Les fondements de la programmation
La programmation orientée objet

Exemples Ruby et Lua

```
# Classe de base
class Vehicle
  attr_reader :name
  attr_reader :speed
  attr_reader :fuel

  # Constructeur
  def initialize(name, speed, fuel)
    @name = name
    @speed = speed
    @fuel = fuel
  end

  # Méthode publique
  def to_s
    "#{@name} (#{@speed} km/h, #{@fuel} L)"
  end

  # Méthode publique
  def info
    puts "Nom : #{@name}"
    puts "Vitesse : #{@speed} km/h"
    puts "Carburant : #{@fuel} L"
  end
end

# Instanciation
car = Vehicle.new("Voiture", 120, 40)
puts car.to_s
car.info
```

```
-- Classe de base
Vehicle = {}
Vehicle.__index = Vehicle

-- Constructeur
function Vehicle.new(name, speed, fuel)
  local self = setmetatable({}, Vehicle)
  self.name = name
  self.speed = speed
  self.fuel = fuel
  return self
end

-- Méthode publique
function Vehicle:to_s()
  return self.name .. " (" .. self.speed .. " km/h, " .. self.fuel .. " L)"
end

-- Méthode publique
function Vehicle:info()
  print("Nom : " .. self.name)
  print("Vitesse : " .. self.speed .. " km/h")
  print("Carburant : " .. self.fuel .. " L")
end

-- Instanciation
car = Vehicle.new("Voiture", 120, 40)
print(car:to_s())
car:info()
```

Héritage de classes primitives

```
# Classe de base
class Vehicle
  attr_reader :name
  attr_reader :speed
  attr_reader :fuel

  # Constructeur
  def initialize(name, speed, fuel)
    @name = name
    @speed = speed
    @fuel = fuel
  end

  # Méthode publique
  def to_s
    "#{@name} (#{@speed} km/h, #{@fuel} L)"
  end

  # Méthode publique
  def info
    puts "Nom : #{@name}"
    puts "Vitesse : #{@speed} km/h"
    puts "Carburant : #{@fuel} L"
  end
end

# Classe dérivée
class Car < Vehicle
  attr_reader :doors

  # Constructeur
  def initialize(name, speed, fuel, doors)
    super(name, speed, fuel)
    @doors = doors
  end

  # Méthode publique
  def to_s
    super + " [#{@doors} portes]"
  end

  # Méthode publique
  def info
    super
    puts "Portes : #{@doors}"
  end
end

# Instanciation
car = Car.new("Voiture", 120, 40, 4)
puts car.to_s
car.info
```

```
-- Classe de base
Vehicle = {}
Vehicle.__index = Vehicle

-- Constructeur
function Vehicle.new(name, speed, fuel)
  local self = setmetatable({}, Vehicle)
  self.name = name
  self.speed = speed
  self.fuel = fuel
  return self
end

-- Méthode publique
function Vehicle:to_s()
  return self.name .. " (" .. self.speed .. " km/h, " .. self.fuel .. " L)"
end

-- Méthode publique
function Vehicle:info()
  print("Nom : " .. self.name)
  print("Vitesse : " .. self.speed .. " km/h")
  print("Carburant : " .. self.fuel .. " L")
end

-- Classe dérivée
Car = {}
Car.__index = Car
setmetatable(Car, Vehicle)

-- Constructeur
function Car.new(name, speed, fuel, doors)
  local self = setmetatable(Vehicle.new(name, speed, fuel), Car)
  self.doors = doors
  return self
end

-- Méthode publique
function Car:to_s()
  return Vehicle.to_s(self) .. " [" .. self.doors .. " portes]"
end

-- Méthode publique
function Car:info()
  Vehicle.info(self)
  print("Portes : " .. self.doors)
end

-- Instanciation
car = Car.new("Voiture", 120, 40, 4)
print(car:to_s())
car:info()
```

Héritage de classes prototypes