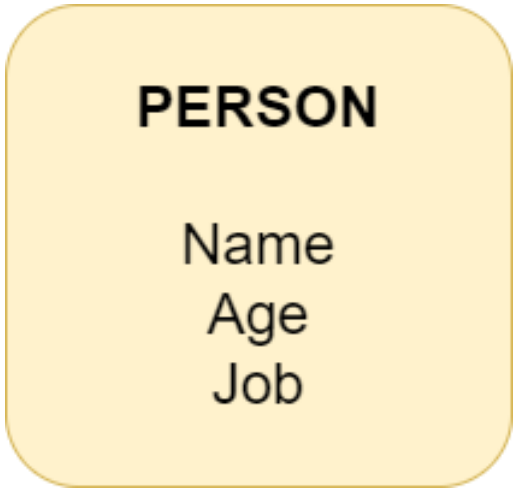
Programming for Social Scientists

Class inheritance

Johan A. Dornschneider-Elkink

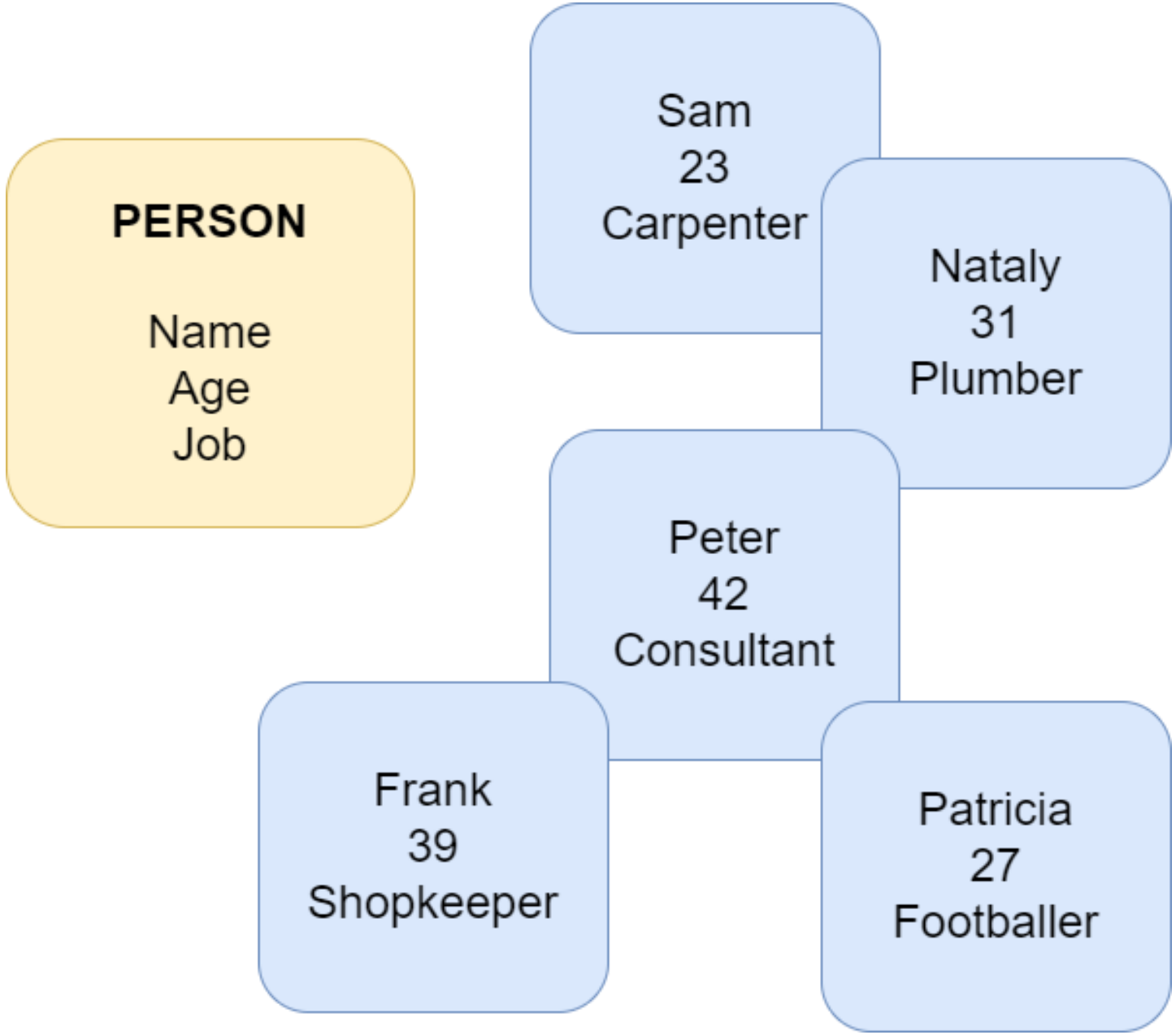


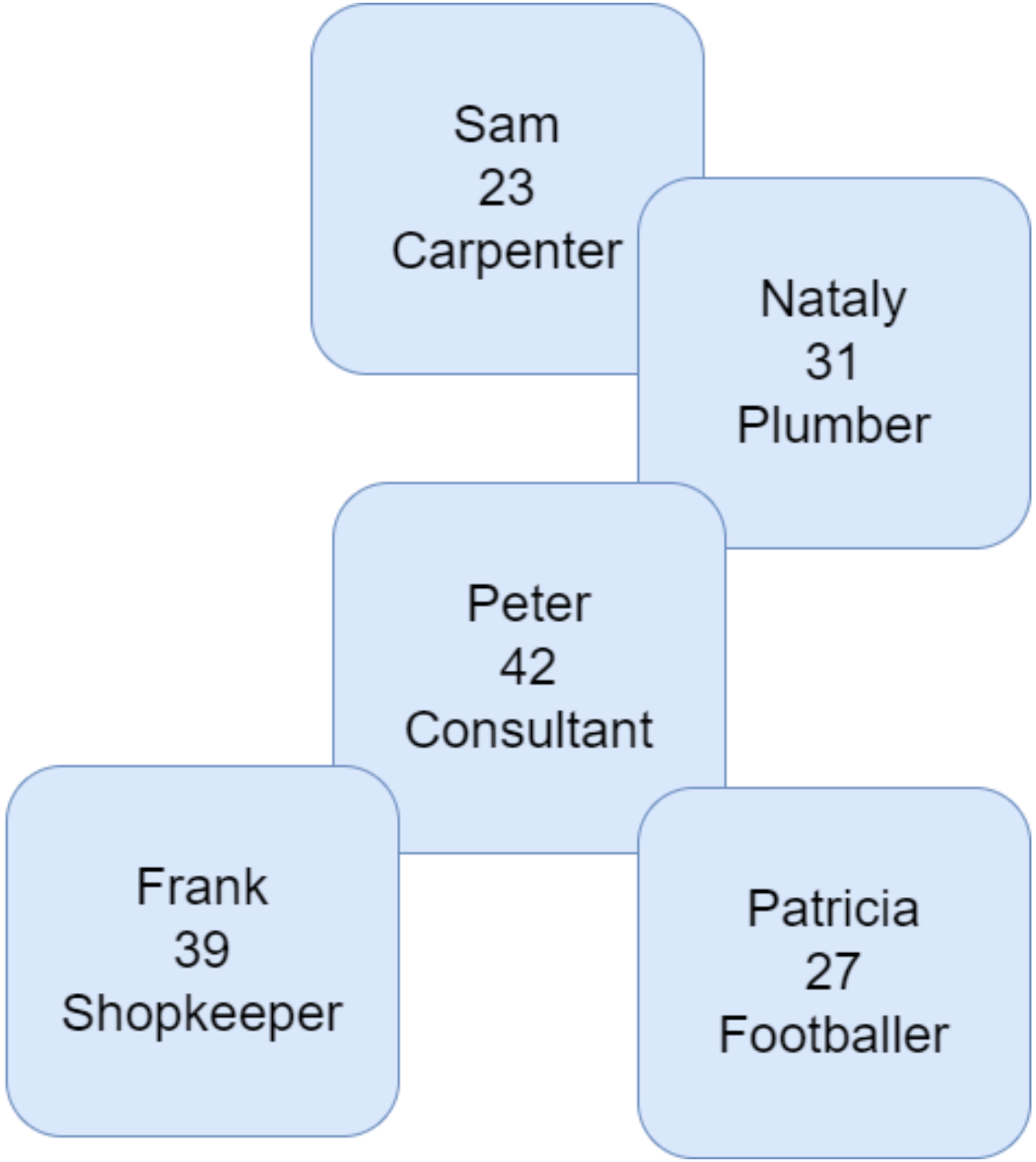
Class

Blueprint or template for user-defined data.

Defines data and functionality to be associated with each instance.

Does not yet reserve any memory space for data.


PERSON
Name
Age
Job
Sam
23
Carpenter
Nataly
31
Plumber
Peter
42
Consultant
Frank
39
Shopkeeper
Patricia
27
Footballer



Object

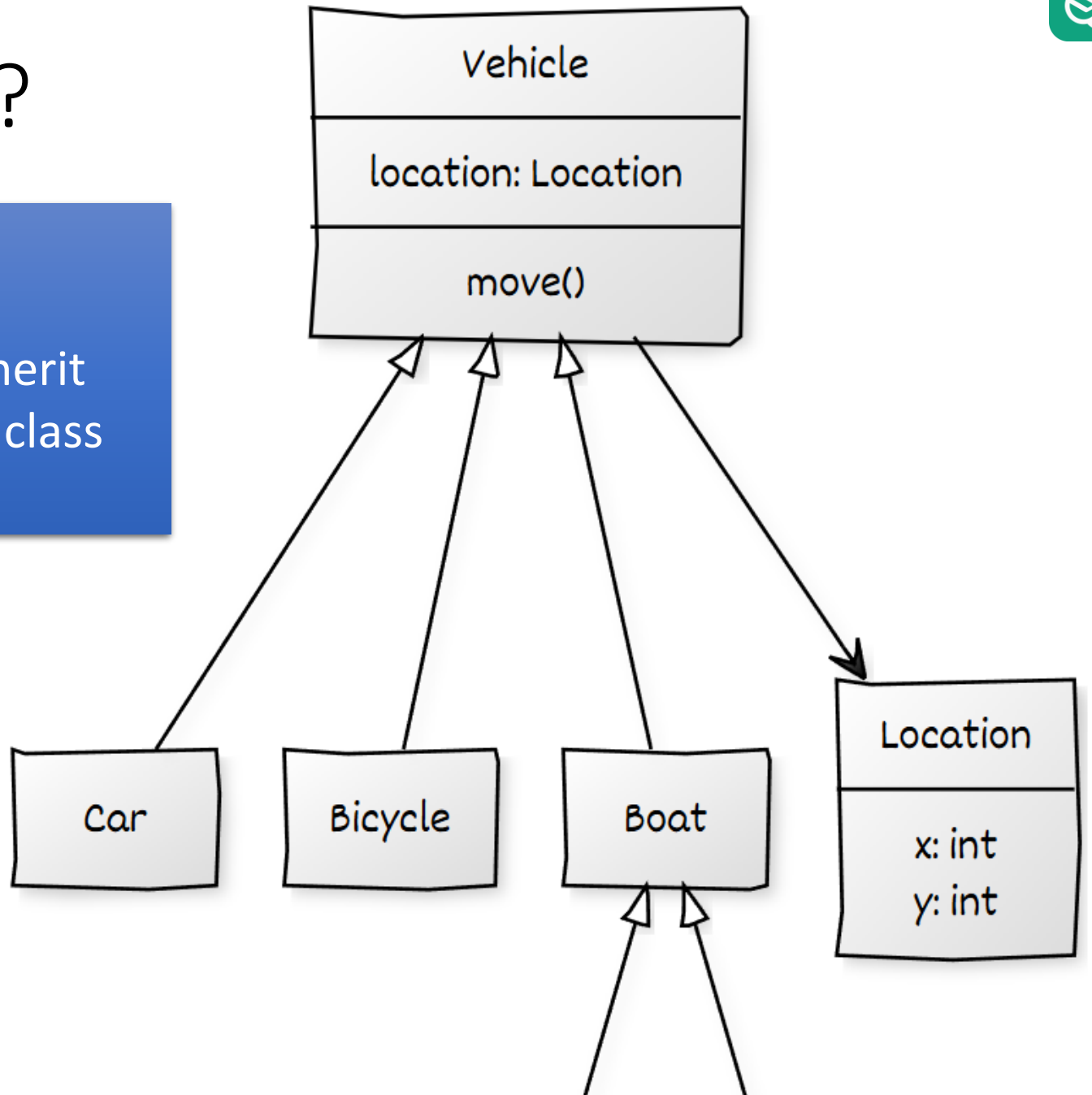
Instance of a specific object, based on the class definition.

Reserves specific memory space for the data, as any other variable type.

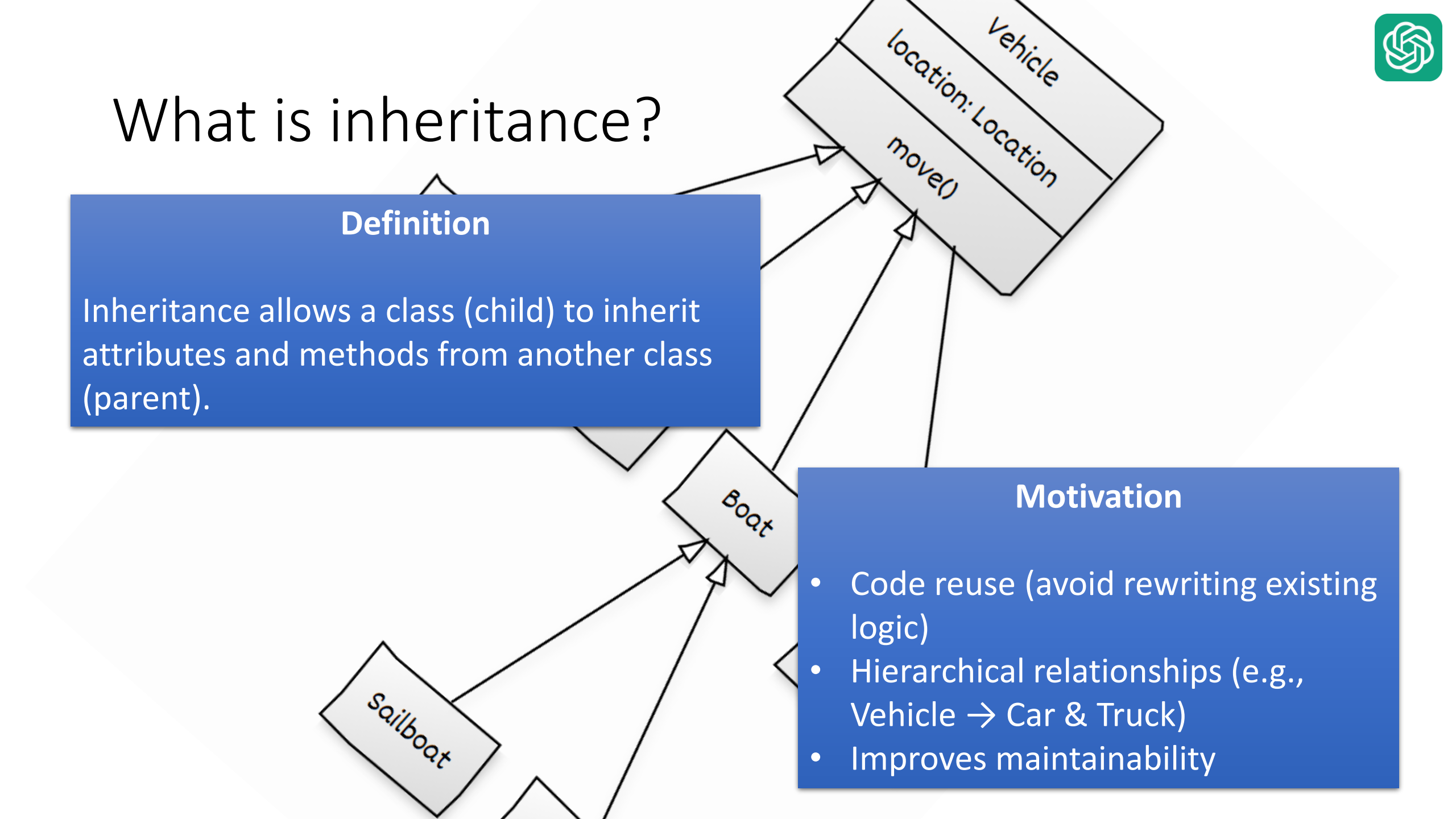
What is inheritance?

Definition

Inheritance allows a class (child) to inherit attributes and methods from another class (parent).



What is inheritance?



Definition

Inheritance allows a class (child) to inherit attributes and methods from another class (parent).

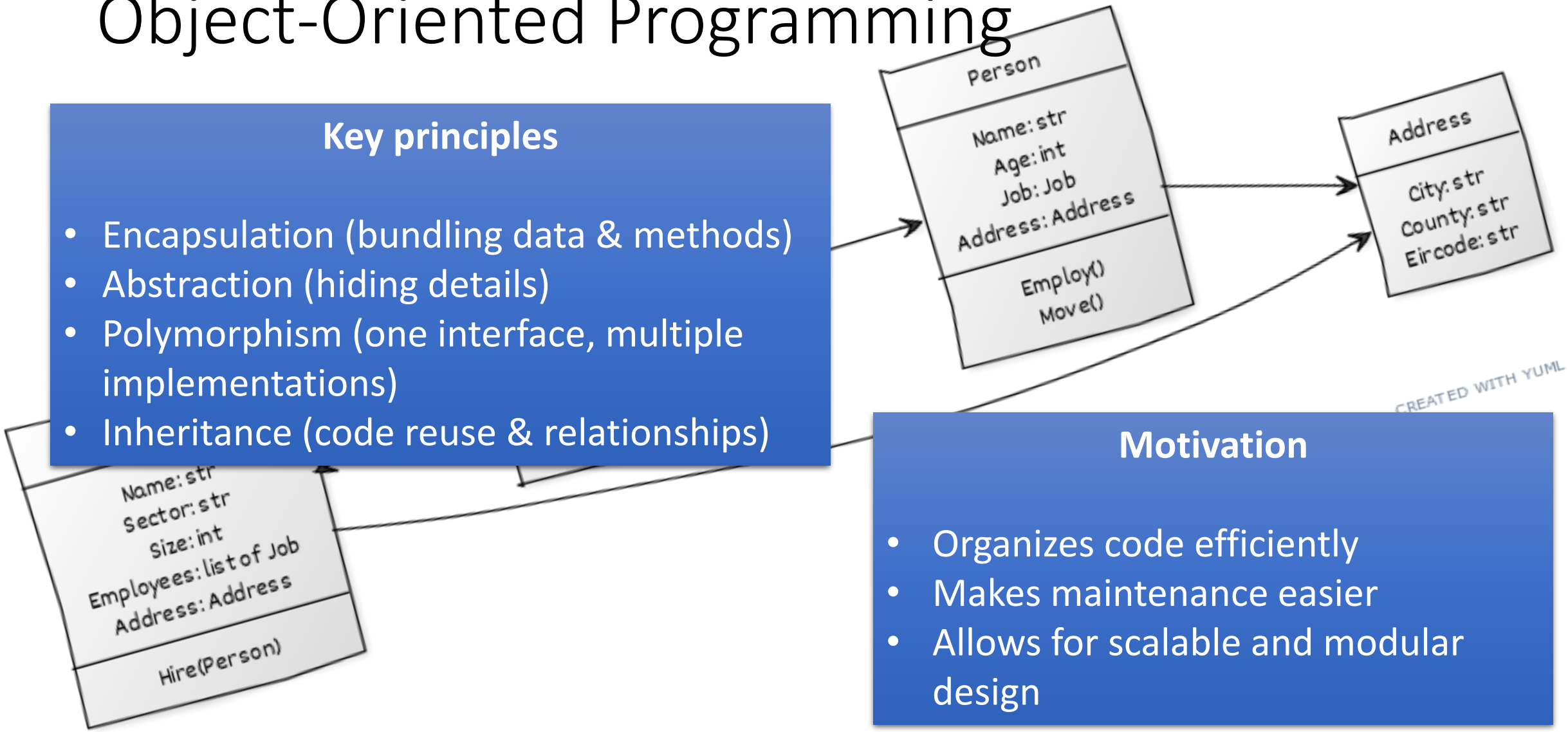
Motivation

- Code reuse (avoid rewriting existing logic)
- Hierarchical relationships (e.g., Vehicle → Car & Truck)
- Improves maintainability

Object-Oriented Programming

Key principles

- Encapsulation (bundling data & methods)
- Abstraction (hiding details)
- Polymorphism (one interface, multiple implementations)
- Inheritance (code reuse & relationships)

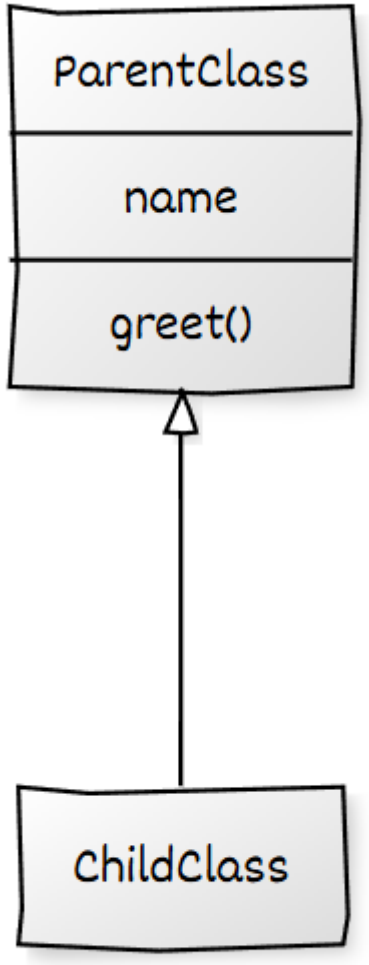


Motivation

- Organizes code efficiently
- Makes maintenance easier
- Allows for scalable and modular design

Inheritance in Python

```python
class ParentClass:
    def __init__(self, name):
        self.name = name

    def greet(self):
        print(f"Hello, my name is {self.name}")

class ChildClass(ParentClass):
    pass

obj = ChildClass("Alice")
obj.greet()  # Inherits greet() from ParentClass
```



CREATED WITH YUML

Lab experiment

Designer Works out how to approach the test. What kind of objects, functions, files do we need? What would be the input and output?

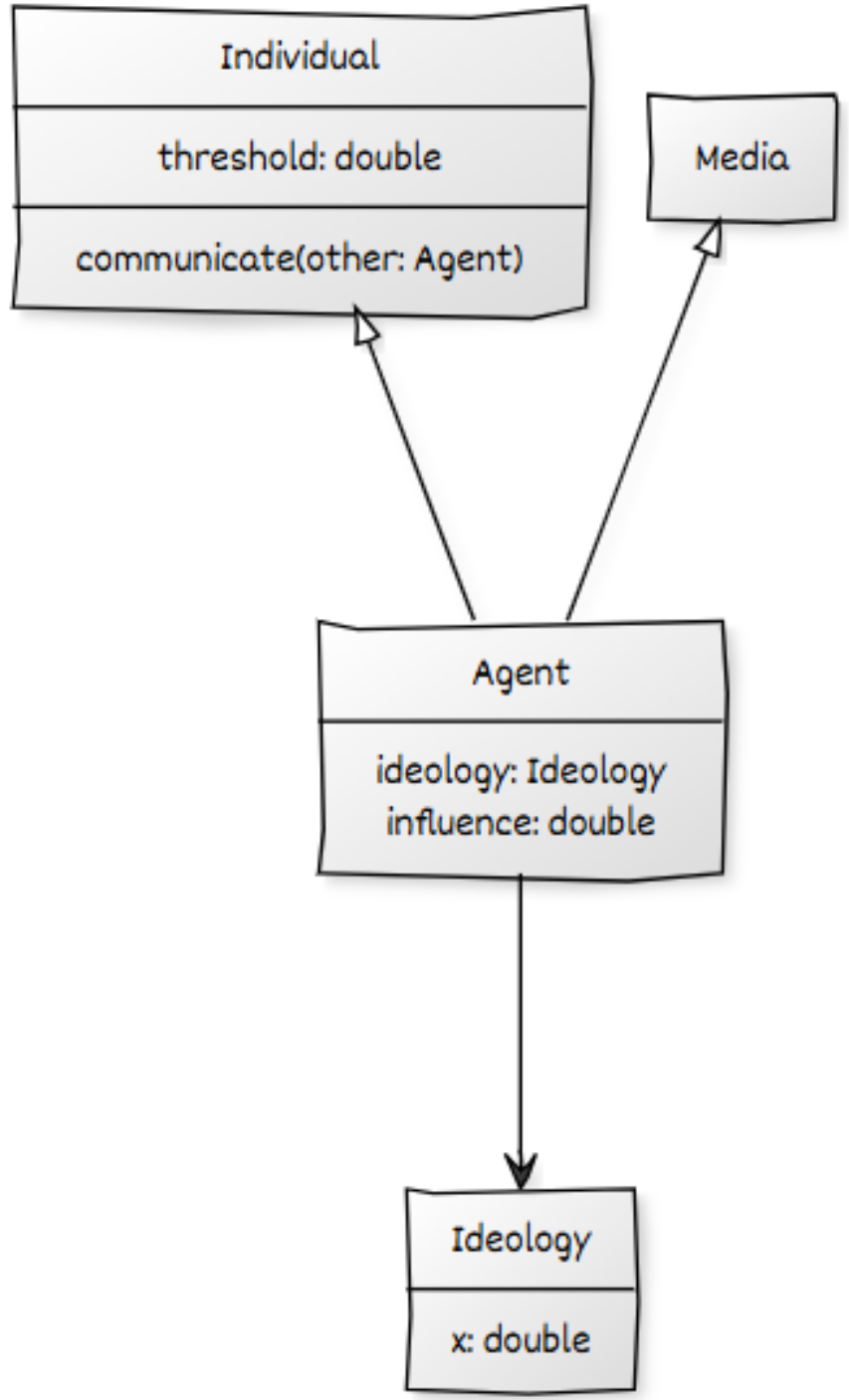
Coder Types up the code implementing the above, with support from the designer.

Tester Works out how to test the new code and implements Python code that tests the functionality.

Reporter Makes sure to understand everything that is being implement and reports back to the class what the design decisions were and what the main challenges are.

Critic Plays advocate of the devil and reviews the design and coding of the coder and the tester, asking critical questions.

```
class Agent:

  def __init__(self, ideology):
    self.ideology = ideology
    self.influence = random.random()
```



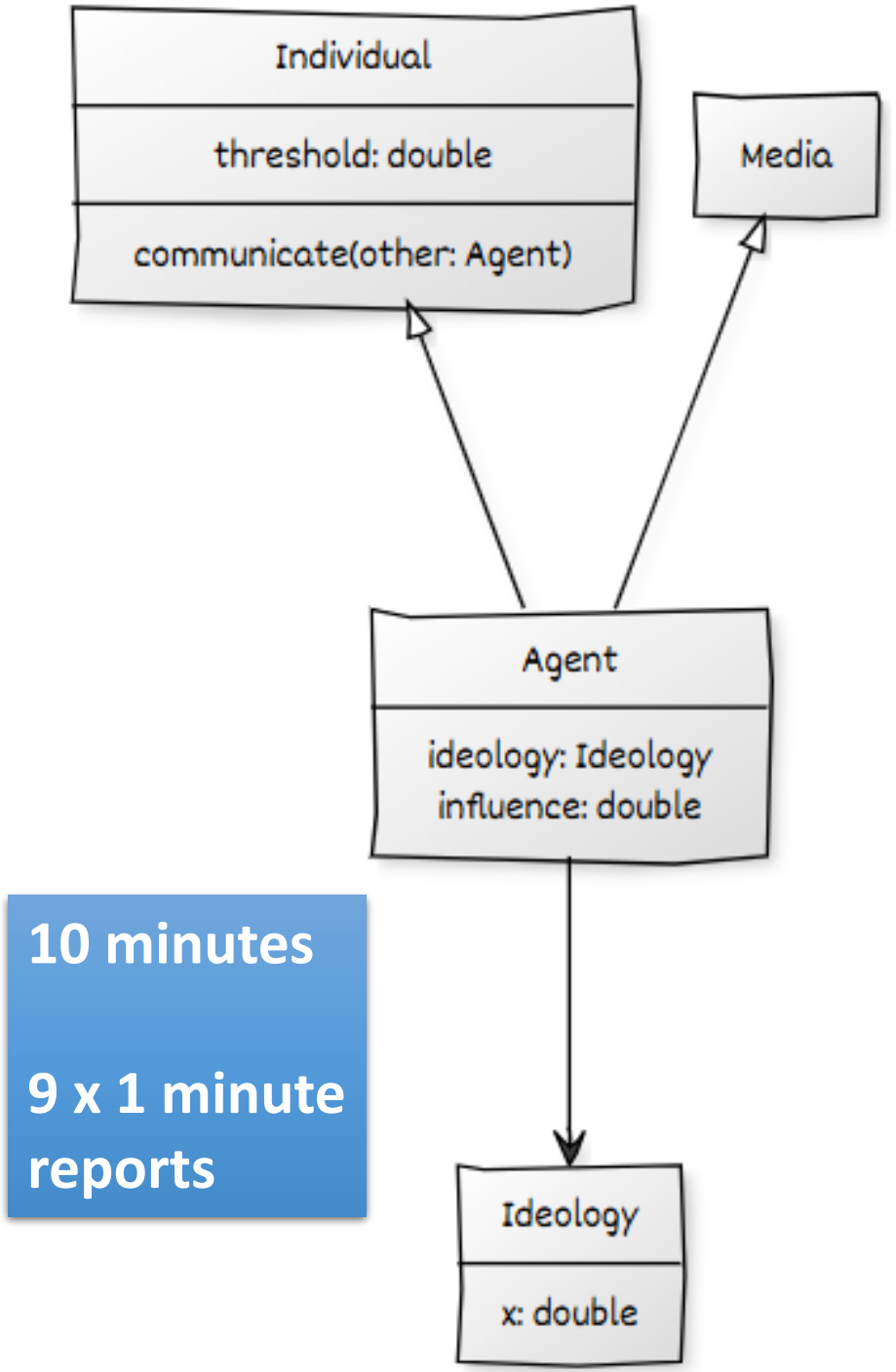
```
class Agent:

  def __init__(self, ideology):
    self.ideology = ideology
    self.influence = random.random()
```

TASK 1

Implement the above code in a file agent.py

Implement the Media class given the above Agent specification and the diagram in media.py

Create a media object of the Media class in main.py

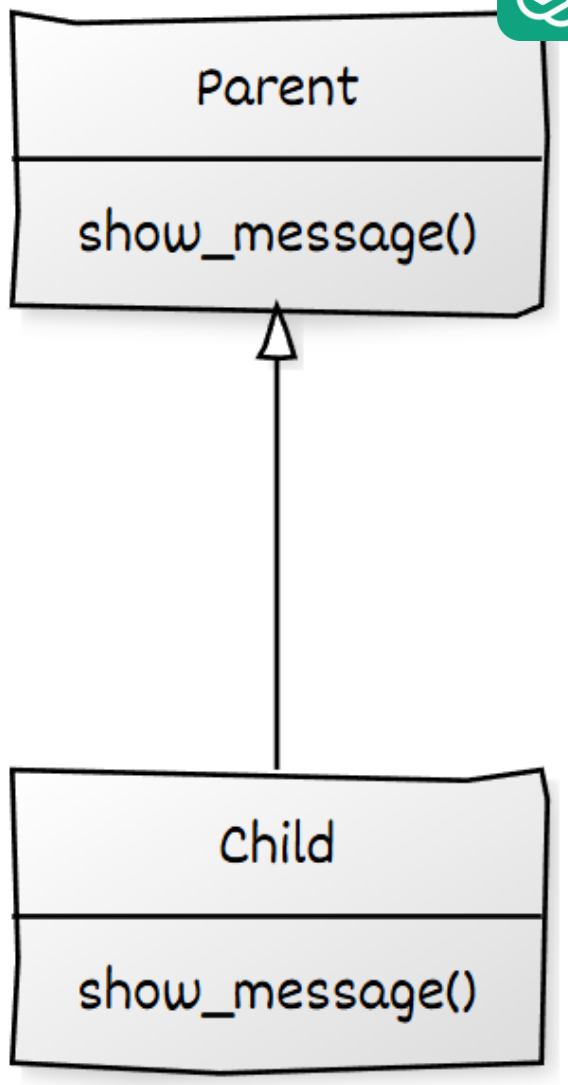


10 minutes

9 x 1 minute reports

Overriding methods

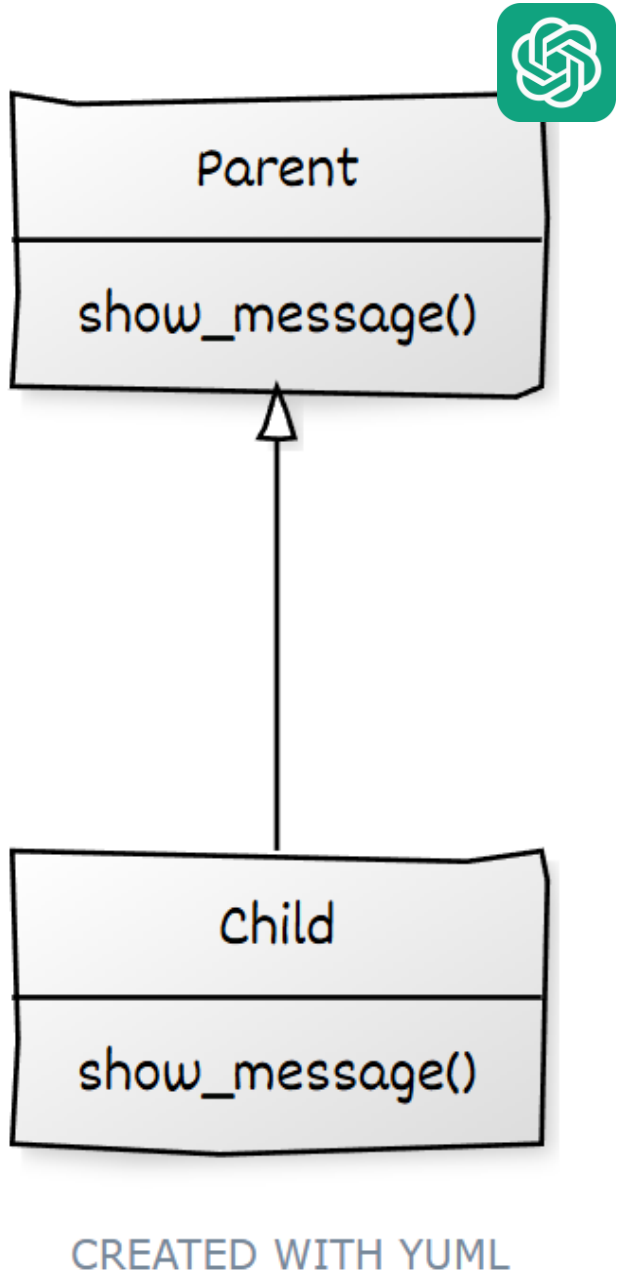
```python
class Parent:
    def show_message(self):
        print("This is a message from the Parent class.")

class Child(Parent):
    def show_message(self):  # Overriding
        print("This is a message from the Child class.")

obj = Child()
obj.show_message()  # Calls the overridden method
```



CREATED WITH YUML

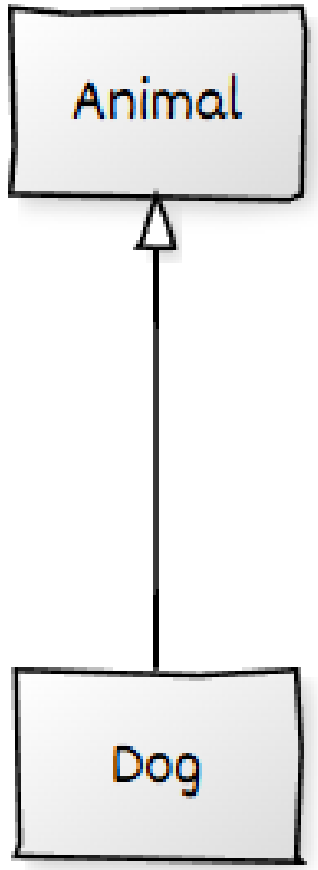
Using super()

```python
class Parent:
    def show_message(self):
        print("This is a message from the Parent class.")

class Child(Parent):
    def show_message(self):
        super().show_message()  # Calls Parent's method
        print("This is a message from the Child class.")

obj = Child()
obj.show_message()
```



Using super() in the constructor

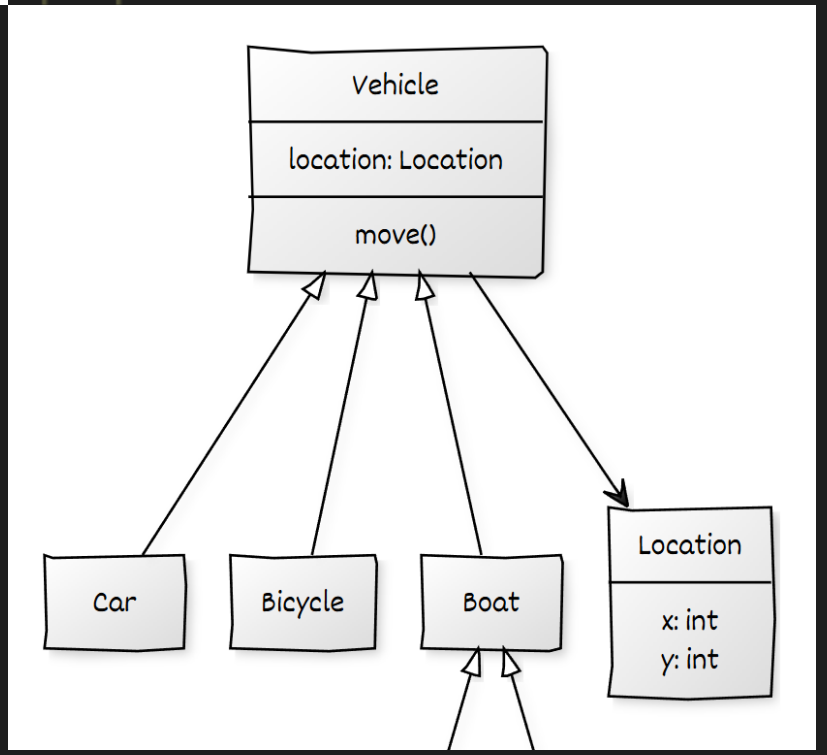
```python
class Animal:
    def __init__(self, name):
        self.name = name

class Dog(Animal):
    def __init__(self, name, breed):
        super().__init__(name)  # Call parent's __init__
        self.breed = breed

dog = Dog("Buddy", "Golden Retriever")
print(dog.name)   # Inherited from Animal
print(dog.breed)  # Defined in Dog
```



Example in Python

```python
class Location:

  def __init__(self):
    self.x = 0
    self.y = 0
```

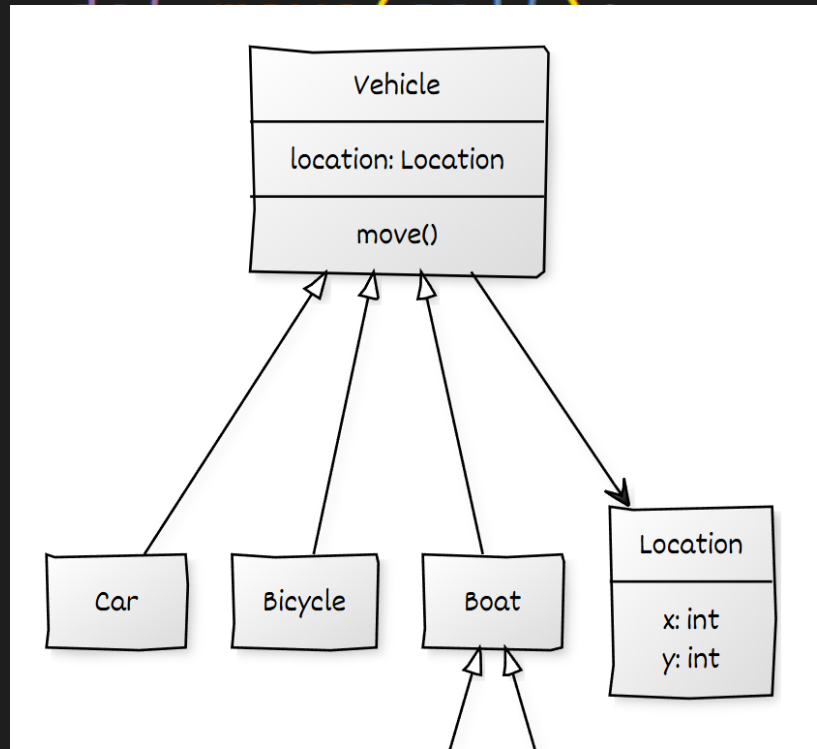
```python
class Vehicle:

  def __init__(self):
    self.location = Location()
    self.speed = 1
    self.direction = 0

  def move(self):
    self.location.x += self.speed * math.cos(self.direction)
    self.location.y += self.speed * math.sin(self.direction)
```



Example in Python

```python
class Car(Vehicle):

    def __init__(self):
        super().__init__()
        self.fuel = 100

    def drive(self):
        if self.fuel > 0:
            self.move()
            self.fuel -= 1
        else:
            print("Out of fuel!")
```

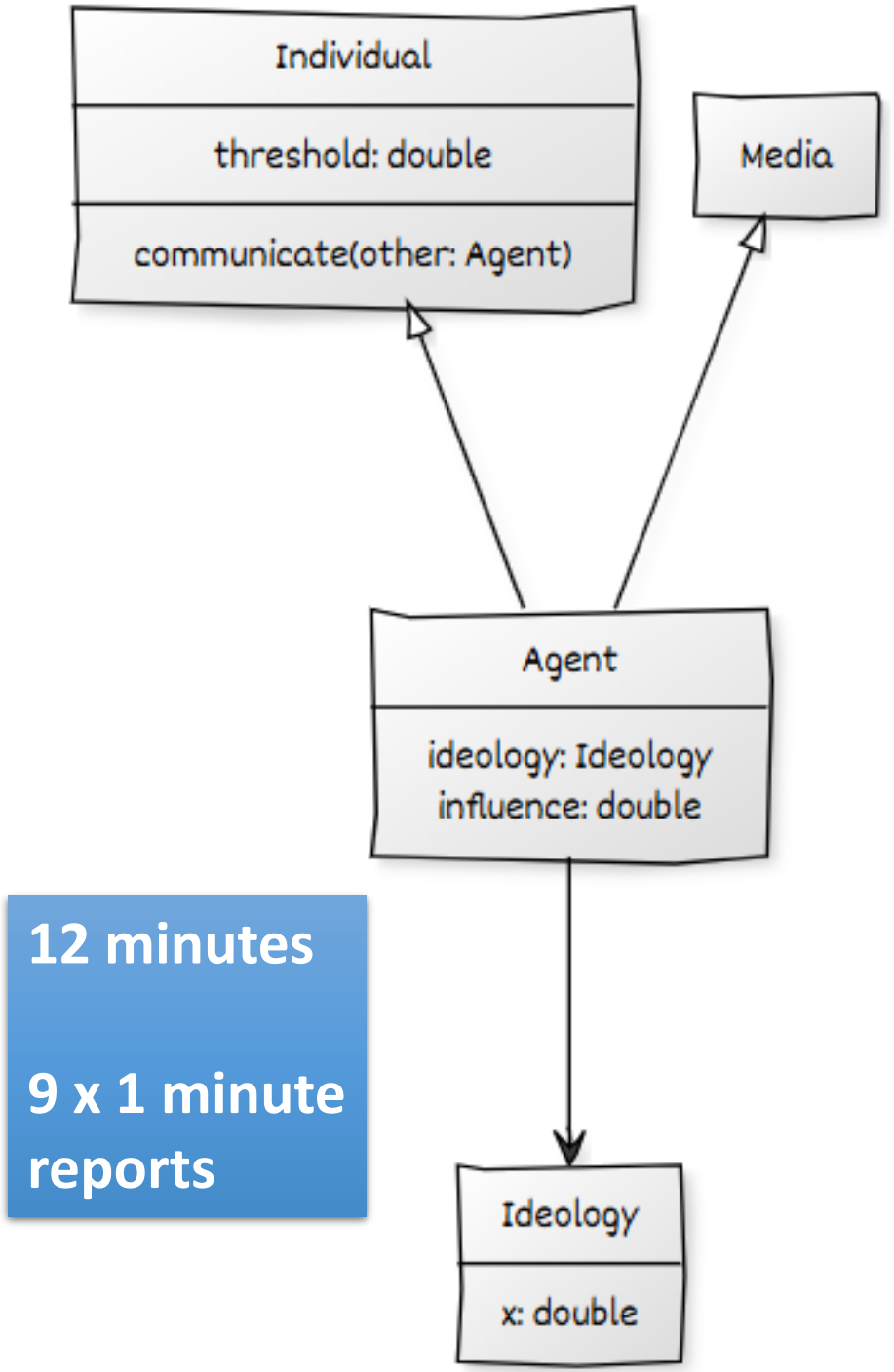
```python
class Vehicle:

    def __init__(self):
        self.location = Lo
        self.speed = 1
        self.direction = 0
```



```
class Agent:

  def __init__(self, ideology):
    self.ideology = ideology
    self.influence = random.random()
```

TASK 2

Implement the Individual class given the above Agent specification and the diagram (without the communicate() method).

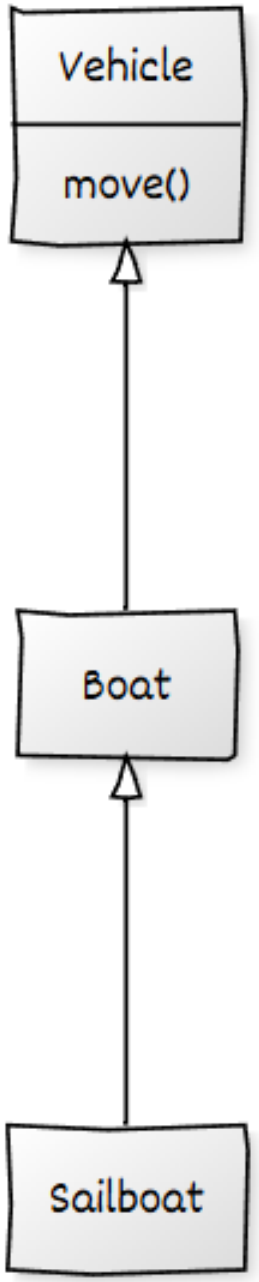


12 minutes

9 x 1 minute reports

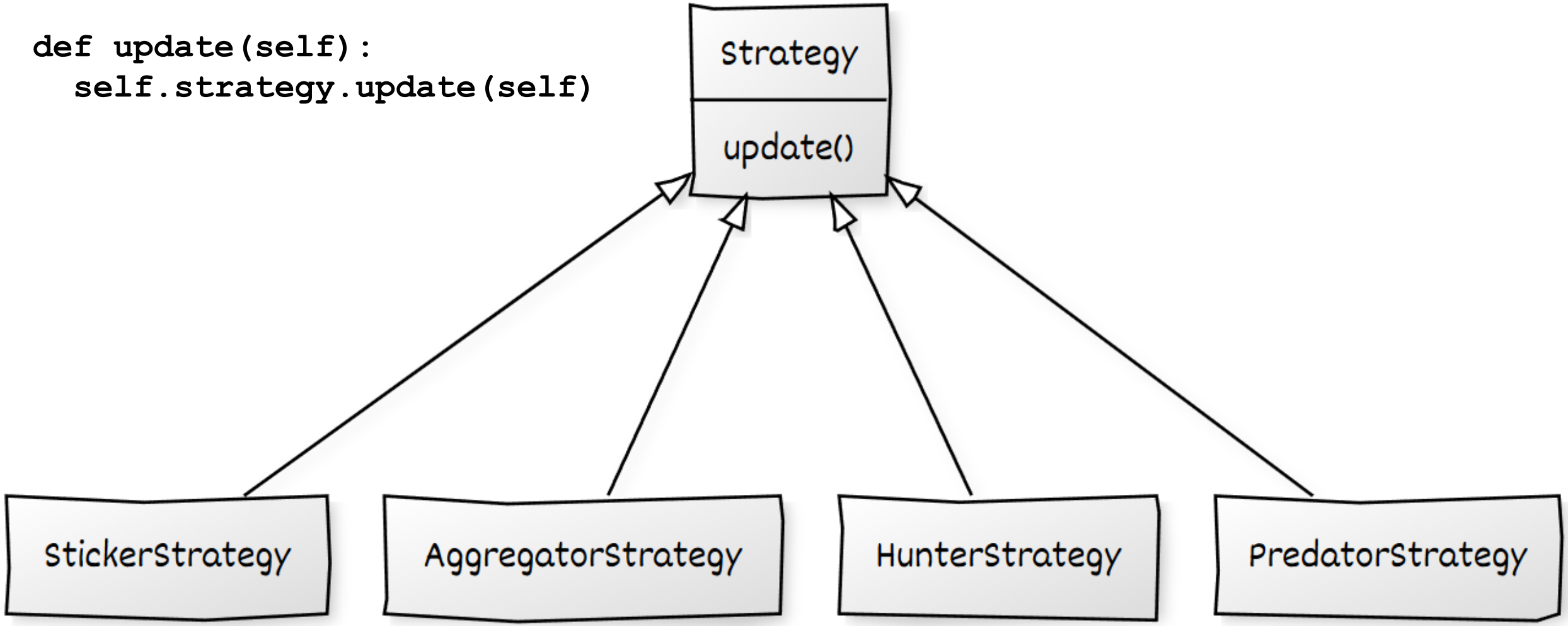
Multi-level inheritance

```python
class Vehicle:
    def move(self, distance):
        print(f"Vehicle moved {distance} meters")

class Boat(Vehicle):
    pass

class Sailboat(Boat):
    pass

mySailboat = Sailboat()
mySailboat.move(100)
# Output: Vehicle moved 100 meters
```



CREATED WITH YUML

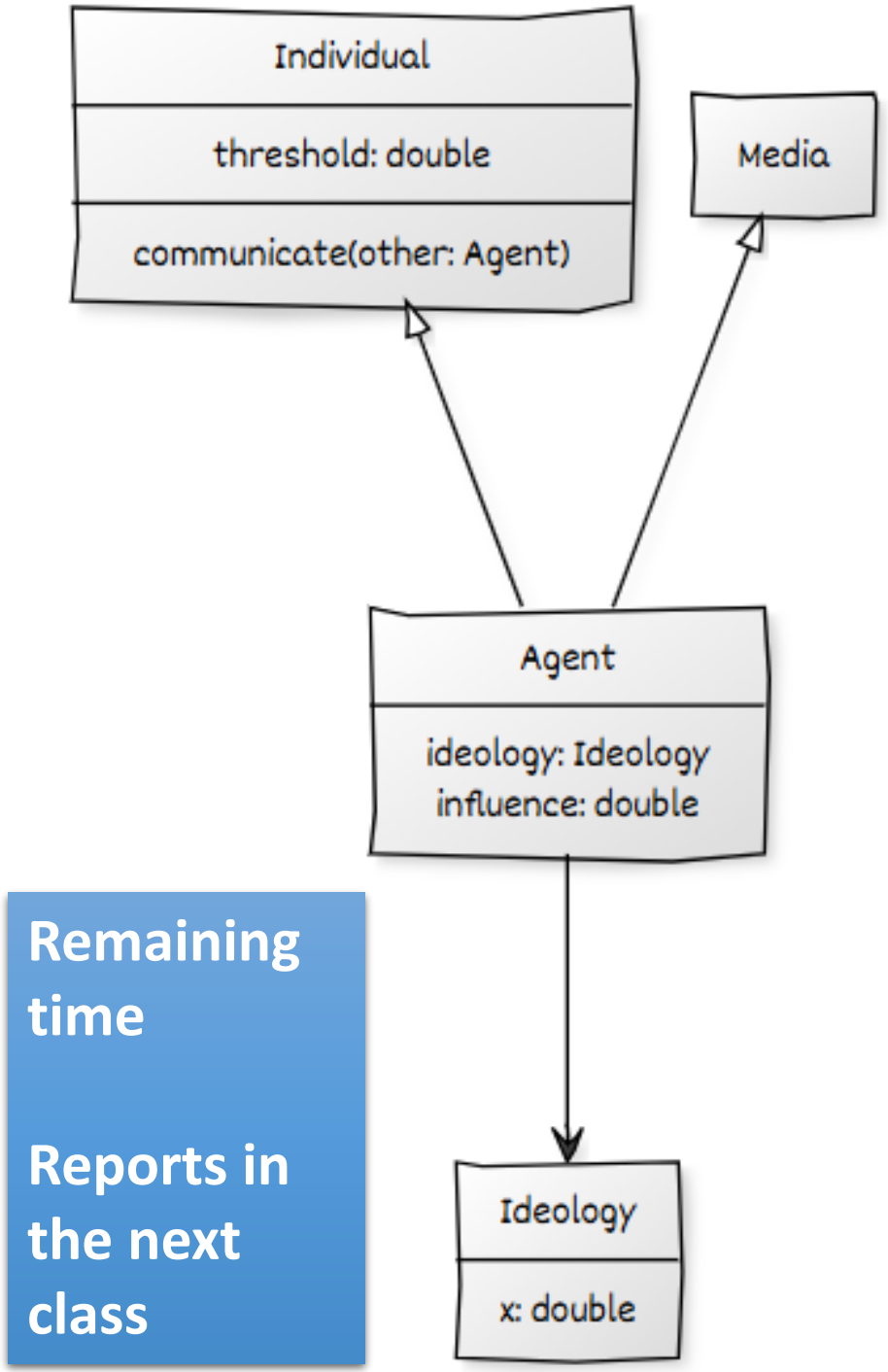
```
class Party:

  def __init__(self, strategy):
    self.strategy = strategy

  def update(self):
    self.strategy.update(self)
```



```
class Agent:

  def __init__(self, ideology):
    self.ideology = ideology
    self.influence = random.random()
```

TASK 3

Implement the communicate() method in the Individual class. See Lab 4, Q. 2 as well as Lab 6, Q. 6.



Remaining time

Reports in the next class