



Quem se prepara, não para.

Programação Orientada a Objetos

Prática 4

3º período

Professora: Michelle Hanne

Orientações

- 1) Criar uma conta no GitHub - <https://github.com/>
- 2) Criar um repositório Público no GitHub com o nome **Prática4_OO**
- 3) Subir os arquivos para o repositório criado
- 4) Enviar o link do repositório na tarefa do Canvas.

OBS: O envio deverá ser realizado individualmente

Questão 1 – Conta Bancária (Guiada)

- Criar um Projeto com o nome **ContaAbstract**
- Criar uma **classe abstrata** com o nome **Conta**
 - Atributo: `private double saldo;`
 - Métodos `setSaldo` e `getSaldo` públicos
 - Método: `public abstract void imprimeExtrato();` //sem conteúdo
- Criar uma classe com o nome **ContaPoupança que estende Conta**
 - `public class ContaPoupança extends Conta`
 - Importar duas bibliotecas de data e hora
 - `import java.text.SimpleDateFormat;`
 - `import java.util.Date;`

Questão 1 – Conta Bancária (Guiada)

- Criar um Projeto com o nome **ContaAbstract**
- Criar uma **classe abstrata** com o nome **Conta**
 - Atributo: `private double saldo;`
 - Métodos `setSaldo` e `getSaldo` públicos
 - Método: `public abstract void imprimeExtrato();` //sem conteúdo
- Criar uma classe com o nome **ContaPoupança que estende Conta**
 - `public class ContaPoupança extends Conta`
 - Importar duas bibliotecas de data e hora
 - `import java.text.SimpleDateFormat;`
 - `import java.util.Date;`

Questão 1 – Conta Bancária (Guiada)

- Criar o método `ImprimeExtrato`, conforme abaixo:

```
public void imprimeExtrato() {  
    System.out.println("### Extrato da Conta ###");  
  
    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/aaaa  
HH:mm:ss");  
    Date date = new Date();  
  
    System.out.println("Saldo: "+this.getSaldo());  
    System.out.println("Data: "+sdf.format(date));  
  
}
```

Questão 1 – Conta Bancária (Guiada)

Criar no main:

```
Conta cp = new ContaPoupanca();  
cp.setSaldo(2121);  
cp.imprimeExtrato();
```

Questão 1 – Conta Bancária (Guiada)

- Criar um Projeto com o nome **ContaInterface**
- Criar uma **Interface** com o nome **Conta**
 - void depositar(double valor);
 - void sacar(double valor);
 - double getSaldo();

Questão 1 – Conta Bancária (Guiada)

- Criar a Classe **ContaCorrente** que implementa **Conta**:

```
public class ContaCorrente implements Conta{
    private double saldo;
    private double taxaOperacao = 0.45;
    @Override
    public void depositar(double valor) {
        this.saldo += valor - taxaOperacao;
    }
    @Override
    public double getSaldo() {
        return this.saldo;
    }
    @Override
    public void sacar(double valor) {
        this.saldo -= valor + taxaOperacao;
    }
}
```

Questão 1 – Conta Bancária (Guiada)

- Criar a Classe **ContaPoupança** que implementa **Conta**:

```
public class ContaPoupança implements Conta{  
    private double saldo;  
  
    @Override  
    public void depositar(double valor) {  
        this.saldo += valor;  
    }  
  
    @Override  
    public double getSaldo() {  
        return this.saldo;  
    }  
  
    @Override  
    public void sacar(double valor) {  
        this.saldo -= valor;  
    }  
}
```

Questão 1 – Conta Bancária (Guiada)

- Criar a Classe Pública **GeradorExtratos**:

```
public class GeradorExtratos {  
  
    public void geradorConta(Conta conta){  
        System.out.println("Saldo Atual: "+conta.getSaldo());  
    }  
}
```

Questão 1 – Conta Bancária (Guiada)

- Agora faça os testes no main:

```
ContaCorrente cc = new ContaCorrente();  
cc.depositar(1200.20);  
cc.sacar(300);
```

```
ContaPoupanca cp = new ContaPoupanca();  
cp.depositar(500.50);  
cp.sacar(25);
```

```
GeradorExtratos gerador = new GeradorExtratos();  
gerador.geradorConta(cc);  
gerador.geradorConta(cp);  
}
```

Questão 2 – Loja Comercial

Uma loja comercial tem 2 tipos de funcionários: vendedores e administrativos.

Para todos a empresa precisa ter o registro do nome e RG do funcionário. Os vendedores têm um salário base, mas ganham também comissão de suas vendas. Os administrativos têm um salário fixo, mas podem ganhar horas extras adicionais.

Faça uma hierarquia de classes que tenha uma classe ancestral que implemente o que for comum aos dois tipos de funcionários e uma classe descendente para cada tipo. Os vendedores devem ter um método que acumule o total de vendas durante o mês e um método que imprima seu salário total considerando que a comissão é de 5%.

Questão 2 – Loja Comercial

Os vendedores devem ter um método que acumule o total de vendas durante o mês e um método que imprima seu salário total considerando que a comissão é de 5%.

Para os administrativos as horas extras é que são acumuladas e pagas com o valor de um centésimo do salário por hora. Nos dois casos, o método que imprime o salário a receber zera os valores acumulados.

****Fazer o main com uma interface de entrada de dados e testar os métodos das classes*

Questão 3 – Automóvel

Criar uma classe com o nome **automóvel** com os **atributos lógicos**: abs, airBag, alarme, som, conjunto elétrico e computador de bordo. Também, deverá ser criado o atributo **preço base do automóvel** sem estes opcionais. Baseado nisto, a classe receberá os seguintes métodos:

- a) construtor: receber os atributos definidos;
- b) métodos set e get: para todos os atributos inseridos;
- c) método **calcularPrecoFinal**: informa o preço do carro baseado nas seguintes regras:

Questão 3 – Automóvel

Criar uma classe com o nome **automóvel** com os **atributos lógicos**: abs, airBag, alarme, som, conjunto elétrico e computador de bordo. Também, deverá ser criado o atributo **preço base do automóvel** sem estes opcionais. Baseado nisto, a classe receberá os seguintes métodos:

- a) construtor: receber os atributos definidos;
- b) métodos set e get: para todos os atributos inseridos;
- c) método **calcularPrecoFinal**: informa o preço do carro baseado nas seguintes regras:

Questão 3 – Automóvel

Opcional	Regra
abs	aumenta em 15% o valor do preço base
air bag	aumenta em 10% o valor do preço base
alarme	aumenta em 3% o valor do preço base
som	aumenta em 2% o valor do preço base
conjunto elétrico	aumenta em 5% o valor do preço base
computador de bordo	aumenta em 10% o valor do preço base

Atenção: caso o automóvel tenha mais do que um opcional, devem ser somados todos os valores percentuais de aumento para, ao final, aplicar o valor desta soma sobre o preço base.

Questão 3 – Automóvel

Criar uma classe chamada **popular** que **herda automóvel**, a qual receberá como parâmetro um percentual de desconto. Criar um método **calcularPrecoFinalPopular** que vai ser responsável por imprimir o preço final com o desconto.

Main: instanciar a classe automóvel e a classe popular, testar os parâmetros e os métodos.