

# Linguagens de Programação

Java Orientado a Objetos

Prof. Waldeck Lindoso Jr.

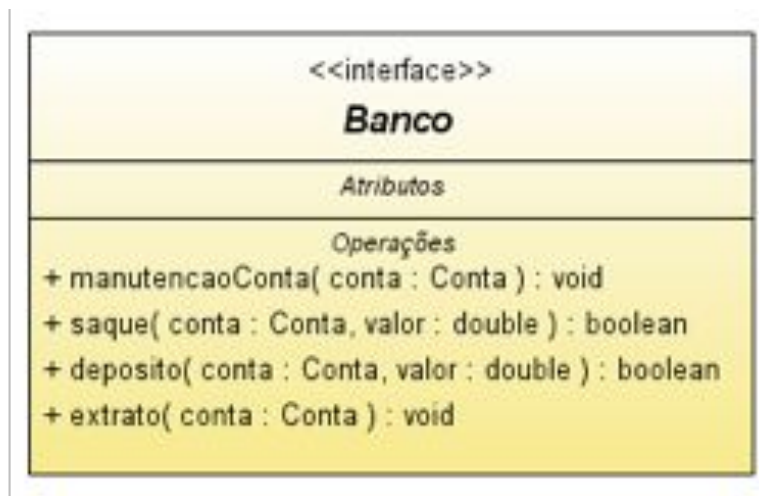
# Interfaces

- Interface é um recurso da linguagem Java que apresenta inúmeras vantagens no sentido da modelagem e instanciação de objetos, porém deve-se entender claramente os conceitos básicos da orientação a objetos a fim de utilizá-la plenamente.
- Uma interface é similar a um contrato, através dela podemos especificar quais métodos as classes que implementam esta interface são obrigados a implementar.



# Interfaces

- Exemplo de interface UML:



# Interfaces

- Em Java esta interface fica do seguinte modo:

```
1 package interfaces;
2 /**
3  * Interface utilizada para representar os métodos mínimos que
4  * os bancos precisam implementar.
5  */
6 public interface Banco {
7     public abstract void manutencaoConta(Conta conta);
8     public abstract boolean saque(Conta conta, double valor);
9     public abstract boolean deposito(Conta conta, double valor);
10    public abstract void extrato(Conta conta);
11 }
```

# Interfaces (*Palavra-chave* **implements**)

- Para criarmos uma interface em Java, utilizamos a palavra-chave **implements** antes do seu nome.

```
1 package interfaces;  
2  
3 public class BancoETEPD implements Banco {  
4  
5 }
```

# Interfaces


- Dentro de uma interface nós não podemos:
  - implementar método
  - construtor
  - estender classe
  - implementar outra interface
  - não pode ser final



# Interfaces

- Quando uma classe implementa uma interface, esta classe obrigatoriamente precisa implementar todos os métodos declarados na interface, apenas quando usamos classes abstratas implementando interface que não precisamos obrigatoriamente implementar todos os métodos (classes abstratas serão vistas mais para frente).

```
1 package interfaces;  
2  
3 public class BancoETEPD implements Banco {  
4  
5 }
```



The context menu shows two options:

- Add unimplemented methods
- Create new JUnit test case for 'BancoETEPD.java'

## 4 methods to implement:

- interfaces.Banco.manutencaoConta(...)
- interfaces.Banco.saque(...)
- interfaces.Banco.deposito(...)
- interfaces.Banco.extrato(...)

# Interfaces

- Vamos criar dois Bancos que implementam a interface Banco com maneiras diferentes de tratar cada método .

```
3 public class BancoETEPD implements Banco {  
4     private Conta contaBancoETEPD;
```

```
3 public class BancoETEPDMOD2 implements Banco {  
4     private Conta contaBancoETEPDMOD2;
```



# Interfaces

```
21 @Override
22 public boolean saque(Conta conta, double valor) {
23     //Verifica se tem saldo suficiente para fazer o saque
24     if(conta.getSaldo() ≥ valor) {
25         //Realiza o saque na conta.
26         double novoValor = conta.getSaldo() - valor;
27         conta.setSaldo( novoValor );
28         System.out.println("Saque efetuado!!!");
29
30         //Toda vez que fizer um saque faz cobra a manutenção da conta.
31         manutencaoConta(conta);
32         return true;
33     } else {
34         System.out.println("Não conseguiu fazer o saque!!!");
35         //Se não conseguir fazer o saque, mostra o extrato da conta.
36         extrato(conta);
37         return false;
38     }
39 }
```

# Interfaces

```
25 @Override
26 public boolean saque(Conta conta, double valor) {
27     //Verifica se tem saldo suficiente para fazer o saque
28     if(conta.getSaldo() ≥ valor) {
29         //Realiza o saque na conta.
30         double novoValor = conta.getSaldo() - valor;
31         conta.setSaldo( novoValor );
32         System.out.println("Saque efetuado!!!");
33
34         //Toda vez que fizer um saque faz cobra a manutenção da conta.
35         manutencaoConta(conta);
36         return true;
37     } else {
38         System.out.println("Não conseguiu fazer o saque!!!");
39         //Se não conseguir fazer o saque, mostra o extrato da conta.
40         extrato(conta);
41         return false;
42     }
43 }
```

# Interfaces

- Agora vamos criar nosso programa BancoTeste

```
3 public class BancoTeste {
4
5     public static void main(String[] args) {
6         Banco bancoETEPD = new BancoETEPD();
7         Conta conta1 = new Conta();
8         conta1.setNomeProprietario("Vegeta");
9         conta1.setNumero(1);
10        conta1.setSaldo(500);
11
12        bancoETEPD.deposito(conta1, 40.99);
13        bancoETEPD.saque(conta1, 300);
14        bancoETEPD.extrato(conta1);
15
16        Banco bancoETEPDMOD2 = new BancoETEPDMOD2();
17        Conta conta2 = new Conta();
18        conta2.setNomeProprietario("Son Goku");
19        conta2.setNumero(1);
20        conta2.setSaldo(1000);
21
22        bancoETEPDMOD2.deposito(conta2, 150.50);
23        bancoETEPDMOD2.saque(conta2, 500);
24        bancoETEPDMOD2.extrato(conta2);
25    }
```

# Interfaces

- A saída do código anterior será:

```
<terminated> BancoTeste [Java Applicatic
Deposito efetuado!!!
Saque efetuado!!!
Deposito efetuado!!!

-- BANCO ETEPD --

-> EXTRATO CONTA

Nome: Vegeta
Numero: 1
Saldo: 240.74

-----

Deposito efetuado!!!
Saque efetuado!!!

-- BANCO ETEPDMOD2 --

-> EXTRATO CONTA

Nome: Son Goku
Numero: 1
Saldo: 650.48

-----
```

Obrigado!

