# Polimorfismo com classes abstratas

Padrões de Projeto Orientado a Objetos

Profa. Danielle Martin
Prof. Pedro Toledo
Universidade de Mogi das Cruzes

# Polimorfismo

Polimorfismo é a característica da orientação a objetos que desacopla a **referência** de um objeto da **implementação** do mesmo, permitindo que um **tipo comum** (abstrato ou genérico) possa utilizar um objeto de qualquer classe polimórfica sem diferenciação.

Exemplo:

```
Animal objeto = new Cachorro();
```

Variavel de referencia do tipo Animal

Objeto instancia do tipo Cachorro

# Polimorfismo

É possível implementar o polimorfismo usando:

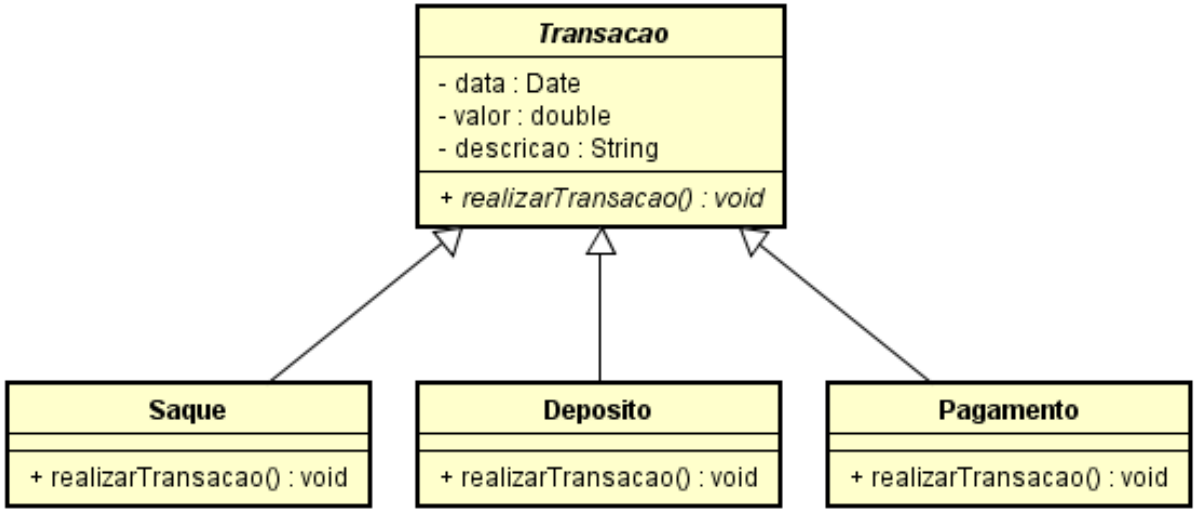
- Herança
- Classes abstratas
- Interfaces

# II. CLASSES ABSTRATAS

# Classe Abstrata

- Classe abstrata é uma classe que **não permite** que se crie instâncias (objetos) da mesma, por se tratar de um conceito, e não uma entidade.

- Podem ser usadas como superclasses de classes concretas.



# Método Abstrato

- **Método abstrato** é um método que não contém implementação. Apenas a assinatura dele é declarada. Sua implementação é obrigatória nas subclasses.

- Apenas classes abstratas podem conter elementos abstratos.
- Classes abstratas também podem ter elementos concretos.
- Usa-se a palavra reservada **abstract** para definir elementos abstratos.

```
public abstract class Transacao {

    private Date data;
    private double valor;
    private String descricao;

    public abstract void registrarTransacao();

    public String toString() {
        return "Transacao [data=" + data
                + ", valor=" + valor
                + ", descricao=" + descricao + "]";
    }
}
```

# Características de classes abstratas

- Toda operação abstrata utiliza **late binding**, ou acoplamento dinâmico (o método a ser executado é definido em tempo de execução).
- Subclasses de classes abstratas são **obrigadas** a sobrescrever todos os métodos abstratos da classe mãe.
- A herança de uma classe abstrata também é feita com a palavra **extends**.

# Polimorfismo com classes abstratas

```
public class Saque extends Transacao {

    public void registrarTransacao() {
        System.out.println("Saque realizado em " + getData());
        System.out.println("Descrição: " + getDescricao());
        System.out.println("Valor: R$" + getValor());
    }
}
```

```
public class Conta {

    private long numero;
    private double saldo;
    private ArrayList<Transacao> transacoes = new ArrayList<Transacao>();

    public Conta(long numero, double saldo) {
        this.numero = numero;
        this.saldo = saldo;
        System.out.println("Conta " + numero + ", saldo R$ " + saldo);
    }

    public void novaTransacao(Transacao transacao) {
        transacoes.add(transacao);
        transacao.registrarTransacao();
    }
}
```

# Polimorfismo com classes abstratas

```java
public class App {

    public static void main(String[] args) {

        Conta conta = new Conta(1234, 100.0);

        Saque saque = new Saque();
        saque.setData(new Date());
        saque.setDescricao("Saque ATM");
        saque.setValor(50.0);

        Deposito deposito = new Deposito();
        deposito.setData(new Date());
        deposito.setDescricao("Salario");
        deposito.setValor(5000.0);

        conta.novaTransacao(saque);
        conta.novaTransacao(deposito);
    }
}
```

Problems | Javadoc | Declaration | Annotations | Console

```
<terminated> App [Java Application] C:\Program Files (x86)\IBM\SDP\jdk\bin\javaw.exe
Conta 1234, saldo R$ 100.0
Saque realizado em Wed Feb 10 19:48:21 BRST 2016
Descrição: Saque ATM
Valor: R$50.0
Deposito realizado em Wed Feb 10 19:48:21 BRST 2016
Descrição: Salario
Valor: R$5000.0
```

# Construtores em classes abstratas

- É possível que uma classe abstrata tenha construtores definidos, mas não é possível instanciar objetos usando os mesmos. Se uma classe abstrata tiver construtores, eles devem ser obrigatoriamente acionados pelos costrutores das subclasses.

```
public abstract class CartaoBancario {

    private Conta conta;

    public CartaoBancario(Conta conta) {
        this.setConta(conta);
    }

    public abstract void registrarToken();

    public Conta getConta() {
        return conta;
    }

    public void setConta(Conta conta) {
        this.conta = conta;
    }
}
```

```
public class CartaoDeCredito extends CartaoBancario {

    public CartaoDeCredito(Conta conta) {
        super(conta);
    }

    public void registrarToken() {
        //registra o token de segurança do cartão
    }
}
```

# Construtores em classes abstratas

- Qual das seguintes instruções é válida para instanciação de um objeto?

```
public class CartaoDeCredito extends CartaoBancario {

    public CartaoDeCredito(Conta conta) {
        super(conta);
    }

    public void registrarToken() {
        //registra o token de segurança do cartão
    }
}
```

```
public abstract class CartaoBancario {

    private Conta conta;

    public CartaoBancario(Conta conta) {
        this.setConta(conta);
    }

    public abstract void registrarToken();

    public Conta getConta() {
        return conta;
    }

    public void setConta(Conta conta) {
        this.conta = conta;
    }
}
```

```
public class App {

    public static void main(String[] args) {

      Conta conta = new Conta(1234, 100.0);

      A) CartaoBancario cartao = new CartaoBancario();
      B) CartaoBancario cartao = new CartaoBancario(conta);
      C) CartaoBancario cartao = new CartaoDeCredito();
      D) CartaoBancario cartao = new CartaoDeCredito(conta);
    }
}
```

# Construtores em classes abstratas

- Qual das seguintes instruções é válida para instanciação de um objeto?

```
public class CartaoDeCredito extends CartaoBancario {

    public CartaoDeCredito(Conta conta) {
        super(conta);
    }

    public void registrarToken() {
        //registra o token de segurança do cartão
    }
}
```

```
public abstract class CartaoBancario {

    private Conta conta;

    public CartaoBancario(Conta conta) {
        this.setConta(conta);
    }

    public abstract void registrarToken();

    public Conta getConta() {
        return conta;
    }

    public void setConta(Conta conta) {
        this.conta = conta;
    }
}
```

```
public class App {

    public static void main(String[] args) {

      Conta conta = new Conta(1234, 100.0);

      A) CartaoBancario cartao = new CartaoBancario();
      B) CartaoBancario cartao = new CartaoBancario(conta);
      C) CartaoBancario cartao = new CartaoDeCredito();
      D) CartaoBancario cartao = new CartaoDeCredito(conta);
    }
}
```