UML Basics

Contents

Overview

A UML(Unified Modeling Language) is a type of status structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations(or methods), and relationships between objects. Using a UML diagram is a good way to visualising the classes in your system before you start to actually code them.

Why do we need UML diagrams

- Planning and modeling ahead of time
- Having a plan makes programming much easier.
- When someone wants to build a house, they don't just grab a hammer and get to work. They need to have a blueprint — a design plan — so they can analyse & modify their system.
- Some technical/language-specific knowledge is needed in order to understand it.

How to

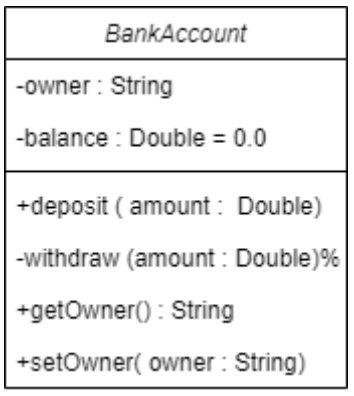
A class is represented by 3 compartments as shown below

- The first compartment represents the class name
- The middle compartment contains the class attributes
- The last compartment contains the class methods

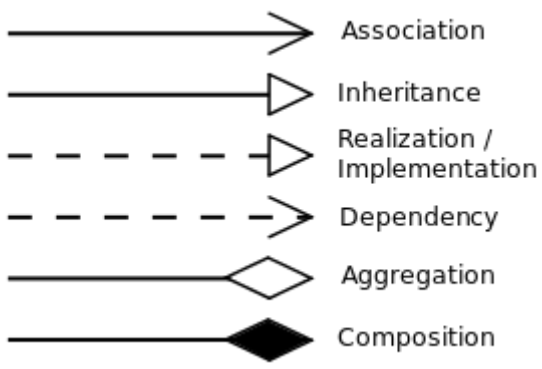
Visibility of class members

Access modifier	Symbol	Description
Public	+	Anywhere in the program and may be called by any object within the system

Access modifier	Symbol	Description
Protected	#	The class that defines it or a subclass of that class. Can also be accessed from within the same package
Package	~	Instances of other classes within the same package
Private	-	The class that defines it



Relationships



Aggregation

A special form of association which is a unidirectional (one way) relationship between classes. One way to look at this is say it **has a** relationship. For instance we have two classes `Wallet` & `Money`, a `Wallet` **has** money but `Money` doesn't need to have a `Wallet` and so it can be considered a one directional relationship.

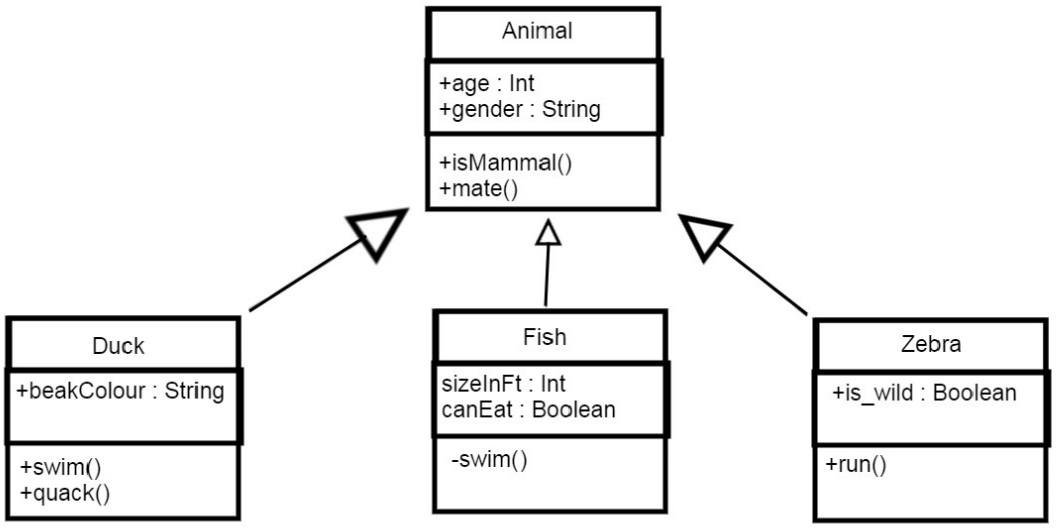
Composition

A restricted form of Aggregation in which two entities(classes) are highly dependent on each other. One way to look at this is say it **is a part of**
For example if we had two classes `Human` & `Heart`, a `human` needs a `heart` and a `heart` needs an `human` in order to function. Both `Human` and `Heart` depend on each other neither can survive without the other.

Inheritance

Indicates that a subclass(child class) is considered to be a specialised form of the super class (parent class).

Below you can see the `Animal` parent class with public methods. there are also arrows coming form `Duck`, `Fish` and `Zebra` child classes.
This indicates that the three subclasses inherit all the members from the `Animal` class, but also implement their own unique member fields.
You can see the `Duck` class has a `swim()` and `quack()` method.



Realisation/Implementation

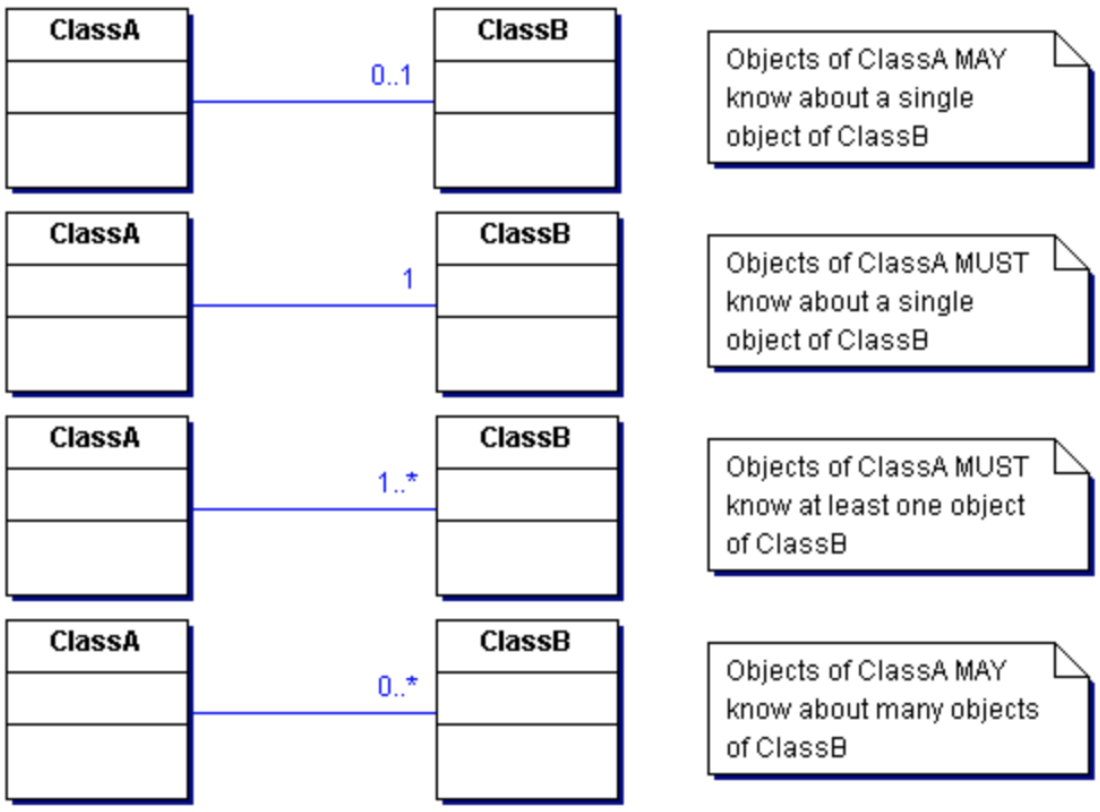
A relationship between two model elements, in which one model element implements/executes the behaviour that the other model element specifies.

Association

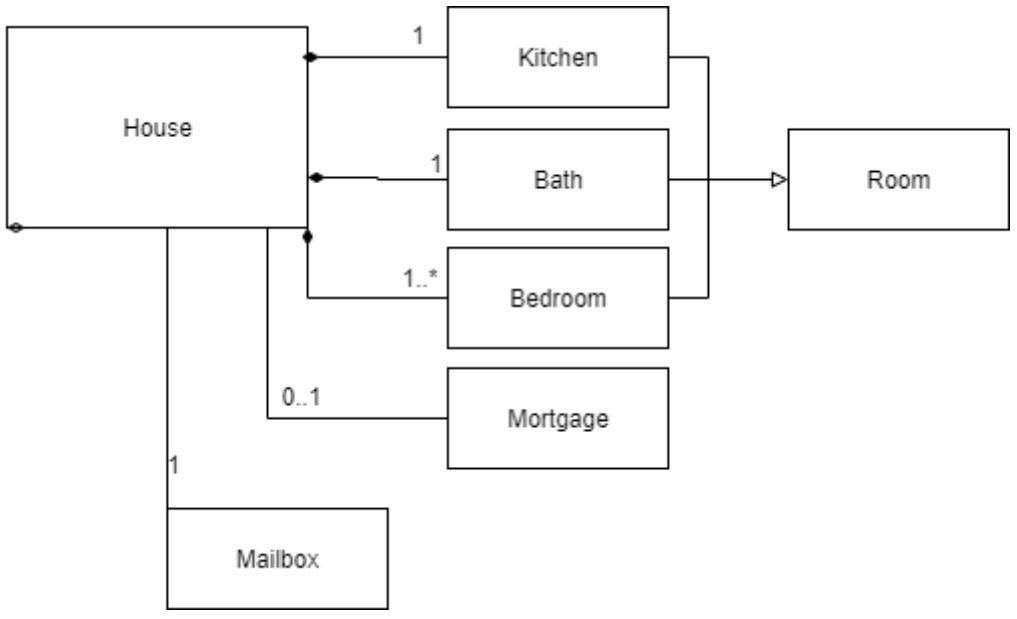
An association states that there is some kind of link between two classes.

Multiplicity

After specifying the type of association relationship by connecting the classes, you can also declare the cardinality between the related entities.



Below is an examples of a house and, how we can use multiplicity in practice. The house has exactly one kitchen, one bathroom and one mailbox. The house also has at least one bedroom but could have many and at most one or zero mortgages.

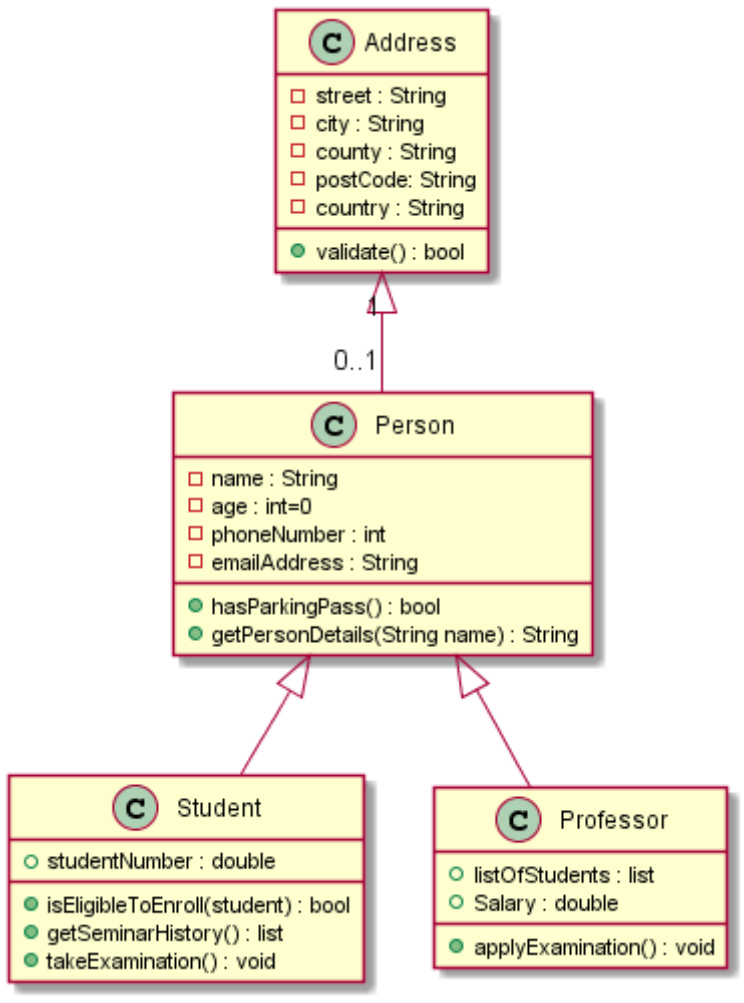


Tutorial

plantUML example

This UML diagram was created using PlantUML. The code and link to the documentation also also be found below.

Student Example



▶ Task Details

How to Run plantUML

PlantUML Documentation

Exercises

1. Trainer UML model exercise
   ▶ Task Details