PROGRAMAÇÃO ORIENTADA A OBJETOS
POLIMORFISMO

FAPESC – DESENVOLVEDORES PARA TECNOLOGIA DA INFORMAÇÃO

HABNER FABRÍCIO BOESING
habner.boesing@unoesc.edu.br

OBJETIVOS

- Pilares da Orientação a Objetos:
  - Polimorfismo
  - Assinatura do método
  - Sobrecarga
  - Sobreposição

ASSINATURA DO MÉTODO

- Todo método tem uma assinatura, que são definidos pela **quantidade** e os **tipos** do parâmetros.
- Mais de um parâmetro pode ter o mesmo nome desde que o tipo do parâmetro ou a quantidade parâmetros sejam diferentes entre si.

Método 1: 2 parâmetros float

Método 2: 2 parâmetros float

Logo: Método 1 possui a mesma assinatura do Método 2.

Método 3: 3 parâmetros float. Logo este possui uma assinatura diferente dos demais.

```java
public float calcularSoma(float n1, float n2) {
    float soma;
    soma = n1 + n2;
    return soma;
}
public double calcularSoma(float n1, float n2) {
    double soma;
    soma = n1 + n2;
    return soma;
}
public float calcularSoma(float n1, float n2, float n3) {
    float soma;
    soma = n1 + n2 + n3;
    return soma;
}
```

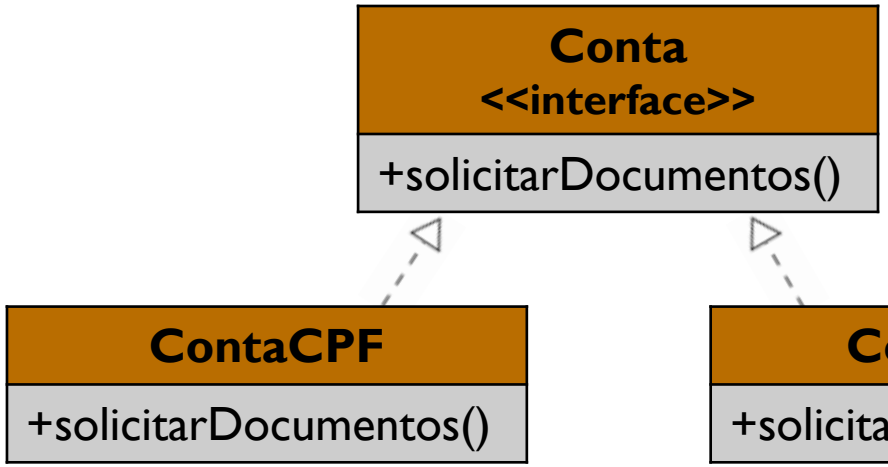
UNOESC

POLIMORFISMO

- O principal objetivo do polimorfismo é quando precisamos recriar o comportamento de um método já existente, atribuindo novos comportamentos a ele, mas mantendo o mesmo nome. Isso costumeiramente é utilizado quando diferentes Classes precisam realizar uma mesma ação, mas com comportamentos diferentes ou a própria Classe implementa a mesma ação por diversas vezes, mas que possui parâmetros e resultados diferentes em cada um deles.
- O Polimorfismo pode ocorrer de duas formas:
- **Polimorfismo de Sobreposição**: importa um método da superclasse e mantém a mesma assinatura, no entanto o implementa de forma diferente.
- **Polimorfismo de Sobrecarga**: reutiliza um método da própria subclasse, no entanto, com assinaturas diferentes.
- Ao utilizar sobreposição, pode-se utilizar tanto uma classe através do comando extends, quanto uma interface utilizando o comando implements.
- Exemplo:


```
public class ContaCNPJ implements Conta{
```

POLIMORFISMO SOBREPOSIÇÃO - REPRESENTAÇÃO



Lembrando que uma interface não possui atributos, apenas especifica os métodos que deverão ser implementados pelas subclasses.

POLIMORFISMO SOBREPOSIÇÃO - EXEMPLO

- Interface Conta

```
public interface Conta {
    public void solicitarDocumentos();
}
```

POLIMORFISMO SOBREPOSIÇÃO - EXEMPLO

- Subclasse ContaCPF

```java
public class ContaCPF implements Conta {
    @Override
    public void solicitarDocumentos() {
        System.out.println("Documentos necessários Pessoa Física:");
        System.out.println("-Identidade\n-nº CPF\n-Comprovante de Residência");
    }
```

POLIMORFISMO SOBREPOSIÇÃO - EXEMPLO

- Classe principal Banco

```
public class Banco {
    public static void main(String[] args) {
        ContaCPF fisica = new ContaCPF();
        ContaCNPJ juridica = new ContaCNPJ();
        //exemplo polimorfismo de sobreposição
        //mesmo método, em objetos diferentes, implementados de formas diferentes
        fisica.solicitarDocumentos();
        juridica.solicitarDocumentos();
```

POLIMORFISMO SOBRECARGA - REPRESENTAÇÃO

ContaCPF
+consultarExtrato()
+consultarExtrato (String: mes)
+consultarExtrato (int: período)

No Polimorfismo de Sobrecarga o mesmo método pode ser implementado na mesma classe por diversas vezes utilizando assinaturas diferentes.

POLIMORFISMO SOBRECARGA - EXEMPLO

- Subclasse ContaCPF

```
public class ContaCPF implements Conta {

    public void consultarExtrato(){
        System.out.println("Impressão extrato dos últimos 7 dias");
    }
    public void consultarExtrato(String mes){
        System.out.println("Impressão extrato do mês de "+mes);
    }
    public void consultarExtrato(int periodo){
        System.out.println("Impressão extrato dos últimos "+periodo+" dias");
    }
```

POLIMORFISMO SOBREPOSIÇÃO - EXEMPLO

- Classe principal Banco

```java
public class Banco {
    public static void main(String[] args) {
        ContaCPF fisica = new ContaCPF();

        //exemplo polimorfismo de sobrecarga
        //mesmo método, no mesmo objeto, implementado de formas diferentes
        fisica.consultarExtrato();
        fisica.consultarExtrato( mes: "Novembro");
        fisica.consultarExtrato( periodo: 60);
    }
```

ATIVIDADE PRÁTICA

- I) Crie uma interface para representar qualquer forma geométrica, definindo um método para cálculo da área de figuras geométricas que deve implementar como retorno a área calculada; em seguida, crie a classe que represente os retângulos, recebendo o tamanho da base e da altura no construtor; desenvolva uma classe que represente os quadrados, que deve receber apenas o tamanho do lado; desenvolva uma classe para representar um círculo, cujo seu construtor deverá receber o tamanho do raio. Estas três classes devem implementar a interface.
- O programa principal deverá perguntar ao usuário qual forma ele deseja criar (quadrado, retângulos ou círculo), em seguida deve solicitar os dados necessários para criar a forma. Após criar a forma, deve ser executado o método para calcular a área da forma e deve exibir o resultado na tela.

ATIVIDADE PRÁTICA

2) Faça uma classe Animal com um método abstrato "falar"

- Faça as classes Homem, Cão e Gato, herdando de animal, redefinindo o método "falar" para retornar "Oi", "Au au" e "Miau", respectivamente
- Para testar crie um vetor de 10 Animais e instancie Homens, Cães e Gatos nesse vetor.
- Faça um loop por todos os animais do vetor, pedindo para eles falarem.