Design
Decomposing User Stories

Thomas Fritz

Many thanks to Reid C. Holmes and Elisa Baniassad

Agenda

1. Diagrams Part II (Class & Sequence Diagrams)
2. Decomposing User Stories
3. Class Exercise

Note:

- *Midterm 10.04 in KOH-B-10 at 10:00; Safe Browser Exam, BYOD, closed book*
- *Mock exam 20.03 at 18:00, mandatory, HAH-E-3 (functional test the days beforehand)*

Examinable skills

By the end of this class, you should be able to…

- Understand, create and critique diagrams in software development
- Decompose a user story and create UML diagrams to depict the design that supports the user story
  - Identify elements and behaviours in user stories
  - Distinguish between fields and classes
  - Identify sequences of behavior
  - Identify relationships (associations)
  - (Potentially improve the user story)
- Analyze/critique lower-level designs created from a set of user stories

Design Recap & Diagrams – Part II

Learning Objectives

Be able to:

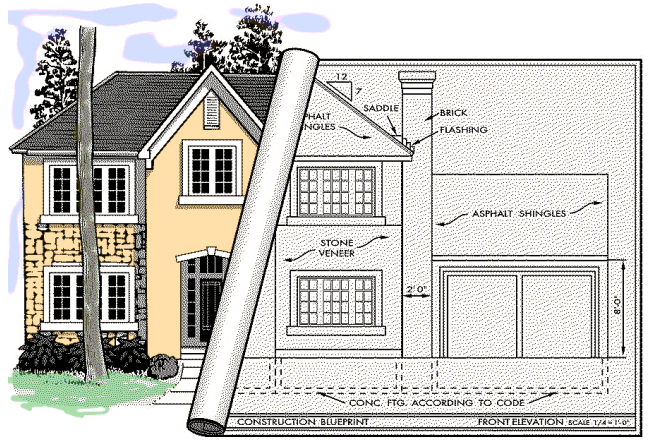
- Explain the importance of software design
- Read and create basic UML diagrams

Design recap

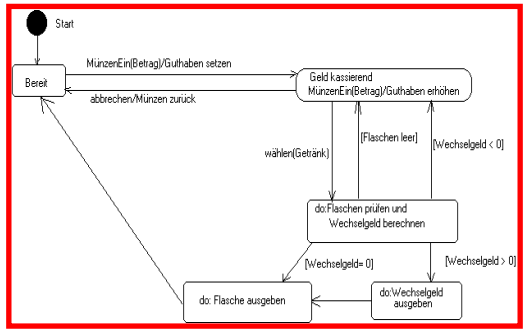
- Design consists of multiple views (both static & dynamic)
- Diagrams are communication tools
- Designing is an iterative refinement process
  - Start “rough”, flexible then clean up; use paper, discuss, iterate, use UML tools later on

Unified Modeling Language (UML)

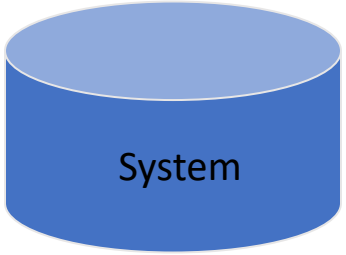
A “diagram language” to visualize, specify, construct and document a software system (historically object-oriented)



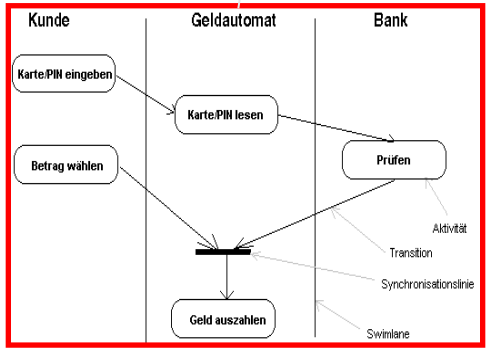
Behavior / Dynamic



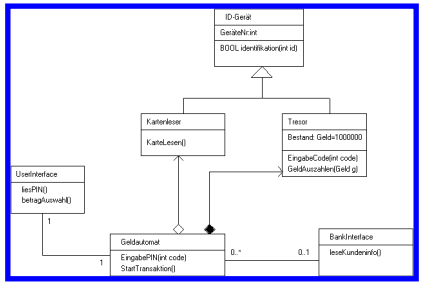
State machine diagram



...

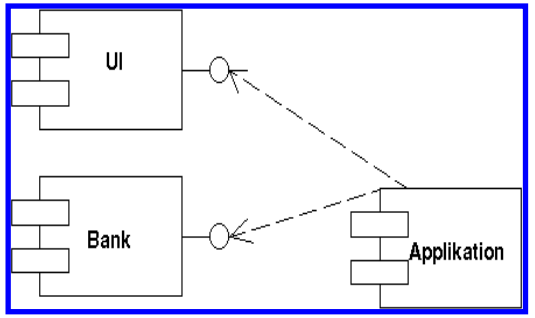


Activity diagram



Class diagram

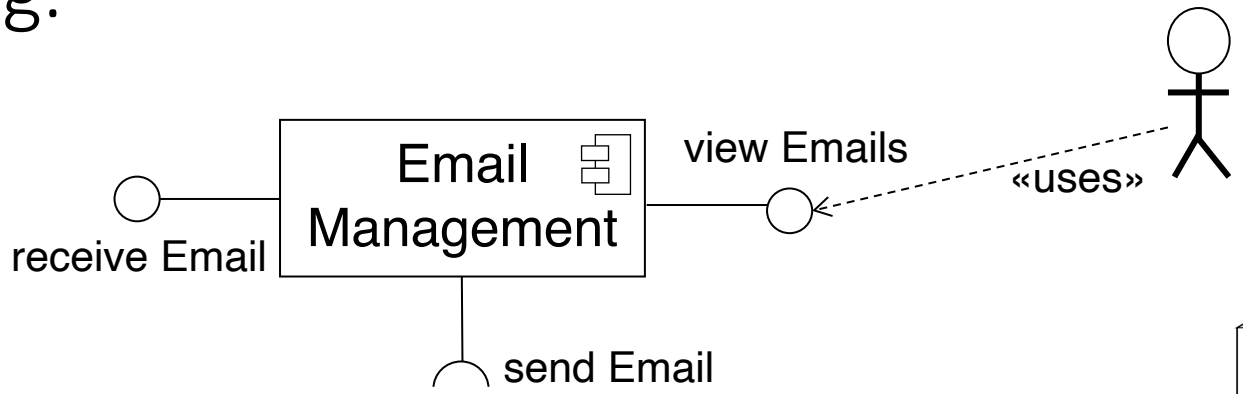
Structural / Static

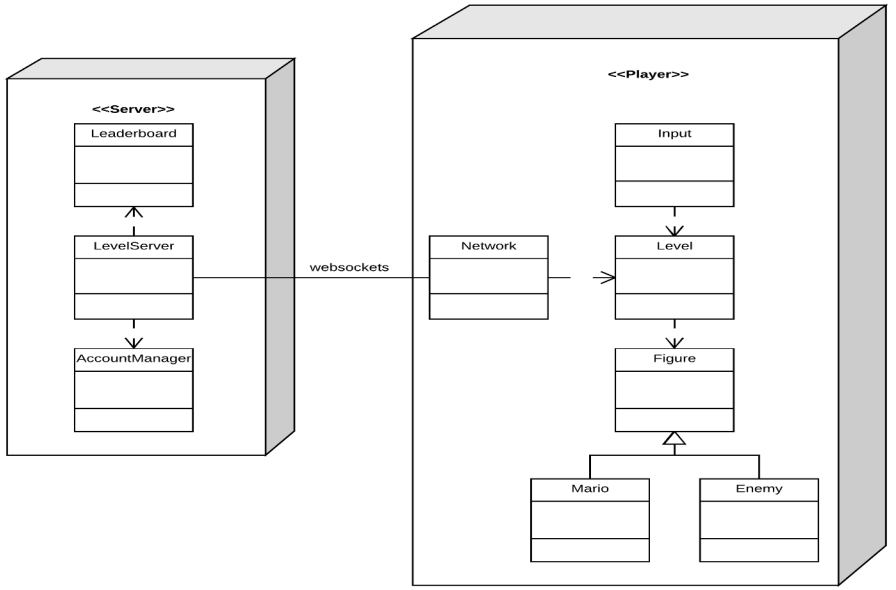


Component diagram

Last week – component & deployment …

e.g.





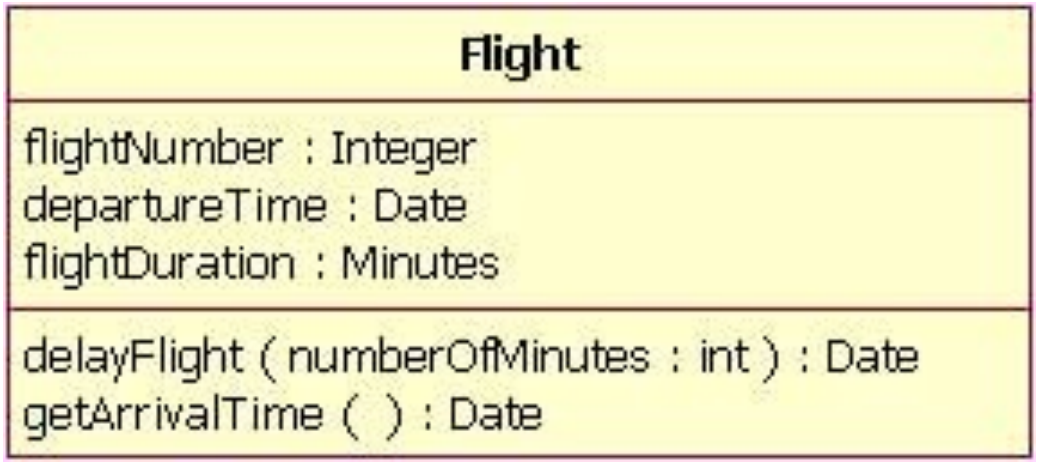
Another view – UML Class Diagram
(brief recap – Software Construction)

- Focus on core classes your design requires
- Focus on relationships between the classes

- Some of the symbols to use (there are also others)
  - three part rectangle for a class, inheritance (extends), implements (realizes), association,…

Class diagram – class (recap & Question)

- Class name (*italics* means abstract)
- Attributes (fields)
  - Name : Type
- Operations (methods)
  - Parameter : return type
- Can also be used for interfaces (without fields or just constants)

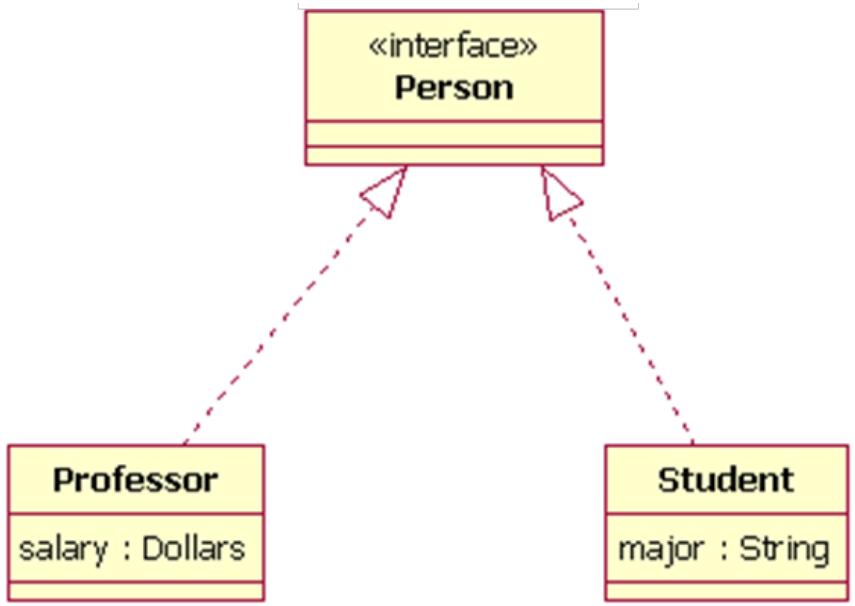


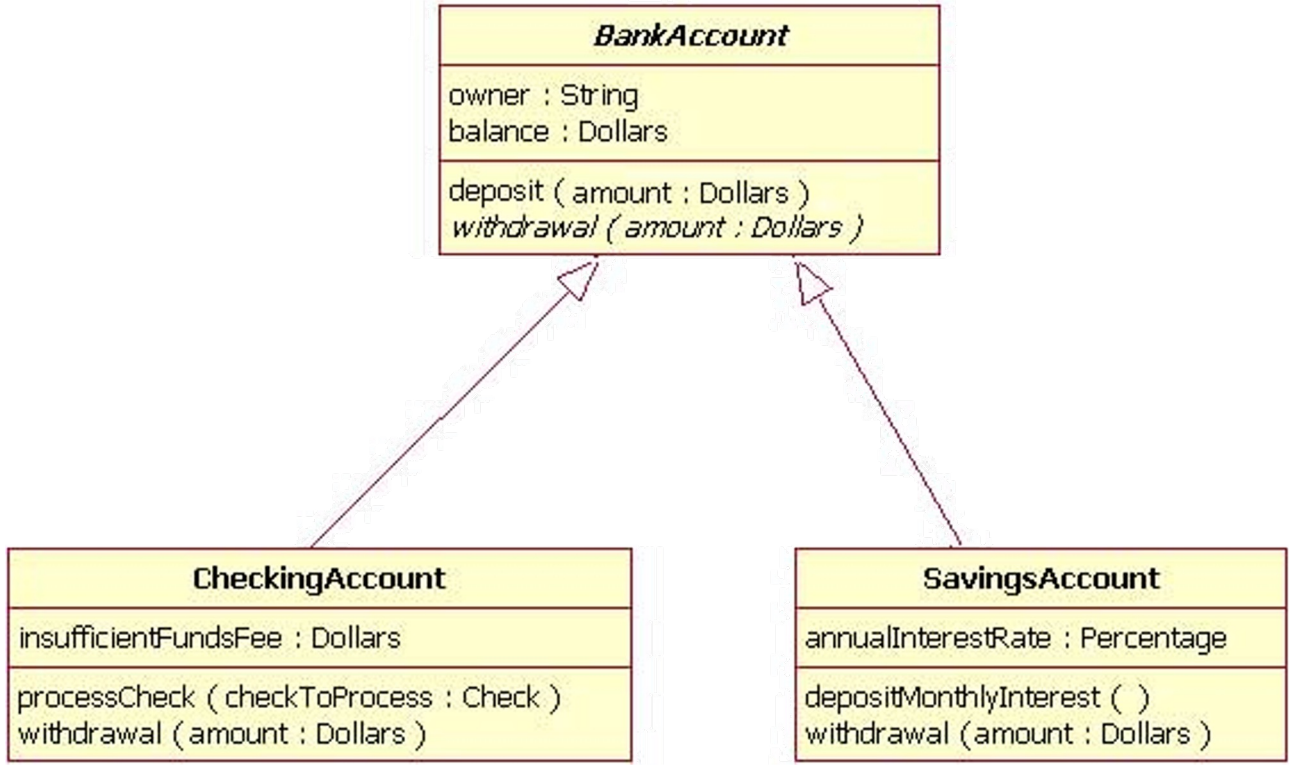
Questions:

- *What is an abstract class?*
- *What is the difference to an interface?*

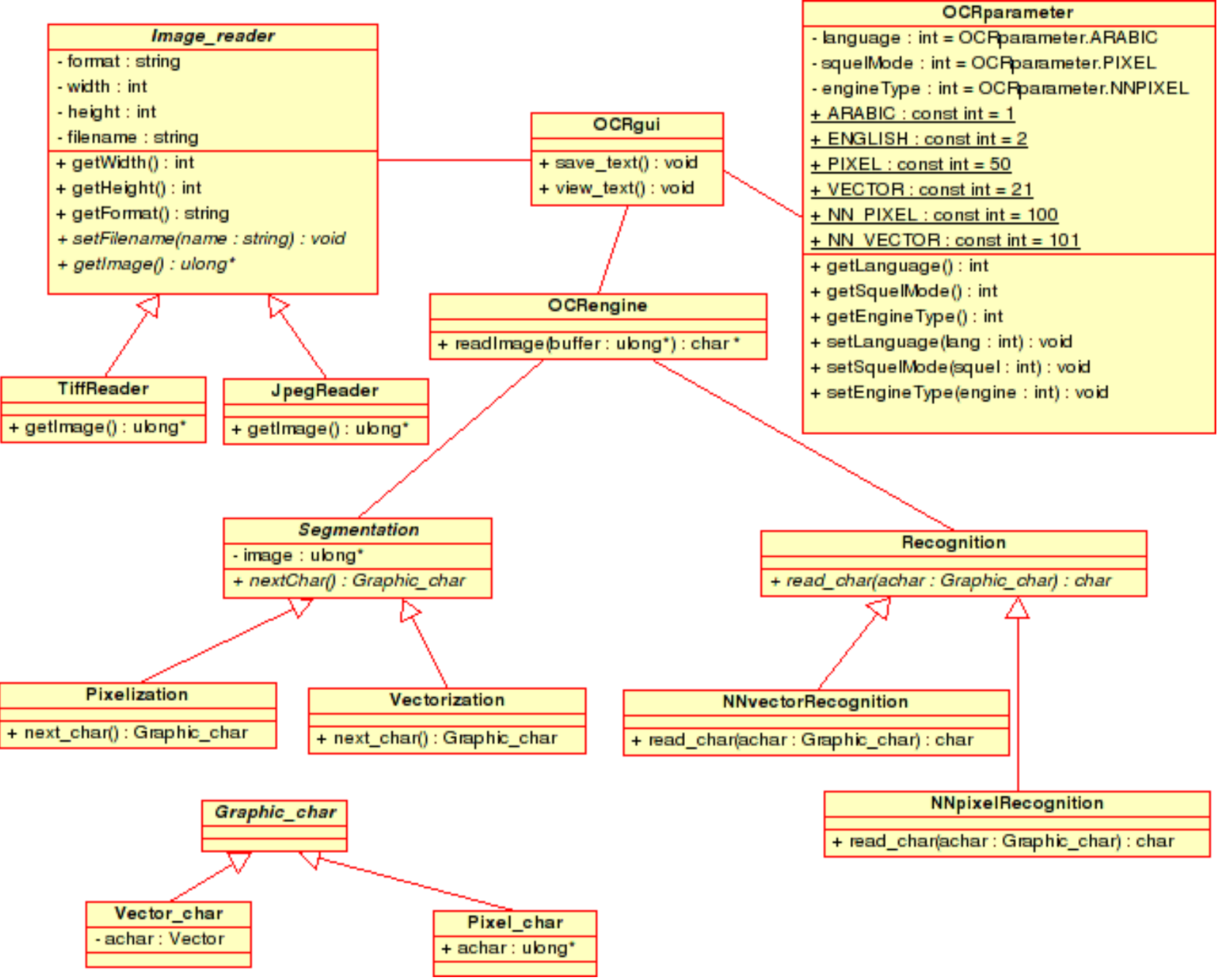
Class diagram – generalization

Used for inheritance and interface implementation





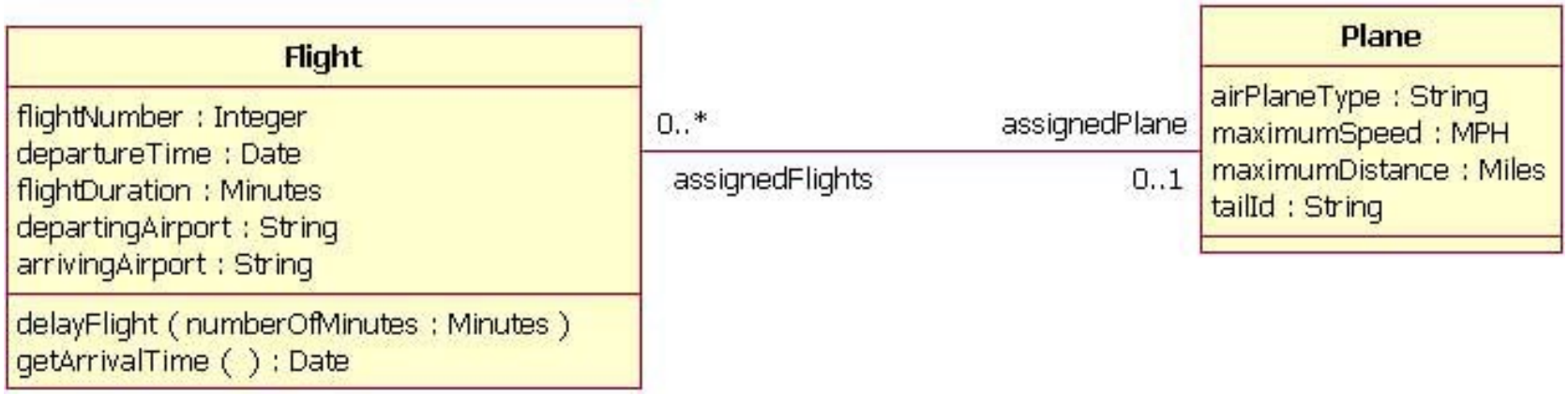
Question: *What is the difference between an interface and a super class?*



Questions:

- *How many super-classes does Vectorization have?*
- *How many fields can you be sure that OCRgui has?*

Class diagrams – association

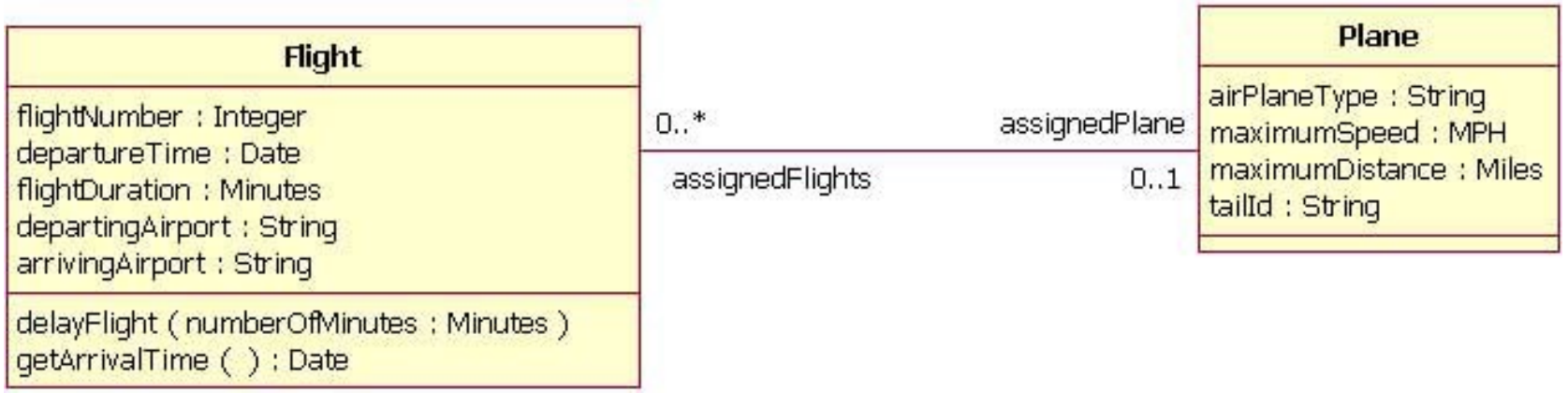


Bi-directional: both classes are aware of each other

Role: usually maps to a field name

Multiplicity: indicates how many instances can be linked (e.g. a list of…)

Question – association implementation (3-5min) [https://bit.ly/3Pq5tzL]

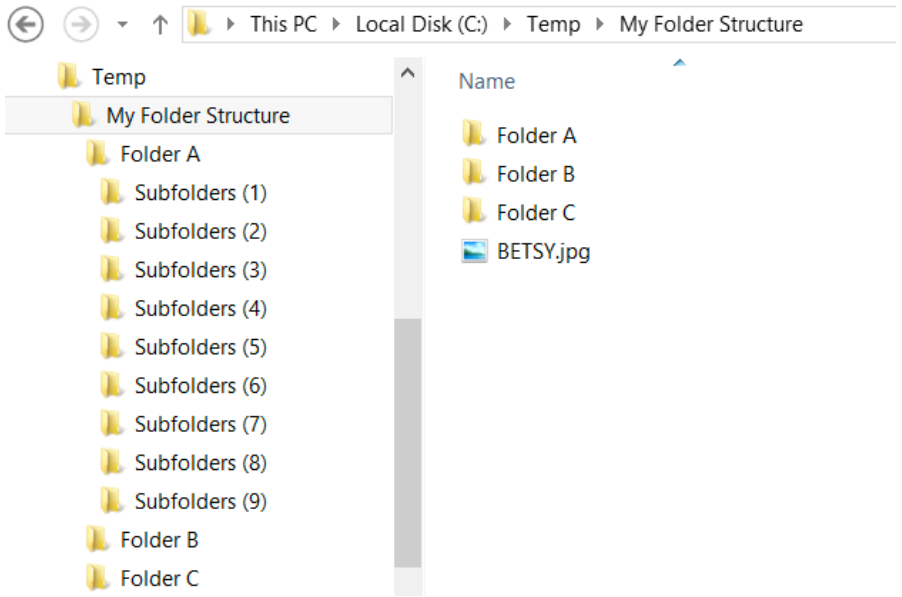


How to implement this association (in Java)?

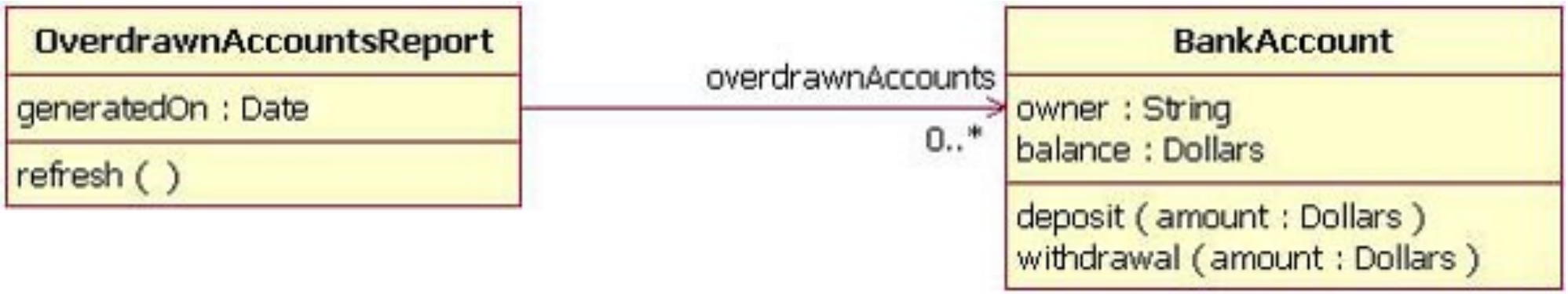
Question – association example

How to depict the following?

In the file system, a folder can again contain folders



Questions – uni-directional association



Question:

- *What is the difference to a bi-directional one?*
- *Why is it easier to change code after if it's only a uni-directional vs a bidirectional association?*

Class diagram – aggregation

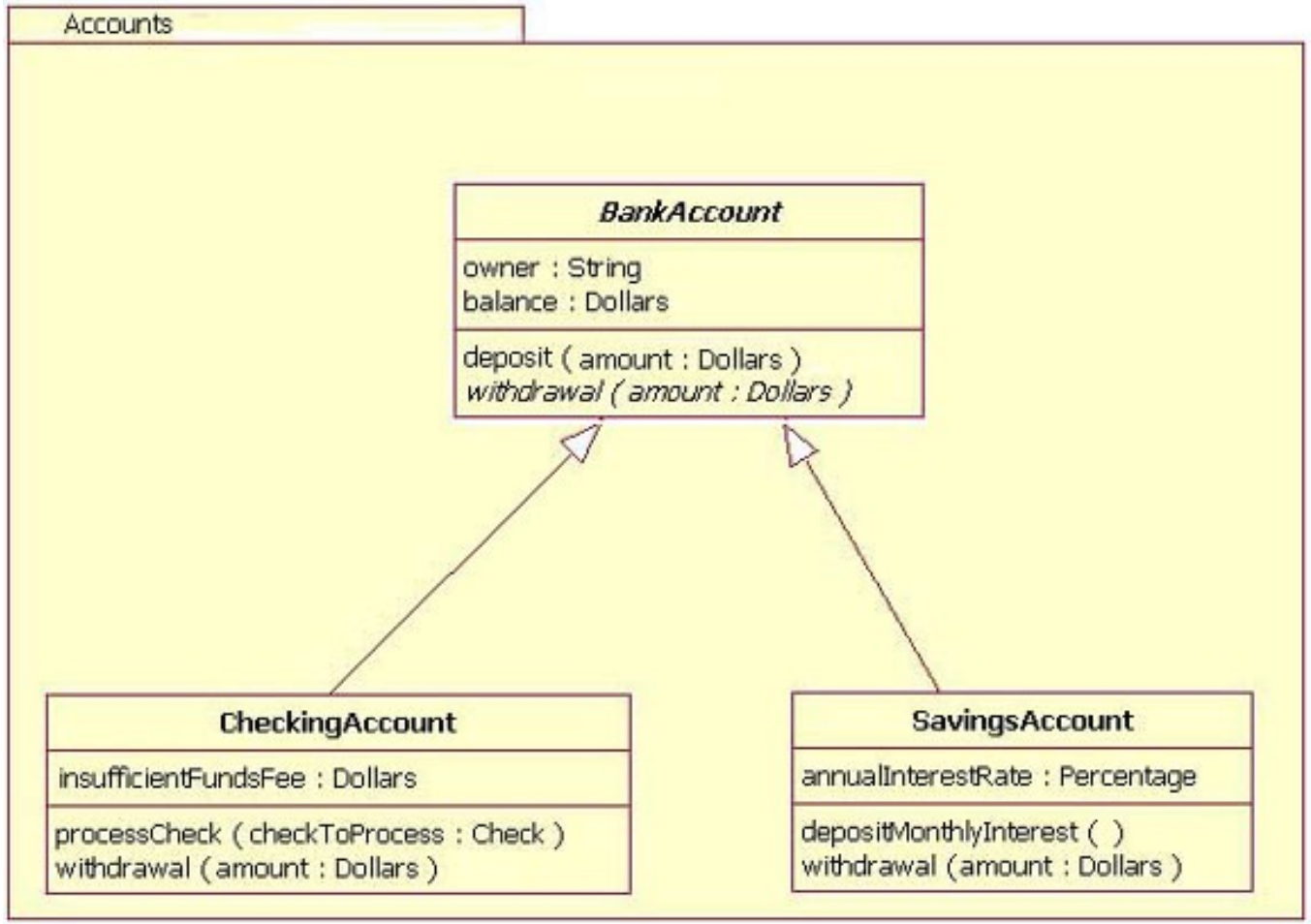
- Advanced type of association
- Contained object ***is part*** of container
- Two types
  - Basic aggregation: children can outlive parent
  - Composite aggregation: children life depends on parent

Question:

- *Should car – wheel be aggregation or composition?*

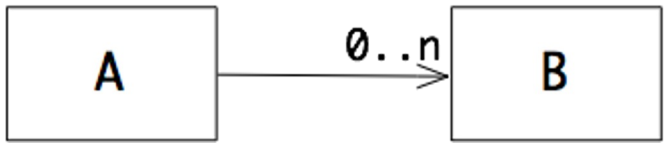
Class diagrams – packages

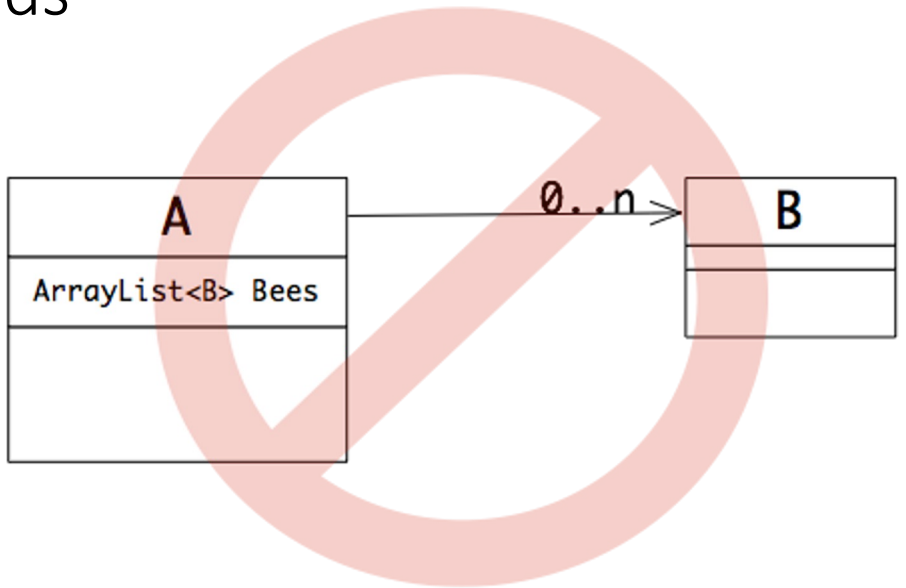
Group classes together



Class diagrams – comments

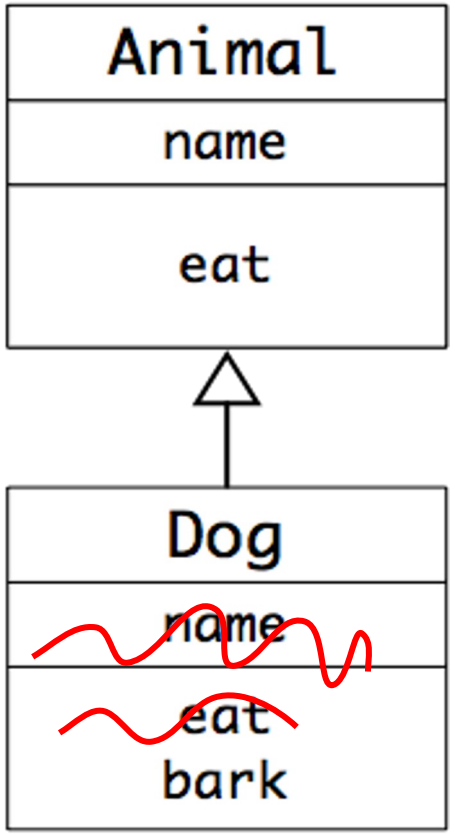
- Think about association/aggregation/composition as well as direction
- Specify roles unless obvious (especially if there are two relations between two classes)
- Put lists as associations, not as fields





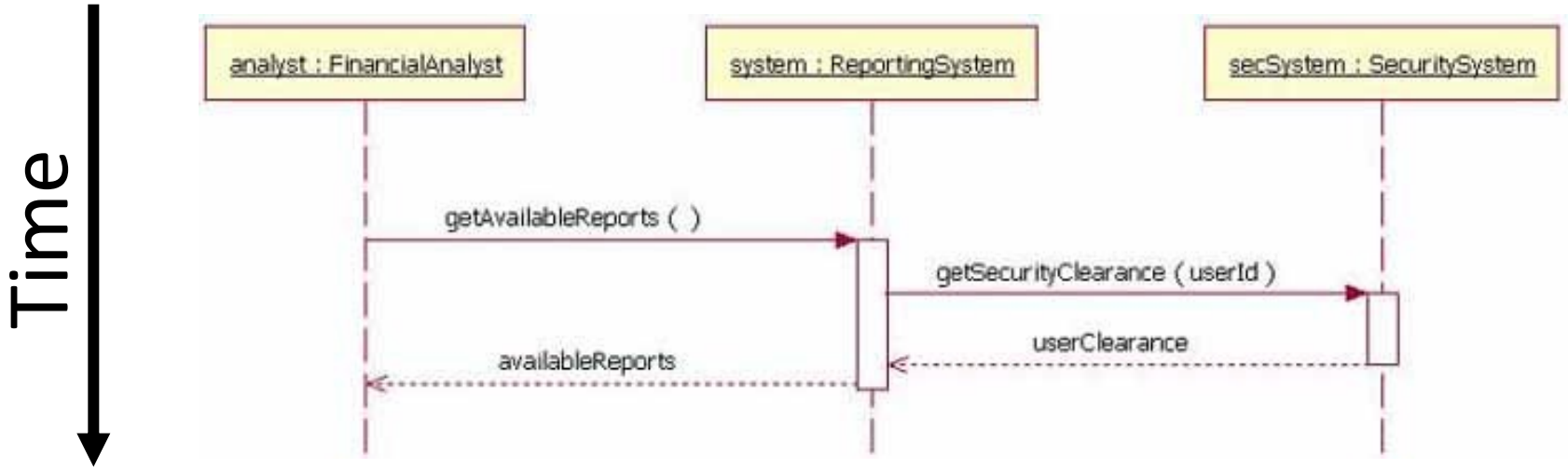
Class diagrams – comments

- Do not put inherited fields and methods in subtype unless you are overriding them
- Class names: singular nouns or doers (start with capital letter)
- Method names: verbs, actions or isX checks (start with lowercase letter, camelCase)



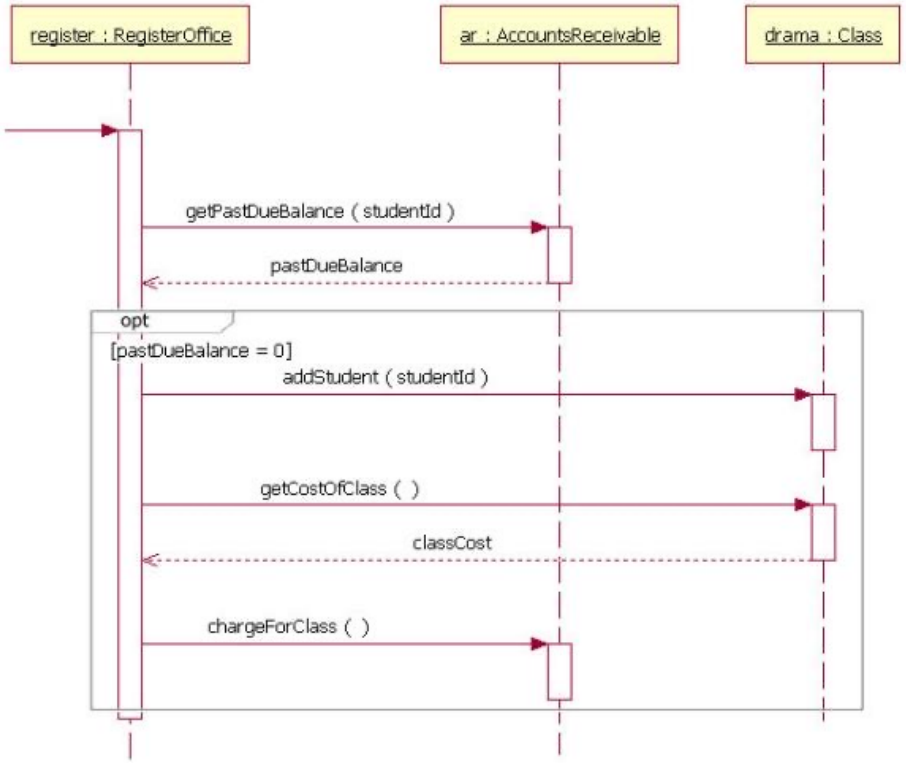
Sequence diagram – recap & questions

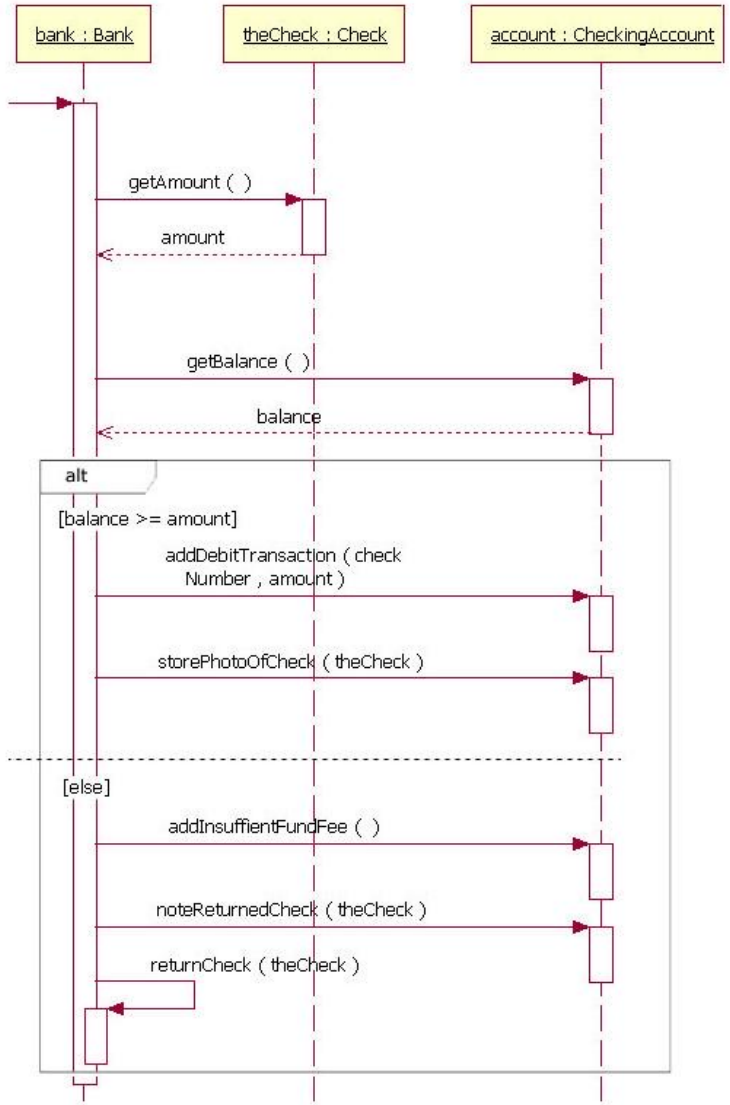
Low-level design tool to describe sequences of invocations between objects



- Roles or object instances that participate in the sequence being modeled
- Box in lifeline indicates activation
- Full arrow: synchronous (blocking until message returns);
  asynchronous: can continue processing

Sequence diagram – ‘if’

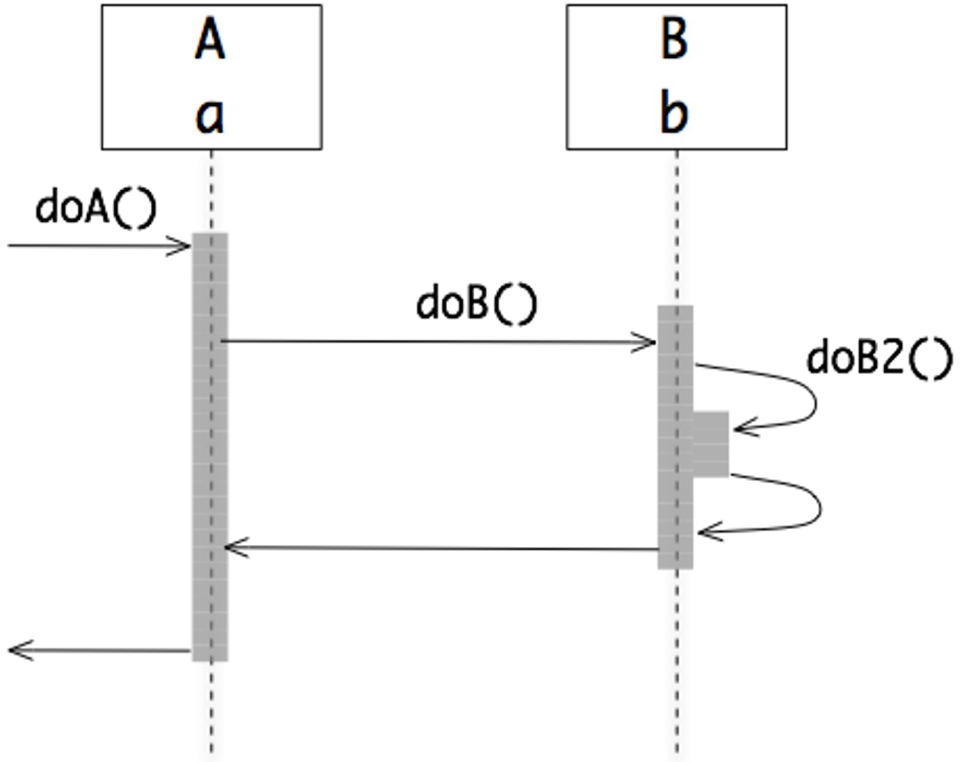




Question: *Is an option just an if without an else branch?*

From: http://www.ibm.com/developerworks/rational/library/3101.html

Sequence diagram – to code




```
a.doA(b);


public class A {
    public void doA(B b) {
        b.doB();
    }
}

public class B {
    public void doB() {
        doB2();
    }
    public void doB2() { }
}
```

Class Activity

Decomposing User Stories (Thanks to Elisa Baniassad)

Learning Objectives

Be able to:

- Decompose a user story and create UML diagrams to depict the design that supports the user story

Take a User Story

value statement: As someone who wants to keep track of their hierarchy of folders and files, I want to be able to see a tree view that updates so that I can tell immediately whenever a change is made to the system

acceptance criteria:

- the folders must be able to contain both folders and files
- the view must be organised in tree format reflecting the folder hierarchy
- the view must update when a change is made to the file system (new file or folder created, or file or folder renamed, etc)

Notice this user story is NOT about actually MAKING the change to the file system — just about displaying it!

Let’s decompose!

value statement: As someone who wants to keep track of their hierarchy of folders and files, I want to be able to see a tree view that updates so that I can tell immediately whenever a change is made to the system

acceptance criteria:

- the folders must be able to contain both folders and files
- the view must be organised in tree format reflecting the folder hierarchy
- the view must update when a change is made to the file system (new file or folder created, or file or folder renamed, etc)

NOTE: DECOMPOSITION is NOT NEATLY SEQUENTIAL!!!
I am just pointing out some of ACTIVITIES involved in decomposing to get you started! (This is the early stages of “design”, which is actually a highly personal, and instinctual process. There is no formula!)

Activities when decomposing...

THIS IS NOT AN ORDERED SET OF STEPS

NOUNS Identify actors (objects/classes)

VERBS Identify behaviours and associate those with actors

OWNERSHIP/ KNOWELDGE Identify relationships between actors (fields)

SEQUENCES Identify calls between behaviours

Reid's video: find entities, link entities, bind actions, [prototype, formalize, implement]

Identify “actors”, “entities” and “responsibilities”

value statement: As someone who wants to keep track of their hierarchy of folders and files, I want to be able to see a tree view that updates so that I can tell immediately whenever a change is made to the system

acceptance criteria:

- the folders must be able to contain both folders and files
- the view must be organised in tree format reflecting the folder hierarchy
- the view must update when a change is made to the file system (new file or folder created, or file or folder renamed, etc)

Another Tiny Example: Where's the party at? (5mins on your own, create class diagram)

Description: a mobile app that tells you where all the cool people are hanging out (for some definition of cool)

User Story: As a senior citizen, I would like to be able to identify all the open games close to where I am currently located, so that I can reserve a seat at one.

Architectural Decisions: Mobile application. user client can register games, which are then saved on the server. The server holds the location and times of games such that they can be queried by the client.

Definitions of Done: User is on the search screen, and enters a keyword game type into the search area, and is shown a list in a games view, where games are sorted by nearness to where they live. User selects a game, taking them to the "reserve game" view.

Give it a go: Work through nouns, verbs, fields, calls

Question: what are the classes?

Definitions of Done: User is on the search screen, and enters a keyword game type into the search area, and is shown a list in a games view, where games are sorted by nearness to where they live. User selects a game, taking them to the “reserve game” view.

Questions & Class Exercise

Designing a social networking app
(Note: user stories are still a bit rough)

Learning Objectives

Be able to:

- Decompose a user story and create UML diagrams to depict the design that supports the user story

Questions to discuss

1. Diffferent people might decompose a user story differently, how do you ensure the project is not too diverse in the end?

2. Is it always possible to decompose user stories into small enough elements for one sprint?

3. Can you decompose a user story the wrong way?

Decomposing User Stories Exercise (~20mins, if time allows) [https://bit.ly/3vnWgkw]

- In teams sketch a design using a UML class diagram

As a user, I want to be able to create a profile with my name and a password in order to be part of the social networking app.

DoD: users can register with their name and password; profile will be created; users with a profile can login to the social networking app;

As a logged in user, I want to be able to add posts to my own messaging board in order to share them with my friends.

DoD: users can login and will see their own messaging board; users can add a post to their own messaging board and can either choose to post a text, a picture, or a link to a website; once the post is submitted/sent, friends will be notified about the post.

As a logged in user, I want to be able to befriend others in order to see their posts.

DoD: users can login and search for other users; they can befriend other users by clicking the befriend button of other users that are no friends yet; once befriended, they will be listed in the friend list of the user

Quiz & more

Next week:
Designing APIs & REST APIs

- Listen to specified videos and readings

- Fill in Quiz

- Mock exam: today!
- Midterm (in person): 10.04.2023