METODOLOGIA E LINGUAGEM DE PROGRAMAÇÃO AVANÇADA

Revisão Java - POO

Ciências da Computação

Material Baseado nas Notas do Prof. Ms. Daniel Brandão

Prof. Esp. Renato Leite

renato.leite@unipe.br

OBJETIVOS

- Revisar os conceitos de POO

- Revisar Java

REVISÃO JAVA OO

- Paradigma que procura compor modelos de forma mais próxima às interações existentes no mundo real, cujas primeiras propostas datam da década de 60;

- A POO define que objetos se comunicam através da troca de mensagens para promover a troca de serviços;

REVISÃO JAVA OO





ANALOGIA

- Problema: Ter e usar um carro.
- Mas antes disso, o que precisa?
- Alguém tem que projetar...
- Engenheiros elaboram o projeto

O QUE PROJETAR?

- No Projeto temos:
- Definição partes que fazem um carro funcionar!
- Quais são elas?

O QUE PROJETAR?

- Freios, suspensão, motor, carroceria…

- Funcionalidades e características!

ANALOGIA

- Podemos dirigir o PROJETO de um carro?

ANALOGIA

- Para guiar precisamos contruir o carro seguindo o projeto pedido!

ANALOGIA

- Estar construído ainda não é o suficiente, como se di

RESUMINDO

- O projeto define o que o carro possui, o que ele é;
- O carro é a realização do projeto construído;
- Os pedais e as funcionalidades do carro são os métodos;
- As características são os atributos.

RESUMINDO

- Projeto = Classe;
- Carro = Objeto;
- Pedais e Funcionalidades = Métodos;
- Cor, quantidade de portas, modelo de rodas = Atributos
- A classe é a entidade responsável por definir os atributos (características) e os métodos (serviços) que serão oferecidos.

RESUMINDO

- Para executar tarefa (rotinas) em um programa é necessário um **método**;
- O método descreve os mecanismos que realmente realizam suas tarefas;
- **Ocultando** de seu usuário as tarefas complexas que este realiza...
- Assim como...
  - Os pedais do nosso carro!

OUTRO EXEMPL

- Primeiro criamos uma unidade de programa chamada classe para abrigar esses tais métodos;

- Você pode fornecer um ou mais métodos que são projetados para realizar as tarefas da classe;

- Por Exemplo....

- Uma classe **Conta** pode abrigar os métodos (tarefas) depositar, debitar, "solicitar saldo atual''...

CONTA

- Um objeto tem **atributos** que são portados consigo quando este é utilizado em um programa;

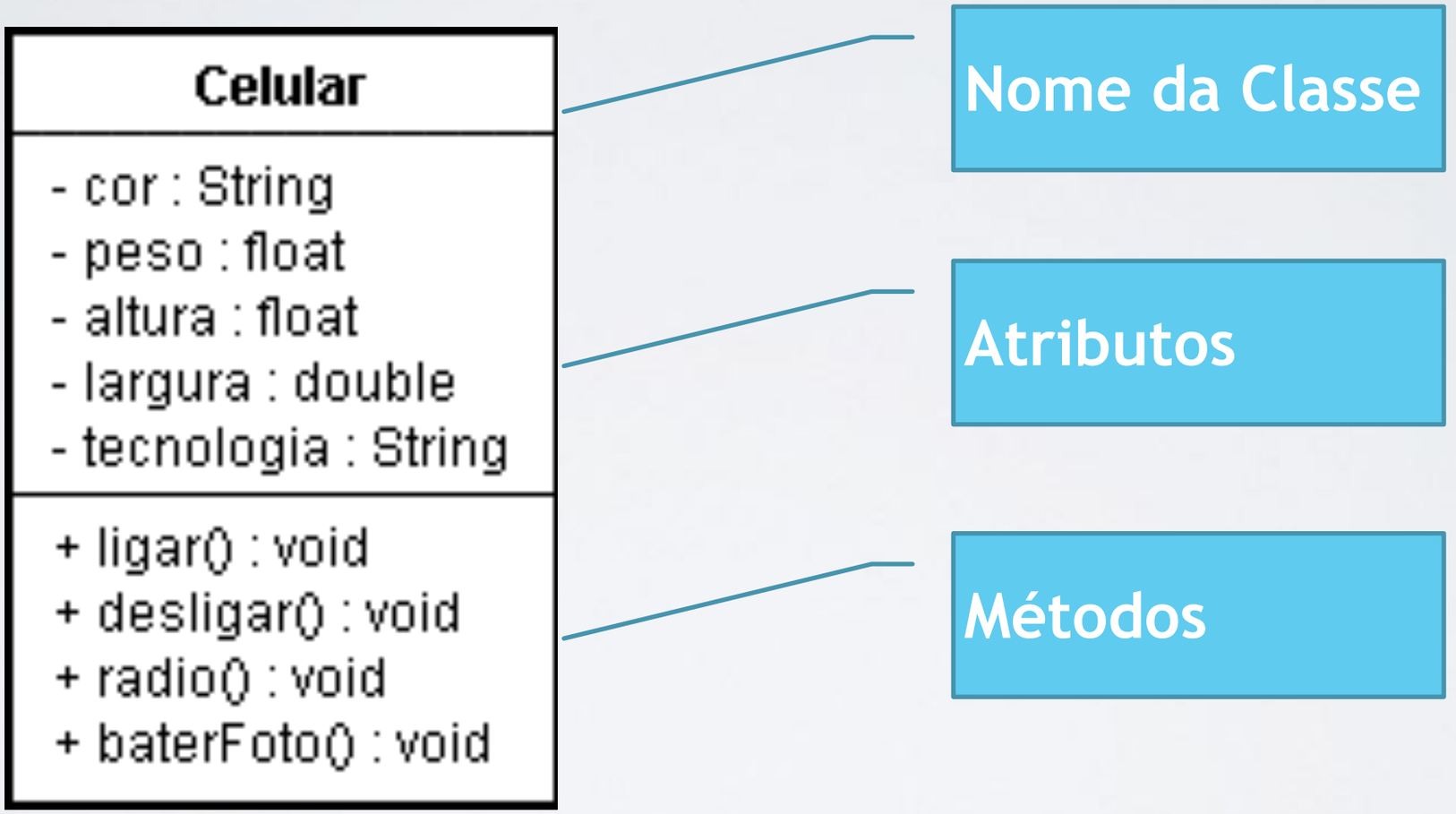
- Por Exemplo:
  - Uma classe conta possui:
  - Número, Saldo, Tipo…
- Esses atributos são especificados pelas chamadas variáveis de instância.

CONTA

- Assim como não podemos dirigir um projeto de um carro, não podemos executar métodos de uma classe;

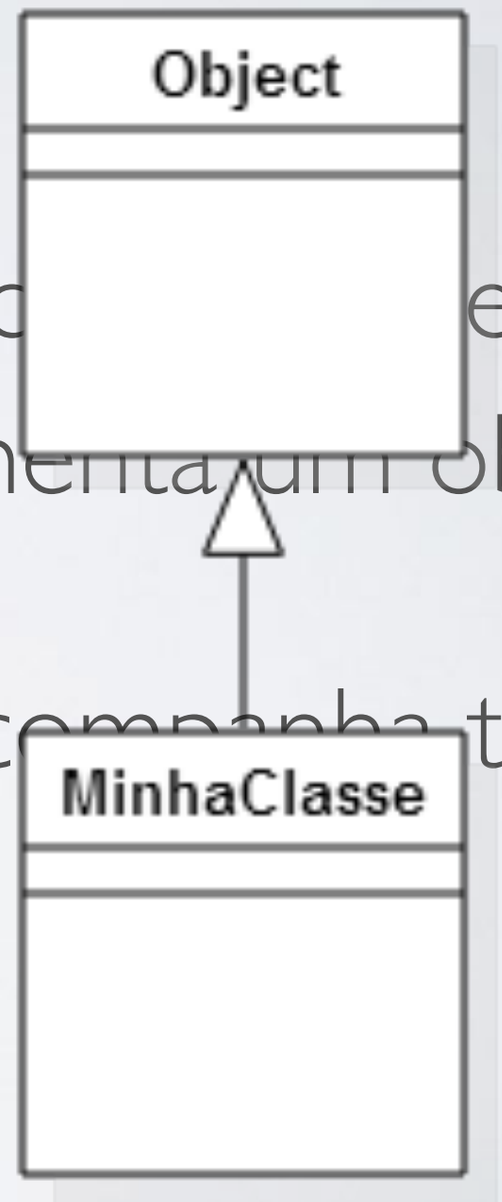
- Assim como alguém tem que construir esse carro a partir do seu projeto, você deve construir um objeto de uma classe antes de fazer um programa realizar as tarefas que a classe descreve como fazer;

CLASSE


Celular
- cor : String
- peso : float
- altura : float
- largura : double
- tecnologia : String
+ ligar() : void
+ desligar() : void
+ radio() : void
+ baterFoto() : void
Nome da Classe
Atributos
Métodos

CLASSE

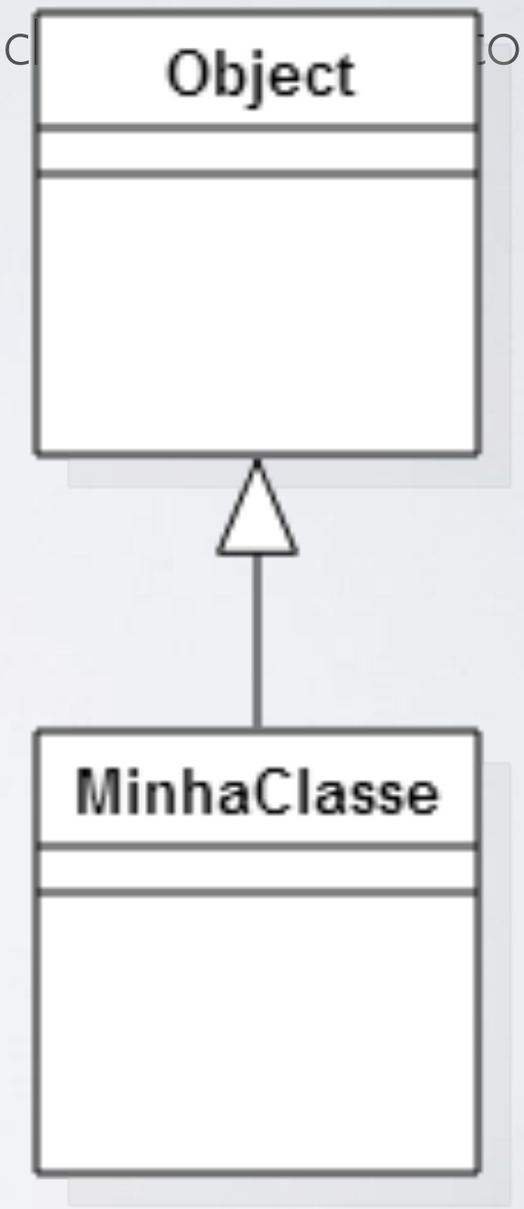
- Toda e qualquer classe criada herd e "Object", ou seja, define e implementa um objeto;



- É a herança mais simples e que acompanha todos as classes!

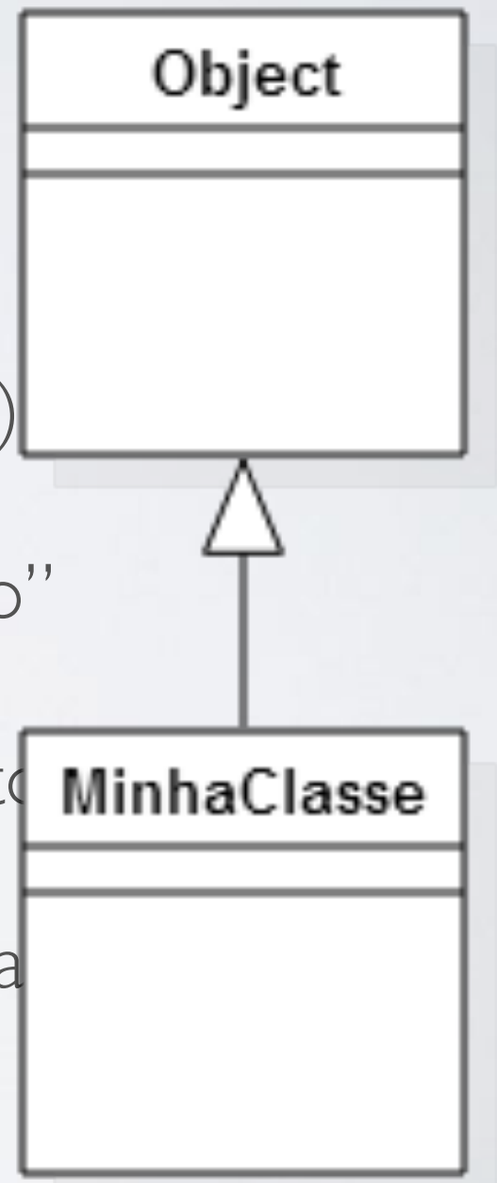
O QUE SE HERDA?

- A classe Object define um conjunto de métodos padrão, inc tor padrão;
- Padrão de inicialização dos atributos:
  - boolean: false
  - char: ‘ ’
  - float: 0.0
  - double: 0.0
  - int: 0
  - Referência: null



O QUE SE HERDA?

- Possui 11 métodos:
  - clone: Realiza uma cópia superficial do objeto.
  - equals: Compara objetos (retorna true ou false)
  - finalize: Prepara o objeto para o "coletor de lixo"
  - getClass: Retorna informações do tipo do objeto
  - hashCode: Retorna um valor hash (código) para
  - toString: Retorna a versão String de um objeto



COMO DEFINIR UMA CLASSE

- Antes de definir uma classe é necessário determinar quais serão os "modificadores de acesso", podendo ser:
  - public: a classe é de todo o projeto
  - private: apenas a classe a qual pertence
  - protected: dentro do pacote e em subclasses
  - default/package: acesso dentro do pacote

MODIFICADORES DE ACESSO

- Promovem a definição do encapsulamento, que pode ocorrer:
  - Na classe;
  - No atributo;
  - No construtor;
  - No método.

EXERCÍCIOS

- Exercício 1
  - Crie uma classe Conta, contendo um método (nomei-o de exibirMensagem) para exibir a mensagem: "Bem vindo à Conta!"

EXERCÍCIOS

```java
package aulaRevisao;

public class Conta {

    static void exibirMensagem() {
        System.out.println("Bem-vindo à Conta");
    }

}
```

EXERCÍCIOS

- Exercício 1
  - Crie uma classe Conta, contendo um método (nomei-o de exibirMensagem) para exibir a mensagem: "Bem vindo à Conta!"

EXERCÍCIOS

- Como chamar os métodos?

- Precisamos instanciar um objeto da classe definida!

- Mas… Como se instancia?

  - R.: Chamando o construtor da classe.
    Publicamente conhecido como método new()

EXERCÍCIOS

- Exercício 2
  - Defina uma classe chamada ContaTeste que contenha um método main, instancie (construa) o objeto Conta e realize uma camada ao método exibirMensagem().

EXERCÍCIOS

```
1  package aulaRevisao;
2
3  public class ContaTeste {
4
5      public static void main(String[] args) {
6          Conta ct = new Conta();
7          ct.exibirMensagem();
8
9      }
10 }
```

EXERCÍCIOS

- Temos uma classe Conta instanciada, mas...
- E seus atributos? Saldo, número...

EXERCÍCIOS

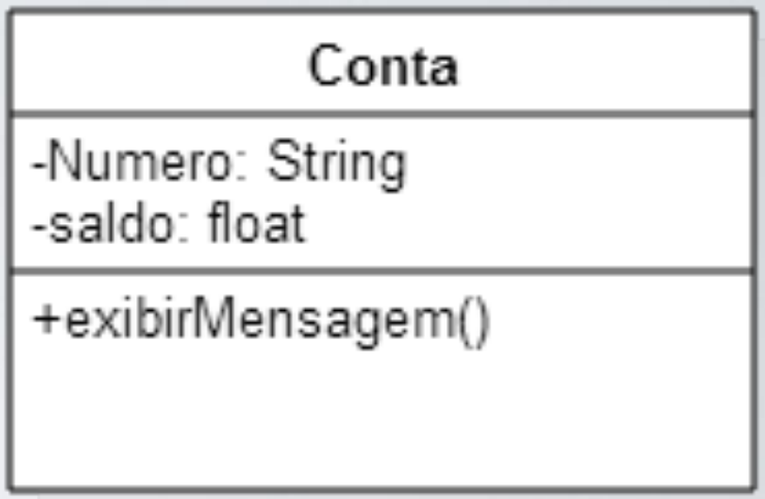
- Exercício 3

- Defina os atributos numero (String) e saldo (float) para a classe Conta.

EXERCÍCIOS

```java
package aulaRevisao;

public class Conta {

    String numero;
    float saldo;

    static void exibirMensagem() {
        System.out.println("Bem-vindo à Conta");
    }



}
```

CONTA



- Temos uma classe Conta:

- Acabamos de definir que os atributos da classe serão "private", ou seja, apenas a classe possui autorização para acesso;

- Se algum outro objeto necessitar visualizar o "estado" desse objeto ?!

CONTA

- Getters and Setters
  - Definiremos métodos com acessibilidade "public" para retornar (**get**) e/ou alterar (**set**) o valor dos atributos, conforme as regras de negócio;
- A classe terá acessibilidade "**public**", dessa forma, outras classe poderão enxergar a classe Conta.
- A POO chama isso de "Encapsulamento".

EXERCÍCIOS

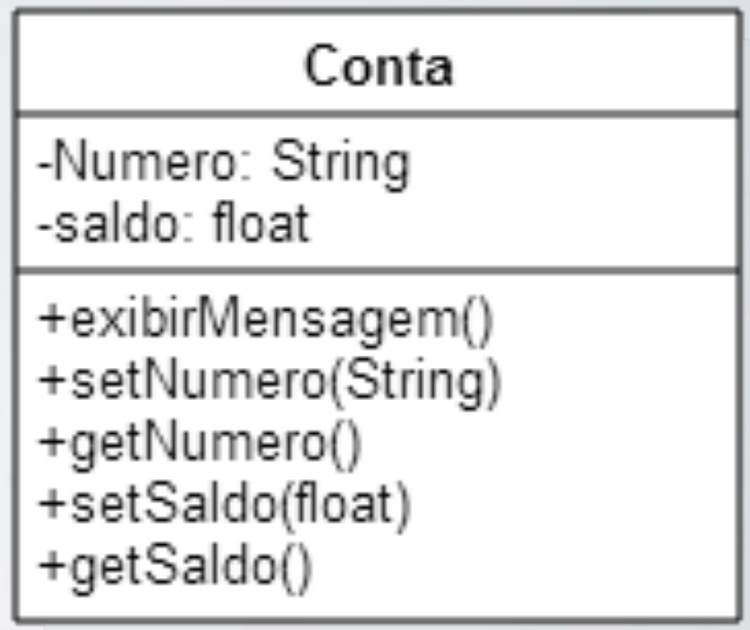
- Exercício 4
  - Defina os métodos públicos **get** e **set** para cada atributo da casse Conta.

EXERCÍCIOS

```
3  public class Conta {
4
5      String numero;
6      float saldo;
7
8      static void exibirMensagem() {
9          System.out.println("Bem-vindo à Conta");
10     }
11
12     public String getNumero() {
13         return numero;
14     }
15
16     public void setNumero(String numero) {
17         this.numero = numero;
18     }
19
20     public float getSaldo() {
21         return saldo;
22     }
23
24     public void setSaldo(float saldo) {
25         this.saldo = saldo;
26     }
```

CONTA

- Agora temos uma classe Conta:



- Vamos utilizar os recursos que a classe nos oferece!

EXERCÍCIOS

- Exercício 5

  - Na classe ContaTeste, modifique o valor de seus atributos, e imprima os novos valores destes.

EXERCÍCIOS

```
1  package aulaRevisao;
2
3  public class ContaTeste {
4
5      public static void main(String[] args) {
6          Conta ct = new Conta();
7          ct.exibirMensagem();
8
9          ct.setNumero("12345");
10
11         ct.setSaldo(200);
12
13     }
14 }
```

POO

- A POO defende que cada objeto é responsável por armazenar os valores de seus atributos (estado do objeto);
- Reflexão:
  - Não seria mais interessante delegar ao objeto responsabilidade de imprimir o seu "estado" ?!

POO

- Toda classe herda de Object, a classe Object tem um método "toString()" que converte o "estado" do objeto para uma versão textual (String);

- Então… Conta também tem um método toString() !!

- Problema resolvido ?! Vamos testar …

SOBRECARGA/ POLIMORFISMO

- O método "toString()" da classe Object converte o "estado" do objeto para uma versão textual (String), porém, com o seguinte formato:

- pacote.classe@representação_hexa_hashCode

- Não seria mais interessante exibir o número da conta?

- Seria possível reescrever o método "toString()" na classe Conta ?!

SOBRECARGA/ POLIMORFISMO

- Resposta:
- Sim ! É possível reescrever o método "toString()" na classe Conta;
- A POO chama essa sobrescrita ou sobrecarga de métodos de "polimorfismo".

EXERCÍCIOS

- Exercício 6

- Sobrescreva o método toString() da classe Conta para imprimir o número da conta seguido do saldo.

EXERCÍCIOS

```
27
28      public String toString(){
29          return "Número: " + this.getNumero() +
30                  "\nSaldo: R$ " + this.getSaldo();
31
32      }
33
```

POO

- Agora vamos "turbinar" nossa classe Conta:
- Quais serviços esse objeto deve prover?
  - Creditar
  - Debitar

- Exercício 7
    - Implemente os métodos Creditar e Debitar na classe Conta.

BIBLIOGRAFIA

- DEITEL, H.; DEITEL, P. **Java: como programar. 6. ed**. São Paulo: Pearson Prentice Hall, 2005. (Disponível na Biblioteca Digital Pearson) - https://centrodeinformacao.unipe.br/bibliotecas-digitais/livros

- HORSTMANN, C. S.; CORNELL, G. **Core java: recursos avançados**. São Paulo: Makron Books, 2000, v. 2.

- SIERRA, K.; BATES, B. **Head first java. 2. ed.** Cambridge: O´Reilly & Assoc., 2005.

DÚVIDAS?

- Se apresentem: qual seu nome? Quais suas expectativas pra disciplina?