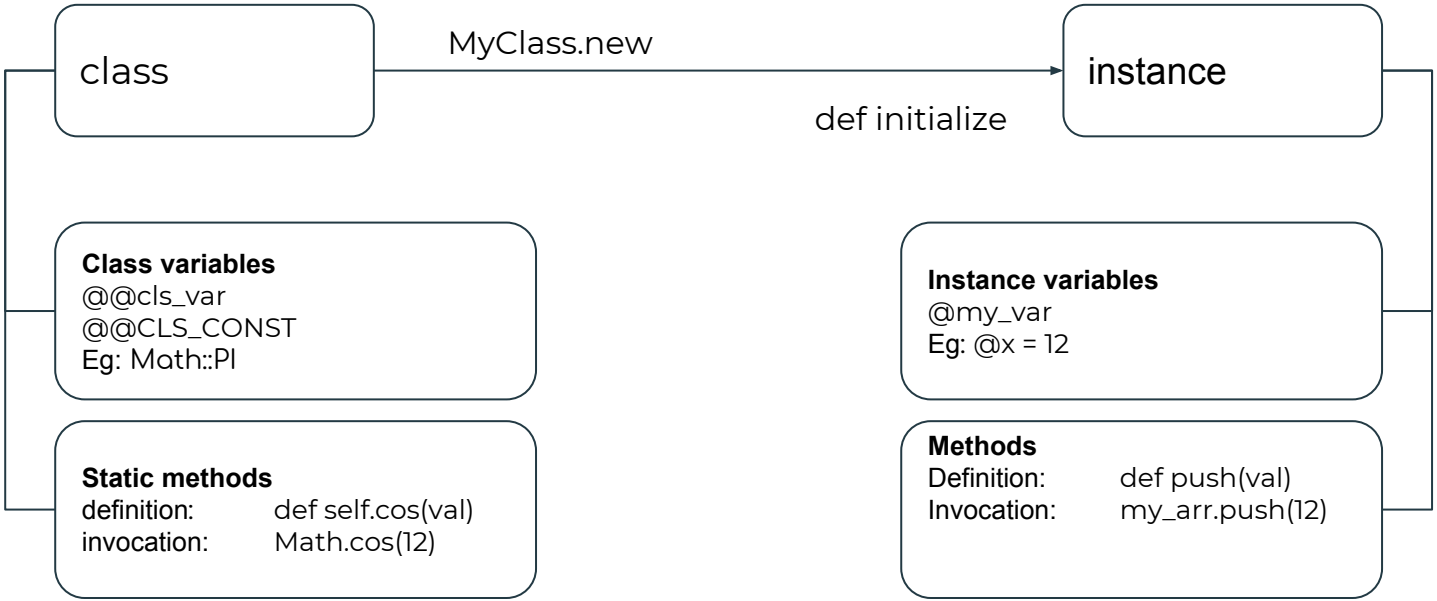
Laboratory of Ruby

OOP recap & some classes

Martin Brugnara - Marco Frego

Object Oriented Programming - OOP

Classes & Objects



Incapsulamento

“L'*incapsulamento* è la proprietà per cui i dati che definiscono lo stato interno di un oggetto e i metodi che ne definiscono la logica sono accessibili ai metodi dell'oggetto stesso, mentre non sono visibili ai *client*. Per alterare lo stato interno dell'oggetto, è necessario invocarne i metodi pubblici, ed è questo lo scopo principale dell'incapsulamento” - it.wikipedia.org

TL;DR;

- Astrarre implementazione da funzionalità
  - Vantaggio 1: non serve “sapere” i dettagli implementativi di Math.cos per poterla utilizzare.
  - Vantaggio 2: l’implementazione di Math.cos può essere cambiata senza dover modificare altro codice.
- Variabili e metodi privati vs pubblici
  - Privati: utilizzabili solo all’interno della classe per l’implementazione di altri metodi.
  - Pubblici: utilizzabili sia internamente che fuori dalla definizione della classe.

Ereditarietà

```
class Person
      @status = ‘hold’
      def move()
            @status = ‘moving’
      end
end

class Pilot < Person
      def drive()
            @status = ‘driving’
      end
end
```

```
a = Person.new
b = Pilot.new

a.move
b.move

a.drive()     # → Error: drive is not defined for
Person
b.drive()
```

- Person e’ **superclasse** di Pilot
- Pilot e’ **sotto-classe** di Person

Polimorfismo

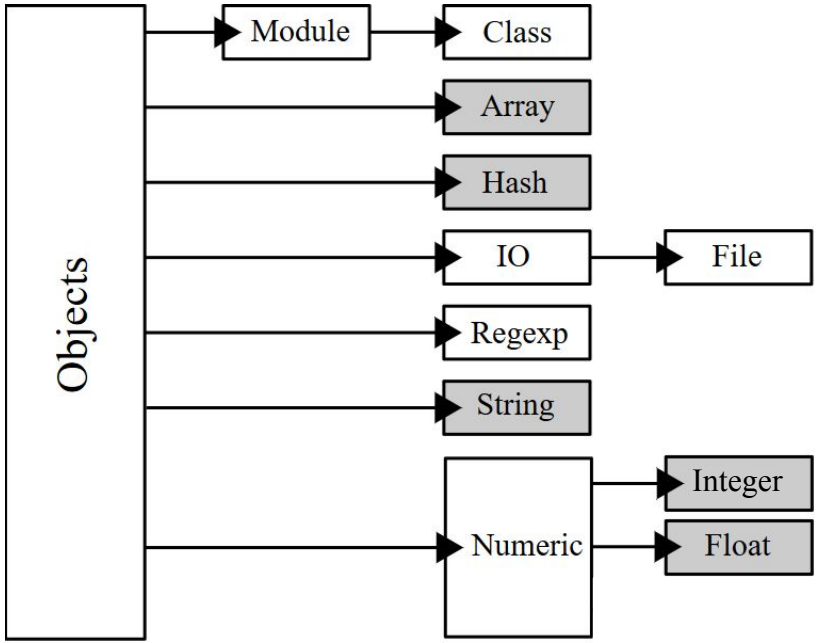
E’ possibile utilizzare un oggetto di una sottoclasse al posto di uno di una superclasse.

- Utile per specializzare azioni ma offrire interfacce uniformi.

```
class Person
        ...
        def get_signature()
                return @surname + ‘ ‘ + @name
        end
end

class Doctor < Person
        def get_signature()
                return ‘Doc. ‘ + super()
        end
end
```

Everything inherits from Object



Some useful classes (from ruby's stdlib)

- Numeric and Array - https://www.ragni.me/ncalc/lecture6b/
- Strings and Hash - https://www.ragni.me/ncalc/lecture6c/
- Files - https://www.ragni.me/ncalc/lecture7/#i-file

Exercise 1

Write a program that reads a list of values from a file and print them sorted on the screen.

- The input file name is provided as command line argument
- Each line of the file will contain a single number

Exercise 2

Write a program that reads a list of values from a file and print for each value how many times it appears on the screen.

- The input file name is provided as command line argument
- Each line of the file will contain a single number

Optional:

- Print the values sorted accordingly to their usage