

1a AVALIAÇÃO BIMESTRAL - 4SA - POO 2024

- **Curso:** ENGENHARIA DE SOFTWARE
- **Série:** 4 S
- **Turma:** A
- **Turno:** NOTURNO
- **Professor(a):** JOÃO CHOMA NETO
- **Horário:** 19:00 – 20:30
- **Disciplina:** PROGRAMAÇÃO ORIENTADA A OBJETOS
- **BIMESTRE:** 1
- **VALOR:** 5,0 PONTOS

INSTRUÇÕES PARA REALIZAÇÃO DA PROVA:

- É vedado, durante a prova, o porte e/ou o uso de aparelhos sonoros, fonográficos, de comunicação ou de registro eletrônico ou não, tais como: notebooks, celulares, tablets e similares.
- A prova é individual e sem consulta, é permitido o git da disciplina.
- É obrigatória a permanência do acadêmico 50 MINUTOS em sala de aula após o início da prova
- Não será permitida a entrada na sala de aula após 10 minutos do início da prova.
- É obrigatória a assinatura da lista de presença impressa na qual constam RA, nome e curso.
- O valor de cada questão está ao lado da mesma.
- Em caso de qualquer irregularidade comunicar ao Professor ou fiscal de sala.
- Ao término da prova, levante o braço e aguarde atendimento do professor ou do fiscal.

E-mail *

ra-23059884-2@alunos.unicesumar.edu.br

NOME DO ACADÊMICO

Gabriel Mendonça Mello Teixeira

REGISTRO ACADÊMICO - inserir o - *

23059884-2

PROBLEMA A SER RESOLVIDO

Problema: Sistema de Gerenciamento de Vagas de Estacionamento em um Shopping

Você foi contratado para desenvolver um sistema de gerenciamento de vagas de estacionamento para um shopping que deseja otimizar a utilização de suas vagas e fornecer um serviço mais eficiente para seus clientes.

Requisitos do Sistema:**1. Tipos de Vagas:**

- O shopping possui diferentes tipos de vagas:
 - **Vaga Regular:** Vaga comum para veículos de pequeno porte.
 - **Vaga VIP:** Reservada para clientes VIP que possuem cartão de fidelidade.
 - **Vaga para Veículos Elétricos:** Exclusiva para veículos que necessitam de carregamento.
 - **Vaga para Deficientes:** Exclusiva para pessoas com deficiência.

2. Gerenciamento de Entrada e Saída:

- O sistema deve registrar a entrada e a saída dos veículos, associando-os ao tipo de vaga utilizado.
- O tempo de permanência no estacionamento deve ser calculado e o valor a ser cobrado deve seguir as seguintes regras:
 - **Vaga Regular:** Tarifa de R\$ 10,00 por hora.
 - **Vaga VIP:** Tarifa de R\$ 5,00 por hora para clientes VIP.
 - **Vaga para Veículos Elétricos:** Tarifa de R\$ 12,00 por hora, com uma tarifa adicional pelo uso do carregador elétrico.
 - **Vaga para Deficientes:** Gratuita, mas requer comprovação de documentação.

3. Reservas de Vagas:

- Clientes VIP podem reservar vagas VIP com antecedência.
- Veículos elétricos podem reservar vagas com carregador elétrico.
- Vagas regulares não podem ser reservadas, sendo ocupadas por ordem de chegada.

4. Gerenciamento de Fidelidade:

- Clientes VIP têm acesso a benefícios como tarifas reduzidas.

Requisitos Funcionais:

- O sistema deve permitir:
 - Registrar entrada e saída de veículos, associando o tipo de vaga utilizada e calculando o tempo de permanência.
 - Gerenciar reservas de vagas VIP e para veículos elétricos.
 - Implementar diferentes comportamentos para os tipos de vaga, sem duplicação de código (aplicando polimorfismo e herança).

Regras de Negócio:

1. **Vaga VIP:** Somente clientes VIP podem reservar essas vagas, e eles têm acesso a tarifas reduzidas.
2. **Vaga para Deficientes:** Gratuita, mas exige comprovação de documentação no momento da entrada.
3. **Vaga para Veículos Elétricos:** Aplica-se uma tarifa adicional para o uso do carregador, além da tarifa de estacionamento.

Requisitos para avaliação:

1. **IMPLEMENTE** uma estrutura de classes que represente os diferentes tipos de eventos e as funcionalidades compartilhadas entre eles (2,0 pontos).
2. **ELABORE** um fluxo de execução no método **main** que represente o sistema (1,0 ponto).
3. **USE** boas práticas de orientação a objetos, incluindo herança E/OU encapsulamento E/OU polimorfismo (2,0 pontos).
4. Toda a lógica deve ser implementada.

Pontos de Avaliação:

- Estrutura e clareza do código.
- Aplicação correta dos conceitos de orientação a objetos (encapsulamento, herança, polimorfismo).
- Capacidade de implementar lógica de negócios.

Para apoiar a implementação segue casos de teste.

Não é obrigatório a utilização dos testes.

Esses testes não cobrem o sistema por completo.

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

class EstacionamentoTest {

    Estacionamento estacionamento = new Estacionamento();

    @Test
    void testClienteVIPodeReservarVagaVIP() {
        Cliente clienteVIP = new Cliente(true);
        Vaga vagaVIP = new VagaVIP();

        boolean resultadoReserva = estacionamento.reservarVaga(vagaVIP, clienteVIP);
        assertTrue(resultadoReserva);
    }

    @Test
    void testClienteNaoVIPNaoPodeReservarVagaVIP() {
        Cliente clienteNaoVIP = new Cliente(false);
        Vaga vagaVIP = new VagaVIP();

        boolean resultadoReserva = estacionamento.reservarVaga(vagaVIP, clienteNaoVIP);
        assertFalse(resultadoReserva);
    }

    @Test
    void testClienteComVeiculoEletricoPodeReservarVagaEletrica() {
        Cliente clienteEletrico = new Cliente(true);
        Vaga vagaEletrica = new VagaEletrica();

        boolean resultadoReserva = estacionamento.reservarVaga(vagaEletrica, clienteEletrico);
        assertTrue(resultadoReserva);
    }

    @Test
    void testClienteNaoVIPodeReservarVagaEletrica() {
        Cliente clienteNaoVIP = new Cliente(false);
        Vaga vagaEletrica = new VagaEletrica();

        boolean resultadoReserva = estacionamento.reservarVaga(vagaEletrica, clienteNaoVIP);
        assertTrue(resultadoReserva);
    }
}
```

```
@Test
void testNaoPodeReservarVagaRegular() {
    Cliente clienteVIP = new Cliente(true);
    Vaga vagaRegular = new VagaRegular();

    boolean resultadoReserva = estacionamento.reservarVaga(vagaRegular, clienteVIP);
    assertFalse(resultadoReserva);
}
}
```

ESPAÇO 01 - PARA INCLUIR AS CLASSES *

```
public class Cliente {
    private boolean vip;
    private boolean eletrico;
    private boolean deficiente;

    public Cliente(boolean vip, boolean eletrico, boolean deficiente) {
        this.vip = vip;
        this.eletrico = eletrico;
        this.deficiente = deficiente;
    }

    public boolean isVip() {
        return vip;
    }

    public boolean isEletrico() {
        return eletrico;
    }

    public boolean isDeficiente() {
        return deficiente;
    }

    public boolean podeReservarVagaVIP() {
        return vip;
    }

    public boolean podeUsarVagaEletrica() {
        return eletrico;
    }
}
```

ESPAÇO 02 - PARA INCLUIR AS CLASSES

```
public class Estacionamento {
    public static void main(String[] args) {
        Cliente clienteVIP = new Cliente(true, false, false);
        Cliente clienteEletrico = new Cliente(false, true, false);
        Cliente clienteRegular = new Cliente(false, false, false);

        VagaVIP vagaVIP = new VagaVIP(5);
        VagaRegular vagaRegular = new VagaRegular(10);
        VagaEletrica vagaEletrica = new VagaEletrica(8);

    }

    public static boolean reservarVaga(Vaga vaga, Cliente cliente) {
        if (cliente.podeReservarVagaVIP() && vaga instanceof VagaVIP && vaga.isPodeReservar()) {
            System.out.println("Vaga VIP Reservada");
            return true;
        } else if (cliente.isEletrico() && vaga instanceof VagaEletrica && vaga.isPodeReservar()) {
            System.out.println("Vaga Eletrica Reservada");
            return true;
        } else if (vaga instanceof VagaRegular && vaga.isPodeReservar()) {
            System.out.println("Vaga Regular Reservada");
            return true;
        } else if (vaga instanceof VagaPDC && cliente.isDeficiente() && ((VagaPDC) vaga).isDocumento() &&
vaga.isPodeReservar()) {
            System.out.println("Vaga PDC Reservada");
            return true;
        } else {
            System.out.println("Vaga Não pode ser reservada");
            return false;
        }
    }
}

abstract class Vaga {
    private int tempoDePermanencia;
    private double tarifa;
    private boolean podeReservar;

    public Vaga(int tempoDePermanencia) {
        this.tempoDePermanencia = tempoDePermanencia;
        this.podeReservar = true;
    }

    public int getTempoDePermanencia() {
        return tempoDePermanencia;
    }

    public double getTarifa() {
```



```
        return tarifa;
    }

    public boolean isPodeReservar() {
        return podeReservar;
    }

    public void setPodeReservar(boolean podeReservar) {
        this.podeReservar = podeReservar;
    }

    public void setTarifa(double tarifa) {
        this.tarifa = tarifa;
    }
}
```

ESPAÇO 03 - PARA INCLUIR AS CLASSES

```
class VagaEletrica extends Vaga {
    public VagaEletrica(int tempoDePermanencia) {
        super(tempoDePermanencia);
        setTarifa(12);
    }

    @Override
    public void setTarifa(double tarifa) {
        super.setTarifa(tarifa);
        System.out.println("Valor R$ " + tarifa);
    }
}

public class VagaPDC extends Vaga {
    private boolean documento;

    public VagaPDC(int tempoDePermanencia, boolean documento) {
        super(tempoDePermanencia);
        this.documento = documento;
        setTarifa(0);
    }

    public boolean isDocumento() {
        return documento;
    }

    @Override
    public void setTarifa(double tarifa) {
        super.setTarifa(tarifa);
        System.out.println("Valor R$ " + tarifa);
    }
}
```

ESPAÇO 04 - PARA INCLUIR AS CLASSES

```
class VagaRegular extends Vaga {  
    public VagaRegular(int tempoDePermanencia) {  
        super(tempoDePermanencia);  
        setTarifa(10);  
    }  
  
    @Override  
    public void setTarifa(double tarifa) {  
        super.setTarifa(tarifa);  
        System.out.println("Valor R$ " + tarifa);  
    }  
}
```

ESPAÇO 05 - PARA INCLUIR AS CLASSES

```
class VagaVIP extends Vaga {  
    public VagaVIP(int tempoDePermanencia) {  
        super(tempoDePermanencia);  
        setTarifa(5);  
    }  
  
    @Override  
    public void setTarifa(double tarifa) {  
        super.setTarifa(tarifa);  
        System.out.println("Valor R$ " + tarifa);  
    }  
}
```

Este conteúdo não foi criado nem aprovado pelo Google.

Google Formulários