INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA BAHIA Campus Irecê	**Instituto Federal de Educação, Ciência e Tecnologia da Bahia – *Campus* Irecê** **Disciplina: Linguagem Técnica I** **Prof° Jonatas Bastos**

Nome: _____

LISTA DE EXERCÍCIO 5 – OO e Herança

1) Crie uma classe Conta, que possua um saldo, e os métodos para pegar saldo, depositar, e sacar.

a) Crie a classe `Conta`:

```
class Conta {

}
```

b) Adicione o atributo `saldo`

```
class Conta {

    private double saldo;
}
```

c) Crie os métodos `getSaldo()`, `deposita(double)` e `saca(double)`

```
class Conta {

    private double saldo;

    void deposita(double valor) {
        this.saldo += valor;
    }

    void saca(double valor) {
        this.saldo -= valor;
    }

    double getSaldo() {
        return this.saldo;
    }
}
```

2) Adicione um método na classe Conta, que atualiza essa conta de acordo com uma taxa percentual fornecida.

```
class Conta {

    private double saldo;

     // outros métodos aqui também ...

    void atualiza(double taxa) {
        this.saldo += this.saldo * taxa;
    }
}
```

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.

Além disso, a ContaCorrente deve reescrever o método deposita, afim de retirar uma taxa bancária de dez centavos de cada depósito.

- Crie as classes `ContaCorrente` e `ContaPoupanca`. Ambas são filhas da classe `Conta`:

```
class ContaCorrente extends Conta {

}

class ContaPoupanca extends Conta {
}
```

- Reescreva o método `atualiza` na classe `ContaCorrente`, seguindo o enunciado:

```
class ContaCorrente extends Conta {

    void atualiza(double taxa) {
        this.saldo += this.saldo * taxa * 2;
    }
}
```

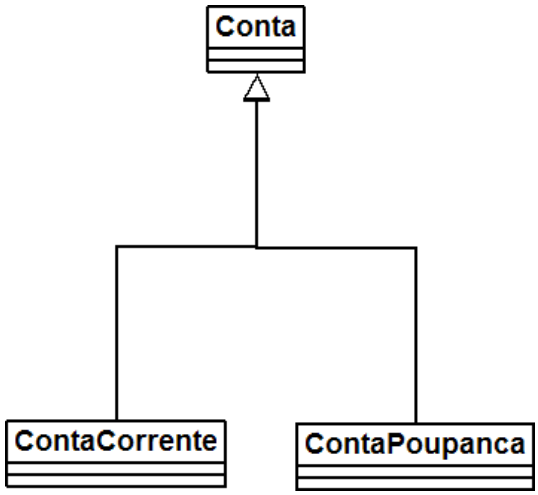
  Repare que, para acessar o atributo saldo herdado da classe `Conta`, você vai precisar trocar o modificador de visibilidade de saldo para `protected`.

- Reescreva o método `atualiza` na classe `ContaPoupanca`, seguindo o enunciado:

```
class ContaPoupanca extends Conta {

    void atualiza(double taxa) {
        this.saldo += this.saldo * taxa * 3;
    }
}
```

- Na classe `ContaCorrente`, reescreva o método `deposita` para descontar a taxa bancária de dez centavos:

```
class ContaCorrente extends Conta {

    void atualiza(double taxa) {
        this.saldo += this.saldo * taxa * 2;
    }

    void deposita(double valor) {
        this.saldo += valor - 0.10;
    }
}
```



4) Crie uma classe com método main e instancie essas classes, atualize-as e veja o resultado. Algo como:

```
class TestaContas {

    public static void main(String[] args) {
        Conta c = new Conta();
        ContaCorrente cc = new ContaCorrente();
        ContaPoupanca cp = new ContaPoupanca();

        c.deposita(1000);
        cc.deposita(1000);
        cp.deposita(1000);

        c.atualiza(0.01);
        cc.atualiza(0.01);
        cp.atualiza(0.01);

        System.out.println(c.getSaldo());
        System.out.println(cc.getSaldo());
        System.out.println(cp.getSaldo());

    }
}
```

Após imprimir o saldo (`getSaldo()`) de cada uma das contas, o que acontece?

5) O que você acha de rodar o código anterior da seguinte maneira:

```
Conta c = new Conta();

Conta cc = new ContaCorrente();
Conta cp = new ContaPoupanca();
```

. 6) Crie uma classe que seja responsável por fazer a atualização de todas as contas bancárias e gerar um relatório com o saldo anterior e saldo novo de cada uma das contas.

```
class AtualizadorDeContas {

    private double saldoTotal = 0;
    private double selic;

    AtualizadorDeContas(double selic) {
        this.selic = selic;
    }

    void roda(Conta c) {
        // aqui voce imprime o saldo anterior, atualiza a conta,
        // e depois imprime o saldo final
        // lembrando de somar o saldo final ao atributo saldoTotal
    }

    // outros métodos, colocar o getter para saldoTotal!
}
```

7) No método main, crie algumas contas e depois rode o programa:

```
class TestaAtualizadorDeContas {

    public static void main(String[] args) {
        Conta c = new Conta();
        Conta cc = new ContaCorrente();
        Conta cp = new ContaPoupanca();

        c.deposita(1000);
        cc.deposita(1000);
        cp.deposita(1000);

        AtualizadorDeContas adc = new AtualizadorDeContas(0.01);

        adc.roda(c);
        adc.roda(cc);
        adc.roda(cp);

        System.out.println("Saldo Total: " + adc.getSaldoTotal());
    }

}
```

8) Use a palavra chave super nos métodos atualiza reescritos, para não ter de refazer o trabalho.

9) Se você precisasse criar uma classe ContaInvestimento, e seu método atualiza fosse complica- díssimo, você precisaria alterar a classe AtualizadorDeContas?

10) Crie uma classe Banco que possui um array de Conta. Repare que num array de Conta você pode colocar tanto ContaCorrente quanto ContaPoupanca. Crie um método void adiciona(Conta c), um método Conta pegaConta(int x) e outro int pegaTotalDeContas(), muito similar a relação anterior de Empresa-Funcionario.

.