Artigo

Polimorfismo, Classes abstratas e Interfaces: Fundamentos da POO em Java

Veja neste artigo alguns dos conceitos fundamentais da Programação Orientada a Objetos: Polimorfismo, Classes Abstratas e Interfaces. Serão mostradas, além da teoria, exemplos de implementação na linguagem Java.

O objetivo deste artigo é apresentar alguns componentes fundamentais da **Programação Orientada a Objetos:** Polimorfismo, Classes Abstratas e Interfaces. Serão apresentados os conceitos e implementações na linguagem Java para facilitar o entendimento.

Polimorfismo

O **polimorfismo** permite que classes abstratas consigam receber comportamentos através de classes concretas. Por exemplo, um dispositivo USB, podemos considerar que o USB seria uma classe abstrata enquanto os dispositivos (Pen Driver, Ipad, Câmeras, etc) seriam as classes concretas. Ou seja, o USB é uma

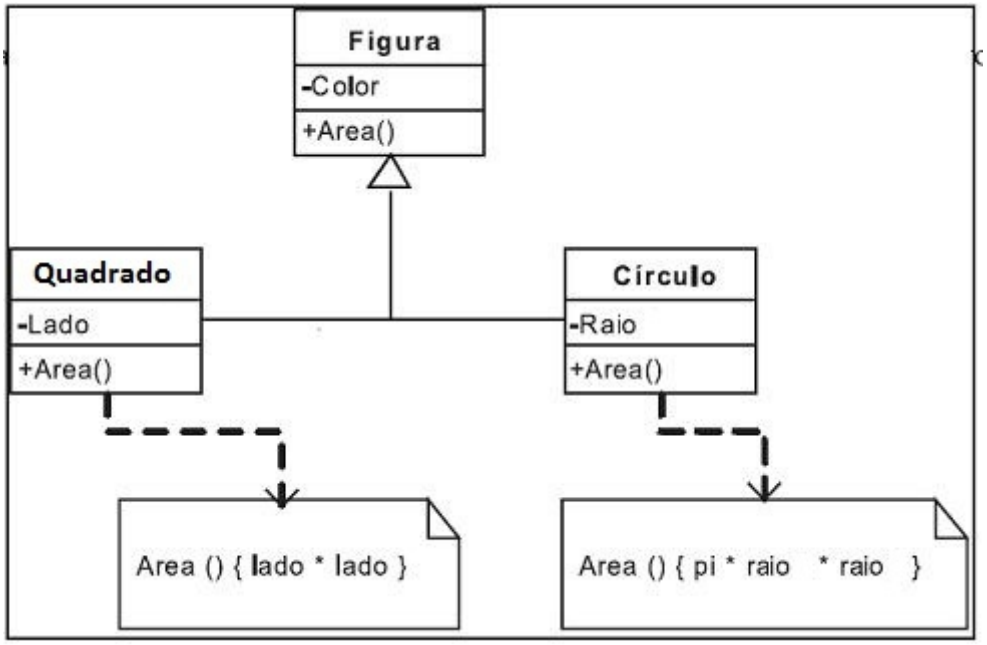

Figura 1. Exemplo de Polimorfismo

Sobrescrita de método

A seguir temos um exemplo onde uma classe Soma herda de uma outra OperacaoMatematica, sobrescrevendo seu método calcular para implementar a lógica adequada à sua função. A **sobrescrita de métodos** é muito utilizada na implementação de polimorfismo para que uma classe filha possa definir seu próprio comportamento, baseada na estrutura da classe mãe.

```
public class OperacaoMatematica {

    public double calcular(double x, double y){
        return 0;
    }
}
```

```
        return x + y;
    }
}
```

Listagem 2. Classe Soma herdando de OperacaoMatematica

Neste exemplo da Listagem 3 é mostrado porque o polimorfismo é conhecido como “possuir muitas formas”. No método “calculaOperacao” acontece o polimorfismo, pois quando invocado, o argumento do tipo “OperacaoMatematica” é diferente para cada operação.

```
public class TestaOperacaoMatematica {

    //EXECUTA A OPERACAO - COM POLIMORFISMO
    public static void calculaOperacao(OperacaoMatematica o, double x, double y){
        System.out.println(o.calcular(x, y));
    }

    public static void main(String[] args) {
        calculaOperacao (new Soma(), 2500, 200);
        calculaOperacao (new Multiplicacao(), 10, 10);
    }
}
```

Listagem 3. Classe Testadora de polimorfismo

Classes abstratas

Pode-se dizer que as **classes abstratas** servem como “modelo” para outras classes que dela herdem, não podendo ser instanciada por si só. Para ter um objeto de

Por exemplo, é definido que a classe “Animal” seja herdada pelas subclasses “Gato”, “Cachorro”, “Cavalo”, mas ela mesma nunca pode ser instanciada.

```
abstract class Conta {

    private double saldo;

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

    public double getSaldo() {
        return saldo;
    }

    public abstract void imprimeExtrato();

}
```

Listagem 4. Classe abstrata Conta

No exemplo da Listagem 5, o método “ `imprimeExtrato()` ” tem uma annotation conhecida como `@Override` , significando que estamos sobrescrevendo o método da superclasse. Entende-se em que na classe abstrata “Conta” os métodos que são abstratos têm um comportamento diferente, por isso não possuem corpo. Ou seja, as subclasses que estão herdando precisam desse método mas não de forma genérica, aonde permite inserir as particularidades de cada subclasse.

```
import java.text.SimpleDateFormat;
import java.util.Date;
```

```
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/aaaa HH:mm:ss");
        Date date = new Date();

        System.out.println("Saldo: "+this.getSaldo());
        System.out.println("Data: "+sdf.format(date));

    }
}
```

Listagem 5. Classe ContaPoupanca

```
public class TestaConta {
    public static void main(String[] args) {
        Conta cp = new ContaPoupanca();
        cp.setSaldo(2121);
        cp.imprimeExtrato();

    }
}
```

Listagem 6. Classe Testadora para classes abstratas

Interfaces

As **interfaces são padrões definidos através de contratos ou especificações.** Um contrato define um determinado conjunto de métodos que serão implementados nas classes que assinarem esse contrato. Uma interface é 100% abstrata, ou seja, os seus métodos são definidos como `abstract`, e as variáveis por padrão são sempre constantes (static final).

Uma interface é definida através da palavra reservada "`interface`". Para uma

questão, pois bem se sabe que uma classe pode ser herdada apenas uma vez, mas pode implementar inúmeras interfaces. As classes que forem implementar uma interface terão de adicionar todos os métodos da interface ou se transformar em uma classe abstrata, veja nos exemplos abaixo.

Na Listagem 7 a interface Conta tem seus métodos sem corpo, apenas com os parâmetros e o tipo de retorno.

```
interface Conta{
    void depositar(double valor);
    void sacar(double valor);
    double getSaldo();
}
```

Listagem 7. Declaração de uma interface

Neste exemplo da Listagem 8 os métodos são sobrepostos através da interface Conta.

```
public class ContaCorrente implements Conta {
    private double saldo;
    private double taxaOperacao = 0.45;

    @Override
    public void deposita(double valor) {
        this.saldo += valor - taxaOperacao;
    }

    @Override
    public double getSaldo() {
```




```
    }

}
```

Listagem 8. Classe Conta Corrente

```
public class ContaPoupanca implements Conta {
    private double saldo;

    @Override
    public void deposita(double valor) {
        this.saldo += valor;
    }

    @Override
    public double getSaldo() {
        return this.saldo;
    }

    @Override
    public void sacar(double valor) {
        this.saldo -= valor;

    }

}
```

Listagem 9. Classe ContaPoupanca com os métodos sobrepostos da interface Conta

O método “geradorConta”, da Listagem 10, mostra a entrada de um parâmetro do tipo Conta, essa função será útil para a saída de um resultado.

```
}
```

Listagem 10. Classe GeradorExtratos

Na Listagem 11 são instanciadas as classes e o gerador de extratos. Na classe "GeradorExtratos" é invocado o método que aceita como parâmetro um tipo de "Conta".

```
public class TestaContas {

    public static void main(String[] args) {

        ContaCorrente cc = new ContaCorrente();
        cc.deposita(1200.20);
        cc.sacar(300);

        ContaPoupanca cp = new ContaPoupanca();
        cp.deposita(500.50);
        cp.sacar(25);


        GeradorExtratos gerador = new GeradorExtratos();
        gerador.geradorConta(cc);
        gerador.geradorConta(cp);
    }

}
```

Listagem 11. Classe TestaContas

Conclusão

Por **Thiago**

Em 2012

Poste aqui sua dúvida ou comentário que nossa equipe responderá o mais rápido possível.

Marco Cunha




21

métodos comuns ou somente os métodos abstratos? Terceira pergunta: O construtor de uma classe abstrata

é privado? por isso que ela não pode ser estanciada?

há +1 ano

Santos DevMedia

Opa Marco, beleza?

Vamos lá:

1 - Sim, uma classe abstrata podem ter tantos métodos abstratos quanto métodos, vide o artigo que mostra um método para setar o valor do saldo ao atributo saldo da mesma classe.

2 - Somente os métodos abstratos. Os métodos comuns você pode utilizar normalmente.

3 - Sobre os construtores, apenas as classes que herdam das abstratas podem instancia-las. Normalmente, elas não possuem construtor (só o padrão que a JVM cria), mas você pode utilizar um construtor normalmente.

Abraços!!

há +1 ano