02. Super e reescrita de métodos

Date	@25/09/2022
Categoria	Java
Curso	Java Polimorfismo: entenda herança e interfaces

Tópicos

- Herança no diagrama de classes
- Reescrita de métodos
- Visibilidade
- Sobrescrita
- Super com métodos
- Dominando herança
- Private x Protected
- Mãos na massa: Reescrita de método
- Para saber mais: Sobrecarga
- O que aprendemos?
- Arquivos do projeto atual

Dominando herança

Sobre herança em Java, julgue as seguintes afirmativas:

1) Uma classe pode ter várias filhas, mas apenas uma mãe. (Verdadeiro)

2) A partir de uma instância de uma classe filha, podemos chamar qualquer método público que tenha sido declarado na classe mãe.(Falso)

3) Na classe filha, podemos escolher o que herdar da classe mãe.(Verdadeiro)

4) No exemplo abaixo, `Cachorro` também herda tudo da classe `Animal`: (Verdadeiro)

```
class Animal {
    // atributos e métodos
}

class Mamifero extends Animal {
    // atributos e métodos
}

class Cachorro extends Mamifero {
    // atributos e métodos
}
```

Private x Protected

Só a própria classe enxerga atributos/métodos `private`, enquanto `protected` é visto pela própria classe mais as classes filhas.

Para saber mais: Sobrecarga

Existe um outro conceito nas linguagens OO que se chama de **sobrecarga** que é muito mais simples do que a *sobrescrita* e nem dependente da herança. Por exemplo, na nossa classe `Gerente`, imagine um outro novo método `autentica` que recebe além da `senha` também o `login`:

```
public class GerenteextendsFuncionario {
  private int senha;

  public void setSenha(int senha) {
        this.senha = senha;
  }
```

```
  public boolean autentica(int senha) {
    if(this.senha == senha) {
      return true;
    }else {
      return false;
    }
  }

  //novo método, recebendo dois params
  public boolean autentica(String login,int senha) {
        //implementacao omitida
  }

    //outros métodos omitidos
}
```

Repare que criamos uma nova versão do método `autentica`. Agora temos dois métodos `autentica` na mesma classe que variam na quantidade ou tipos de parâmetros. Isso se chama **sobrecarga** de métodos. A sobrecarga não leva em conta a visibilidade ou retorno do método, apenas os parâmetros e não depende da herança.

O que aprendemos?

Nessa aula já entramos mais a fundo na herança. Aprendemos:

- que classe mãe é chamada de super ou base class
- que a classe filha também é chamada de sub class
- como aumentar a visibilidade de um membro (atributo, método) através do `protected`
- como acessar ou chamar um membro (atributo, método) através do `super`
- como redefinir um método através da sobrescrita