Object-orient programming

Software Requirements and Design
T-216-GHOH

Skúli Arnlaugsson | 6. september 2019

Object-oriented programming

- Object-orient programming, OOP, (í. hlutbundin forritun) is a programming paradigm based on the idea of objects that have attributes (í. eigindi) and methods (í. aðferðir).

Object-oriented programming

- Object-orient programming (í. hlutbundin forritun) is a programming paradigm based on the idea of objects, that have attributes (í. eigindi) and methods (í. aðferðir).
  - Objects have **attributes** that define the instance of the object.

Object-oriented programming

- Object-orient programming (í. hlutbundin forritun) is a programming paradigm based on the idea of objects, that have attributes (í. eigindi) and methods (í. aðferðir).
  - Objects have attributes that define the instance of the object.
  - Objects have **methods** that define how they interact.

Object-oriented programming

- Object-orient programming (í. hlutbundin forritun) is a programming paradigm based on the idea of objects, that have attributes (í. eigindi) and methods (í. aðferðir).
  - Objects have attributes that define the instance of the object.
  - Objects have methods that define how they interact.
  - Every object is an **instance** of a class.

What is an class?

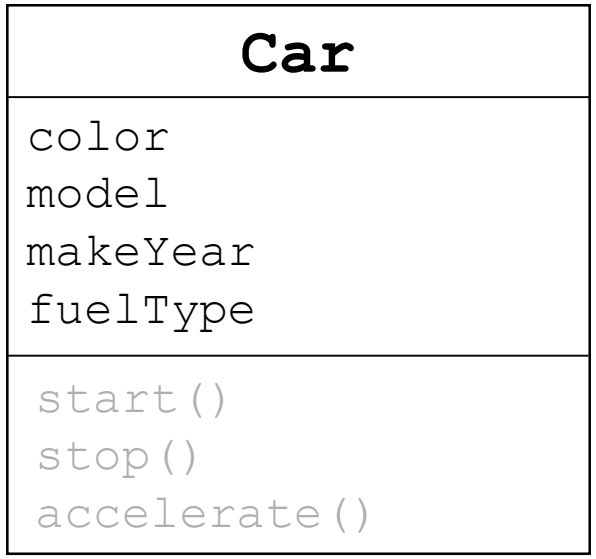
- A class (í. klasi) is a groupping of similar traits of objects.
  - Traits: attributes, methods
- Objects are instances (í. tilvik) of classes.

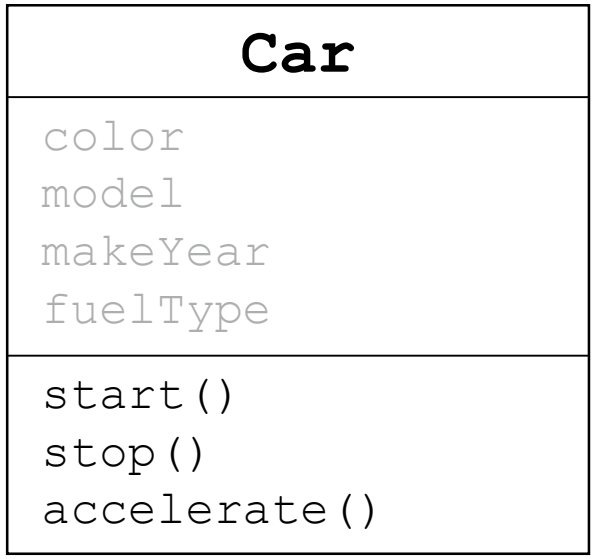
- A house **class** describes what houses are and do.
- An **instance** of a house class defines what that particular house looks like.

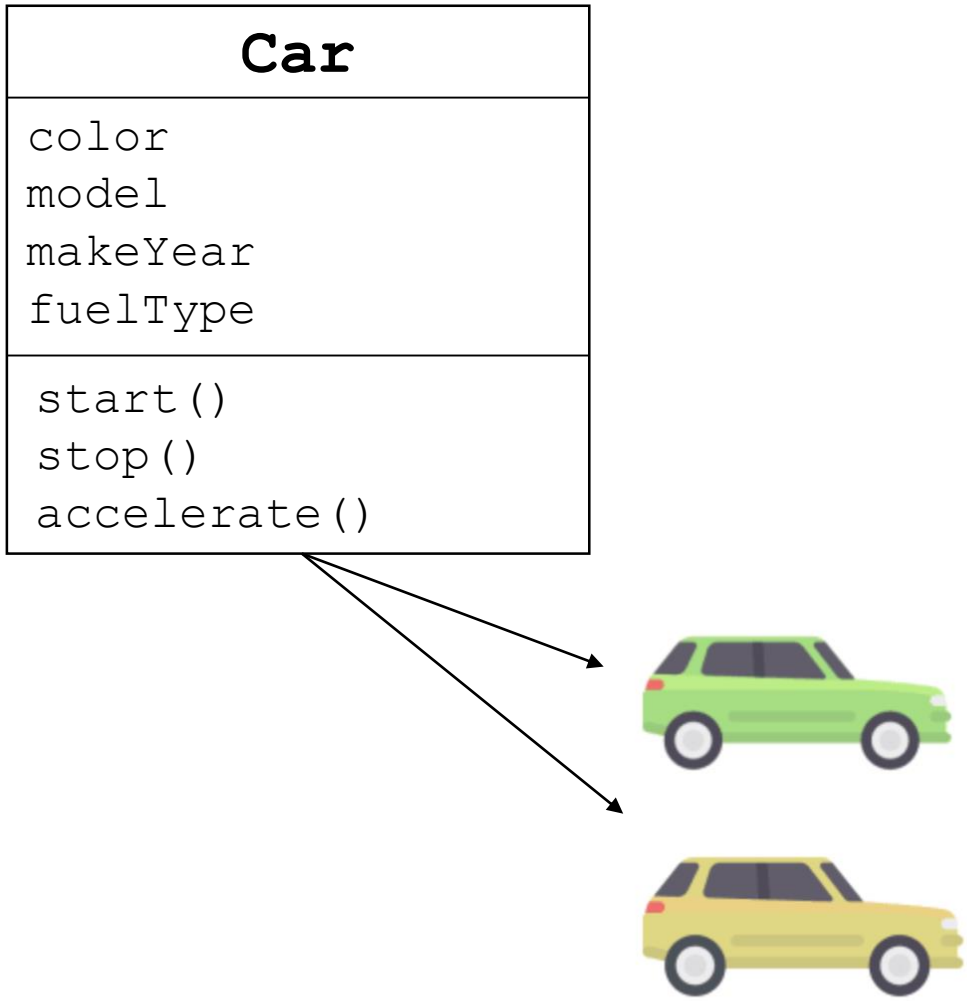
Attributes vs. methods

Attributes are values that describe the object.

Methods are functions that describe what the object can do (and how it does it).







Classes vs. instances

- Each object is an instance of a class.
- A green car is an instance of the car class, and so is an yellow car.
- The green car might be of the same model as the yellow, or a different one. But the color is different at least.
- The methods are the same. How the object can interact.

The four pillars of OOP - encapsulation

Encapsulation (í. hjúpun)

- Encapsulation bundles data and methods that work on data within one unit.
- Part or all of the class attributes and methods *can be* made non-accessible to other parts of the program.
- *Public* method and attributes can be used and accessed.
- *Private* methods and attributes cannot be accessed and used except only the Instance itself.
- *Protected* methods and attributes have limitations regarding who and what can access them, but wider than *private*.

The four pillars of OOP - abstraction

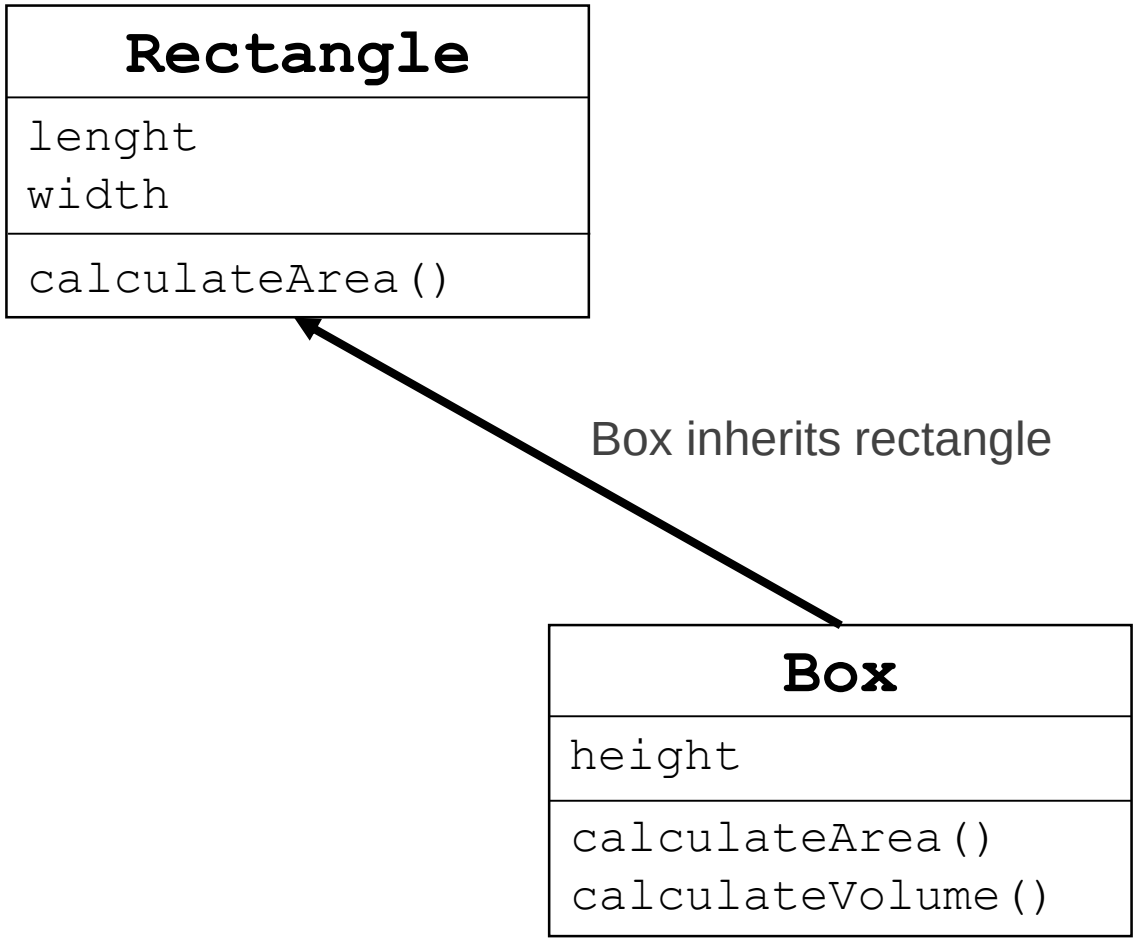
Abstraction (í. útdráttur)

- Abstraction helps to hide unnecessary details from the user.
- An extension of encapsulation.
- You don’t know how the car engine works, just that you know how to use it in your car. What happens beneath the hood is hidden from the user.

The four pillars of OOP - inheritance

Inheritance (í. erfðir)

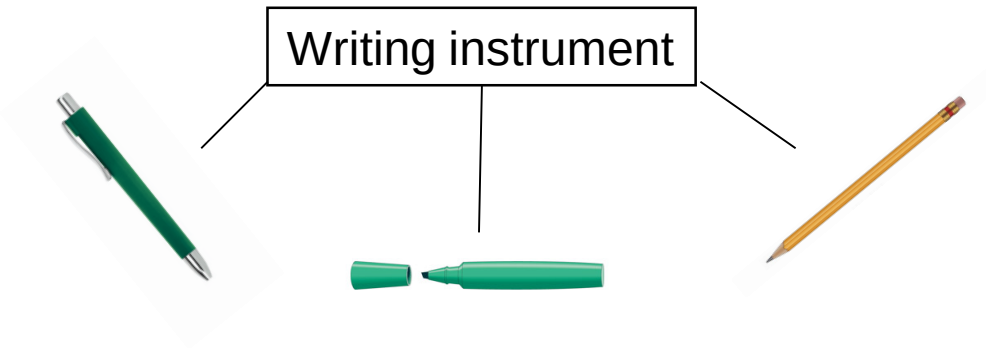
- Finding the common traits in related object types.
- Classes (object types) are often very similar, share common attributes and methods. But some might differ slightly.
- Take a square and a rectangle. All squares are rectangles, but not all rectangles are squares.



The four pillars of OOP - polymorphism

Polymorphism (í. fjölvirkni)

- Common behavior is defined in a base class, and overridden in derived classes.
- The box in our example from before now has to implement another way of calculating the surface area.
- So our box class overrides the calculateArea method.
- Derived classes can also add new attributes and methods as needed.
- The Student class, that is derived from the person class, could have the enrollToClass method for example.
- Another example:



OOP and UML

- UML has several types of diagrams that are used to describe classes and operations involving classes:
  - Class diagrams – one of the most common diagrams used.
  - Object diagrams – shows a snapshot of the detailed state of a system in a point of time. Similar to a class diagram, but with values instead of abstractions.
  - Sequence diagrams – shows interactions between objects.


HÁSKÓLINN Í REYKJAVÍK
REYKJAVÍK UNIVERSITY