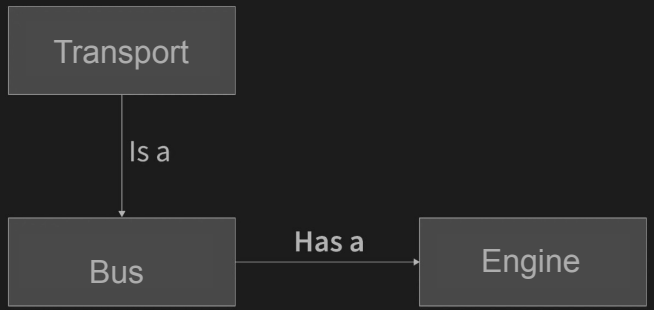
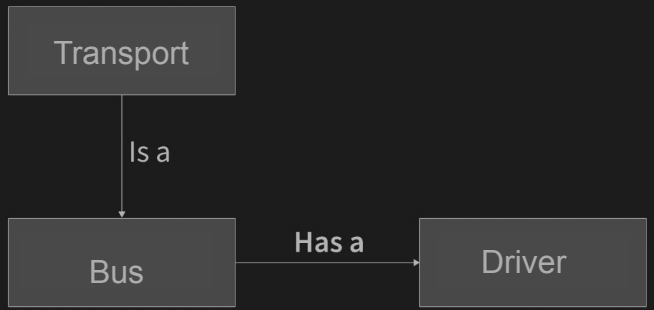
Отношения между классами

Виды отношений

Композиция



Агрегация



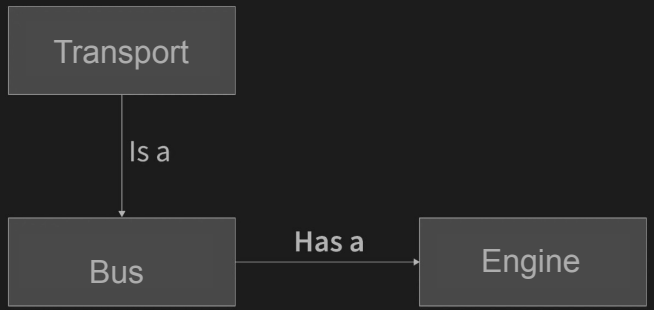
С вероятностью 99,9% между классами в наших приложениях будут существовать некая связь

В ООП выделяют два вида связей между классами:

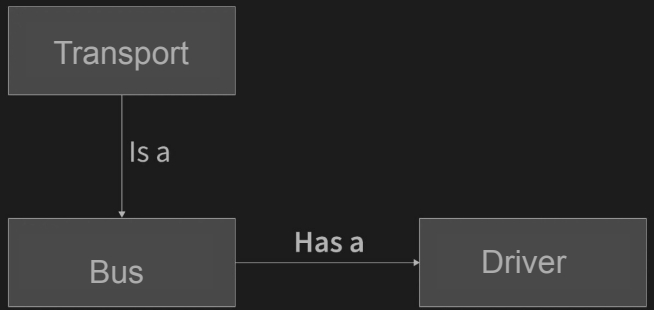
- IS-A (является)
- HAS-A (имеет)

Примеры

Композиция



Агрегация



Например:

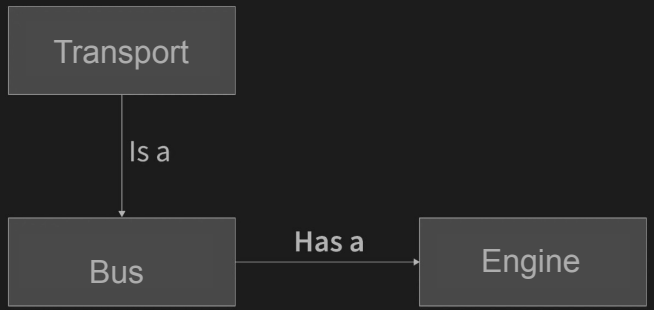
- Autobus IS-A Transport
- Driver IS-A Person
- Passenger IS-A Person

- Autobus HAS-A RegistrationDoc
- Autobus HAS-A Driver
- Autobus HAS-A Passengers

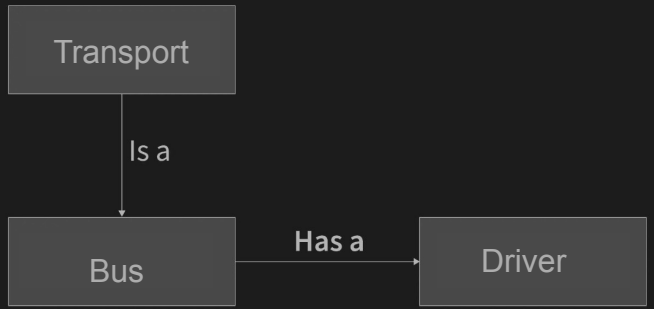
- Mother IS-A Woman
- Mother HAS-A Child

Класификация отношений

Композиция



Агрегация

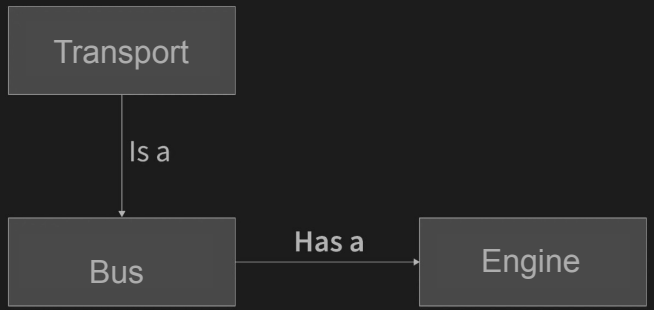


В объектно-ориентированных языках программирования существует три способа организации взаимодействия между классами:

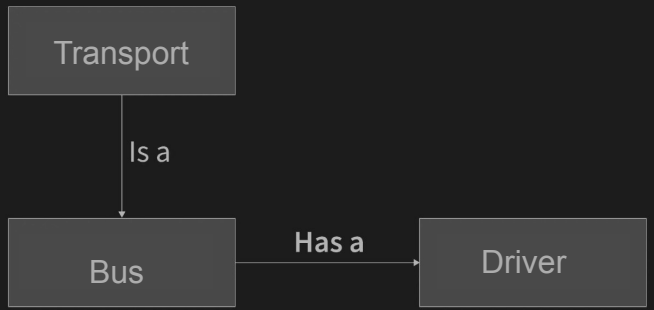
- Наследование (IS-A)
- Ассоциация (HAS-A)
  - Агрегация
  - Композиция

Ассоциация

Композиция



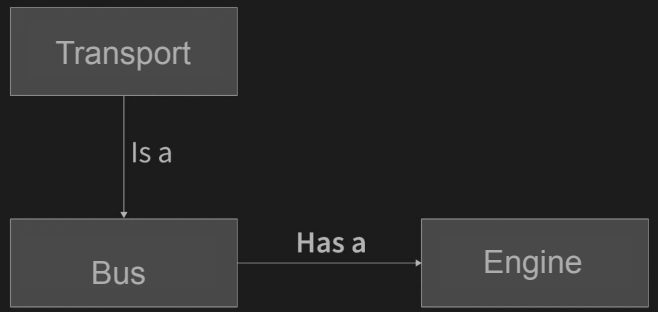
Агрегация



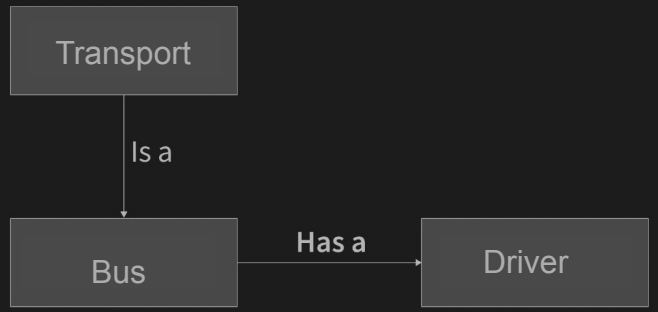
Агрегация и **композиция** частными случаями **ассоциации**. Это более конкретизированные отношения между объектами.

Агрегация

Композиция



Агрегация



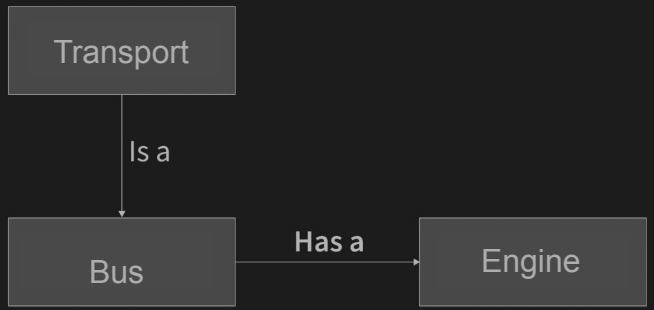
Агрегация — отношение когда один объект является частью другого.

Например, Студент входит в Группу любителей физики, у Автобуса есть Водитель.

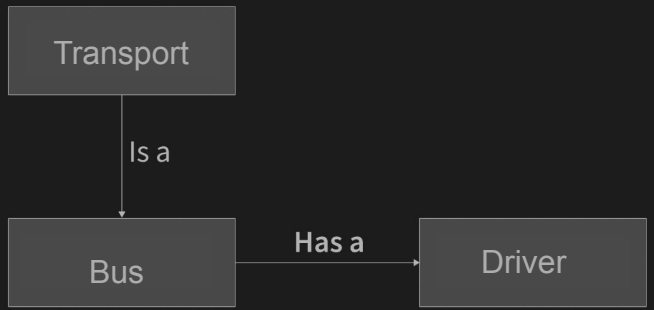
При агрегации объекты рассматриваемых классов могут существовать независимо друг от друга.

Агрегация

Композиция



Агрегация



Композиция — более “жесткое” отношение, когда объект не только является частью другого объекта, но и вообще не может принадлежать еще кому-то.

Например когда двигатель не существует отдельно от автомобиля. Двигатель создается при создании автомобиля и полностью управляется автомобилем.

В типичном примере, экземпляр двигателя будет создаваться в конструкторе автомобиля. В отличие от студента, который может входить и в другие группы тоже.

Наследование

```
class Transport {
   private String model;
   private int year;

   public Transport(String model, int year) {
       this.model = model;
       this.year = year;
   }
}

class Car extends Transport {
   private final int passengersCount;
   private int mileage;

   public Car(String model, int year, int pC)
{
       super(model, year);
       this.passengersCount = pC;
       this.mileage = 10000;
   }
}
```

Наследование — это когда класс-наследник имеет все поля и методы родительского класса, и, как правило, добавляет какой-то новый функционал и/или поля.

Наследование – это один из основных принципов объектно-ориентированного программирования, который позволяет создавать новый класс на основе существующего класса.

Наследование основывается на связи IS-A (является).

Наследование

```
class Transport {
    private String model;
    private int year;

    public Transport(String model, int year) {
        this.model = model;
        this.year = year;
    }
}

class Car extends Transport {
    private final int passengersCount;
    private int mileage;

    public Car(String model, int year, int pC)
{
        super(model, year);
        this.passengersCount = pC;
        this.mileage = 10000;
    }
}
```

Наследование в коде программы обозначается ключевым словом extends.

Также следует знать, что класс, от которого наследуются, называется родителем (родительским классом, класс-родитель).

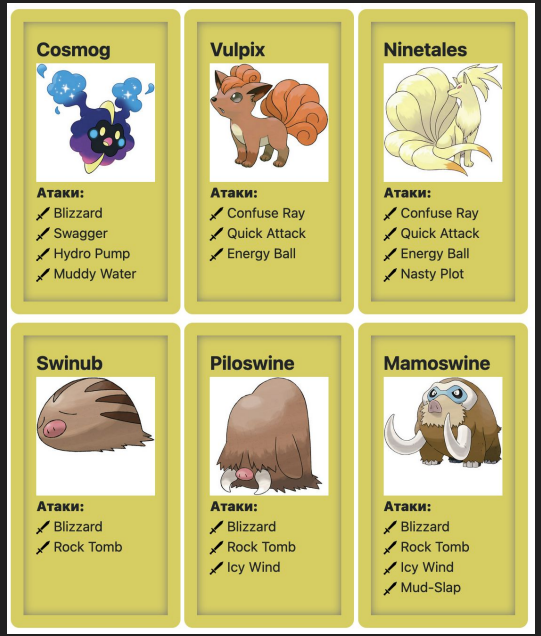
Класс, который наследует, соответственно класс-потомок.

В Java каждый класс может наследоваться только от одного класса (множественного наследования нет)

Домашнее задание

Вариант 1: взять 17 домашку и для каждого класса из этой домашки написать родительский класс в соответствии с названием подпакета (animal, people, transport).

Вариант 2:
Реализуейте дочерние классы покемонов по этому набору:



The end