Aula 14 - Interface

Interface é um tipo que define um conjunto de operações que uma classe deve implementar. A interface estabelece um contrato que a classe deve cumprir.

A interface possui algumas características diferentes, são elas:

1. Suporta “Herança Múltipla”: Uma classe pode implementar múltiplas interfaces;

2. Permite apenas métodos sem implementação, apenas com assinatura;

3. Não contém atributos;

4. Não contém construtor;

5. Uma interface não é herdada e sim implementada;

6. Serve como um “contrato” a ser cumprido pelas classes que a implementam;

7. Deve ser utilizada para criar sistemas com baixo acoplamento e flexíveis.

Uma interface deve ser utilizada quando várias classes diferentes compartilham apenas a assinatura de seus métodos. Em outras palavras, podemos ter várias classes sem qualquer relação entre si, mas se elas compartilharem as funcionalidades oferecidas pela interface que estamos criando, então faz total sentido que essas classes a implementem.

As interfaces existem para ditar o que uma classe deve fazer, ajudando a definir quais habilidades as classes que assinarem esse "contrato" devem possuir.

Vamos ver um exemplo de **uma classe implementando uma interface**:

Conta.java (Interface)

```
public interface Conta {

    public void Depositar(double valor);

    public void Depositar(double valor, String nomeDepositante);

    public void Sacar(float valor);
}
```

ContaCorrente.java (Classe comum)

```
public class ContaCorrente implements Conta{
    private String agencia;
    private String numero;
```

```
    private Double saldo;
    private double chequeEspecial;

    public String getAgencia() {
        return agencia;
    }

    public void setAgencia(String agencia) {
        this.agencia = agencia;
    }

    public String getNumero() {
        return numero;
    }

    public void setNumero(String numero) {
        this.numero = numero;
    }

    public Double getSaldo() {
        return saldo;
    }

    public void setSaldo(Double saldo) {
        this.saldo = saldo;
    }

    public double getChequeEspecial() {
        return chequeEspecial;
    }

    public void setChequeEspecial(double chequeEspecial) {
        this.chequeEspecial = chequeEspecial;
    }

    public ContaCorrente() {
    }

    public ContaCorrente(String agencia, String numero, double saldo,
double chequeEspecial) {
        this.agencia = agencia;
        this.numero = numero;
        this.saldo = saldo;
```

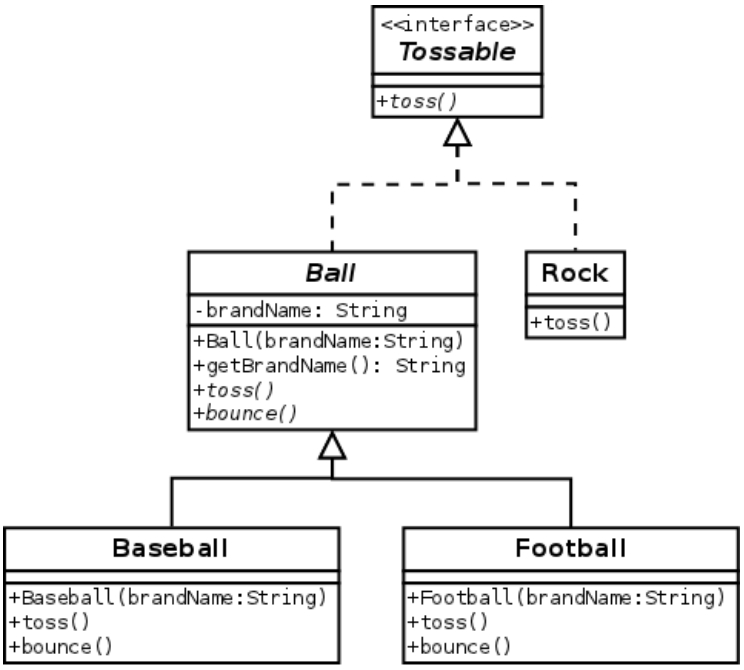
```
        this.chequeEspecial = chequeEspecial;
    }

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

    public void Depositar(double valor, String nomeDepositante){
        this.saldo += valor;
    }

    public void Sacar(float valor){
        this.saldo -= valor;
    }
}
```

UML



Agora vamos ver um outro exemplo de **uma classe implementando duas interfaces**:

Conta.java (Interface)

```
public interface Conta {

    public void Depositar(double valor);

    public void Depositar(double valor, String nomeDepositante);
```

```
    public void Sacar(float valor);
}
```

ContaBasica.java (Interface)

```
public interface ContaBasica {

    public void enviarMensagemAoGerente(String mensagem);

}
```

ContaCorrente.java (Classe comum)

```
public class ContaCorrente implements Conta, ContaBasica{
    private String agencia;
    private String numero;
    private Double saldo;
    private double chequeEspecial;

    public String getAgencia() {
        return agencia;
    }

    public void setAgencia(String agencia) {
        this.agencia = agencia;
    }

    public String getNumero() {
        return numero;
    }

    public void setNumero(String numero) {
        this.numero = numero;
    }

    public Double getSaldo() {
        return saldo;
    }

    public void setSaldo(Double saldo) {
        this.saldo = saldo;
    }
```

```
    public double getChequeEspecial() {
        return chequeEspecial;
    }

    public void setChequeEspecial(double chequeEspecial) {
        this.chequeEspecial = chequeEspecial;
    }

    public ContaCorrente() {
    }

    public ContaCorrente(String agencia, String numero, double saldo,
double chequeEspecial) {
        this.agencia = agencia;
        this.numero = numero;
        this.saldo = saldo;
        this.chequeEspecial = chequeEspecial;
    }

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

    public void Depositar(double valor, String nomeDepositante){
        this.saldo += valor;
    }

    public void Sacar(float valor){
        this.saldo -= valor;
    }

    public void enviarMensagemAoGerente(String mensagem) {

    }
}
```

UML

Agora vamos ver um último exemplo de **uma classe herdando de uma outra classe e implementando duas interfaces**:

Conta.java (Super classe)

```java
public class Conta {
    protected String agencia;
    protected String numero;
    protected double saldo;

    public String getAgencia() {
        return agencia;
    }

    public void setAgencia(String agencia) {
        this.agencia = agencia;
    }

    public String getNumero() {
        return numero;
    }

    public void setNumero(String numero) {
        this.numero = numero;
    }

    public double getSaldo() {
        return saldo;
    }

    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }

    protected String exibirMensagem() {
        return "Bem vindo ao Internet Banking do Prof. Joseffe!";
    }

    Conta(){

    }

    Conta(String agencia, String numero, double saldo){
        this.agencia = agencia;
        this.numero = numero;
        this.saldo = saldo;
    }
```

```
    public void Depositar(double valor) {
        this.saldo += valor;
    }

    public void Depositar(double valor, String nomeDepositante) {
        this.saldo += valor;
    }

    public void Depositar(String nomeDepositante, double valor, String
telefoneDepositante) {
        this.saldo += valor;
    }
}
```

ContaBancaria.java (Interface)

```
public interface ContaBancaria {

    public void Sacar(float valor);
}
```

ContaBasica.java (Interface)

```
public interface ContaBasica {

    public void enviarMensagemAoGerente(String mensagem);
}
```

ContaCorrente.java (Classe comum)

```
public class ContaCorrente extends Conta implements ContaBancaria,
ContaBasica{
    private double chequeEspecial;

    public double getChequeEspecial() {
        return chequeEspecial;
    }

    public void setChequeEspecial(double chequeEspecial) {
        this.chequeEspecial = chequeEspecial;
    }

    public ContaCorrente() {
    }
```

```
    public ContaCorrente(String agencia, String numero, double saldo,
double chequeEspecial) {
        super(agencia, numero, saldo);
        this.chequeEspecial = chequeEspecial;
    }

    public void Sacar(float valor){
        this.saldo -= valor;
    }

    public void enviarMensagemAoGerente(String mensagem) {

    }
}
```

UML

https://docs.google.com/document/d/1lhfmbNxGS970dSVj6kyDqYbKUeEuxr1Q_blq6cRhnN0/edit