PROGRAMAÇÃO DE COMPUTADORES

Técnico Integrado em
INFORMÁTICA

Professor Jefferson Chaves
jefferson.chaves@ifc-araquari.edu.br

OBJETIVOS DA AULA

- Dizer o que é herança e quando utilizá-la;
- Reutilizar código escrito anteriormente;
- Criar classes filhas e reescrever métodos;
- Compreender polimorfismo.

PROGRAMAÇÃO DE COMPUTADORES

REPETINDO CÓDIGO?

Jefferson de Oliveira Chaves

- Em nosso banco, além de um funcionário comum, há também outros cargos, como os gerentes.
- Os gerentes guardam a mesma informação que um funcionário comum, mas:
  - possuem outras informações;
  - tem funcionalidades um pouco diferentes.

- Um gerente no nosso banco possui também uma senha numérica que permite o acesso ao sistema interno do banco, além do número de funcionários que ele gerencia.

```
class Gerente {
   $nome;
   $cpf;
   $salario;
   $senha;
   $numeroDeFuncionariosGerenciados;

   public function autentica(int $senha) {
        if ($this->senha == $senha) {
            echo "Acesso Permitido!";
            return true;
        } else {
           echo "Acesso Negado!";
           return false;
        }
   }

   // outros métodos
}
```

- **Precisamos mesmo de outra classe?**

- **Poderíamos ter deixado a classe Funcionario mais genérica, mantendo nela senha de acesso, e o número de funcionários gerenciados. Caso o funcionário não fosse um gerente, deixaríamos estes atributos vazios.**

- **Essa é uma possibilidade, porém podemos começar a ter muito atributos opcionais, e a classe ficaria estranha. E em relação aos métodos? A classe Gerente tem o método autentica, que não faz sentido existir em um funcionário que não é gerente.**

- Existe um jeito, de relacionarmos uma classe de tal maneira que uma delas herda tudo que a outra tem;
- Isto é uma relação de classe mãe e classe filha;
- No nosso caso, gostaríamos de fazer com que o Gerente tivesse tudo que um Funcionario tem, gostaríamos que ela fosse uma extensão de Funcionario;
- Fazemos isto através da palavra chave extends;

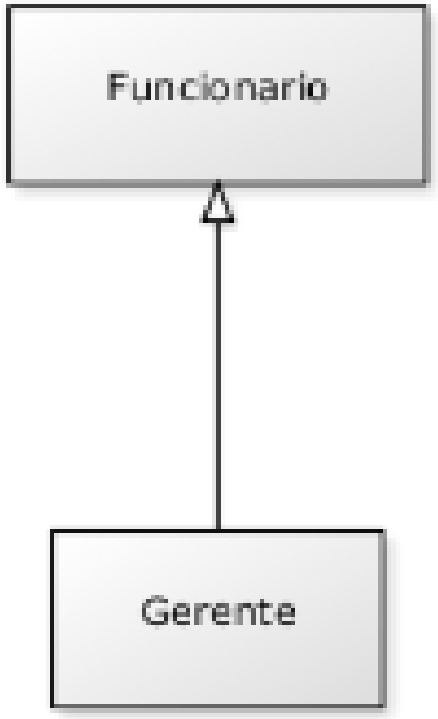
- Existe um jeito, de relacionarmos uma classe de tal maneira que uma delas herda tudo que a outra tem;
- Isto é uma relação de classe mãe e classe filha;
- No nosso caso, gostaríamos de fazer com que o Gerente tivesse tudo que um Funcionario tem, gostaríamos que ela fosse uma extensão de Funcionario;
- Fazemos isto através da palavra chave extends;

```
class Gerente extends Funcionario {
  $senha;
  $numeroDeFuncionariosGerenciados;

  public function autentica(int $senha) {
    if ($this->senha == $senha) {
          echo "Acesso Permitido!";
          return true;
    } else {
          echo "Acesso Negado!";
          return false;
    }
  }

  // setter da senha omitido
}
```

PROGRAMAÇÃO DE COMPUTADORES | HERANÇA



Super e Sub classe

- A nomenclatura mais encontrada é que Funcionario é a superclasse de Gerente, e Gerente é a subclasse de Funcionario. Dizemos também que todo Gerente é um Funcionário.
- Outra forma é dizer que Funcionario é classe mãe de Gerente e Gerente é classe filha de Funcionario.

- E o modificador de acesso?
- Sempre usar o *#protected*?

PROGRAMAÇÃO DE COMPUTADORES

REESCRITA DE MÉTODO

Jefferson de Oliveira Chaves

- Todo fim de ano, os funcionários do nosso banco recebem uma bonificação. Os funcionários comuns recebem 10% do valor do salário e os gerentes, 15%.

```
class Funcionario {
  protected $nome;
  protected $cpf;
  protected $salario;

  public function getBonificacao() {
    return $this->salario * 0.1;
  }
  // métodos
}
```

- Se deixarmos a classe Gerente como ela está, ela vai herdar o método getBonificacao;
- Quando herdamos um método, podemos alterar seu comportamento.
- Podemos reescrever (reescrever, sobrescrever, override) este método:

```
class Gerente extends Funcionario {
  $senha;
  $numeroDeFuncionariosGerenciados;

  public function getBonificacao() {
    return $this->salario * 0.15;
  }
  // ...
}
```

PROGRAMAÇÃO DE COMPUTADORES

POLIMORFISMO

Jefferson de Oliveira Chaves

- Na herança, vimos que todo Gerente é um Funcionario, pois é uma extensão deste.
- Podemos nos referir a um Gerente como sendo um Funcionario.
- Se alguém precisa falar com um Funcionario do banco, pode falar com um Gerente! Porque? Pois Gerente é um Funcionario.
- **Essa é a semântica da herança.**

- Polimorfismo é a capacidade de um objeto poder ser referenciado de várias formas.
- **Cuidado, polimorfismo não quer dizer que o objeto fica se transformando, muito pelo contrário, um objeto nasce de um tipo e morre daquele tipo, o que pode mudar é a maneira como nos referimos a ele;**

- **Como ficaria uma classe que controlasse quanto em bonificações o banco pagou?**

REESCRITA DE MÉTODO

Parâmetro Polimórfico

```
class ControleDeBonificacoes {
  private $totalDeBonificacoes = 0;

  public function registra(Funcionario $funcionario) {
    $this->totalDeBonificacoes += $funcionario->getBonificacao();
  }

  public function getTotalDeBonificacoes() {
    return $this->totalDeBonificacoes;
  }
}
```

- Qual será o valor resultante?
- Não importa que dentro do método registra do ControleDeBonificacoes receba Funcionario ;
- Quando ele receber um objeto que realmente é um Gerente, o seu método reescrito será invocado.
- **Reafirmando:** não importa como nos referenciamos a um objeto, o método que será invocado é sempre o que é dele.

- No dia em que criarmos uma classe Secretaria, por exemplo, que é filha de Funcionario, precisaremos mudar a classe de ControleDeBonificacoes? Não.
- **É exatamente esse o poder do polimorfismo, juntamente com a reescrita de método;**
- Repare que quem criou ControleDeBonificacoes pode nunca ter imaginado a criação da classe Secretaria ou Engenheiro. Contudo, não será necessário reimplementar esse controle em cada nova classe: reaproveitamos aquele código.

PROGRAMAÇÃO DE COMPUTADORES | Exercícios: Herança e Polimorfismo

1. Modifique sua classe **Conta**, definindo seus atributos com o privados ou protegidos. Analise cada atributo e método.
2. Adicione um método na classe **Conta**, que atualiza o saldo da **Conta**, de acordo com uma taxa percentual fornecida.
3. Crie duas subclasses da classe Conta: **ContaCorrente** e **ContaPoupanca**. Ambas terão o método atualiza reescrito: A ContaCorrente deve atualizar-se com o dobro da taxa e a ContaPoupanca deve atualizar-se com o triplo da taxa.
4. ContaCorrente deve reescrever o método deposita, a fim de retirar uma taxa bancária de dez centavos de cada depósito.