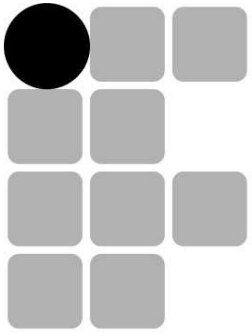


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO SUL
Câmpus Feliz

Modificadores de acesso e Construtores de classe

Prof. Moser Fagundes

Programação II

Sumário

- Revisão
  - Convenções da orientação a objetos
  - Pacotes
  - Importando classes de pacotes
- Atributos privados
- Geração de métodos get e set
- Construtores
- Exercícios

Pilares da OO

- Abstração
- **Encapsulamento**
- Herança
- Polimorfismo

Convenções

- **Nomes de classes** são normalmente **substantivos**. Por convenção, o nome de cada palavra existente no nome da classe se escreve começando por **maiúsculo**.

Convenções

- **Atributos** costumam ser **substantivos** que descrevem características enquanto **métodos** normalmente são **verbos** representando comportamento.

- O nome dos atributos, métodos ou variáveis se escrevem com cada palavra começando por **maiúsculo**, **exceto a primeira letra**.

Convenções

- **Indente** o seu código!

- **NÃO** use acentos, cedilha e caracteres especiais em nomes de projetos, classes, atributos, variáveis, pacotes, etc.

Pacotes

- Use letras **minúsculas** para nomes de pacote.
- Você pode criar **pacotes dentro de pacotes** para melhorar a estruturação do seu programa.

- **Exemplo**:
  - Crie o pacote **`lista2`**
  - Crie o subpacote **`lista2.exercicio1`**
  - Crie o subpacote **`lista2.exercicio2`**
  - E assim sucessivamente...

Pacotes

- Note que o **nome do pacote** ao qual a classe pertence é a primeira coisa que aparece no arquivo da classe.

```
package nomeDoPacote;
```

- Veja que os pacotes estão estruturados em pastas no seu **workspace**.

Pacotes

- **Exemplo** de estruturação em pacotes e classes (**pacotes** em vermelho e **classes** em azul):

```
jogo
jogo.Futebol
jogo.Equipe
jogo.pessoas
jogo.pessoas.Jogador
jogo.pessoas.Juiz
jogo.pessoas.Treinador
jogo.equipamentos
jogo.equipamentos.Campo
jogo.equipamentos.Bola
```

Importar classes

- Quando uma classe está no **mesmo pacote** de outra classe, **NÃO precisamos importar** uma das classes para ser usada na outra.
- **Exemplo**: se quisermos criar um objeto da classe **`Equipe`** dentro de um objeto da classe **`Futebol`**, não precisamos importar equipe.
  - Nos exemplos anteriores, cite outros casos nos quais não necessitamos fazer importação.

Importar classes

- Quando uma classe **NÃO** está no mesmo **pacote** de outra classe, **precisamos importar** uma das classes para ser usada na outra.
- Para importar, usamos a palavra **`import`**.
  - Exemplo: classe **`Scanner`**

- **Exemplo**: se quisermos criar um objeto da classe **`Jogador`** dentro de um objeto da classe **`Equipe`**, precisamos importar **`Jogador`**.
  - **Crie um atributo `Jogador` na classe `Equipe`**

Atributos privados

- Um atributo **`private`** significa que ele só poderá ser acessado **dentro** do objeto no qual ele está declarado.
  - O atributo **`private`** não poderá ser acessado por nenhum outro objeto (sem visibilidade externa).

- **Exemplo**:
  Considere uma classe **Pessoa** com um atributo **idade** privado e um **nome** público: **não** é possível acessar a idade de fora da classe, apenas o atributo nome será acessível.

Atributos privados

- **Exemplo**: completar a equipe de futebol e o jogo criando **atributos privados**.
  - Jogador, Treinador e Juiz possuem nome
  - Bola tem marca
  - Campo tem largura e comprimento
  - Equipe tem jogadores e treinador
  - Futebol tem equipes, juiz, campo e bola

Métodos get e set

- Quando os atributos são **privados**, precisamos criar métodos para **ler** e **modificar** os valores destes atributos:
  - **`get`** (para leitura)
  - **`set`** (para modificação)

- **Nomenclatura**:
  - se o atributo se chama **`idade`**, os métodos se chamam **`getIdade`** e **`setIdade`**.
  - se o atributo se chama **`nome`**, os métodos se chamam **`getNome`** e **`setNome`**

Métodos get e set

Os métodos **`set`** limitam que tipo de conteúdo que pode ser colocado em uma variável.

- **Exemplo**: Um sistema onde o valor do atributo idade poderá conter apenas valores maiores ou iguais a 18. Neste sistema, colocamos a verificação apenas no setIdade evitando verificações espalhadas pelo código, difíceis de manter e modificar.

Construtores

- Um **construtor** é um **método especial** chamado para **criar** e **inicializar** um objeto.

- Não possui **nenhum tipo de retorno declarado** e tem o **mesmo nome da classe**.
  - **NÃO** podemos usar **return** em construtores!

- Podemos definir mais de um construtor para uma mesma classe, desde que eles possuam **assinaturas diferentes**.

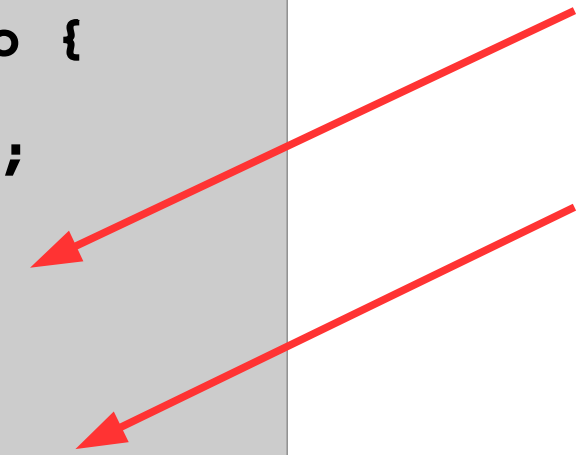
Exemplo de construtor

```
public class Cachorro {

   public String nome;

   public Cachorro(){
      nome = "Lessie";
   }

   public Cachorro(String n){
      nome = n;
   }
}
```



O construtor tem exatamente o mesmo nome da classe.

Pode conter parâmetros ou não. Neste exemplo, o primeiro construtor não tem parâmetro e o segundo possui um parâmetro do tipo String.

```
public class TesteCachorro {
   public static void main(String[] args){
      Cachorro c1 = new Cachorro();
      Cachorro c2 = new Cachorro("Rex");
   }
}
```

Construtores

Em Java, se **nenhum construtor** for explicitamente definido, será criado um construtor *default* sem argumentos.

Até o momento, é o que temos feito...

Vantagem de construtor

```
public class Cachorro {
   public String nome;
   public int idade;

   public Cachorro(String n, int i){
      nome = n;
      idade = i;
   }
}
```

```
Cachorro c1 = new Cachorro("Lessie",8);
Cachorro c2 = new Cachorro("Rex",5);
```

Isso...

..no lugar disso

```
Cachorro c1 = new Cachorro();
c1.nome = "Lessie";
c1.idade = 8;
Cachorro c2 = new Cachorro();
c2.nome = "Rex";
c2.idade = 5;
```

Exercício

Crie uma classe **Computador** com 3 atributos.

Crie **dois** construtores para esta classe: um construtor sem parâmetros e outro com 3 parâmetros (um para cada atributo).

Crie **dois** objetos, um com cada construtor.

Imprima o valor dos atributos usando métodos get.

Exercício

Crie **construtores** para as classes **`Futebol`**, **`Equipe`**, **`Jogador`**, **`Juiz`**, **`Treinador`**, **`Campo`** e **`Bola`** do exemplo do jogo de futebol.

Para concluir a aula...

- Os métodos **`get`** e **`set`** podem ser gerados automaticamente.
- Podemos criar atributos que são objetos de classes já existentes. Isso se chama associação de classes. Os diferentes tipos de associação serão estudados na disciplina de análise no próximo semestre.

Exercícios

Lista de exercícios no Moodle.