

Diogo Carminatti

Java

Programação Orientada a Objetos

*A Jornada do
Programador Jedi*

A Força da POO: Um Novo Começo

Entendendo os Fundamentos da Programação Orientada a Objetos

A Programação Orientada a Objetos (POO) é um paradigma de programação que utiliza "objetos" para representar dados e métodos. Esses objetos são instâncias de "classes", que podem ser vistas como plantas baixas ou moldes para criar objetos.

Exemplo

Imagine que você é um construtor de robôs. Cada robô que você constrói é baseado em um projeto específico. Nesse caso, o projeto é a classe, e cada robô construído a partir desse projeto é um objeto.



Os Jedi e Seus Sabres de Luz: Classes e Objetos

Construindo e Utilizando Classes em Java

Uma classe em Java define um novo tipo de dado, especificando o que os objetos dessa classe podem fazer (métodos) e quais informações eles podem armazenar (atributos).

Exemplo

```
public class Robo {  
    String nome;  
    int anoDeFabricacao;  
  
    void dizerOla() {  
        System.out.println("Olá, eu sou " + nome + "!");  
    }  
}
```

Neste exemplo, Robo é a classe, nome e anoDeFabricacao são atributos, e dizerOla é um método.

Um objeto dessa classe pode ser criado assim:

```
Robo robo1 = new Robo();  
robo1.nome = "R2-D2";  
robo1.anoDeFabricacao = 2020;  
robo1.dizerOla();
```

Protegendo a Ordem: Encapsulamento

Controlando o Acesso aos Dados

Encapsulamento é o princípio de esconder os detalhes internos de um objeto e apenas expor o que é necessário. Isso é feito utilizando modificadores de acesso (private, public, etc.) e métodos acessores (getters e setters).

Exemplo

```
public class Robo {  
    private String nome;  
    private int anoDeFabricacao;  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
  
    public int getAnoDeFabricacao() {  
        return anoDeFabricacao;  
    }  
  
    public void setAnoDeFabricacao(int anoDeFabricacao) {  
        this.anoDeFabricacao = anoDeFabricacao;  
    }  
}
```

A Linhagem Skywalker: Herança

Reutilizando Código com Herança em Java

Herança é um mecanismo que permite criar uma nova classe a partir de uma classe existente. A nova classe (subclasse) herda os atributos e métodos da classe existente (superclasse).

Exemplo

```
public class Robo {  
    private String nome;  
    private int anoDeFabricacao;  
  
    // getters e setters...  
}  
  
public class RoboDeCombate extends Robo {  
    private String arma;  
  
    public String getArma() {  
        return arma;  
    }  
  
    public void setArma(String arma) {  
        this.arma = arma;  
    }  
}
```

Os Lados da Força: Polimorfismo

Entendendo o Polimorfismo em Java

Polimorfismo permite que uma classe ou método tenha muitas formas. Em Java, isso pode ser feito através de métodos sobrecarregados (overloading) ou sobrescritos (overriding).

Exemplo

```
public class Robo {  
    public void mover() {  
        System.out.println("O robô está se movendo");  
    }  
}  
  
public class RoboDeCombate extends Robo {  
    @Override  
    public void mover() {  
        System.out.println("O robô de combate está atacando");  
    }  
}
```

Conselho Jedi: Interfaces

Definindo Contratos em Java

Uma interface em Java é um contrato que uma classe pode implementar. Ela define métodos que a classe deve implementar, mas não fornece a implementação desses métodos.

Exemplo


```
public interface Atacante {  
    void atacar();  
}  
  
public class RoboDeCombate implements Atacante {  
    @Override  
    public void atacar() {  
        System.out.println("Robo de combate atacando");  
    }  
}
```

Segredos da Força: Abstração

Simplificando a Complexidade com Abstração em Java

Abstração é o processo de esconder os detalhes complexos e mostrar apenas a funcionalidade essencial do objeto. Em Java, isso pode ser feito através de classes abstratas.

Exemplo



```
public abstract class Robo {
    public abstract void mover();
}

public class RoboDeCombate extends Robo {
    @Override
    public void mover() {
        System.out.println("Robo de combate atacando");
    }
}
```


Tornando-se um Mestre Jedi da POO

Avançando para os Próximos Níveis

Neste ebook, cobrimos os conceitos básicos da Programação Orientada a Objetos em Java. Agora, você deve praticar esses conceitos e explorar tópicos mais avançados, como design patterns, desenvolvimento de aplicações em grande escala e frameworks populares como Spring.