Classes and objects

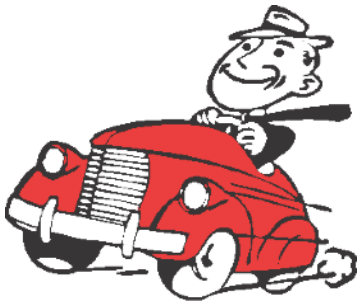
COSC346

Objects in real world

- An object is a thing
- A real-world example is a car

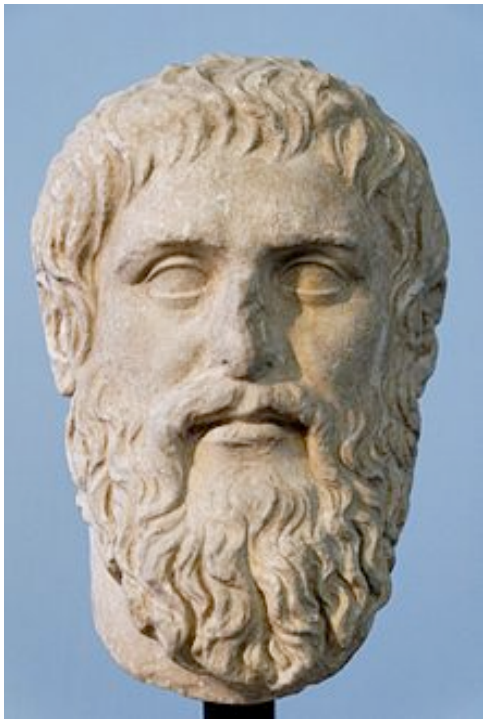
Objects in real world

- Objects have properties
- You can *act* on objects
- Objects interact



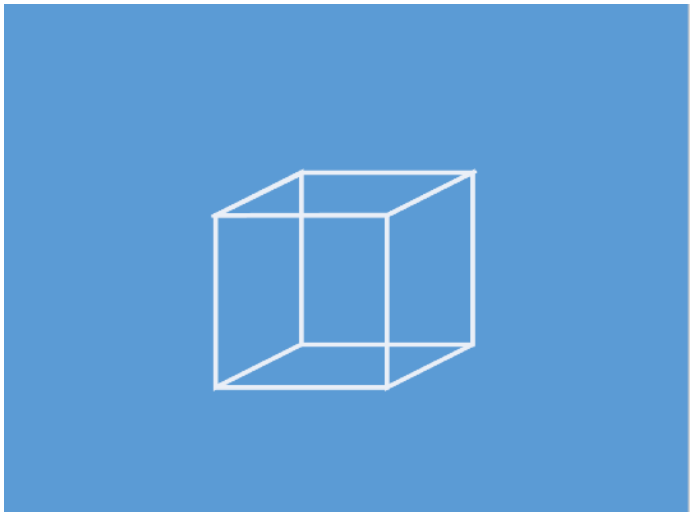
Plato's Theory of Forms

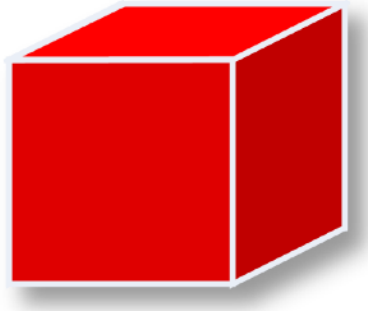
- Objects that we see mimic real Forms
- A Form is an idea, an abstract concept that conveys the essence of an object
- Example:
  - What is the form of a "car"?
  - How does a car you see on the street correspond to its form?



Class and object

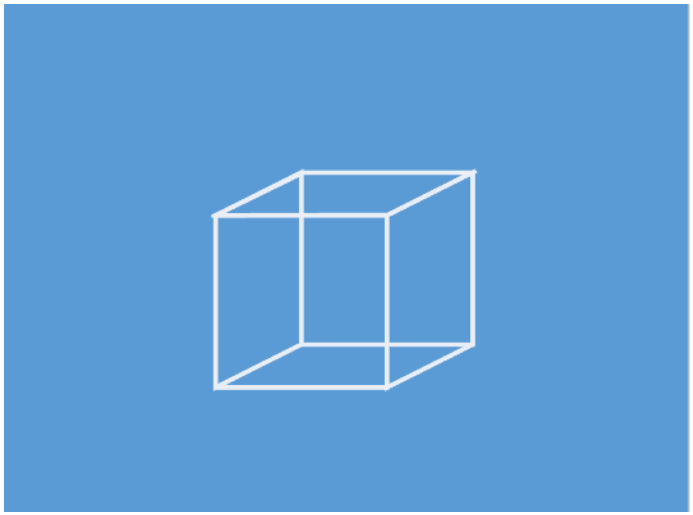
- A class is a specification of how the object is to be built (essence of an object)
- An object is an instance of a class (a "real" object...in computer memory)

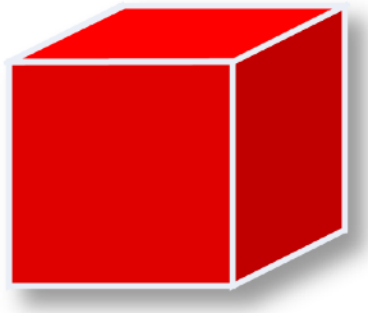




Class

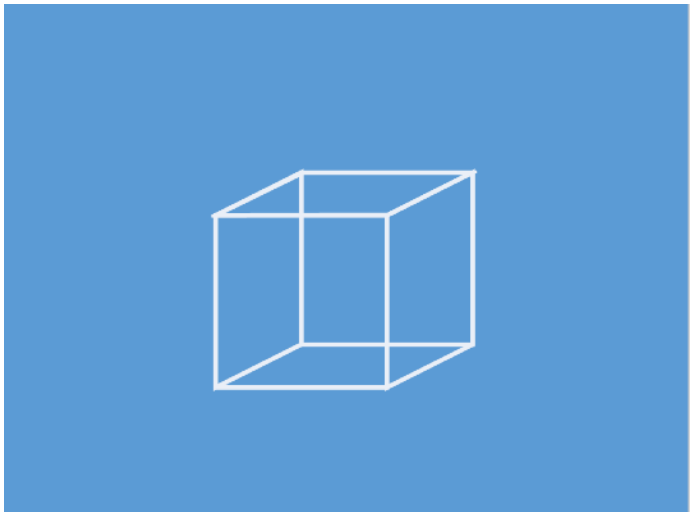
- A class defines a type by specifying:
  - What its state is composed of (its internal variables and properties)?
  - How it behaves (its methods)?

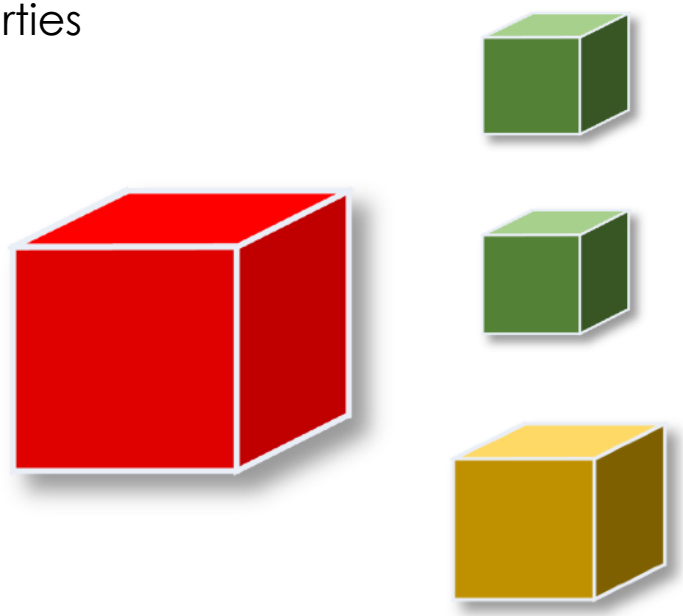




Object instances

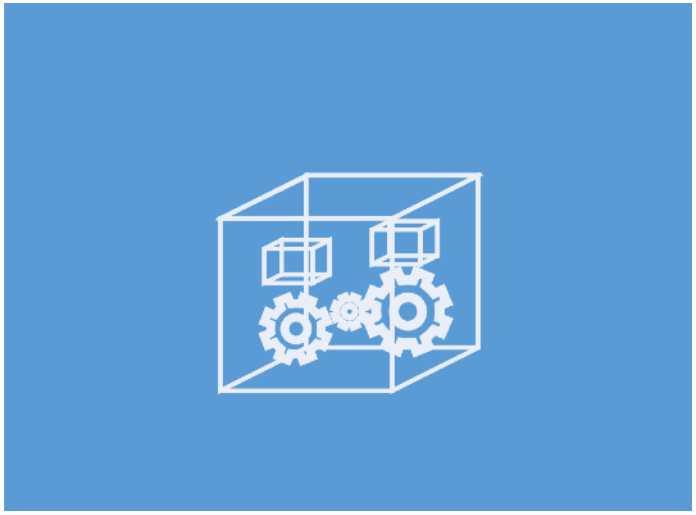
- An object instance is a particular "realisation" of a given class
  - Properties take on specific values
  - Behaviour of a given object may depend on its state and properties
  - Different instances can have different properties

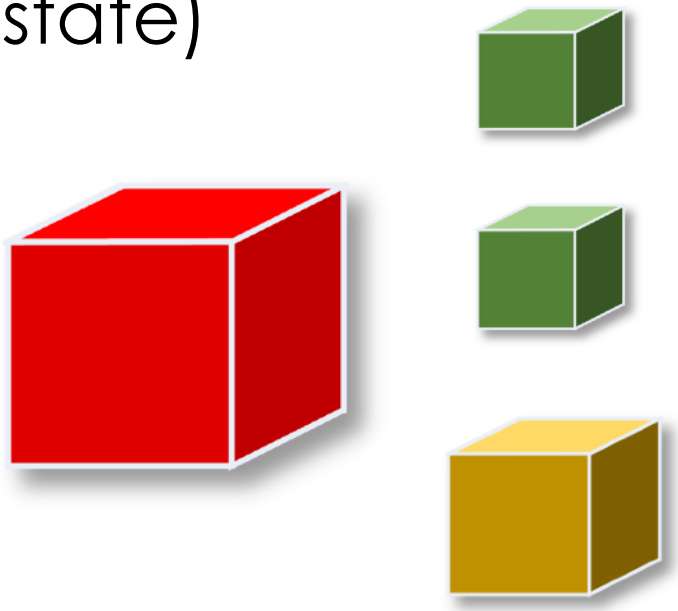




Object state

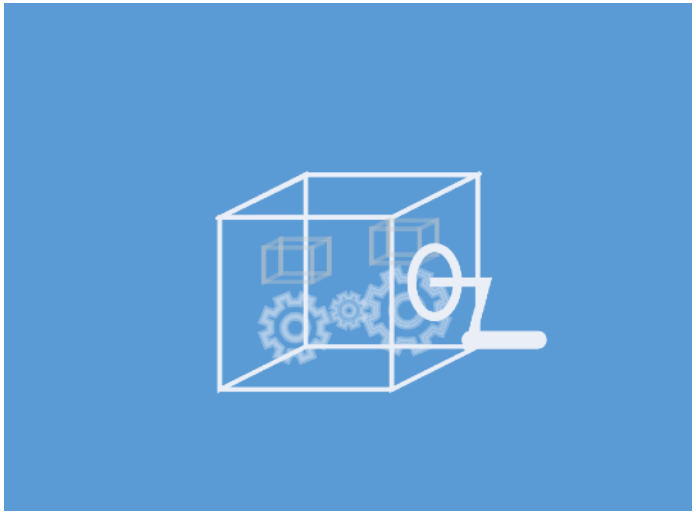
- **Instance variables** specify object's state
- Some of this state is visible to the object user (object properties) …
- … and some is not (internal state)

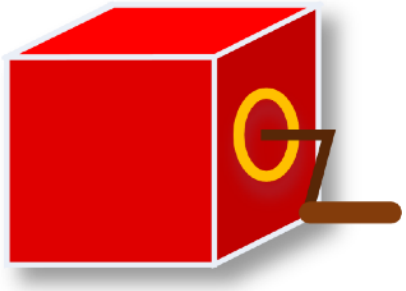




Methods

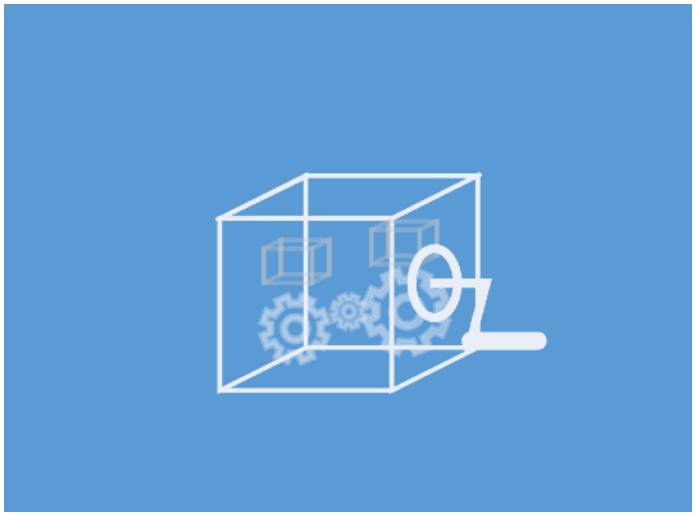
- **Methods** are class specific functions that define what the object does and how it does it





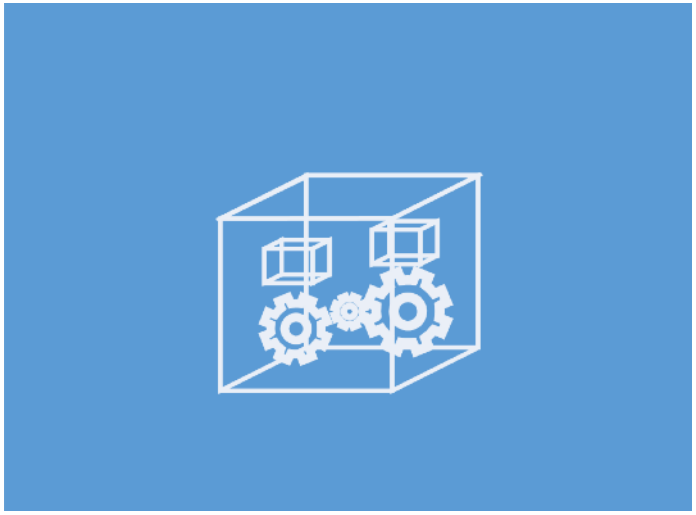
Methods

- **Methods** are class specific functions that define what the object does and how it does it



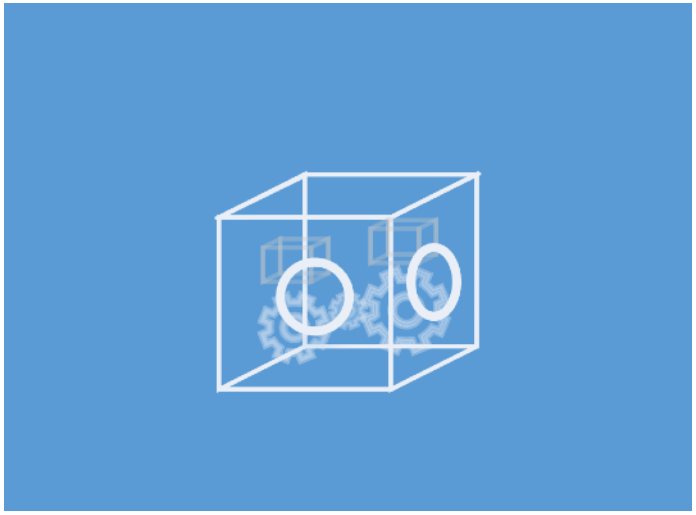
Abstraction

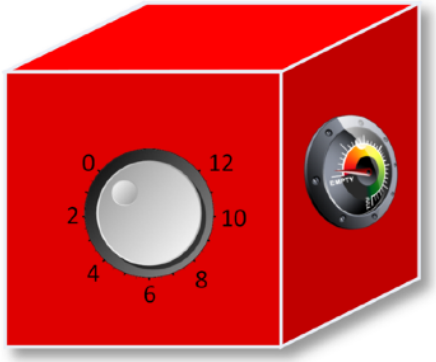
- Knowledge of the inner workings of the object is not required in order to use it
- It's sufficient that user understands object's properties (visible state) and how to use it



Encapsulation

- Internal state may not be directly visible to user, but interface (methods) may be provided to allow user to modify the state
  - Ability to control access to the inner state of the object
- Accessor methods:
  - **Setters** – methods that allow writing to internal variables
  - **Getters** – methods that allow reading of internal variables





Visibility

Mechanic

- Works on the engine, so that car is drivable
- Engine internals are hidden away under the hood

Driver

- Does not need to understand how the engine works
- Does not need to look at the engine
- Needs to use the interface skilfully in order to control the engine and drive

Visibility

Toolmaker

- Works on the implementation of the class, so that its object is usable
- Class internals can be hidden away

Builder

- Does not need to understand details of the class internals
- Does not need to look at the internals
- Needs to instantiate objects of the class and use their methods skilfully in order to co produce desired program logic

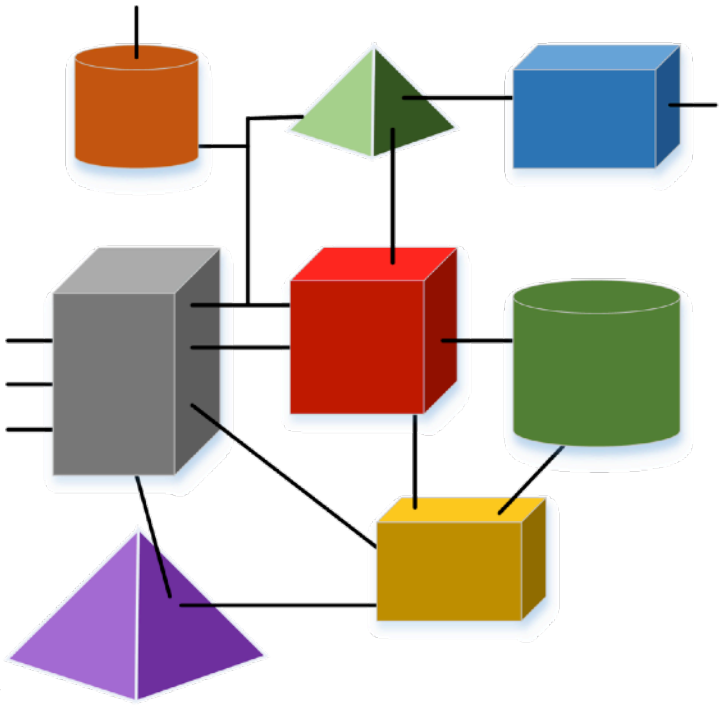
Visibility

- Class creator can decide the degree of visibility into its internals

- Access Control:
  - Private – only visible from within class implementation (internal use)
  - Public – visible to the object user (internal and external use)

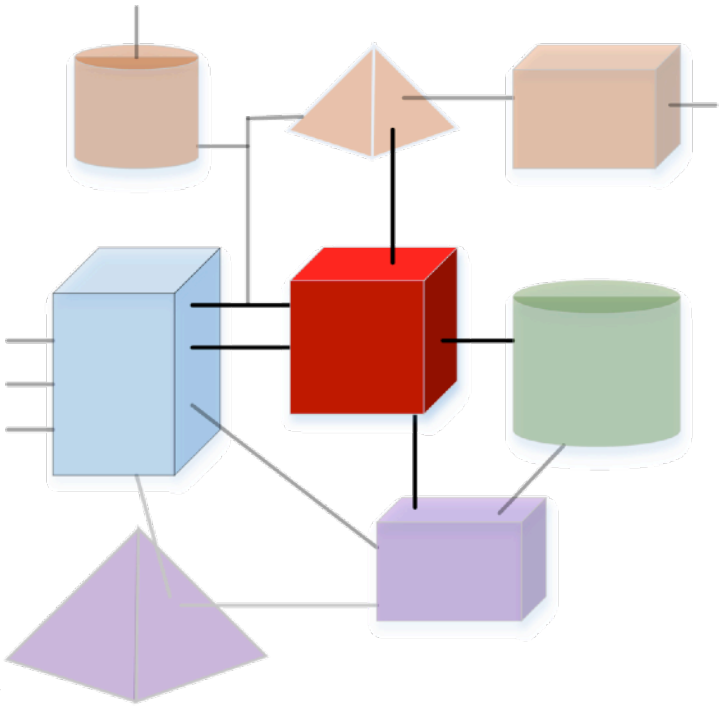
Interchangeability

- Ability to change inner working of an object without affecting its interface and the code depending upon it

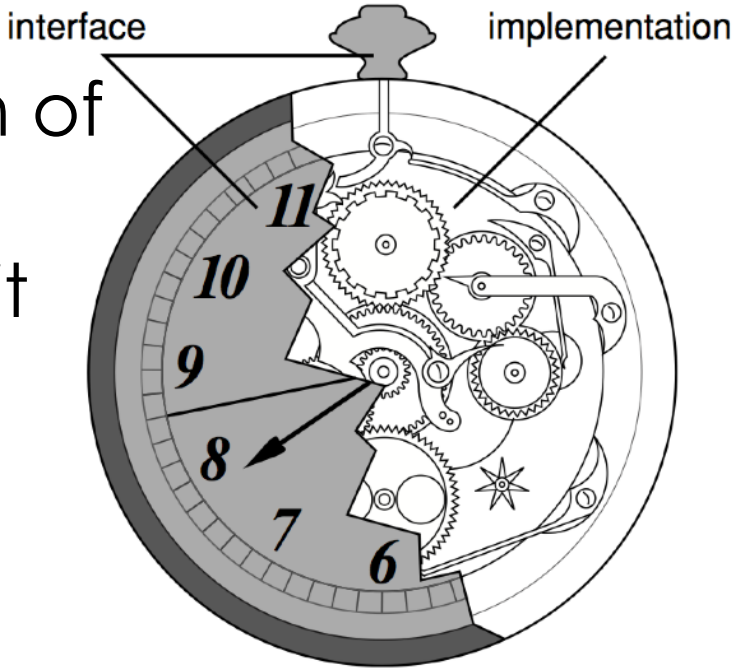


Interchangeability

- Ability to change inner working of an object without affecting its interface and the code depending upon it

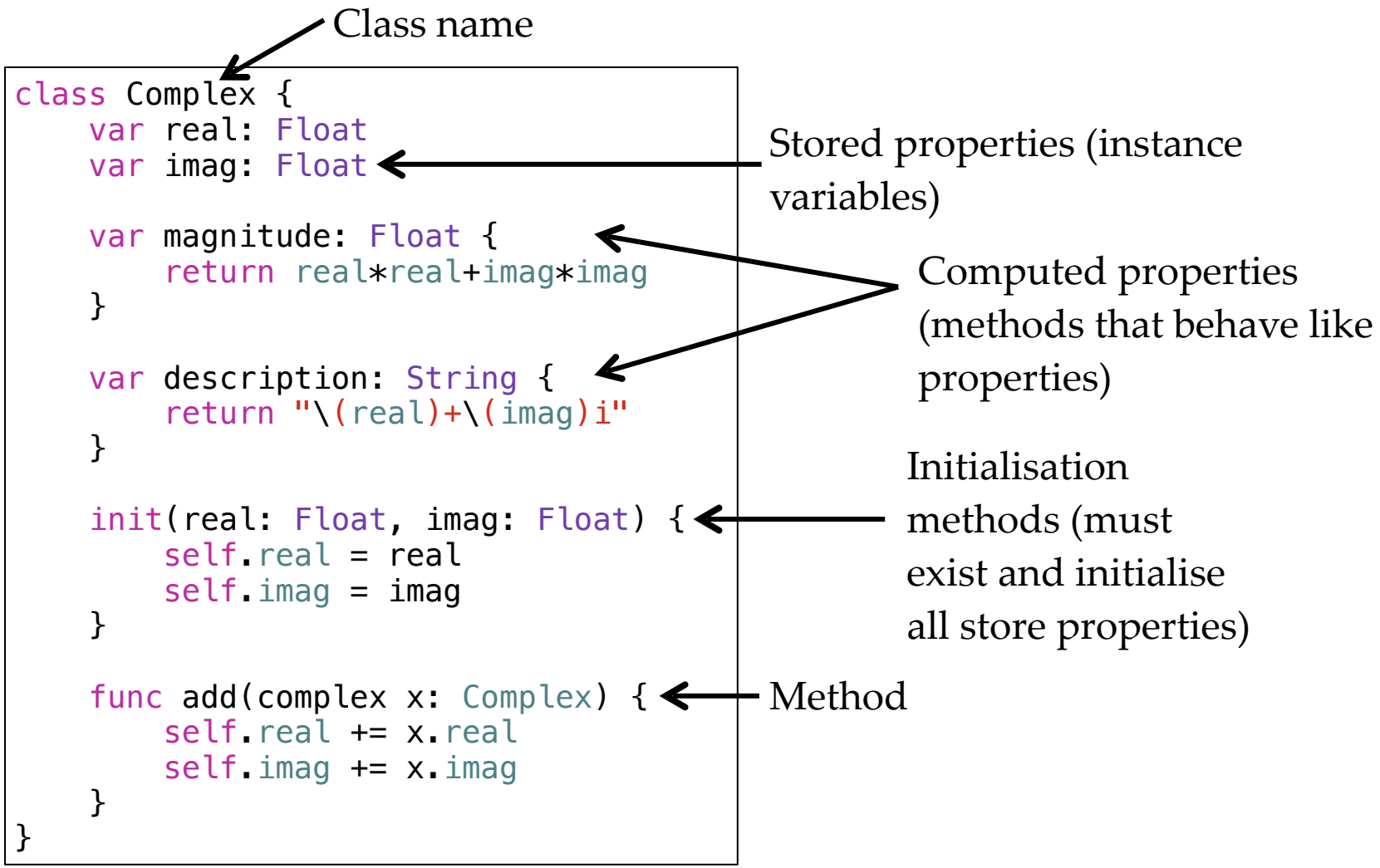


Interface and implementation



- **Interface**—declaration of what the object is and what can be done to it
- **Implementation**—the code that defines the behaviour of the class object
- In many languages class interface and implementation are specified separately (header and implementation files)

How to define a class



How to define a class

Swift

```
class Complex {
    var real: Float
    var imag: Float

    var magnitude: Float {
        return real*real+imag*imag
    }

    var description: String {
        return "\(real)+\(imag)i"
    }

    init(real: Float, imag: Float) {
        self.real = real
        self.imag = imag
    }

    func add(complex x: Complex) {
        self.real += x.real
        self.imag += x.imag
    }
}
```

- Swift doesn't separate interface and implementation: it's all in one place
- Setters & getters can be defined within computed property
- The default setting for access control makes class internals visible to all files in the module/project

How to create an object instance

Swift

Create two instances of Complex objects with different internal state

```
var m: Complex = Complex(real: 3.1, imag: -0.5)
var n: Complex = Complex(real: 1.0, imag: 2.3)
```

Stored properties

```
print("The magnitude of \(m.real)+\(m.imag)i is \(m.magnitude)")
print("The magnitude of \(n.description) is \(n.magnitude)")
```

Computed properties

```
print("(\(m.description))+(\(n.description))=", terminator: "")
m.add(complex:n)
print("\(m.description)")
```

← Invoke a method on 'm' with 'n' passed in as a parameter