Programação Orientada a Objetos

Módulo 1

Vaux Gomes [1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará
Campus Jaguaribe

7 de Agosto de 2022

Sumário

Instanciação

Como já vimos, instanciar significa alocar/separar memória para um determinado objeto. Para isso usamos o **new**.

- Aos tipos primitivos, por padrão, é atribuído o valor zero.
- Aos de tipo String é atribuído **null**.

Para alguns atributos do objeto a memória pode não ser totalmente alocada.

- Aos tributos do tipo classe, é atribuído **null**.
- Estes atributos são, na verdade, ponteiros que apontarão para alguma posição de após o atributo ser inicializado.
- Assim, precisamos alocar todos os objetos do tipo classe que quisermos/precisarmos usar.

Instanciação

Juntamente com o **new** usaremos os **construtores** dos objetos para instanciá-los.

- Toda classe possui ao menos um construtor com o qual é possível instanciar objetos
  - Chamaremos este construtor de **default**.
  - Este construtor não exite nenhum parâmetro

Segue um exemplo de instanciação da classe Carro pelo construtor default:

```
/*
 * Sintaxe
 * <nome da classe> <nome da variavel> =
       new <construdor>(<valores dos parametros>);
 */
Carro carro = new Carro();
```

Construtores

Parte I

- Os construtores são métodos que levam o mesmo nome da classe e são usados para instanciar os objetos
- Construtores não retornam valores, mas podem receber argumentos.
- Caso **não declarado** toda classe já conta com um construtor **default** (padrão) que não recebe argumentos.
  - Caso algum construtor seja declarado o construtor default deixa de existir
- É possível re-declarar o construtor default se for necessário

Construtores

É comum usar construtores para forçar o preenchimento de alguns atributos da classe. Além disso, podemos usar construtores para instanciar os atributos do tipo classe de maneira padronizada.

Construtores

Parte I: Exemplos

Segue exemplos de construtores da nossa classe Carro

```
// Re-escrevendo o construtor default: Modo 1
public Carro() {
    // Faz nada
}

// Re-escrevendo o construtor default: Modo 2
public Carro() {
    this.modelo = "Fusca";
    this.ano = 1967;
    this.combustivel = 0.0;
    this.kms = 3141592;
}
```

Construtores

Parte I: Exemplos

```java
// Construtor com dois parametros
public Carro(String modelo, int ano) {
    this.modelo = modelo;
    this.ano = ano;
    this.combustivel = 0.0;
    this.kms = 0;
}

// Construtor completo
public Carro(String modelo, int ano, float combustivel, int kms) {
    // ...
}
```

Atenção

Não é possível existir dois construtores com a mesma sequência de parâmetros. Pelo mesmo motivo não é possível existir dois métodos de mesmo nome com a mesma sequência de parâmetros.

Construtores

Parte II

Os construtores podem invocar outros construtores, se assim for necessário. Podemos usar este artifício juntamente com restrições de acesso para melhor organizar o nosso código.

```
// Construtor completo (Protegido para pacote)
Carro(String modelo, int ano, float combustivel, int kms) {
    this.modelo = modelo;
    this.ano = ano;
    this.combustivel = combustivel;
    this.kms = kms;
}

// Construtor com dois parametros usando o completo
public Carro(String modelo, int ano) {
    this(modelo, ano, 0.0, 0);
}
```

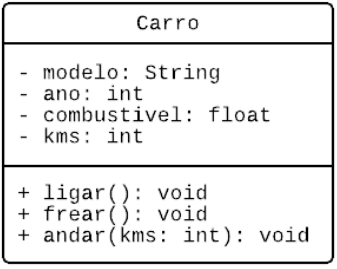
Atenção

A chamada de um construtor a partir de outro deve ocorrer na primeira linha do bloco de código.

Exercício

Parte I

1. Delete o projeto criado na aula passada (removendo todo o conteúdo do disco)
2. Crie um novo projeto Java:
   - `File > New > Java Project`
   - `POOJBE201901`
3. Adicione um novo **package**
   - `File > New > Package`
   - `ifce.jbe.poo.carros`
4. Adicione uma classe chamada **Carro** ao package criado:
   - `Botão direito no package > New > Class`
5. Escreva o código descrito pelo diagrama ao lado.



Exercício

Parte II

Reescreva a função **toString()** como segue:

```
@Override
public String toString() {
  return String.format(
     "Carro:\n" +
     "Modelo:%s\n" +
     "Ano:%d\n" +
     "Combustivel:%f\n" +
     "Kms:%d", modelo, ano, combustivel, kms);
}
```

Esta função facilita no momento de escrever os atributos de nossos objetos. Vamos utilizar este artifício sempre que quisermos observar o **estado** dos nossos objetos.

Exercício

Parte III

1. Crie uma nova classe chamada Main no nosso package
   - `Botão direito no package > New > Class`
   - Marque a opção que escreve o método main
2. Instancie um objeto do tipo Carro usando o construtor default
3. Use `System.out.println()` para imprimir o nosso objeto
4. Rode o programa.
   - O que aconteceu? Explique o valor de cada variável.

Exercício

Parte IV: Quebrando o código

1. Na classe Carro: Implemente um construtor que receba os valores iniciais de **modelo** e **ano** e inicialize os valores de **combustivel** e **kms** para zero
2. Rode novamente o programa
   - Porque o programa quebrou?
3. Implemente o construtor default para “Fazer nada"
4. Rode novamente o programa
   - Voltou a funcionar!
5. Implemente outro construtor igual ao default e inicialize
   - modelo $\rightarrow$ Fusca
   - ano $\rightarrow$ zero
   - combustivel $\rightarrow$ zero
   - kms $\rightarrow$ 1000
6. O que aconteceu e por quê? Como consertar? Conserte!

Exercício

Parte IV: Quebrando o código

7. Implemente um construtor que recebe primeiro **ano** e depois **kms**. Rode o programa
8. Implemente um construtor que recebe primeiro **kms** e depois **ano**. Rode o programa. O que aconteceu?
9. Crie um segundo carro e imprima na tela. Você entende que objetos diferentes estão alocados em partes diferentes da memória?