**Universidade Federal do Paraná  
Curso de Engenharia Elétrica**

**TE 353 - Programação Orientada a Objetos**

## Formulário de entrega PjBL1 – Controle de Estacionamento

**Equipe**

Estudante 1:

Estudante 2:

Estudante 3:

**Link para o vídeo**

Importante:

* O vídeo precisa estar **público**
* Pode estar **não listado**: não aparece no canal, apenas quem tem o link acessa.
* Faça um teste com o link em um navegador não logado no YouTube.

Link para o Vídeo:

**Autoavaliação**

Para cada um dos itens da tabela da próxima página, **escolha marcando um X naquele que mais se adapta ao resultado de sua implementação**.

|  |
| --- |
| A soma dos pontos é: |

|  |
| --- |
| *Comentários a respeito da implementação (o que gostaria de destacar de interessante, algo extra que foi feito e merece uma observação, alguma dificuldade que foi resolvida/aprendida etc.)* |
|  |

**Grade de autoavaliação do trabalho em relação a especificação**

|  |
| --- |
| **Reuso adequado dos métodos *static* na main (classe Principal)**  ( ) 1.0 - Sim, todos os métodos da main estão bem estruturados, com reuso, sendo static e sem duplicação de trechos de código similares  ( ) 0.6 – Mais ou menos, há reutilização e estruturação, porém há código repetido que demanda refatoração e melhor estruturação  ( ) 0.4 – Não. A classe main funciona, porém o código ficou extenso e bem desorganizado, com pouco reuso de código. |
| ***Encapsulamento e métodos das classes Carro, Marca e Modelo empregados adequadamente, com sobrecarga e redefinição, conforme o caso:*** |
| Classe Carro (construção, métodos e uso):  ( ) 1.0 – Sim, a classe Carro possui os atributos pedidos com construtores, getters/setters, toString e método compareTo corretamente implementados e em funcionamento  ( ) 0.6 – Mais ou menos, a classe Carro possui seus métodos, mas alteramos alguns de seus atributos  ( ) 0.4 – Não, a classe Carro funciona, porém alteramos bastante, mudando a especificação, como por exemplo o atributo modelo |
| Classe Marca (construção, métodos e uso):  ( ) 1.0 – Sim, a classe Marca possui os atributos pedidos com construtores, getters/setters e toString, bem como o atributo modelos mantém uma lista de modelos desta Marca  ( ) 0.6 – Mais ou menos, a classe Modelo possui seus métodos, mas alteramos alguns de seus atributos, incluindo a organização dos modelos da Marca  ( ) 0.4 – Não, a classe Modelo funciona, porém alteramos bastante, mudando a especificação da classe para se adaptar a nossa solução (sem ArrayList para modelos e outras mudanças) |
| Classe Modelo (construção, métodos e uso):  ( ) 1.0 – Sim, a classe Modelo possui os atributos pedidos com construtores, getters/setters e toString  ( ) 0.6 – Mais ou menos, a classe Modelo possui seus métodos, mas alteramos alguns de seus atributos  ( ) 0.4 – Não, a classe Modelo funciona, porém alteramos bastante, mudando a especificação |
| **Funcionalidade entrada**  ( ) 1.0 – Opera corretamente, pegando a data e hora do sistema, lendo a placa e permitindo a escolha do modelo a partir da lista de modelos disponíveis bem como aloca a primeira vaga livre do vetor  ( ) 0.6 – Le os dados corretamente, porém o modelo não vem da lista de modelos ou não pega a primeira vaga livre  ( ) 0.4 – Opera diferente do especificado (funciona de outro jeito do esperado).  ( ) 0.0 – Não há esta possibilidade |
| **Funcionalidade saída**  ( ) 1.0 – Opera corretamente, calcula o preco corretamente e incluir o carro que sai na lista histórico  ( ) 0.6 – Opera com alguma funcionalidade diferente do especificado (exemplo: cálculo do preço ou inclusão no histórico ou outro aspecto)  ( ) 0.4 – Opera diferente do especificado (funciona de outro jeito do esperado)  ( ) 0.0 – Não há esta possibilidade |
| **Funcionalidade cadastro de marca**  ( ) 1.0 – Opera corretamente como opção de menu, incluindo a marca no Array de Marcas definido da Main como um objeto  ( ) 0.6 – Funciona, porém de outra forma, não usando o objeto Marca dentro do ArrayList static  ( ) 0.0 – Não há esta possibilidade |
| **Funcionalidade cadastro de modelo**  ( ) 1.0 – Opera corretamente como opção de menu, incluindo a modelo como novo objeto em uma instância de Marca que está presente no Array definido da Main  ( ) 0.6 – Funciona, porém de outra forma, não usando o objeto Marca dentro do ArrayList static  ( ) 0.0 – Não há esta possibilidade |
| **Funcionalidade incluir modelo e/ou marca quando da entrada de um veículo** ( ) 1.0 – Sim, é possível incluir um modelo e/ou marca quando ao cadastrar um veículo que entra, percebe-se que seu modelo ainda não está disponível na lista cadastrada ( ) 0.6 – Mais ou menos, há uma função, porém, não estão usando os objetos e listas determinadas na especificação  ( ) 0.0 – Não há esta possibilidade |
| **Funcionalidade mostrar ocupação do estacionamento (100 posições do vetor)**  ( ) 1.0 – Sim, é possível consultar o relatório, para cada vaga é mostrado Livre ou Ocupado, quando ocupado, mostra-se também o modelo e a placa do carro e dd/mm/aaaa - hh:mm da entrada  ( ) 0.6 – Mais ou menos, lista as vagas, mas com diferenças do que foi especificado (sem todas as informações pedidas)  ( ) 0.0 – Não há esta possibilidade ou a listagem não corresponde aos carros de fato estacionado |
| **Funcionalidade relatório gerencial consolidado a partir de data fornecida pelo usuário (ordenados por ordem de chegada)**  ( ) 1.0 – Sim, é possível consultar o relatório a partir do array histórico conforme especificação, sendo exibido por ordem de chegada (usando o campo horaEntrada para ordenação natural, interface Comparable<Carro> ( ) 0.6 – Mais ou menos, lista um relatório parcial, com diferenças do que foi especificado  ( ) 0.0 – Não há esta possibilidade |