UML

Unified Modeling Language

Intro

Problem

Как программные системы ... ?

- визуализировать
- специфицировать
- конструировать
- документировать

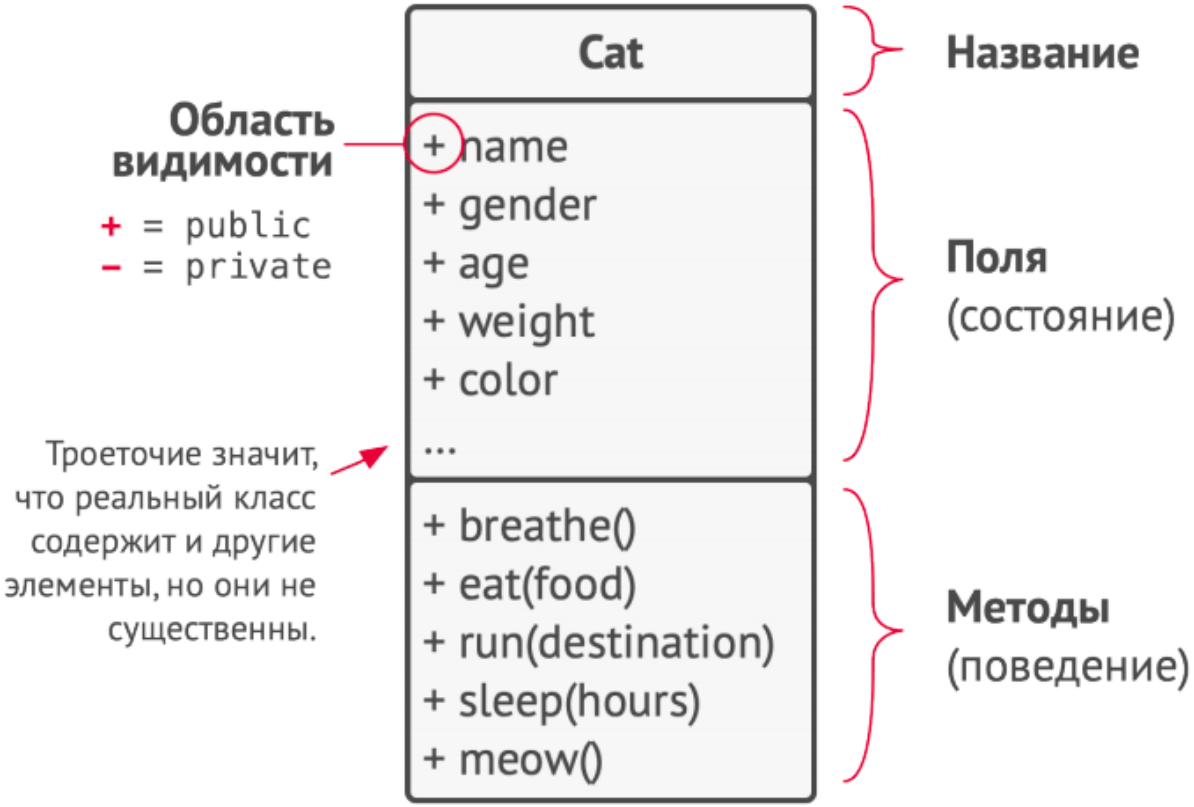
Solution

UML: class-diagram

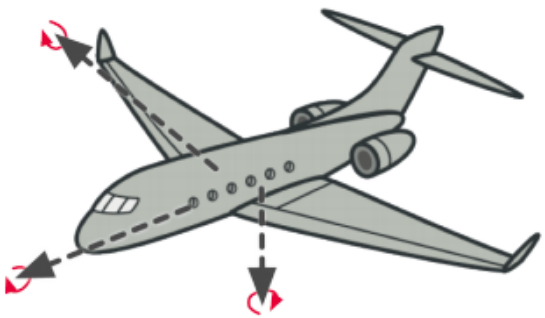
One of many

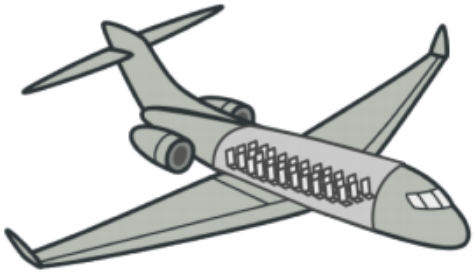
Common

Class


Cat
Название
Область видимости
+ = public
– = private
+ name
+ gender
+ age
+ weight
+ color
...
Поля
(состояние)
Троеточие значит, что реальный класс содержит и другие элементы, но они не существенны.
+ breathe()
+ eat(food)
+ run(destination)
+ sleep(hours)
+ meow()
Методы
(поведение)

OOP: abstraction

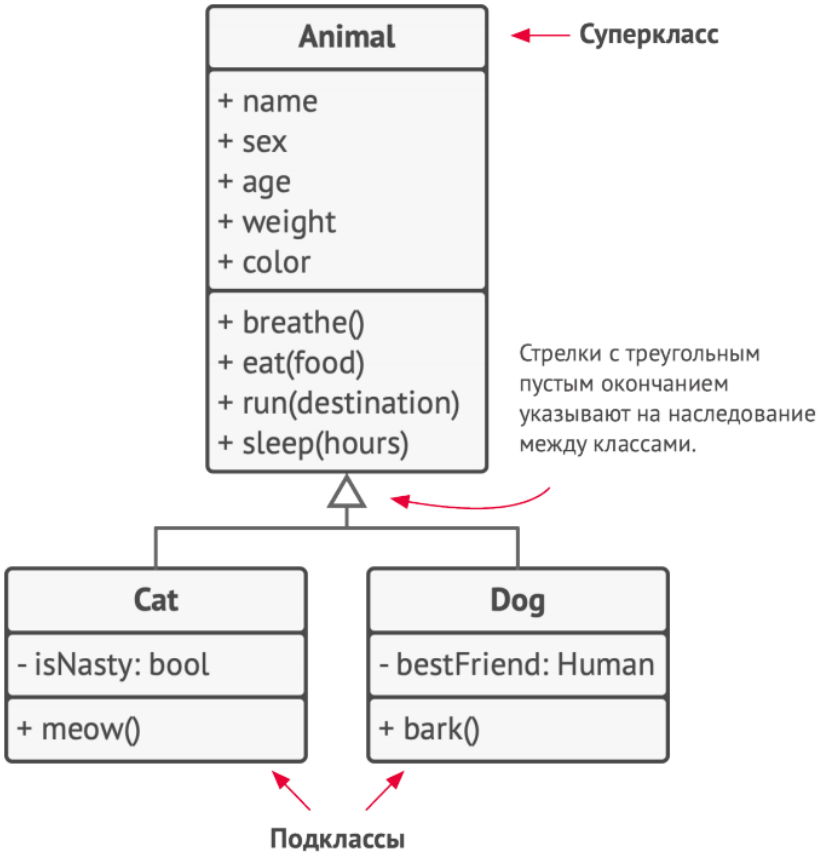




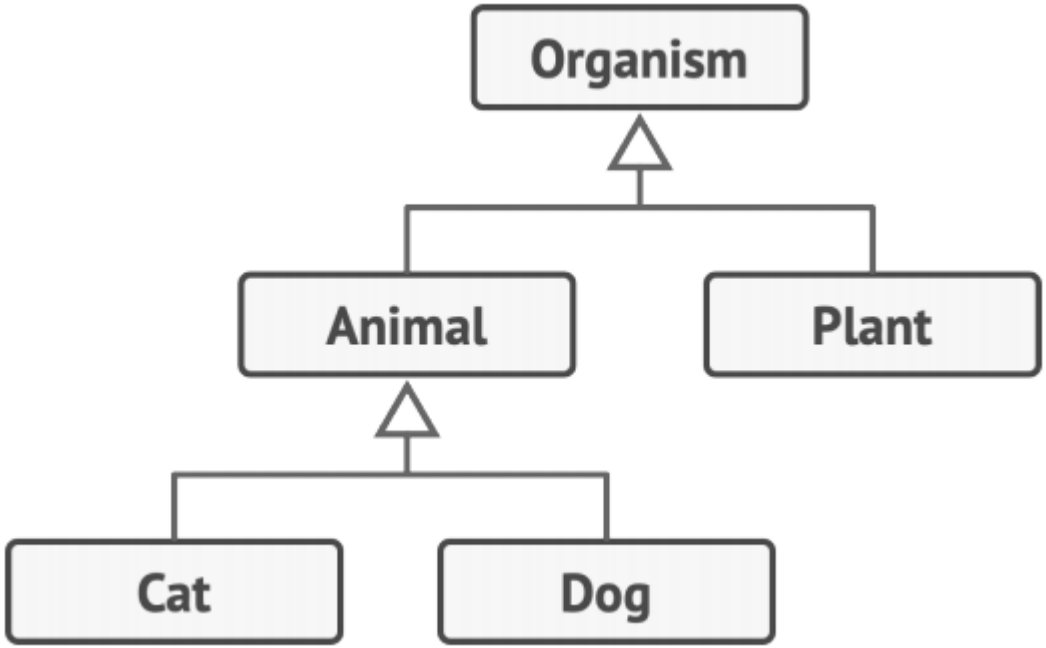
Airplane
- speed - altitude - rollAngle - pitchAngle - yawAngle
+ fly()

Airplane
- seats
+ reserveSeat(n)

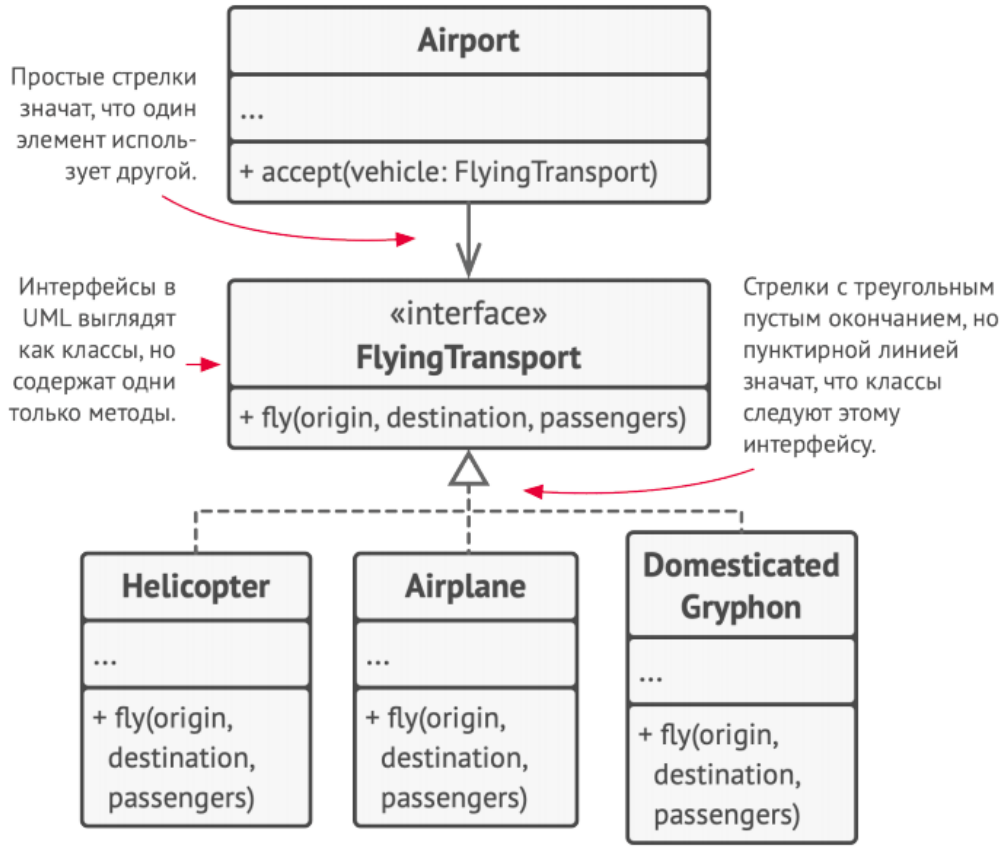
OOP: inheritance



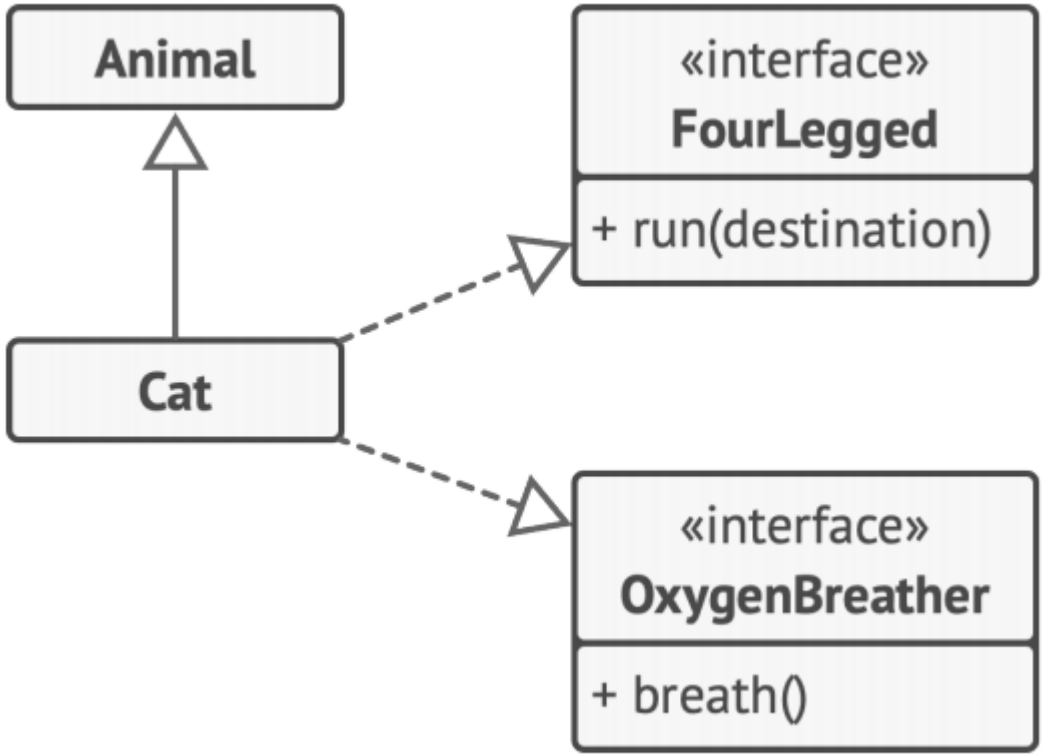
OOP: inheritance



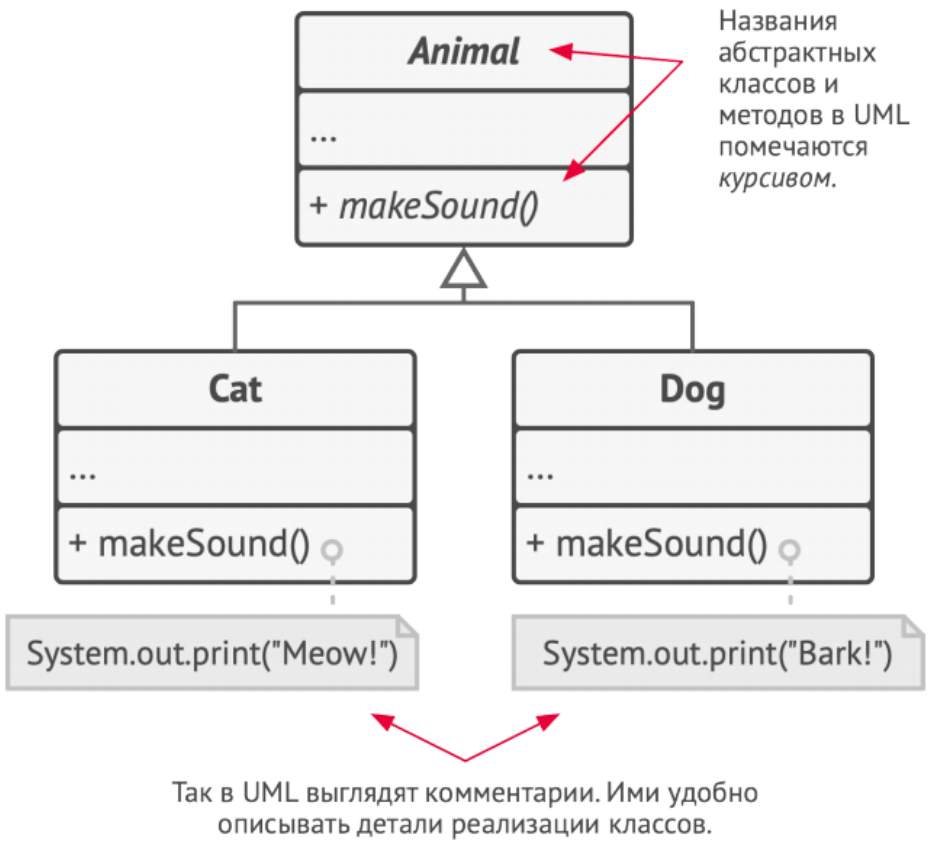
Interface


Airport
...
+ accept(vehicle: FlyingTransport)
Простые стрелки значат, что один элемент использует другой.
Интерфейсы в UML выглядят как классы, но содержат одни только методы.
«interface»
FlyingTransport
+ fly(origin, destination, passengers)
Стрелки с треугольным пустым окончанием, но пунктирной линией значат, что классы следуют этому интерфейсу.
Helicopter
...
+ fly(origin, destination, passengers)
Airplane
...
+ fly(origin, destination, passengers)
Domesticated Gryphon
...
+ fly(origin, destination, passengers)

Inheritance vs. Interface



Abstract Class



Relations

Relation: dependency

Зависимость:

- Класс A могут затронуть изменения в классе B. Другими словами: изменяя класс B (названия полей, методов, их параметры и возвращаемый тип) возможно потребуется делать изменения и в классе A

Relation: association

Ассоциация:

- Класс A зависит от B.
- Объект A знает об объекте B.

Relation: aggregation

Агрегация:

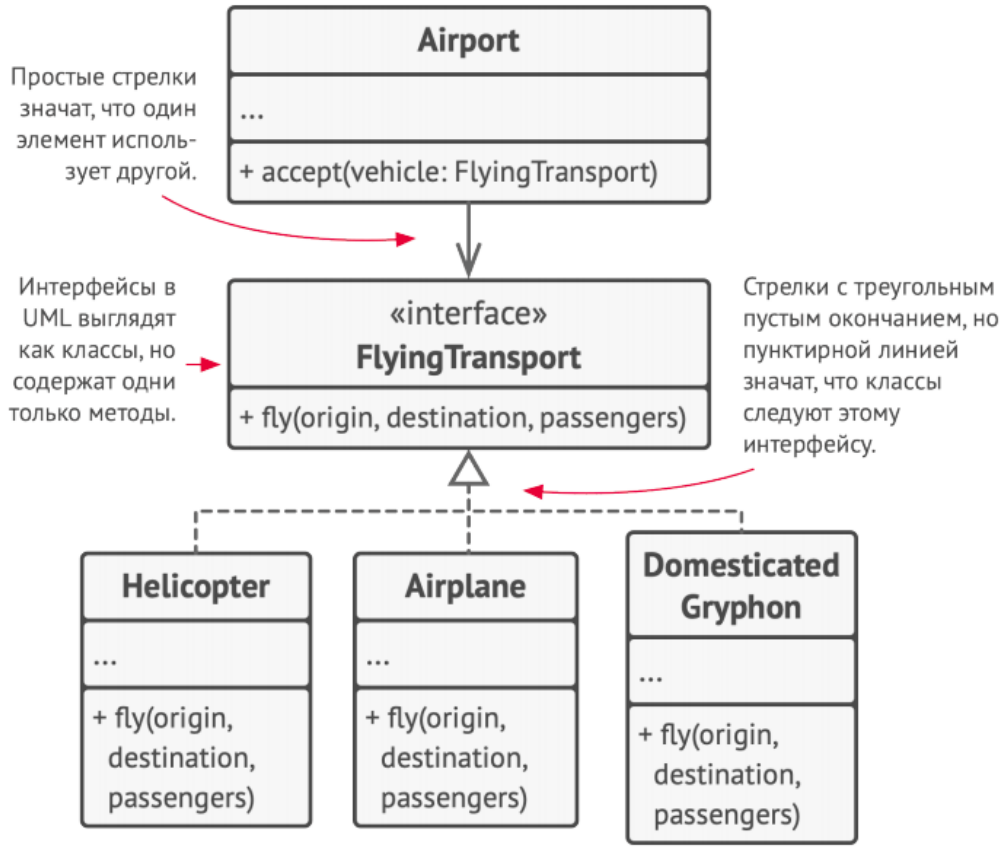
- Класс A зависит от B.
- Объект A знает об объекте B.
- Объект A состоит из объекта B.

Relation: composition

Композиция:

- Класс A зависит от B.
- Объект A знает об объекте B.
- Объект A состоит из объекта B.
- Объект A управляет жизненным циклом объекта B.

Relation: Interface

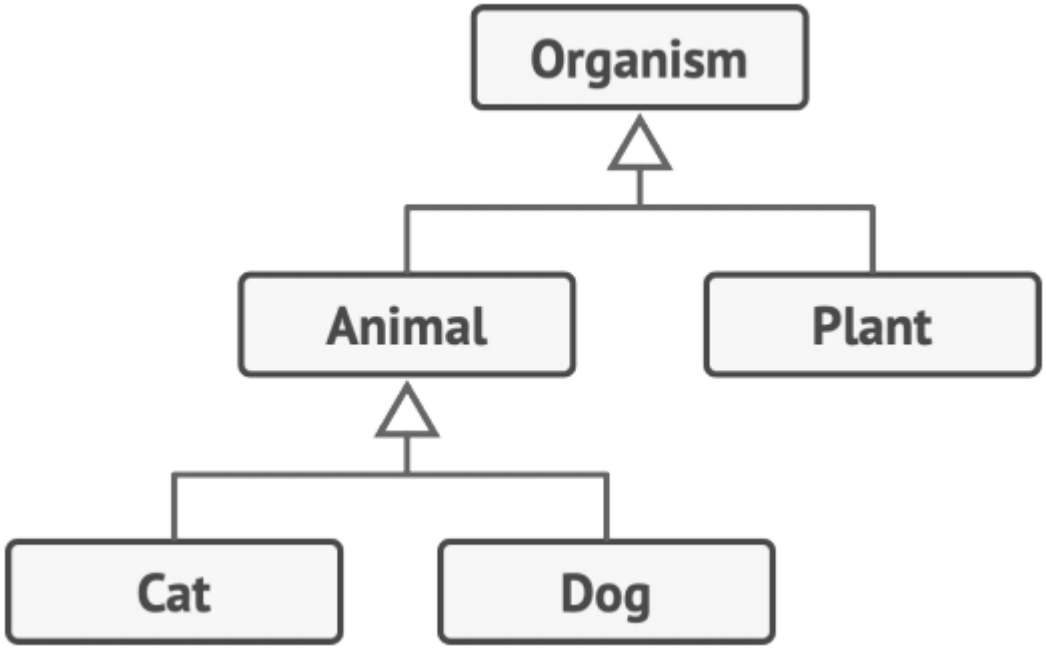


Relation: Interface

Реализация:

- Класс `A` зависит от `B`.
- Класс `A` определяет методы объявленные интерфейсом `B`.
- Объекты `A` можно рассматривать через интерфейс `B`.

Relation: inheritance

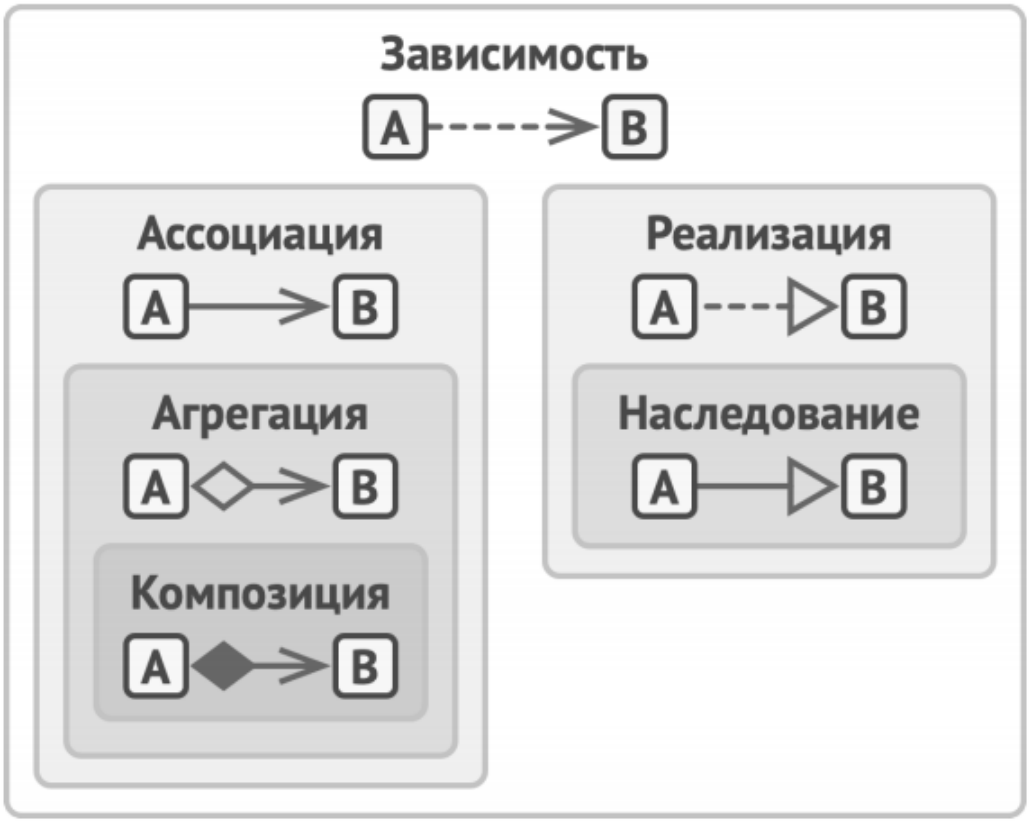


Relation: inheritance

Наследование:

- Класс `A` зависит от `B`.
- Класс `A` наследует интерфейс и реализацию класса `B`, но может переопределить её.
- Объекты `A` можно рассматривать через интерфейс класса `B`.

Relations


Зависимость
A
B
Ассоциация
A
B
Агрегация
A
B
Композиция
A
B
Реализация
A
B
Наследование
A
B