SOFTWARE ENGINEERING MODELS AND MODELING

ANTTI KNUTAS (D.SC.)

Course Q&A

Please ask any question (or post them in Echo360 lecture Q&A area)

Course Q&A

I'll address the first question here: What events and attendance options we have?

Interactive events

- Lectures, in Lappeenranta (you can replace these by watching videos)
- 2x long seminars in Lahti
- Exercises, on both campuses (plus online)

There's a tiny point bonus from coming to work with us on these.

Materials

- Online lectures
- Coursebook
- Other links

How we'll proceed today

- I'll introduce the topic with a short, 30 to 45min lecture
- Then, we'll proceed to work on analyzing the course project
- Lastly, groups are welcome to post their analysis on the course project assignment *before the end of day today* (for a small point reward)
- There will be more in-depth videos posted online (on UML and modeling)

The small point rewards from interactive lectures, weekly exercises, and seminars overlap => it is enough to do one, or part from each

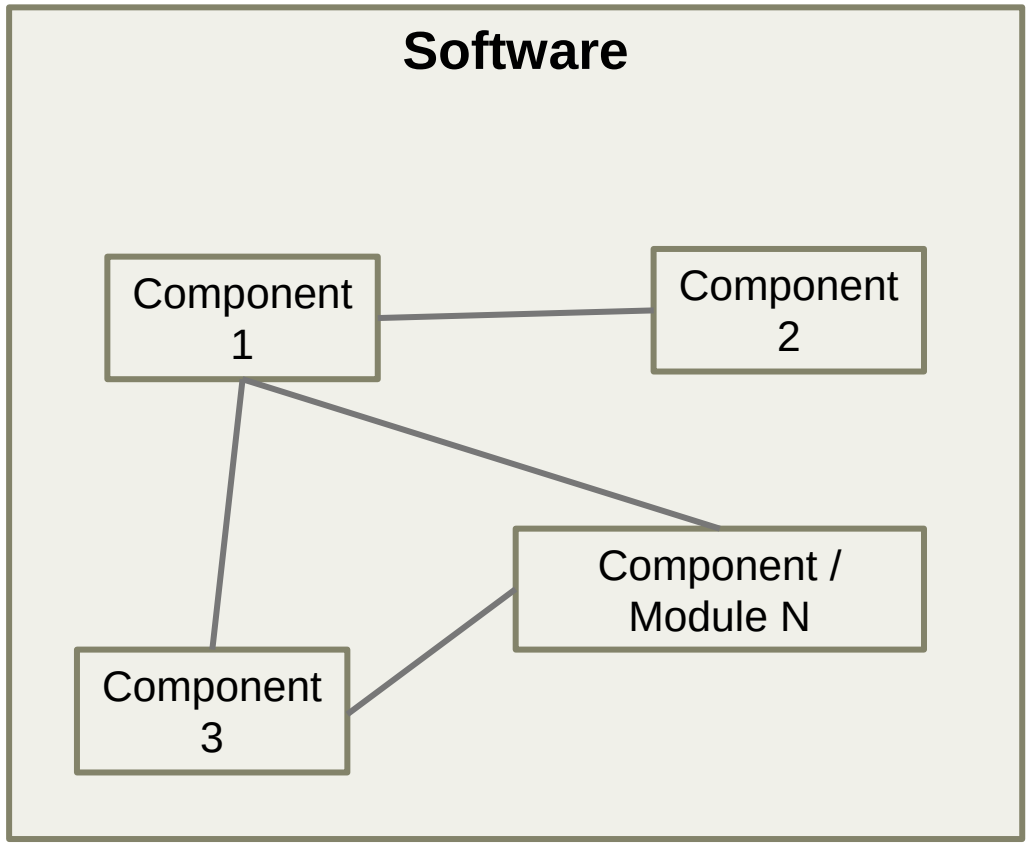
Weekly assignments are the main "big effort" before the course project. (10%)

LUT University

MODELING SOFTWARE WITH OBJECTS

LECTURE 3

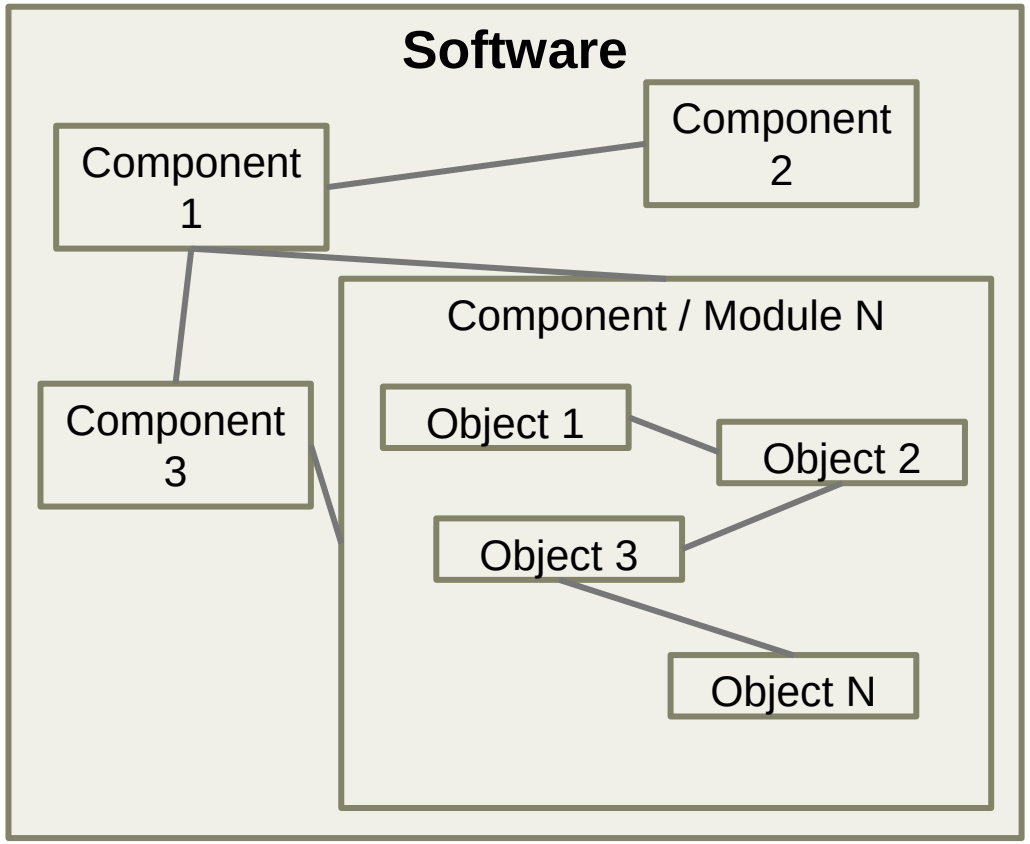
WHAT IS SOFTWARE?



Software: A compilation of interconnected components (also called modules) which exchange data to get things done.

Component (also module): A part of software system, usually responsible for managing one activity or role such as GUI drawing, event listener, networking etc…

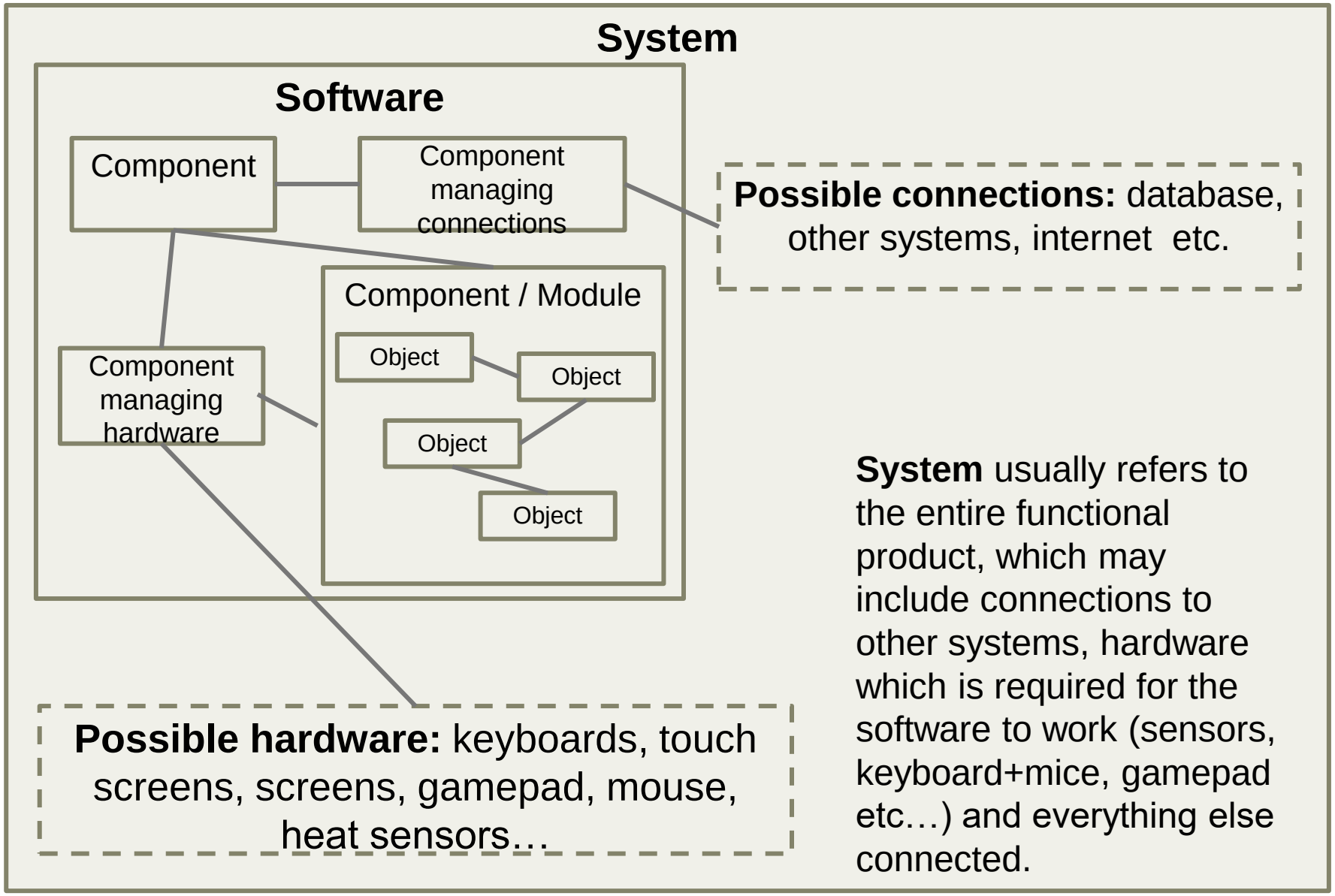
WHAT ARE OBJECTS?



Components (or modules) are also formed from a group of objects, which exchange information to enable the component to work**.**

Objects are formed based on defined classes, which are written in source code, and created by programming work to do **routines** and manage **messages**.

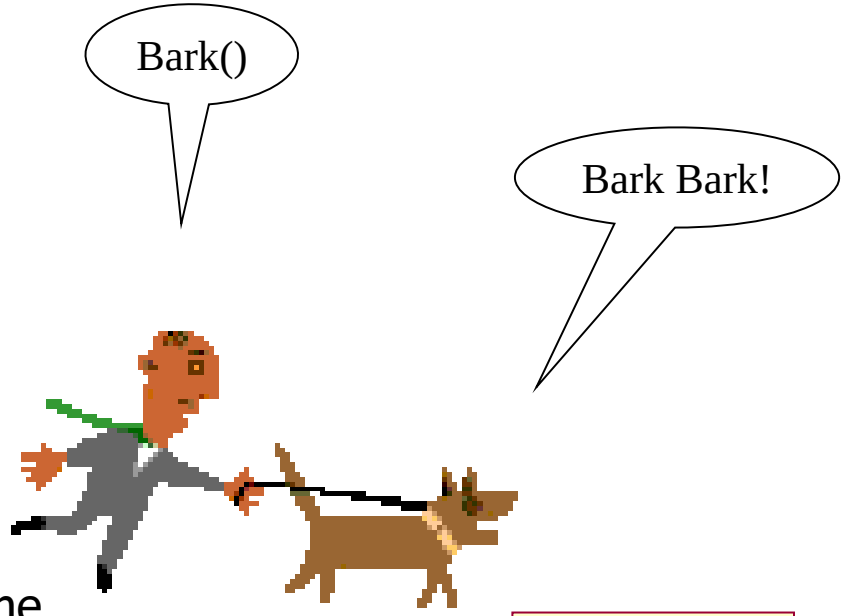
WHAT IS SYSTEM?



Review: object oriented concepts and thinking

- What is an object and what is a class?
- Messages
- Inheritance
- Polymorphism
- Abstraction
- Uses for objects
- Benefits and drawbacks of object orientation
- Object oriented technologies

Class or object?



Class
Attributes, state variables
Services, methods

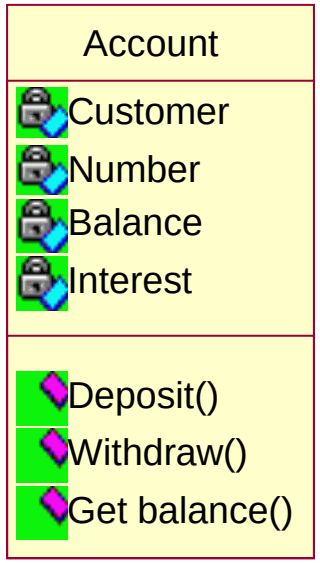
- Class
  - A set of similar objects that have same attributes and behavior
- An object has
  - identity
  - privacy (it acts independently)
  - class
  - state (attribute values ~ member variables)
  - behavior (services ~ operations ~ methods)

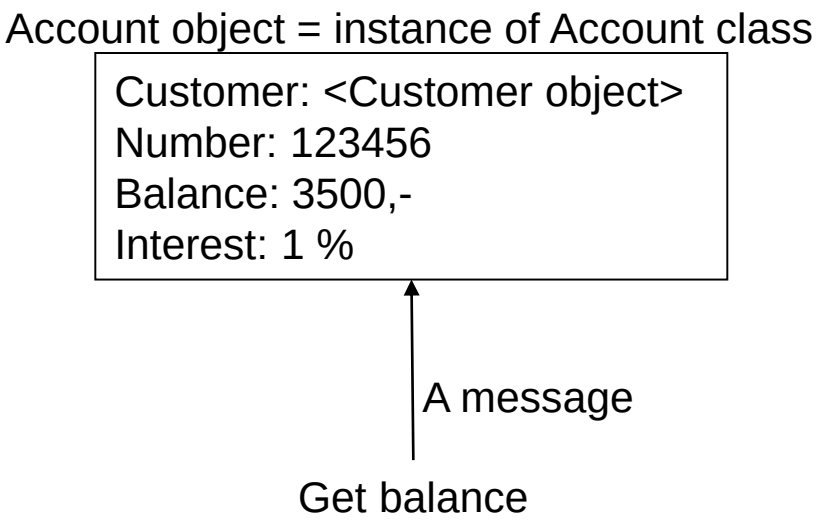
Fido : Dog

Name: Fido
Age: 3 years

Dog
Name Age
Bark()

Objects react to messages





- A message causes an operation execution in an object
  - Operations are stored in class definition
  - Operations are common to all class instances (objects)
  - Each object has its own state = attribute values
  - Method = operation implementation

Encapsulation and information hiding

- Information hiding
  - Hiding of *design decisions* in a computer program that are most likely to change, thus protecting other parts of the program from change if the design decision is changed (Parnas, 1972)
  - A module should implement and hide only one design decision and reveal as little as possible about its inner workings and data
- Related concepts
  - Cohesion
  - Coupling

Cohesion

- The extent to which an individual module is self-contained and performs a single well-defined function
- High cohesion
  - The module has strongly related and focused responsibilities
- Low cohesion
  - The responsibilities are varied and use unrelated sets of data

Coupling

- The degree of interdependence of modules
- High coupling
  - A module modifies or relies on the internal workings of another module
  - Change in a module requires a change in the dependent module
- Low coupling
  - A module interacts with another module through a stable interface and does not need to be concerned with the other module's internal implementation
  - Change in a module does not require a change in the other module

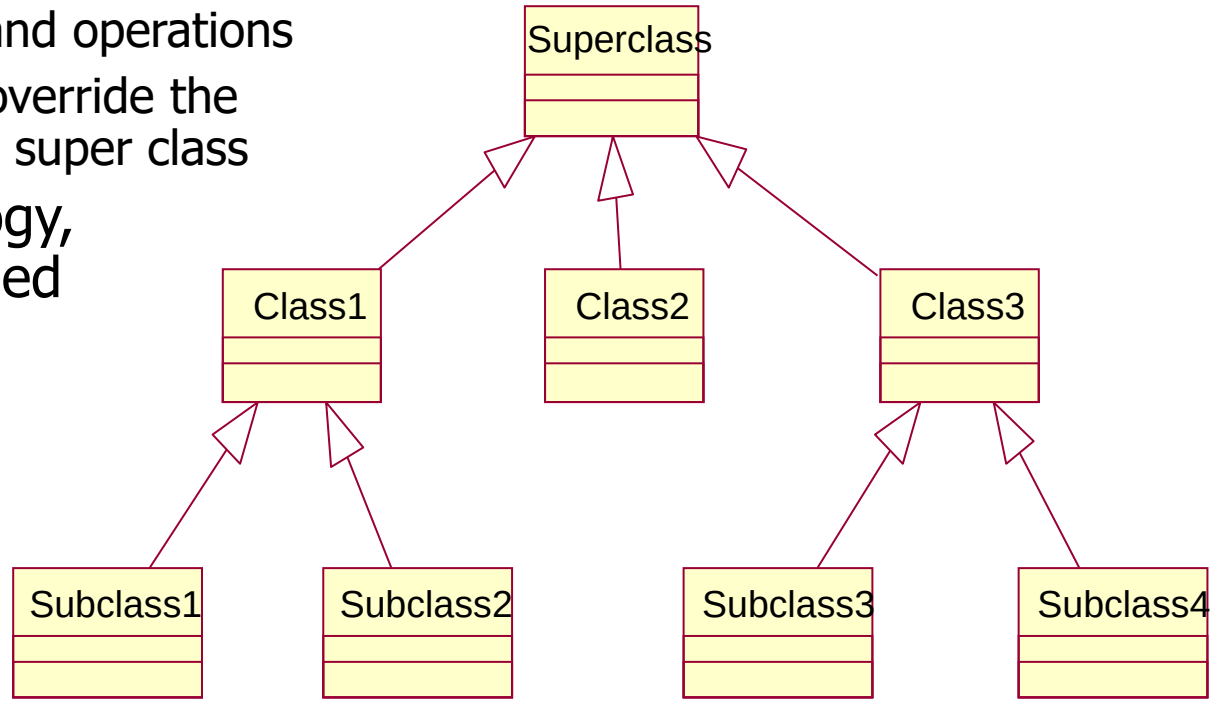
Encapsulation

- A well-designed module should have highest possible cohesion and lowest possible coupling
- Encapsulation
  - An object oriented term for information hiding
  - Binding of data and methods into an object
  - The internal structure of an object is hidden from other objects
  - The object has an interface that it reveals to other objects – everything else is hidden
  - Aims at low coupling and gives possibilities for high cohesion

Inheritance/generalization

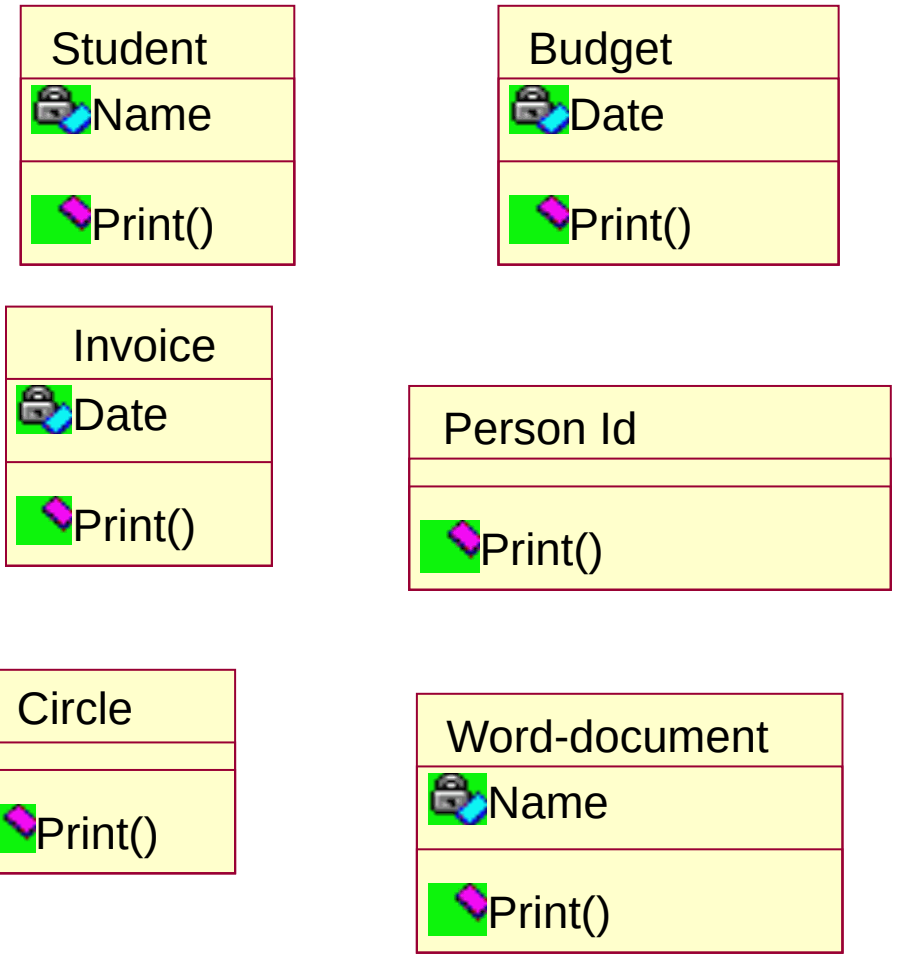
- A subclass inherits all properties of a super class: attributes and operations
- In addition, a subclass has its own attributes and operations
- A subclass can override the operations of its super class

- In UML terminology, inheritance is called generalization



Properties are polymorphic

- Polymorphism
  - The process of using an operator or function in different ways for different given inputs
  - A message does not need to know how the operation will be performed = low coupling
  - Objects of several classes can react to the same message
  - Objects of several classes may have the same attribute



Classes are abstractions

	Real world	Model
Abstract	Concepts	Classes Relationships
Concrete	Events Things Phenomena	Objects/instances Attribute values

- In object oriented design, models are made from the problem domain (often the real world)
  - Business models, data structures, conceptual models, etc

DESIGN WITH OBJECTS

So why did this Software Engineering –course turned into an intro to Object-Oriented programming?

Because objects are a tool which provides us the functionality to design without implementation!

There is no inherent shape, form or look for software component, class or object, so we need a language to express our designs.

Abstraction

- Concentration on essentials
  - Leave out inessentials
  - Purposeful simplification of things
- Forming of general concepts by simplifying things
- Modeling
- Main ways of abstraction
  - Classification – individualization
    - ”Fiat 500, reg.no GLL-86” is an instance of ”Car”
  - Generalization – specialization
    - A ”Car” can be specialized to a ”Sports car”
  - Aggregation (combination) – separation
    - A ”Car” is composed of four ”Wheels”, ”Body”, and ”Engine”

An object has an interface and it encapsulates its properties

- The interface encapsulates object properties
- Information hiding: the data is not accessed directly but through the defined operations in the interface
- Hiding of complexity from the class user
- Enables better change management and reuse
  - Under the interface, the implementation is free

Benefits of object-orientation

- Use of natural human ways of thinking
  - Real world vs. model
- Encapsulation helps change management
- Reuse becomes easier through inheritance and encapsulation
- Iterative development becomes possible
  - Design first the interfaces, then implement the functionality iteratively
- Use of same basic concepts through the development life-cycle
  - From requirements to implementation one speaks about classes and objects

OO technologies: OO programming languages

- Pure OO languages
  - All variables except basic data types (int, float, char, etc.) are objects
  - Typically there is one class hierarch with the root class Object
  - Java, C# (.NET framework) Smalltalk, Eiffel
  - Garbage collection: automatic memory management
  - Late binding
- Hybrid languages
  - Often extensions to more traditional languages
  - It is possible to program both in OO way and procedurally
  - No automatic memory management
  - C++, Python, Visual Basic, Object Pascal (Delphi), Modula, Ada

THIS WEEK'S LECTURE CASE & ANALYSIS

CONCEPTUAL MODEL OF ENVIROSENSE

How we'll proceed

1. I'll review today's tasks
2. Then we analyse in teams or groups
3. Lastly, we'll select one person (the bravest one!) shares the analysis outcomes of their group
4. Group is also welcome to post a mobile phone photo, screenshot or summary of their analysis for a tiny point reward (same point category with exercises and seminars)

Task 1/2

1. Take a quick look at conceptual models: https://www.interaction-design.org/literature/topics/conceptual-models
2. Have a quick look at the EnviroSense description, especially about software or hardware components.
3. Then, proceed with conceptual modeling 2 or 3 **key objects** from the case study description, using the steps presented on the following page.

Task 2/2

Then, proceed with conceptual modeling 2 or 3 **key objects** from the case study description, using the steps presented on the following page. The outcome should look like a mind map (bubbles or boxes with connections).

- **Identify Tasks:** Identify the tasks that your users need to accomplish.
- **Identify Objects:** Once you've identified the tasks, identify the physical objects or concepts that are relevant to those tasks.
  Assign Actions and Attributes to Objects: Once you've identified the objects, assign actions and attributes to them based on how your users will interact with them.

Sharing and review

What were the key technology objects? (2 to 3)

What should they do?
Who is using them?

Let’s discuss the findings of few randomly selected groups, before the lecturer weighs in his answer.

How to continue from here? (What's the use for course project?)

UML class diagrams can be used to present conceptual models in a more formal, standardized way.