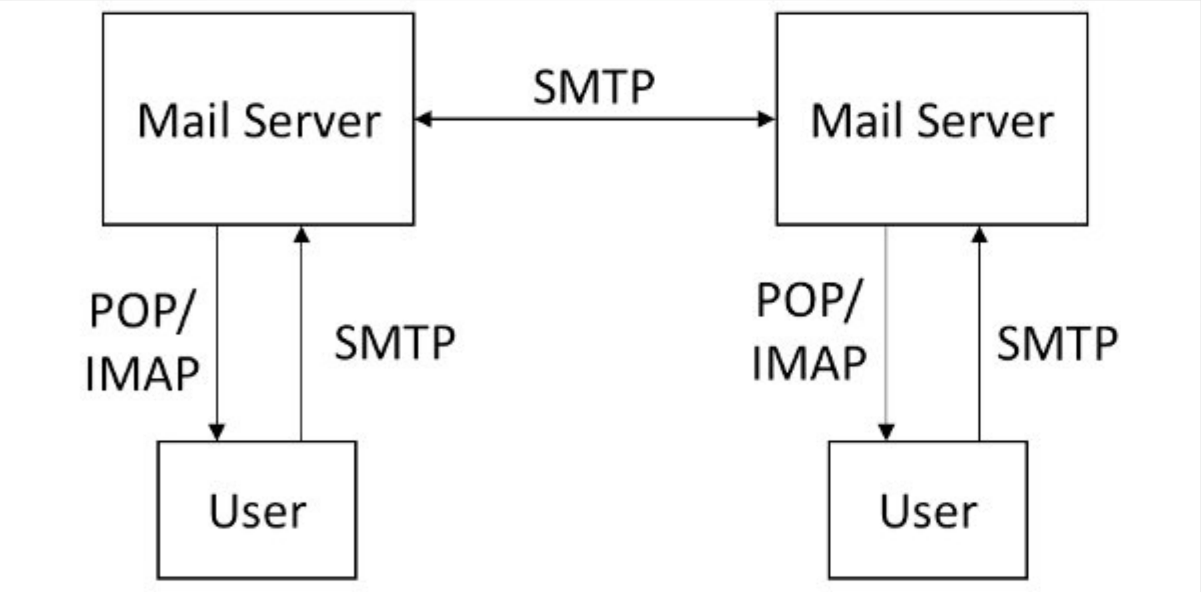


```
Begin
    Until each cell contains exactly one machine, Do
        Identify machines n1 and n2 such that $d_{n1,n2}$ is the minimum.
        Assign n1 and n2 to two different and empty cells.
        Discard machines n1 and n2 from the unassigned machines set.
        If only one cell is remaining then
            Assign n1 to this cell
            Discard machine n1 from the unassigned machines set.
    End Until
    Until unassigned machines set becomes empty, Do
        Identify machines n1 and n2 such that $d_{n1,n2}$ is the maximum
        Assign n1 and n2 to the same cell
    End Until
    Read V (* interactively from the user *)
    Add V% dummy individual machines to each cell, such that the C cell sizes are equal
End
```

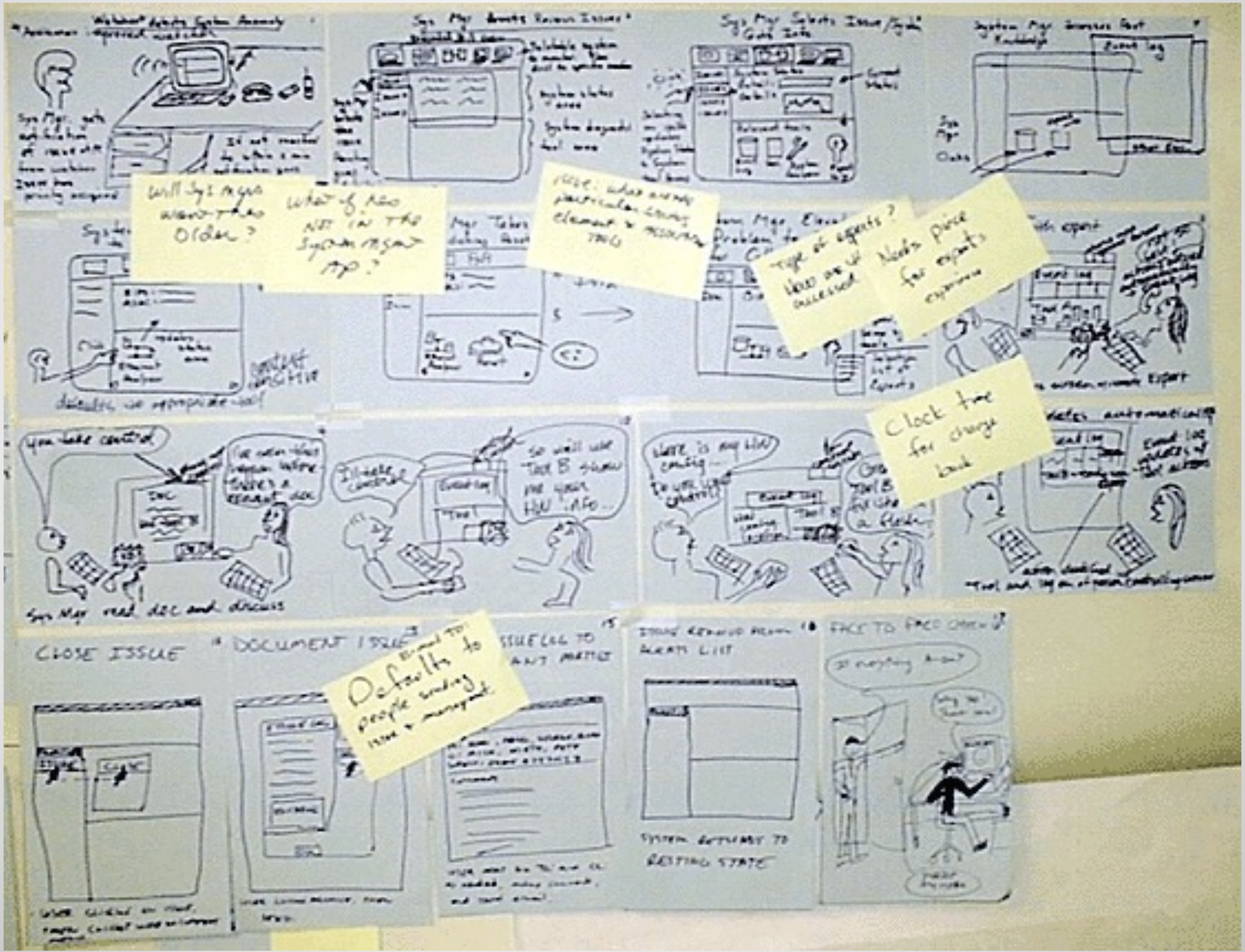
Other Diagrams – Entity Relationship Diagram



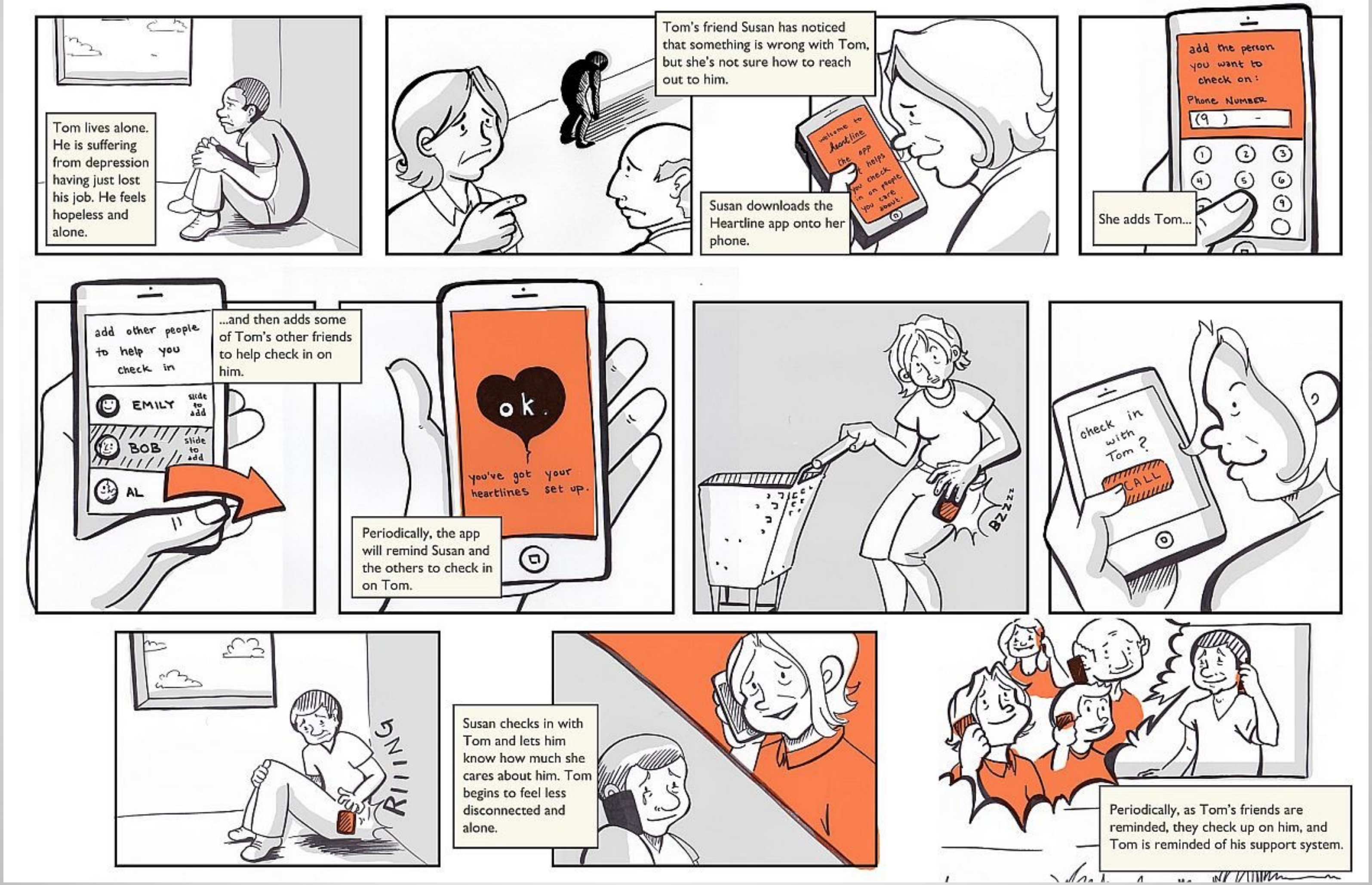
Other Diagrams – Architecture Diagrams



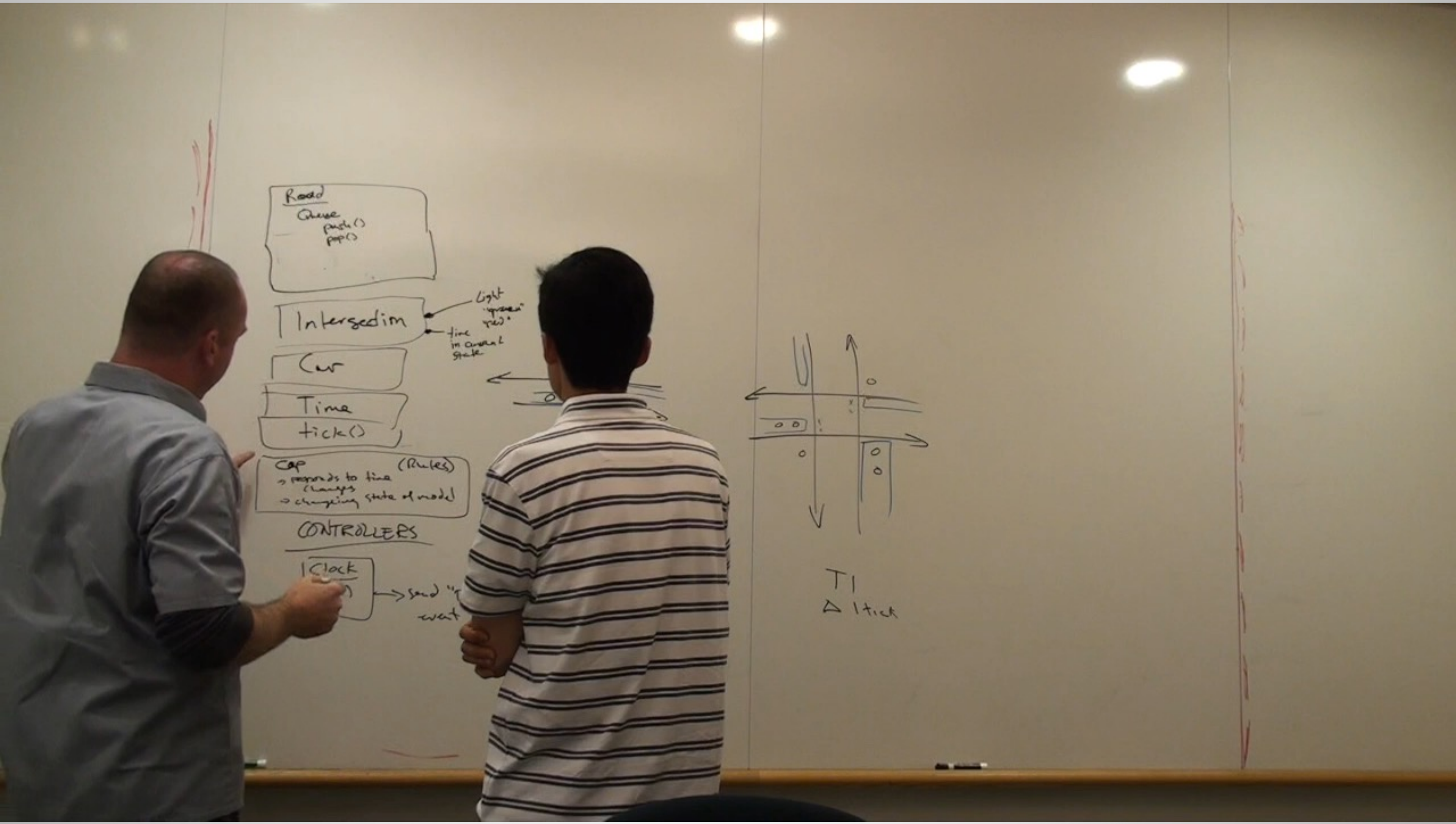
Other Diagrams – Storyboard


Will Sys Mgr want this order?
What if Rec NOT IN THE System Mgr App?
Type of experts? How are we accessed
Needs place for experts experience
Clock time for charge back
You take control
Defaults to people sending issue + management
CLOSE ISSUE
DOCUMENT ISSUE
FACE TO FACE CHECK
SYSTEM RETURNS TO RESTING STATE

Other Diagrams – Storyboard


Tom lives alone. He is suffering from depression having just lost his job. He feels hopeless and alone.
Tom's friend Susan has noticed that something is wrong with Tom, but she's not sure how to reach out to him.
welcome to Heartline the app that helps you check in on people you care about.
Susan downloads the Heartline app onto her phone.
add the person you want to check on:
Phone Number
(9) -
1 2 3 4 5 6 9
She adds Tom...
add other people to help you check in
EMILY slide to add
BOB slide to add
AL
...and then adds some of Tom's other friends to help check in on him.
ok.
you've got your heartlines set up.
Periodically, the app will remind Susan and the others to check in on Tom.
BZZzz
check in with Tom?
CALL
RIIIING
Susan checks in with Tom and lets him know how much she cares about him. Tom begins to feel less disconnected and alone.
Periodically, as Tom's friends are reminded, they check up on him, and Tom is reminded of his support system.

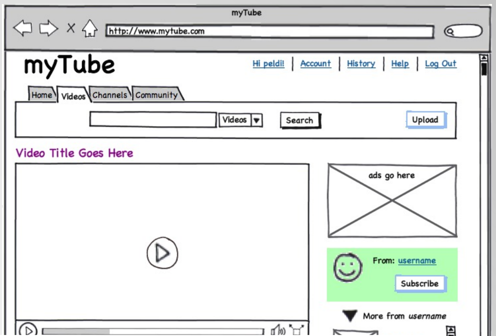
Other Diagrams - Sketches

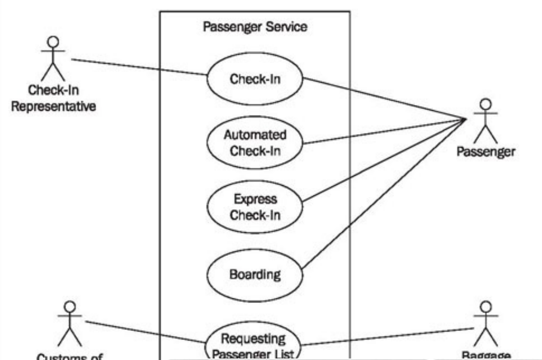


Other Diagrams - Sketches

What is Software Engineering?

Software engineering is the process of building a set of related models *that represent the system-to-be.*





Trigger: Blink on campus. Electrician calls Duty Officer.

The lights flickered.
Ok, I'll forward the message to the head engineer.
Electrician
Duty Officer

Duty officer forwards message to head engineer who also receives a text message from the software.

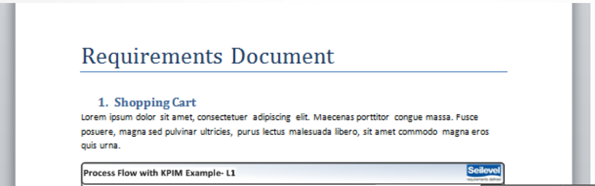
Head Engineer
I'll investigate the issue.
Text msg of alert

Head engineer logs into software to see the alert.

Head Engineer

Calls electrical foreman who sends out an electrician to resolve the issue.

Head Engineer
Here's the issue that needs to be addressed
I'll dispatch an Electrician to fix the issue.
Electrical Foreman


Requirements Document

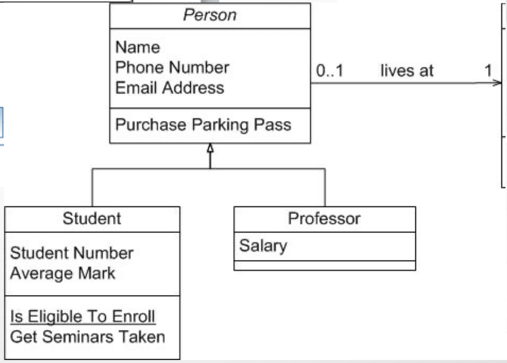
1. Shopping Cart

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Process Flow with KPIM Example- L1

Seilevel
Collect mortgage application information
Organize application information
Send to underwriting for approval
Generate appropriate file format
60 loan applications are organized per day per processor
100% of loans have no errors on first submission
Average five business days from when information is collected to bank submission

```
        state.getActionStateRepository().ge
                .add((FireAction) action);
        ruleExec.update(gui, RuleExecutor.
                "FireDestroyObjectsRuleA",
        destChecker.update(false, gui);
    } else if (action instanceof PurchaseTo
        Vector<SSObject> participants = act
        for (int i = 0; i < participants.si
            SSObject obj = participants.ele
            if (obj instanceof Employee) {
                if (menuText.equals("Purcha
                    ((Employee) obj)
                            .setOverheadTex
                }
            } else if (obj instanceof Custo
                if (menuText.equals("Purcha
                    ((Customer) obj)
                            .setOverheadText("Tool(s) have been purchased!");
                }
            }
        }
        state.getActionStateRepository()
                .getPurchaseToolActionStateRepository()
                .add((PurchaseToolAction) action);
        ruleExec.update(gui, RuleExecutor.UPDATE_ONE,
                "PurchaseToolEffectRuleA", action);
        destChecker.update(false, gui);
    }
```



Today’s Lecture - **How do we structure the software in detail?**

- Design: recap
- Design notations / diagrams
  - UML diagrams
  - Other diagrams
- Design principles

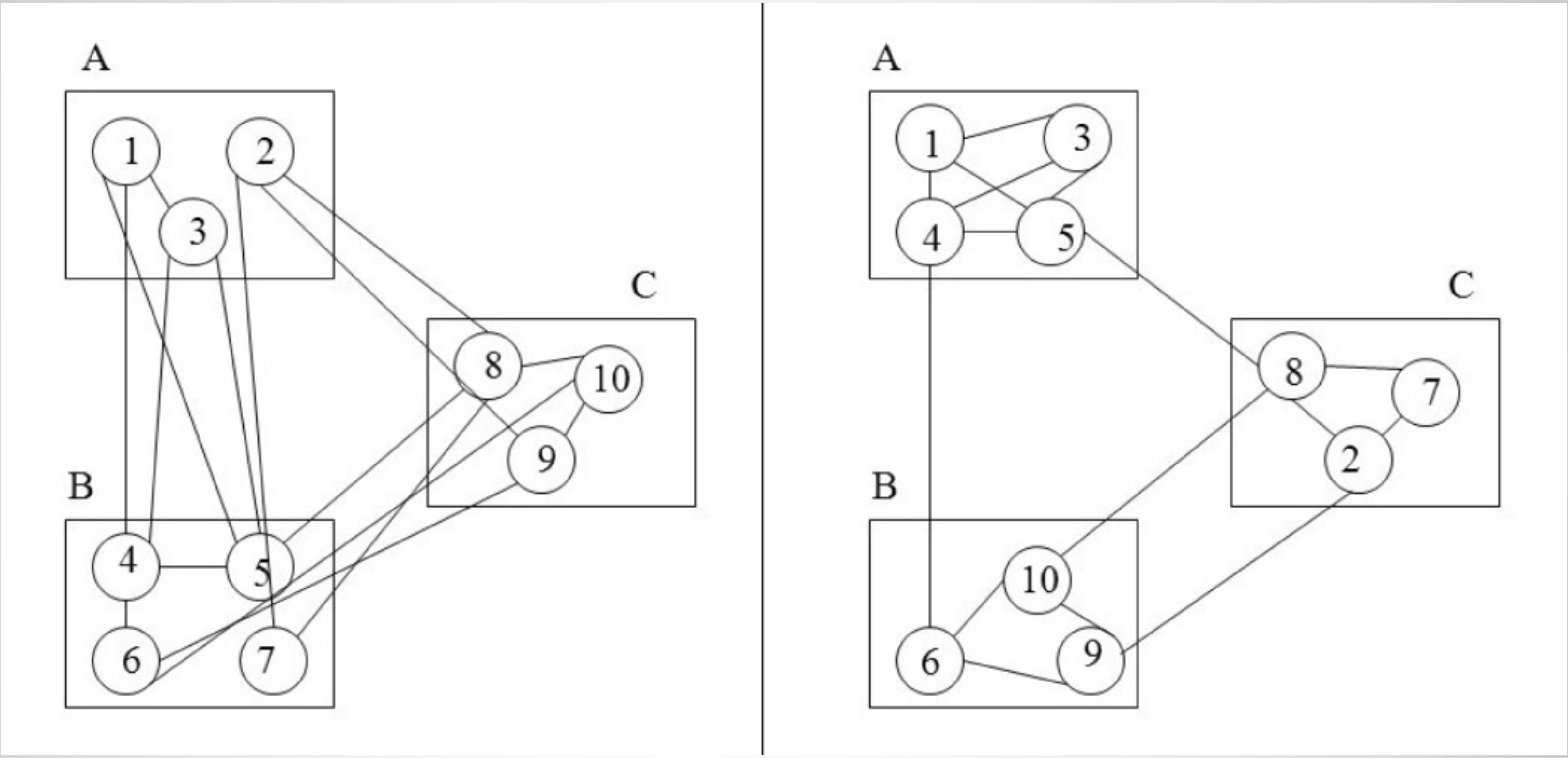
Design Principles

- High cohesion/low coupling
- Information hiding
- …

High Cohesion/ Low Coupling

- High Cohesion: Grouping related functionality
- Low Coupling: Ungrouping unrelated functionality / reducing interdependency
- Effects:
  - Changes don't propagate
  - Reuse is facilitated

Cohesion/Coupling



Low cohesion/high coupling ☹ High cohesion/low coupling ☺

Information Hiding

- Hide design decisions that are most likely to change, thereby protecting other parts of the program from change if the design decision is changed
- *"Showing only those details to the outside world which are necessary for the outside world and hiding all other details from the outside world." -http://cs-study.blogspot.com*

Summary

- Every design notation supports an abstraction
- A design diagram is a statement in a language that has a syntax
  - UML diagrams, UI mockups, pseudo code, ER diagrams, architecture diagrams, storyboards, sketches
- Software engineering is the process of building a set of related models that represent the system-to-be.

Quiz 4

- 6 approaches to software design (architecture, functional decomposition, relational database design, object-oriented design, user interface design, sketching)
- Purposes of designs (think, communicate, prescribe)
- Abstraction
- Diagrams: UML class diagrams, associations, multiplicities
- Design principles: Cohesion/coupling, information hiding

Next Time

- User orientation