

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 *

allangabrielcontato@gmail.com

NOME DO ACADÊMICO

Allan Gabriel da Silva

REGISTRO ACADÊMICO - inserir o - *

23211467-2

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 *

```
package org.example;
```

```
import java.sql.SQLOutput;
```

```
import java.util.Scanner;
```

```
// Press Shift twice to open the Search Everywhere dialog and type `show whitespaces`,  
// then press Enter. You can now see whitespace characters in your code.
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        Scanner ler = new Scanner(System.in);
```

```
        Estacionamento estacionamento = new Estacionamento();
```

```
        boolean resultadoReserva;
```

```
        int horas;
```

```
        int opcao = 0;
```

```
        do {
```

```
            System.out.println("Bem vinda(o)");
```

```
            System.out.println("Digite a opção desejada");
```

```
            System.out.println("1 - Cliente Vip");
```

```
            System.out.println("2 - Cliente não Vip");
```

```
            System.out.println("3 - Sair");
```

```
            opcao = ler.nextInt();
```

```
        switch (opcao){
```

```
            case 1:
```

```
                Cliente clienteVip = new Cliente(true);
```

```
                break;
```

```
            case 2:
```

```
                Cliente cliente = new Cliente(false);
```

```
                break;
```

```
            case 3:
```

```
                System.out.println("Saindo");
```

```
                break;
```

```
        }
```

```
        System.out.println("Qual vaga vc quer?");
```

```
        System.out.println("1 - Vaga Vip");
```

```
        System.out.println("2 - Vaga Regular");
```

```
        System.out.println("3 - Vaga Eletrica");
```

```
        System.out.println("4 - Vaga Deficiente");
```

```
        System.out.println("5 - Sair");
```

```
        opcao = ler.nextInt();
```

```
        System.out.println("Quantas horas?");
```

```
        horas = ler.nextInt();
```



```
switch (opcao){
    case 1:
        Vaga vagaVIP = new VagaVIP();

        resultadoReserva = estacionamento.reservarVaga(vagaVIP, );
    case 2:
        Vaga vagaRegular = new VagaRegular();
        Cliente cliente = new Cliente(false);
        resultadoReserva = estacionamento.reservarVaga(vagaRegular, cliente);
        System.out.println("Vaga reservada no valor de " + vagaRegular.calcularTarifa(horas));
        break;
    case 3:
        Vaga vagaEletrica = new VagaEletrica();
        Cliente cliente1 = new Cliente(false);
        resultadoReserva = estacionamento.reservarVaga(vagaEletrica, cliente1);
        System.out.println("Vaga reservada no valor de " + vagaEletrica.calcularTarifa(horas));
        break;
    case 4:
        Vaga vagaDeficiente = new VagaDeficiente();
        Cliente cliente3 = new Cliente(false);
        resultadoReserva = estacionamento.reservarVaga(vagaDeficiente, cliente3);
        System.out.println("Vaga reservada no valor de " + vagaDeficiente.calcularTarifa(horas));
        break;
    case 5:
        System.out.println("Saindo");
        break;
}
}while (opcao != 5);

} while (opcao != 3);

}
}
```

ESPAÇO 02 - PARA INCLUIR AS CLASSES

```
package org.example;
```

```
public class VagaVIP extends Vaga{  
    private int tarifa = 0;  
}
```

```
package org.example;
```

```
public class VagaRegular extends Vaga {  
    private int tarifa = 10;  
}
```

```
package org.example;
```

```
public class VagaEletrica extends Vaga {  
    private int tarifa = 14;  
  
}
```

```
package org.example;
```

```
public class VagaDeficiente extends Vaga{  
    private int tarifa = 0;  
  
}
```

```
package org.example;
```

```
public class Vaga {  
    private int tarifa;  
    private int horas;  
  
    public int calcularTarifa(int horas){  
        return horas * tarifa;  
    }  
}
```

ESPAÇO 03 - PARA INCLUIR AS CLASSES

```
package org.example;

import java.util.Scanner;

public class Estacionamento {

    private int[] vagas = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20};
    private int[] vagasEletricas = {1, 2, 3, 4, 5};

    Scanner ler = new Scanner(System.in);
    private int opcao = 0;

    public boolean verifVip(Cliente i){
        if(i.isEhvip()){
            return true;
        }else {
            return false;
        }
    }

    public boolean reservarVaga(Vaga b, Cliente i) {
        if(vagas >= 1){
            //vip chato
            if(verifVip(i)){
                System.out.println("Qual vaga vc quer?");
                do {
                    System.out.println("Disponíveis:");
                    for (int a = 0; a == vagas.length; a++) {
                        System.out.println(vagas[a]);
                    }
                    opcao = ler.nextInt();
                }while (opcao!=vagas.length);
                System.out.println("Vaga reservada");
                vagas[opcao] = 0;
                return true;
            }else {
                System.out.println("Vc não é VIP :/");
                return false;
            }
        }
        if (vaga)
    }

}
```

ESPAÇO 04 - PARA INCLUIR AS CLASSES

```
package org.example;
```

```
public class Cliente {  
    private boolean ehvip;  
  
    public Cliente(boolean ehvip) {  
        this.ehvip = ehvip;  
    }  
  
    public boolean isEhvip() {  
        return ehvip;  
    }  
}
```

ESPAÇO 05 - PARA INCLUIR AS CLASSES

OI CHOMA, TUDO BEM?

olha isso aqui :

Este conteúdo não foi criado nem aprovado pelo Google.

Google Formulários