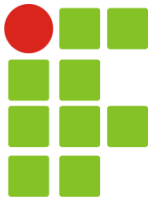


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
CAMPUS JOÃO CÂMARA

PROGRAMAÇÃO ESTRUTURADA E ORIENTADA A OBJETOS - INTERFACE

Nickerson Fonseca Ferreira
nickerson.ferreira@ifrn.edu.br

Introdução

- As vezes necessitamos que um conjunto de classes possuam os mesmos comportamentos, porém não sejam filhos de uma super classe.
- Para isso podemos criar uma interface.
- Interface funciona como um contrato, informando que as classes que implementarem essa(s) interface(s) possuem comportamentos semelhantes.
- Funciona de forma semelhante às Classes Abstratas, porém não possuem o relacionamento “é um”.

Introdução

- Até a versão 7 do Java só era permitido informar os métodos "abstratos" e variáveis na interface.

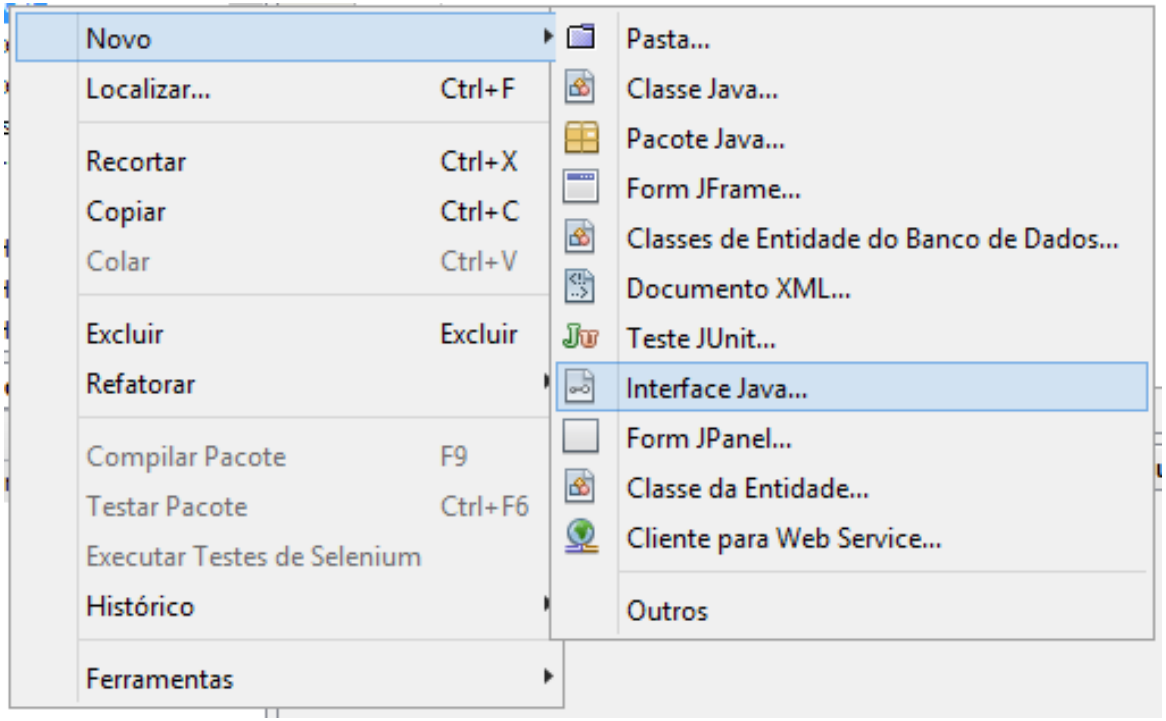
```
public interface ICaixaEletronico {
        void sacar(float valor);
}
```

- Já na versão 8 criaram os métodos **default.**

```
1  public interface ICaixaEletronico {
2          void sacar(float valor);
3          default void verificaFraude(){
4                  System.out.println("Verificação de fraude iniciada");
5          }
6  }
```

Criando uma Interface

- 1° Criamos uma nova Interface Java.



Criando uma Interface

- Depois informamos quais métodos o contrato terá.

```
1  public interface ICaixaEletronico {
2          void sacar(float valor);
3          default void verificaFraude(){
4                  System.out.println("Verificação de fraude iniciada");
5          }
6  }
```

Criando uma Interface

- Para dizer que uma classe possui esse novo contrato utilizamos a palavra reservada **implements**.

```
public class CaixaEletronico implements ICaixaEletronico {
        @Override
        public void sacar(float valor) {
                System.out.println("Vou sacar " + valor);
        }
}
```

Usando tudo isso


```
public class Main {

    public static void main(String[] args) {
        ICaixaEletronico i = new CaixaEletronico();
        i.sacar(100.00);
    }
}
```

Utilizando interfaces

- Podemos utilizar mais de uma interface numa classe.

```
public class CaixaEletronico implements ICaixaEletronico, ICaixaEletronico24h {
        @Override
        public void sacar(float valor) {
                System.out.println("Vou sacar " + valor);
        }
}
```

Exercício

- Crie a seguinte hierarquia de classes:
- Uma interface para representar qualquer forma geométrica, definindo métodos para cálculo do perímetro e cálculo da área da forma;
- Uma classe abstrata para representar quadriláteros. Seu construtor deve receber os tamanhos dos 4 lados e o método de cálculo do perímetro já pode ser implementado;
- Classes para representar retângulos e quadrados. A primeira deve receber o tamanho da base e da altura no construtor, enquanto a segunda deve receber apenas o tamanho do lado;
- Uma classe para representar um círculo. Seu construtor deve receber o tamanho do raio.
- No programa principal, pergunte ao usuário quantas formas ele deseja criar. Em seguida, para cada forma, pergunte se deseja criar um quadrado, um retângulo ou um círculo, solicitando os dados necessários para criar a forma. Todas as formas criadas devem ser armazenadas em um vetor. Finalmente, imprima: (a) os perímetros; e (b) as áreas de todas as formas.

Referências

- **Apostila Caelum:** https://www.caelum.com.br/apostila-java-orientacao-objetos/orientacao-a-objetos-basica
- H.M. Deitel, P.J. Deitel, **Java Como programar.**