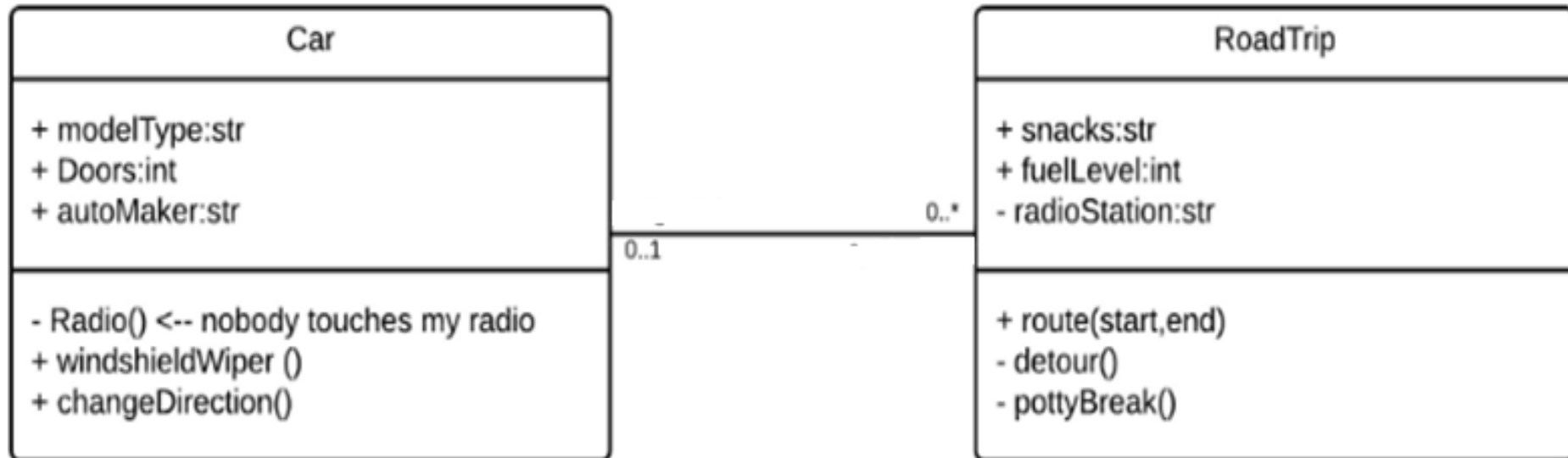


(11224) INTRODUCTION TO SOFTWARE ENGINEERING

Lecture 7: UML Class Diagram (part 2)

Shereen Fouad

Revision

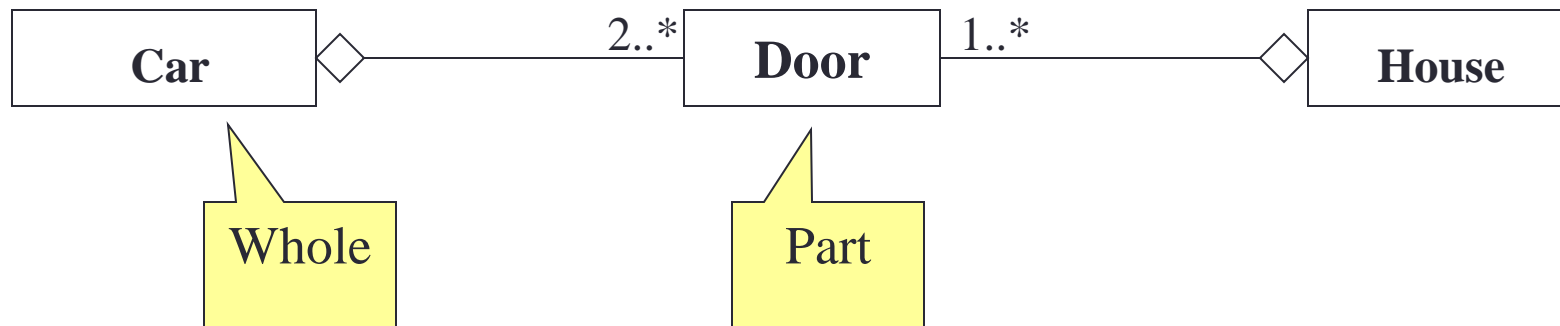


Relationships

- Instance level relationships
 - Association
 - Aggregation
 - Composition
- Class level relationships
 - Generalization
- General relationship
 - Dependency

Aggregation

- A special form of association that models a **whole-part relationship** between an aggregate (the whole) and its parts.
- Can be named and have the same adornments that an association can.
- It must be a binary association.
- Put a hollow diamond on the end of the line next to the “whole”

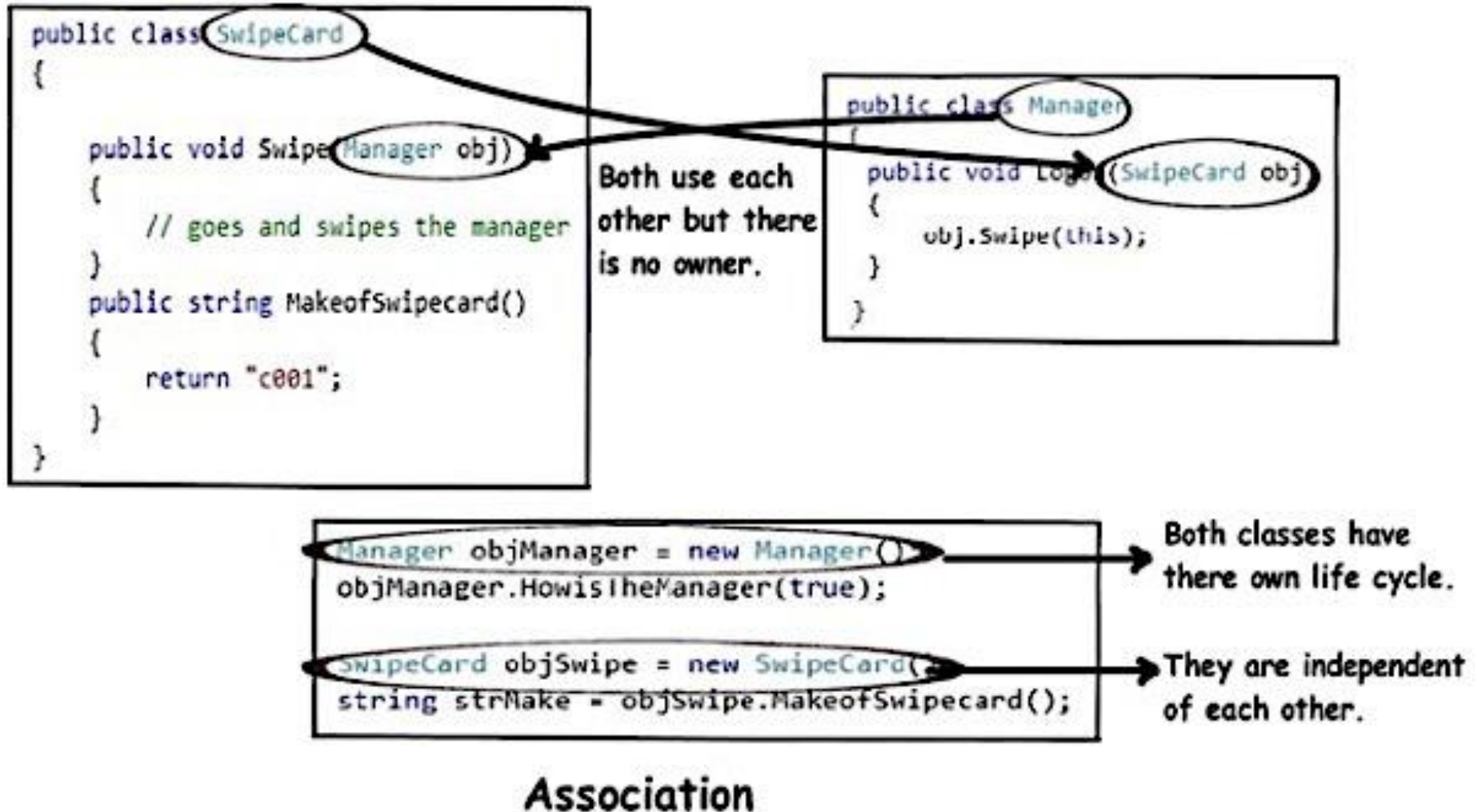


Aggregation (is part of)

- *Aggregation* can occur when a class is a collection or container of other classes
- The contained classes do not have a strong *lifecycle dependency* on the container.
- The contents of the container are not automatically destroyed when the container is.



Difference between association and aggregation



Difference between association and aggregation

```
public class Manager
{
    // Aggregation relation
    public List<Worker> Workers = new List<Worker>();
    .....
}
```

Worker is the child object and manager is the parent object.

```
public class Worker
{
    public string WorkerName = "";
}
```

Worker class cannot belong to other parent object.

```
Worker obj = new Worker();
```

But worker object can have his own life time.

Aggregation

Composition

- A strong form of aggregation
- One class contains a collection of instances of other classes
- The instance of the other class can not exist independently of the collection.
- Parts are copied (deleted, etc.) if the whole is copied (deleted, etc.)
- A part cannot be part of more than one whole
- Put a solid diamond on the end of the line next to the “whole”

Composition (is entirely made of)

- After the student has graduated, a transcript is generated and stored for the student on an archive system, but the student record is deleted from the active student record system.
- When the student record is deleted, all marks for that student should likewise be deleted.



- The master end always has multiplicity “1”

Quiz !!

- A **box office** is a place where tickets are sold to the public for admission to a certain movie theatre. A **Movie** is presented in more than one **theater**.
- Draw a class diagram for this information, and be sure to label all associations with appropriate multiplicities.

Relationships

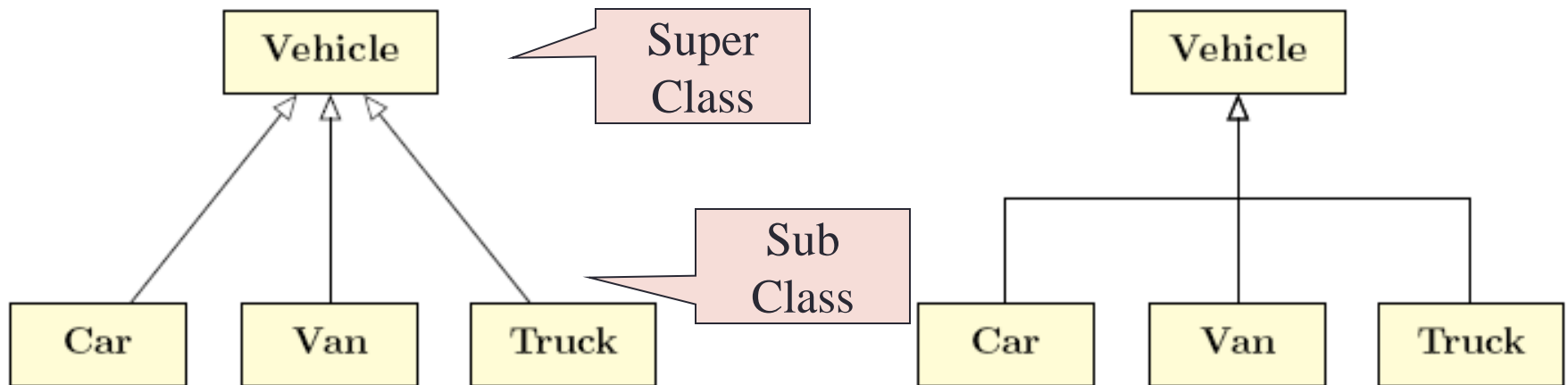
- Instance level relationships
 - Association
 - Aggregation
 - Composition
- Class level relationships
 - **Generalization**
- General relationship
 - Dependency

Generalisation (inheritance)

- Generalisations are relationships between classes, rather than, between objects.
- It indicates that one of the two related classes (the *subclass*) is considered to be a specialized form of the other (the *super type*).
- Any instance of the subtype is also an instance of the superclass.
- In Java, the sub-type class extends the super-type class, i.e. to use inheritance.
- A generalisation is drawn as arrows with open triangular arrow heads from the sub-type class to the super-type class.

Generalisation Example

It can be layed out in one of two ways (the right hand one is considered neater):



Generalisation hierarchies do NOT have names, nor do they have cardinality information on them as associations do.

Relationships

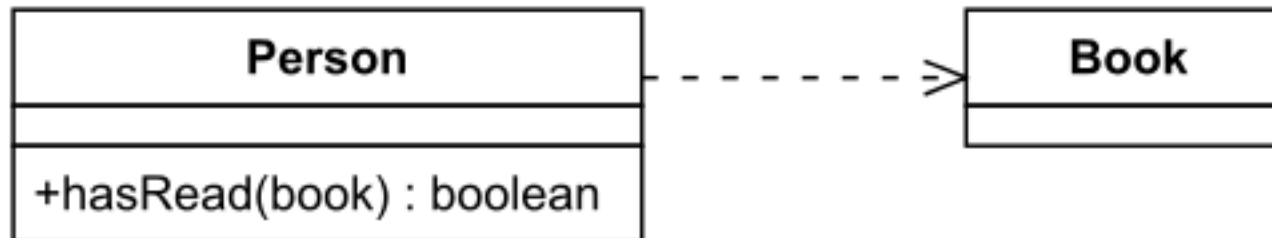
- Instance level relationships
 - Association
 - Aggregation
 - Composition
- Class level relationships
 - Generalization
- General relationship
 - **Dependency**

Dependency

- A dependency is a weaker (loosely coupled) relationship between two elements
- Indicates that one class depends on another because it uses it at some point of time.
- One class depends on another if the independent class is a parameter variable or local variable of a method of the dependent class.
- This is different from an association, where an attribute of the dependent class is an instance of the independent class.

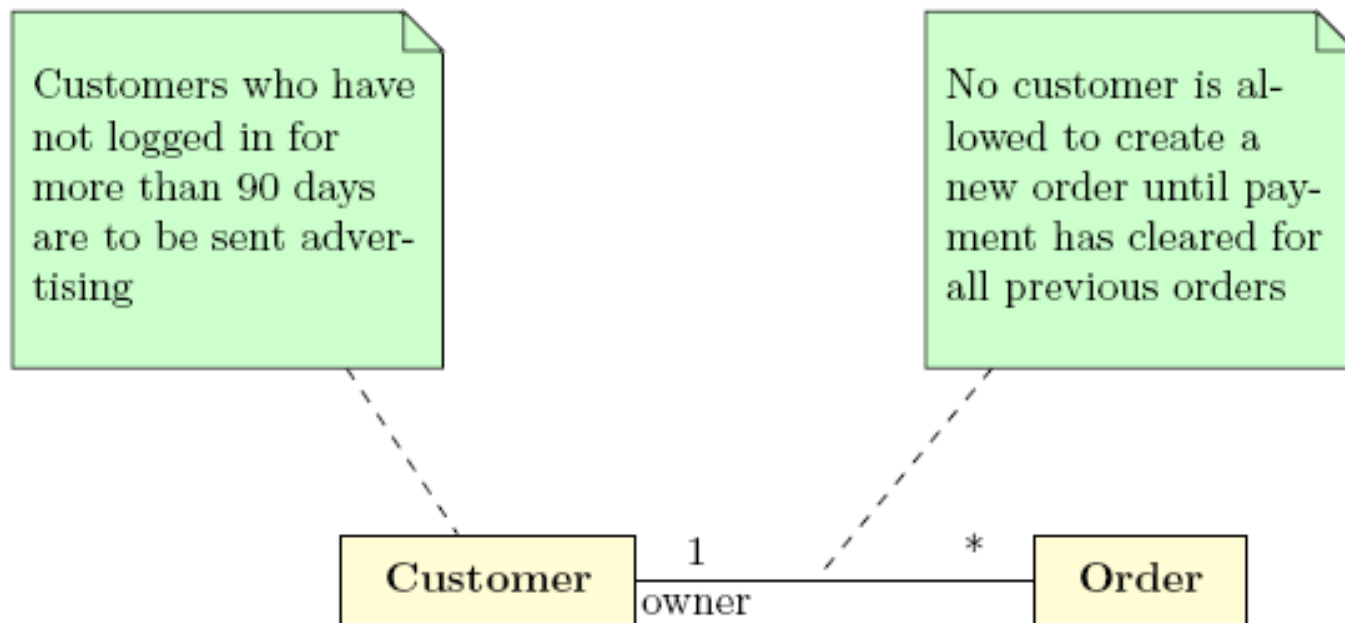
Dependency (uses temporarily)

- If the object is not stored in any instance variable, then this is modeled as a dependency relationship.
- For example, the Person class might have a hasRead method with a **Book passed as a parameter** that returns true if the person has read the book (perhaps by checking some database).
- Usually ignored, **Why?**



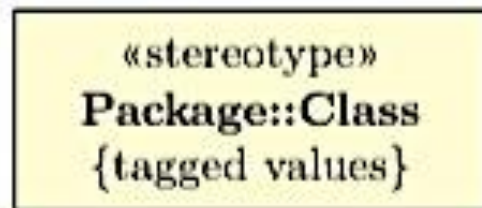
Comments

one can include notes, constraints or comments and attach them to classes or associations.

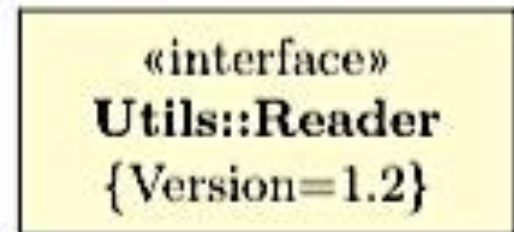


Stereotypes

- Extends the class elements of a UML diagram to new specialized variants Class element.
- Example:
- An Interface, can be made available in the model simply by adding the stereotype “«interface»” to a Class element.
- Syntax: Above name add <<stereotype>> inside (French quotes)

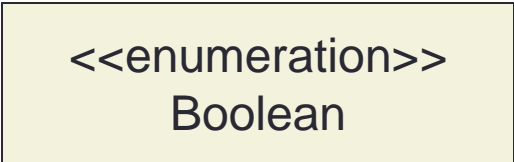


Example:



Stereotypes

- Other typical examples include Enumeration
- Used to provide extra info about the UML modeling construct

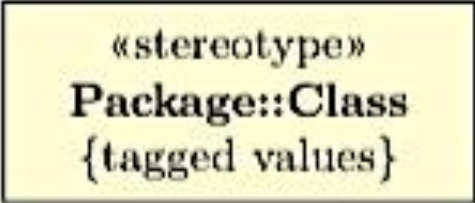


`<<enumeration>>`
Boolean

An *enumeration* is a user-defined data type that consists of a name and an ordered list of enumeration literals.


Tagged Values

- Every modeling element in UML has its set of properties
 - Classes have: name, attributes, operations, etc.
 - What if we want to add our own? (e.g. author?)
- Just add text in curly-brackets, with name=value, and put below the element name
- Note: These tell you something about the model, not about the final system to be built!
 - Often used for code generation, version control, etc.



«stereotype»
Package::Class
{tagged values}

Example:



«interface»
Utils::Reader
{Version=1.2}

Conceptual Modeling

- The conceptual model is an initial early stage design
- It concentrates on developing a good representation for the data of the program.
- It **pins down only the concepts** to be managed and manipulated in the program, **rather than the full details** of every part of the program.
- Not concerned about the details of the design, e.g. details of whether we are going to use an array, an ArrayList or some other collection class to implement a particular part of the program.

UML Class Diagram Exercise

Draw a UML Class Diagram representing the following elements from the problem domain for a hockey league. A hockey league is made up of at least four hockey teams. Each hockey team is composed of six to twelve players, and one player captains the team. A team has a name and a record. Players have a number and a position. Hockey teams play games against each other. Each game has a score and a location. Teams are sometimes lead by a coach. A coach has a level of accreditation and a number of years of experience, and can coach multiple teams. Coaches and players are people, and people have names and addresses.

Draw a class diagram for this information, and be sure to label all associations with appropriate multiplicities.

UML Class Diagram Exercise

Draw a UML Class Diagram representing the following elements from the problem domain for a hockey **league**. A hockey league is made up of at least four hockey **teams**. Each hockey team is composed of six to twelve **players**, and one player **captains** the team. A team has a name and a record. Players have a number and a position. Hockey teams play **games** against each other. Each game has a score and a location. Teams are sometimes lead by a **coach**. A coach has a level of accreditation and a number of years of experience, and can coach multiple teams. Coaches and players are people, and people have names and addresses.

Draw a class diagram for this information, and be sure to label all associations with appropriate multiplicities.

Tools for creating UML diagrams

- Visio is a Microsoft tool for drawing UML diagrams
- Dia is a freeware clone of Visio
- Violet (free)
 - <http://horstmann.com/violet/>
- Rational Rose
 - <http://www.rational.com/>
- Visual Paradigm UML Suite (trial)
 - <http://www.visual-paradigm.com/>
 - (nearly) direct download link:
<http://www.visual-paradigm.com/vp/download.jsp?product=vpuml&edition=ce>

(there are many others, but most are commercial)

References

- A number of slides in this talk is based on:
 - Alan P. Sexton hand-outs (Introduction to Software Engineering. The University of Birmingham. Spring Semester 2014
 - http://en.wikipedia.org/wiki/Class_diagram