SENAI CIMATEC

SISTEMA FIEB

TECNOLOGIA E INOVAÇÃO EM PROL DA INDÚSTRIA

SENAI CIMATEC

SISTEMA FIEB

Curso Técnico em Informática

Desenvolvimento de Sistemas II – 180h

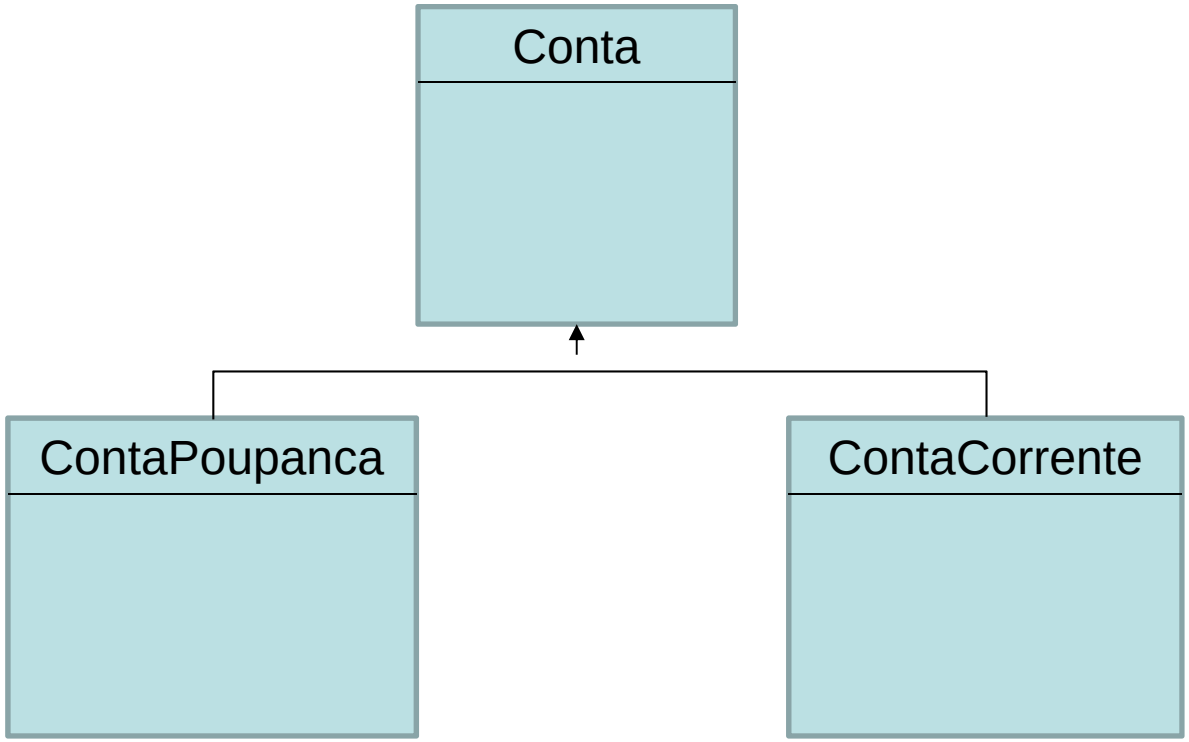
Profª: Francisleide Almeida

https://pt.scribd.com/document/218141650/EXERCICIOS-RESOLVIDOS-JAVA-POLIMORFISMO

Polimorfismo

- Definimos **Polimorfismo** como um princípio a partir do qual as classes derivadas de uma única classe base são capazes de invocar os métodos que, **embora apresentem a mesma assinatura**, **comportam-se de maneira diferente** para cada uma das classes derivadas.
- O **Polimorfismo** é um mecanismo por meio do qual selecionamos as funcionalidades utilizadas de forma dinâmica por um programa no decorrer de sua execução.

- Voltemos ao exemplo da Conta



```
public class Conta {

    protected int numero;
    protected String dono;
    protected double saldo;

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

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

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

    void transfere(Conta destino, double valor)
    {
          this.saldo = this.saldo - valor;
          destino.saldo = destino.saldo + valor;
    }
    void atualiza(double taxa)
    {
       this.saldo += this.saldo * taxa;
    }

}
```

```java
public class ContaPoupanca extends Conta {

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

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

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

    void transfere(Conta destino, double valor)
    {
          this.saldo = this.saldo - valor;
          destino.saldo = destino.saldo + valor;
    }

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

}
```

```java
public class ContaCorrente extends Conta {

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

    void deposita(double quantidade)
    {
        this.saldo += quantidade-0.;
    }

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

    void transfere(Conta destino, double valor)
    {
          this.saldo = this.saldo - valor;
          destino.saldo = destino.saldo + valor;
    }

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

}
```

Testando as classes

```
public class TestaContas {

    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);

        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());

    }
}
```

```
public 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());

    }
}
```

```
Saída - LingProgramacaoIII3 (run)
run:
1010.0
1020.0
1030.0
CONSTRUÍDO COM SUCESSO (tempo total: 2 segundos)
```

```
Saída - LingProgramacaoIII3 (run)
run:
1010.0
1020.0
1030.0
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

- A diferença prática das implementações é a existência do polimorfismo. No primeiro caso ocorre a declaração de uma variável de um tipo menos específico.
- A JVM vai executar o método referente ao tipo do objeto e não como nos referimos a ele.
- no primeiro caso em “cc” seja referido como “Conta” ele é instanciado como “ContaCorrente”.

- Se a gente for criar uma classe que seja responsável por fazer a atualização de todas as contas bancárias, gerando um relatório com o saldo anterior e saldo novo de cada uma das contas.

```
public class AtualizarContas {

    private double TotalSaldo = 0;

    public double getTotalSaldo() {
        return TotalSaldo;
    }
    private double imposto;

    public AtualizarContas(double imposto) {
        this.imposto = imposto;
    }

    public void roda(Conta c) {
        System.out.println("Saldo Anterior: " + c.getSaldo());
        c.atualiza(imposto);
        System.out.println("Saldo Final: " + c.getSaldo());

        TotalSaldo += c.getSaldo();
    }

}
```

```
public class TestarAtualizadorContas {
    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);

        AtualizarContas adc = new AtualizarContas(0.01);

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

        System.out.println("\nSaldo Total: " + adc.getTotalSaldo()
    }
}
```

```
Saída - LingProgramacaoIII3 (run)
run:
Saldo Anterior: 1000.0
Saldo Final: 1010.0
Saldo Anterior: 1000.0
Saldo Final: 1020.0
Saldo Anterior: 1000.0
Saldo Final: 1030.0

Saldo Total: 3060.0
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

- Usar a palavra chave super nos métodos atualiza reescritos, para não ter de refazer o trabalho.

```
public class ContaCorrente extends Conta {

    void atualiza(double taxa)
    {
        super.atualiza(2*taxa);
    }

}
```

```
public class ContaPoupanca extends Conta {

    void atualiza(double taxa)
    {
        super.atualiza(3*taxa);
    }

}
```

- Agora observe a criação de um Banco que possua uma lista de contas, independente do tipo delas

```
public class Banco {

    int totalDeContas=0;

    Conta [] c = new Conta[4];

    public void adicionarConta(Conta a) {
        c[totalDeContas] = a;
        this.totalDeContas++;
    }

    public Conta pegarConta(int indice) {
        return c[indice];
    }

    public int getTotalDeContas() {
        return this.totalDeContas;
    }

    public static void main(String[] args) {

        Banco b = new Banco ();
        Conta a[] = new Conta[4];
        AtualizarContas d = new AtualizarContas(0.01);

        a[0] = new ContaCorrente();   a[0].deposita(500);

        a[1] = new ContaPoupanca();   a[1].deposita(600);

        a[2] = new ContaCorrente();   a[2].deposita(700);

        a[3] = new ContaPoupanca();   a[3].deposita(800);

        for (int i=0; i<4; i++)
        {
            b.adicionarConta(a[i]);
        }

        System.out.println("\nSaldo Total: " + b.getTotalDeContas());

        for (int i=0; i<4; i++)
        {
            d.roda(b.pegarConta(i));
        }

        System.out.println("Saldo Total do Banco: " + d.getTotalSaldo());
    }

}
```

```
Saída - LingProgramacaoIII3 (run)
run:

Saldo Total: 4
Saldo Anterior: 500.0
Saldo Final: 510.0
Saldo Anterior: 600.0
Saldo Final: 618.0
Saldo Anterior: 700.0
Saldo Final: 714.0
Saldo Anterior: 800.0
Saldo Final: 824.0
Saldo Total do Banco: 2666.0
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```