FACEC
Faculdade

Programação II

Polimorfismo

Prof. Emmanuel Neri

2019

FACEC
Faculdade

Cronograma

- Conceitos
- Exercícios

FACEC
Faculdade

Polimorfismo

“A palavra polimorfismo significa muitas formas. Na programação orientada a objetos significa a realização de uma tarefa de formas diferentes”

Peter Jandl Junior

2019

FACEC
Faculdade

Polimorfismo

"O polimorfismo permite programar no geral em vez de programar no específico"

Paul J. Deitel

2019

FACEC
Faculdade

Não polimórfico

```
public double calcularDesconto(TipoPessoa tipoPessoa,
                                        double valor) {

    if(tipoPessoa == TipoPessoa.FISICA) {
        return valor - (valor * 0.10);
    }

    if(tipoPessoa == TipoPessoa.JURIDICA ) {
        return valor - (valor * 0.15);
    }

    return 0;
}
```

FACEC
Faculdade

Polimorfismo

"O polimorfismo permite que programadores tratem de generalidades e deixem que o ambiente de tempo de execução trate as especificidades. Os programadores podem instruir objetos a se comportarem de maneiras apropriadas para esses objetos, sem nem mesmo conhecer os tipos dos objetos"

ROBERTO RUBINSTEIN SERSON

2019

FACEC
Faculdade

Polimorfismo

```
public enum TipoPessoa {
    FISICA {
        @Override
        public double calcularDesconto(double valor) {
            return valor - (valor * 0.10);
        }
    },
    JURIDICA {
        @Override
        public double calcularDesconto(double valor) {
            return valor - (valor * 0.15);
        }
    };

    public abstract double calcularDesconto(final double valor);

}
```

```
pessoa.getTipo().calcularDesconto(1000)
```

FACEC
Faculdade

Polimorfismo

```
public static void main(String args[]) {

    List<Pessoa> pessoas = new ArrayList<>();
    pessoas.add(new Vendedor("Vendedor", "1"));
    pessoas.add(new Vendedor("Vendedor", "2"));
    pessoas.add(new Fornecedor("Fornecedor", "1"));
}
```

FACEC
Faculdade

Polimorfismo

```
public static void main(String args[]) {

    List<Pessoa> pessoas = new ArrayList<>();
    pessoas.add(new Vendedor("Vendedor", "1"));
    pessoas.add(new Vendedor("Vendedor", "2"));
    pessoas.add(new Fornecedor("Fornecedor", "1"));

    for(Pessoa pessoa : pessoas) {
        imprimirNome(pessoa);
    }
}

private static void imprimirNome(Pessoa pessoa) {
    System.out.println(pessoa.getNomeCompleto());
}
```

FACEC
Faculdade

Polimorfismo

```
public static void main(String args[]) {

    List<Pessoa> pessoas = new ArrayList<>();
    pessoas.add(new Vendedor("Vendedor", "1", "Livros"));
    pessoas.add(new Fornecedor("Fornecedor", "1",  "Revistas"));

    for(Pessoa pessoa : pessoas) {
        System.out.println("Nome completo: " + pessoa.getNome());

        if(pessoa instanceof Vendedor) {
            System.out.println("Departamento: "
                               + ((Vendedor) pessoa).getDepartamento());
        }

        if(pessoa instanceof Fornecedor) {
            System.out.println("Especialidade: "
                               + ((Fornecedor) pessoa).getEspecialidade());
        }
    }
}
```

FACEC
Faculdade

Polimorfismo

```
public static void main(String args[]) {

    List<Pessoa> pessoas = new ArrayList<>();
    pessoas.add(new Vendedor("Vendedor", "1", "Livros"));
    pessoas.add(new Fornecedor("Fornecedor", "1",  "Revistas"));

    for(Pessoa pessoa : pessoas) {
        System.out.println(pessoa.toString());
    }
}
```

departamento

especialidade

Vendedor{nome='Vendedor', sobrenome='1', departamento='Livros'}
Fornecedor{nome='Fornecedor', sobrenome='1', especialidade='Revistas'}

FACEC
Faculdade

Conclusão

- Baseia-se nos tipos mais genéricos
- Polimorfismos pode ser usado com classes abstratas e com interface

FACEC
Faculdade

Complementar

- https://www.caelum.com.br/apostila-java-orientacao-objetos/heranca-reescrita-e-polimorfismo/

FACEC
Faculdade

Exercícios

- Criar as classes ContaCorrente e ContaPoupanca de forma que seja possível realizar saque e depósito independente do tipo da conta;

- Criar uma interface Imprimivel que possua o método imprimir() e implemente a interface em duas classes concretas. Após a criação da estrutura de classes, criar uma lista de classes que implementam Imprimivel e itere a lista imprimindo em cada objeto da lista.

- Criar uma estrutura que saiba realizar as operações bases (adição, subtração, multiplicação e divisão) em dois valores;