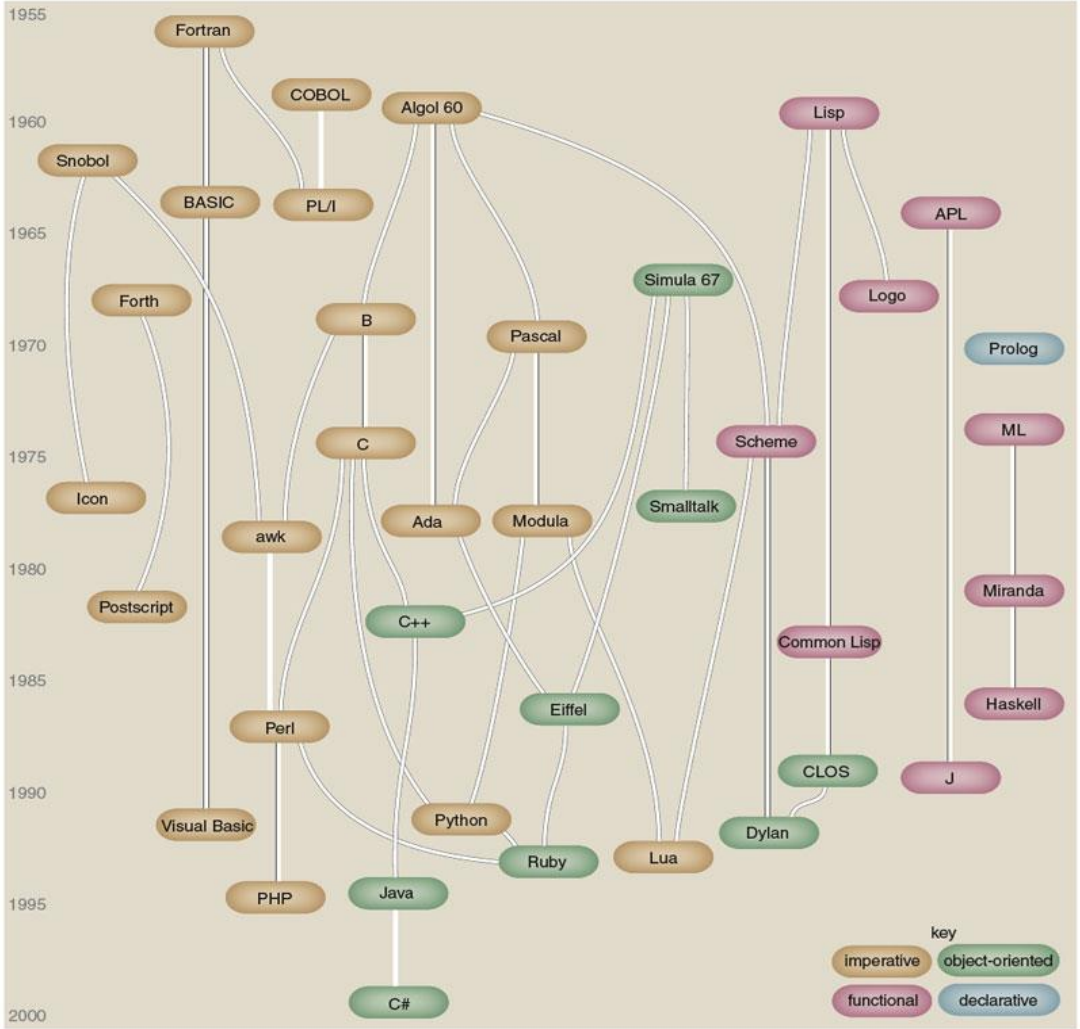
centro universitário FEI

CC8210 – NCA210 Programação Avançada I

Prof. Reinaldo A. C. Bianchi
Prof. Isaac Jesus da Silva
Prof. Danilo H. Perico

Introdução à Programação Orientada a Objetos

Genealogia das principais linguagens de programação



O que é Programação Estruturada

- Paradigma com ênfase em sequência, decisão e, iteração:
  - Procedimentos, Sub-rotinas, laços de repetição, condicionais e, estruturas em bloco.
  - Não utiliza saltos, jumps e go-to, comum nas linguagens como Assembly.
  - Ainda é muito influente pois grande parte das pessoas ainda aprende programação através dela.
  - Para a resolução de problemas simples e diretos, a programação estruturada é bastante eficiente.
- Criado no final de 1950 com a linguagem ALGOL.

Programação Orientada a Objetos

Programação Orientada a Objetos (POO) é um paradigma de programação no qual pode-se abstrair um programa como uma coleção de objetos que interagem entre si.

O conceito formal de Orientação a Objetos foi introduzido em meados de 1960, com a linguagem Simula 67 (Centro Norueguês de Computação em Oslo).

O desenvolvimento completo da POO veio com o Smalltalk 80 (1980).

Algumas Linguagens Orientadas a Objetos

Programação Estruturada versus P.O.O.

- Uma linguagem pode ser tanto estruturada como orientada a objetos:
  - C / C++
  - Python
- Nós já usamos Python como uma linguagem estruturada e como linguagem orientada a objetos...

Exemplo de Python – Paradigma Orientado a Objetos

- O objeto array em NumPy é chamado `ndarray`.
- Podemos criar um objeto ndarray usando a função `array()`:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
```

Saída:
[1 2 3 4 5]
<class 'numpy.ndarray'>

Por que Orientação a Objetos?

- Aproximação do sistema criado ao que é observado no mundo real:
  - Representação de elementos como objetos (classes)
- Reutilização do código:
  - Menos linhas
  - Melhor manutenção
- Organização

Classes e Objetos

Classes

- Representam itens do mundo real:
  - Exemplos:
    - Pessoas
    - Veículos
    - Robôs
- São Compostas de:
  - Atributos (variáveis de instância)
  - Métodos (funções-membro)

Objetos

- Todo objeto pertence a uma classe
- Representam instâncias de entidades no mundo real
- São criados a partir das classes
- São instâncias das classes
- Os objetos associam valores específicos aos atributos

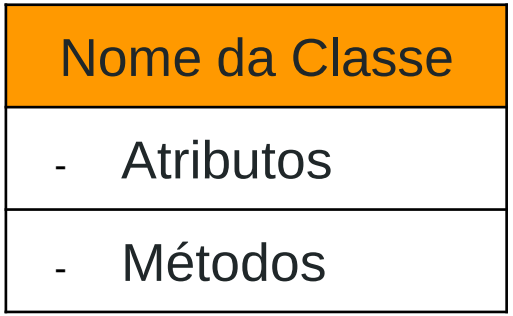
Instanciar (Informática)

- Instanciar é criar um objeto, ou seja, alocar um espaço na memória, para posteriormente poder utilizar os métodos e atributos que o objeto dispõe.

Exemplo: Diferença entre Classe e Objeto

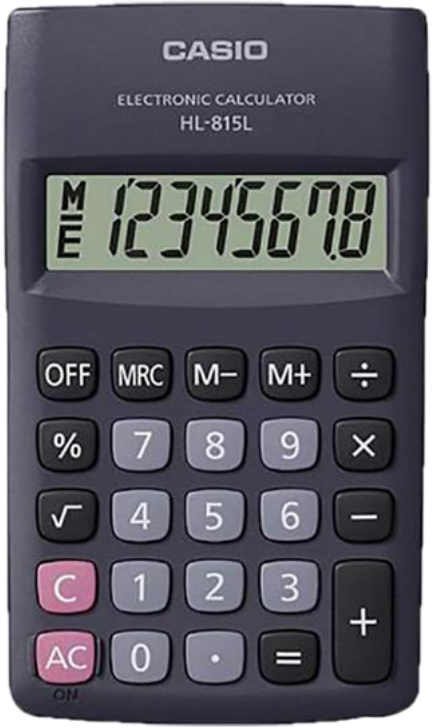
- Classe:
  - É um modelo;
  - De maneira mais prática, é como se fosse a planta de uma casa;
- Objeto:
  - É criado a partir da classe;
  - É como se fosse a própria casa construída
  - Pode-se construir várias casas a partir da mesma planta, assim como podemos instanciar vários objetos de uma só classe

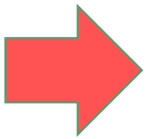
Estrutura de uma Classe



- Atributos são variáveis que armazenam informações do objeto.
- Métodos são as operações (funções) que o objeto pode realizar.

Exemplo de Classe





Calculadora
a: float b: float
Soma() Subtrai() Multiplica() Divide()

Atributos

Métodos

Atributos

- Variáveis de instância
- Atributos são variáveis em que o objeto armazena informações.

Métodos

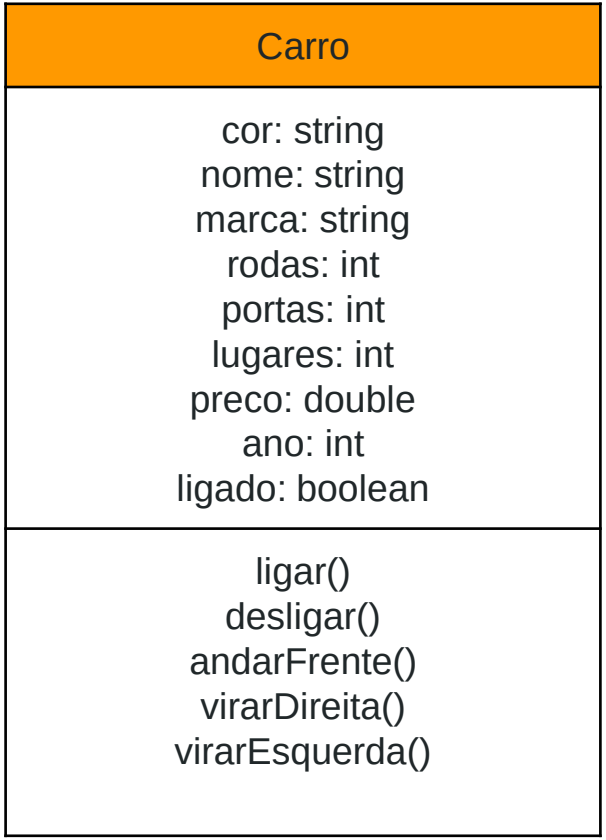
- São sequências de declarações e comandos executáveis encapsulados como se fossem um mini-programa.
- Similares:
  - sub-rotinas
  - procedimentos
  - funções

Exemplo de Classe e Objetos

Como você modelaria um carro?

Quais atributos e métodos você incluiria na sua classe Carro?

Exemplo de Classe e Objetos



Exemplo de Classe e Objetos

Carro
cor: string nome: string marca: string rodas: int portas: int lugares: int preco: double ano: int ligado: boolean
ligar() desligar() andarFrente() virarDireita() virarEsquerda()

Exemplo de Classe e Objetos

Carro
cor: string nome: string marca: string rodas: int portas: int lugares: int preco: double ano: int ligado: boolean
ligar() desligar() andarFrente() virarDireita() virarEsquerda()

Exemplo de Classe e Objetos

Cat
size: float color: string positionX: float positionY: float
moveForward() moveBackward() moveUP() moveDown()

Cat garfield;
Cat tom;
Cat felix ;
Cat scratchy;

Exemplo de Classe e Objetos

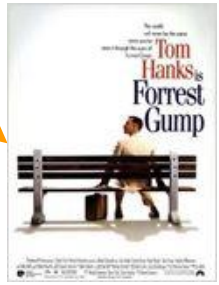
Robot
positionX: float positionY: float direction: float
moveForward() moveBackward() turnLeft() turnRight()

Robot b1;
Robot b2;
Robot b3;

Exemplo de Classe e Objetos

Movie
name: string storyline: string runtime: float
play() stop() pause()

Movie poderosoChefao;
Movie senhorDosAneis;
Movie forrestGump;

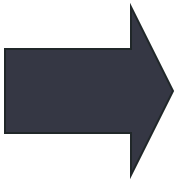


Definindo uma Classe em Python

Em Python uma classe é declarada da seguinte forma:

```
1  class NomeClasse:
2      # atributos
3
4      # métodos
```

Exemplo: Classe Robô



Robot
Name: string positionX : float positionY : float direction: short
moveForward() moveBackward() turnLeft() turnRight() stop()

Declarando uma Classe em Python

```
class Robot:
    name = ""
    positionX = 0.0
    positionY = 0.0
    direction = 0.0

    def moveForward(self):
        print ("Anda para frente")
    def moveBackward(self):
        print ("Anda para tras")
    ...
```

Atributos

Métodos

Declarando uma Classe em Python

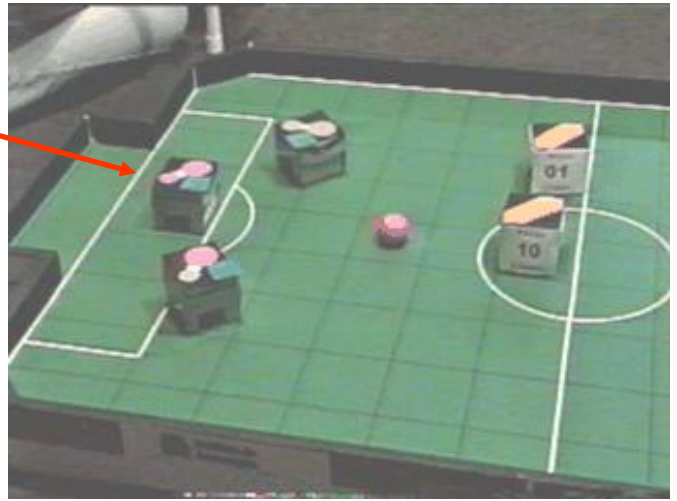
- Todo atributo deve ser inicializado:
  - Se não for inicializado, não tem como o Python saber o tipo do dado.
- Todo método tem como primeiro parâmetro uma variável que aponta para o próprio objeto:
  - O parâmetro `self` é uma referência à instância atual da classe, e é usado para acessar variáveis que pertencem à classe.
  - Ele não tem que ser nomeado `self`: você pode chamá-lo do que quiser, mas tem que ser o primeiro parâmetro de qualquer função na classe.

Exemplo de Instanciação: 3 Objetos Robots

```
C3_PELE       = Robot()
R2D_DUNGA     = Robot()
ROBOMARIO     = Robot()
```

Objetos

```
C3_PELE.moveForward()
R2D_DUNGA.moveBackward()
```



Exemplo de Saída:
Anda para frente
Anda para tras

Inicialização e Finalização de objetos

Construtores:

Métodos usados para inicializar os atributos de um objeto recém criado.

Finalizadores (Destrutores):

Métodos usados para “limpeza” ao final do uso do objeto.

Construtores

- Muitas classes precisam criar novos objetos com um estado inicial predeterminado.
- Para tanto, a classe pode definir um método especial chamado construtor:
  - São métodos com o nome `__init__()`.
  - São chamados automaticamente quando declara-se um novo objeto.
  - Podem executar comandos.
  - Não retornam valores.

Destrutores

- Muitas classes precisam realizar alguma limpeza antes do objeto ser deletado.
- Para tanto, a classe pode definir um método especial chamado de destrutor, finalizador ou destruidor:
  - São métodos com o nome `__del__()`.
  - São chamados automaticamente quando a instância está prestes a ser deletada.
  - Podem executar comandos.
  - Não retornam valores.

Construtores e Destrutores

- USE Construtores para inicializar seu objetos
- NÃO defina um tipo de retorno para os construtores ou destrutores.
- NÃO passe parâmetros para os destrutores.

Construtores e Destrutores em Python

```
class Robot:
    name = "Tiago++"
    positionX = 0.0
    positionY = 0.0
    direction = 0.0

    def __init__(self):
        print("Construindo o Robo. Aguarde :-)")

    def __del__(self):
        print ("Bye bye!!")
```

Atributos

Exemplo

```
C3_PELE     = Robot()
R2D_DUNGA = Robot()
ROBOMARIO = Robot()


C3_PELE.moveForward()
R2D_DUNGA.moveBackward()
ROBOMARIO.turnLeft()
```

Exemplo de Saída:
Construindo o Robo. Aguarde :-)
Construindo o Robo. Aguarde :-)
Construindo o Robo. Aguarde :-)
Anda para frente
Anda para tras
Girando para a esquerda
Bye bye!!
Bye bye!!
Bye bye!!

Construtores

- Muitas classes precisam criar novos objetos com um estado inicial predeterminado.
- Podem ter parâmetros padrão, para dar conta de chamadas com diferentes números de parâmetros.

Construtores e Destrutores em Python

```
class Robot:
    name = ""
    positionX = 0.0
    positionY = 0.0
    direction = 0.0

    def __init__(self, nome = "Tiago++"):
        self.name = nome
        print("Construindo o %s :-)" % self.name)

    def __del__(self):
       print ("Bye bye!!")
```

Atributos

Exemplo

```
C3_PELE     = Robot("PELE")
R2D_DUNGA = Robot("Dunga")
ROBOMARIO = Robot()


C3_PELE.moveForward()
R2D_DUNGA.moveBackward()
ROBOMARIO.turnLeft()
```

Exemplo de Saída:
Construindo o PELE :-)
Construindo o Dunga :-)
Construindo o Tiago++ :-)
Anda para frente
Anda para tras
Girando para a esquerda
Bye bye!!
Bye bye!!
Bye bye!!

Outro exemplo

```
1  class Aluno:
2      nome = ""
3      ra = 0
4
5      def __init__(self, nome="", ra=0):
6          self.nome = nome
7          self.ra = ra
8
9      def mostraAluno():
10         print("Nome: %s" % self.nome)
11         print("R.A.: %d" % self.ra)
```

Outro Exemplo

```python
#coding: utf-8
class Aluno:
    nome = ""
    ra = 0

    def __init__(self, nome="", ra=0):
        self.nome = nome
        self.ra = ra

    def mostraAluno(self):
        print("Nome: %s" % self.nome)
        print("R.A.: %d" % self.ra)
##############################################
aluno = Aluno("Danilo", 1234567890)
aluno.mostraAluno()
```

```
fei CursoFerias2016 $ python exemplo.py
Nome: Danilo
R.A.: 1234567890
fei CursoFerias2016 $
```

Conclusão

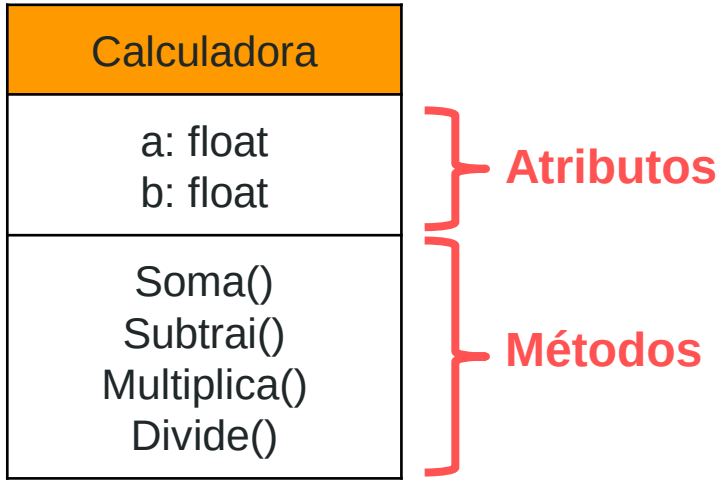
Conclusão

- Foi realizada uma breve apresentação do paradigma de programação Orientado à Objetos.
- 2 elementos:
  - Classes
  - Objetos
- Quase todas as bibliotecas usam POO...

Exercícios

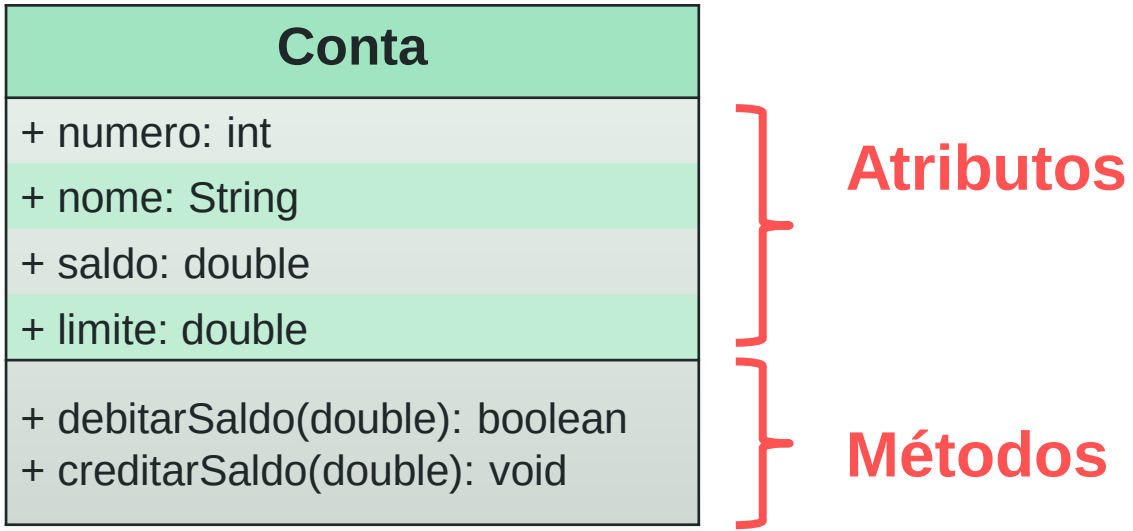
Exercício 01

- Crie uma classe chamada Calculadora conforme Diagrama de classes dado abaixo. Crie um objeto da Classe Calculadora e utilize os métodos da classe.

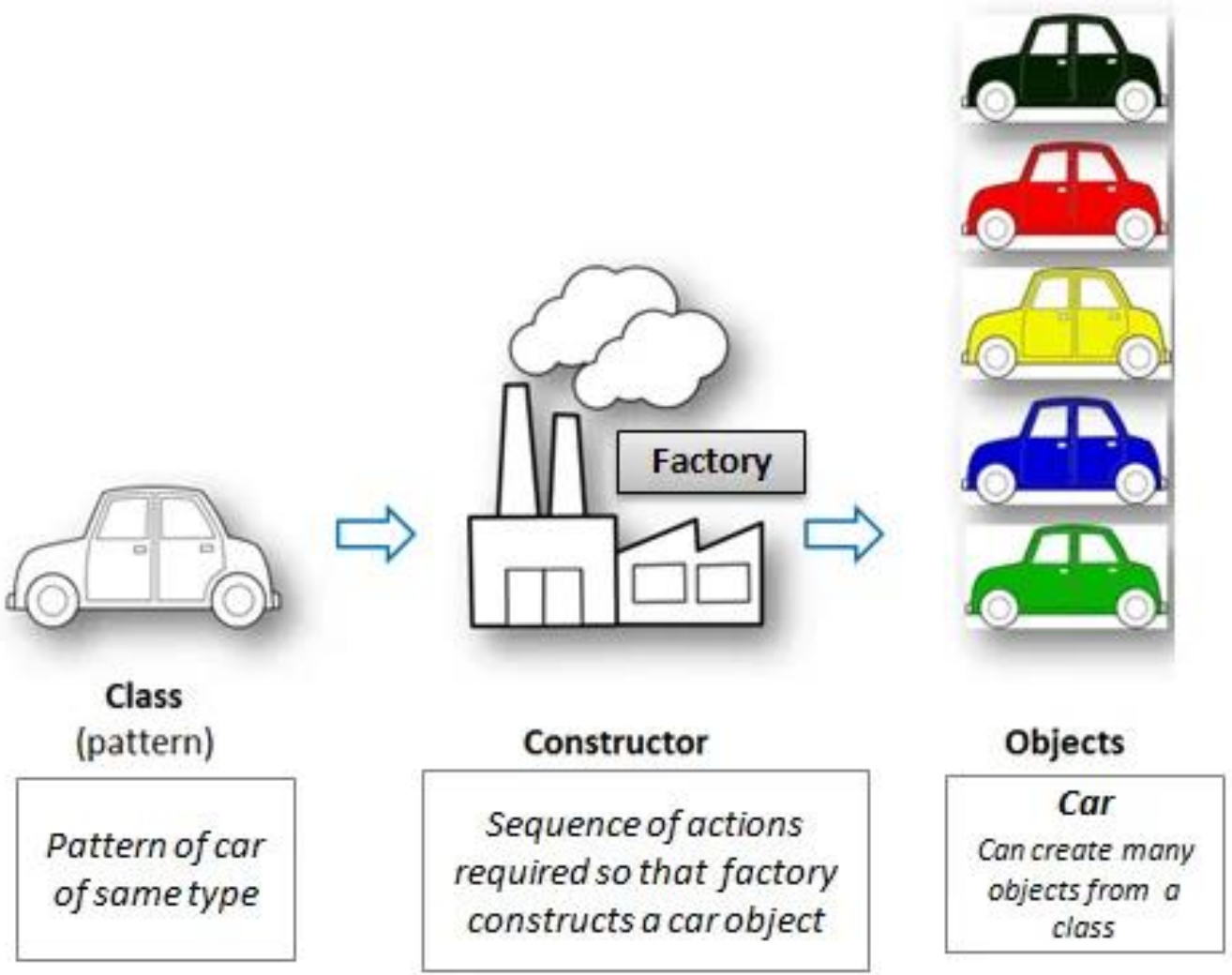


Exercício 02

- Crie uma nova classe chamada Conta. Crie mais alguns objetos da classe conta. Acesse os atributos dos objetos que você criou, e modifique os valores dos atributos. Use o método creditarSaldo(). Use o método debitarSaldo().



Fim


Factory
Class
(pattern)
Pattern of car of same type
Constructor
Sequence of actions required so that factory constructs a car object
Objects
Car
Can create many objects from a class