PROGRAMAÇÃO ORIENTADA A OBJETOS-T01-2024-1

📍 [Checkout de Presença] Módulo 3 - Pilares da Programação Orientada a Objetos.

Professor Especialista: Anderson Viçoso de Araújo.

Fecha: segunda, 6 mai 2024, 23:59

Disciplinas

Formação Profissional em Computação Concluído

Materiais

Neste módulo, você aprendeu sobre os quatro conceitos chamados de “Pilares da POO”:

- Polimorfismo, Abstração, Encapsulamento e Herança.

Agora, vamos ver se tudo ficou bem entendido?

Para esta atividade você deve apresentar um exemplo de código Java que usa cada um dos conceitos:

Polimorfismo, Abstração, Encapsulamento e Herança.

Envie um arquivo PDF com as classes criadas. No mesmo PDF coloque uma página adicional explicando os conceitos aplicados em cada classe.

Polimorfismo:

```
// interface Animal
interface Animal {
    void fazerSom();
}

// implementação da interface Cachorro
class Cachorro implements Animal {
    public void fazerSom() {
        System.out.println("Au au.");
    }
}

// implementação da interface Gato
class Gato implements Animal {
    public void fazerSom() {
```

```
        System.out.println("Miau.");
    }
}

// método principal
public class Main {
    public static void main(String[] args) {
        // Polimorfismo: mesmo método fazendo coisas diferentes dependendo do objeto
        Animal animal1 = new Cachorro();
        Animal animal2 = new Gato();
    }

    animal1.fazerSom(); // Au au.
    animal2.fazerSom(); // Miau.
}
```

Abstração:

```
// classe abstrata Veiculo
abstract class Veiculo {
    int velocidade;

    // método abstrato acelerar/frear
    abstract void acelerar();
    abstract void frear();
}

// classe concreta para Carro que herda do Veiculo
class Carro extends Veiculo {
    void acelerar() {
        System.out.println("Velocidade aumentando...");
    }

    void frear() {
        System.out.println("Reduzindo a velocidade...");
    }
}

// método principal
public class Main {
    public static void Main(String[] args) {
        // Abstração: Oculta a implementação específica.
        Veiculo veiculo = new Carro();

        veiculo.acelera(); // Velocidade aumentando...
        veiculo.frear(); // Reduzindo a velocidade...
    }
}
```

Encapsulamento:

```
// class ContaBancaria encapsulada
class ContaBancaria {
    private double saldo;
    private String numeroConta;

    // método get e set para saber o saldo
    public double getSaldo() {
        return saldo;
    }

    // método para depositar dinheiro
    public void depositar(double valor) {
        saldo += valor;
    }

    // método para sacar dinheiro
    public void sacar(double valor) {
        saldo += valor;
    }

    // método get e set para saber o numero da conta
    public String getNumeroConta() {
        return numeroConta;
    }

    // método para definir número da conta
    public void setNumeroConta(String numeroConta) {
        this.numeroConta = numeroConta;
    }
}

// método principal
public class Main {
    public static void main(String[] args) {
        // Encapsulamento: protege os dados e a implementação da classe
        ContaBancaria conta = new ContaBancaria();

        conta.depositar(1000);
        conta.sacar(500);

        System.out.println("Seu saldo é: " + conta.getSaldo()); // Saldo atual: 500.0
    }
}
```

Herança:

```
// classe pai
class Animal {
    void comer() {
        System.out.println("O animal está comendo...");
    }
}

// subclasse que herda da classe Animal
```

```
class Cachorro extends Animal {
    void latir() {
        System.out.println("O cachorro está latindo...")
    }
}

// método principal
public class Main {
    public static void main(String[] args) {
        // Herança: Reutilização de código e especialização de classes
        Cachorro cachorro = new Cachorro();

        cachorro.comer(); // O animal está comendo...
        cachorro.latir(); // O cachorro está latindo...
    }
}
```

Conceitos Aplicados em Cada Classe:

Polimorfismo:

Nesta classe, temos a interface chamada Animal que define um método fazerSom(). Duas classes, Cachorro e Gato, implementam essa interface e fornecem suas próprias implementações do método fazerSom(), produzindo diferentes sons. No método main(), o polimorfismo está ao criar instâncias de Cachorro e Gato e chamar o método fazerSom(). Mesmo método é capaz de fazer coisas diferentes dependendo do objeto ao qual é referenciado.

Abstração:

Nesta classe, temos a classe abstrata Veiculo que define métodos abstratos acelerar() e frear(). A classe concreta Carro herda de Veiculo e fornece implementações específicas para esses métodos. No método main(), mostra a abstração ao criar uma instância de Carro e chamar os métodos acelerar() e frear(), sem precisar conhecer os detalhes de implementação específicos de Carro.

Encapsulamento:

Nesta classe, temos a classe ContaBancaria que encapsula o saldo e o número da conta como atributos privados. Métodos getters e setters são fornecidos para acessar e modificar esses atributos de forma controlada. No método main(), demonstra o encapsulamento ao criar uma instância de ContaBancaria e usar métodos depositar() e sacar() para interagir com o saldo sem acessar diretamente os atributos privados.

Herança:

Nesta classe, temos uma classe Animal que define um método comer(). A classe Cachorro herda de Animal e adiciona um método latir(). No método main(), demonstra a herança ao criar uma instância de Cachorro e chamar tanto o método comer() da classe Animal quanto o método latir() adicionado pela classe Cachorro. A herança permite a reutilização de código e a especialização de classes.
