PRONTUÁRIOS: \_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_ Início: 28/01/2021 Término: 30/01/2021 (14h00)

|  |  |
| --- | --- |
| * A prova é individual e não é permitido o compartilhamento código-fonte. * Atribui-se nota zero à prova em desacordo com o item acima. * A nota final da prova poderá ser alterada após arguição pelo professor. | * O projeto deve ser nomeado da seguinte forma: PRONTUARIOS\_P2, com “SC”. * Crie e copie um PDF preenchido da prova para dentro do projeto, compacte o projeto todo como um zip e envie pelo Moodle. * Não envie apenas as classes!!! |

Essa prova é um self-service! Você deve propor o contexto no qual demonstrará suas habilidades de Orientação a Objetos usando a linguagem Java e o banco de dados SQLite. O problema proposto deverá conter ao menos quatro classes no modelo, sendo que entre elas deverá haver um relacionamento do tipo “um para um", outro do tipo “um para muitos” e ao menos uma "herança". A classe herdada deverá possuir um método a ser sobrescrito na(s) sua(s) subclasse(s), de forma a permitir a realização de comportamento polimórfico. Você pode indicar quais atributos cada classe deverá conter, bem como os tipos de dados mais adequados a cada atributo. Entretanto, é necessário incluir, em qualquer uma das classes, ao menos um campo do tipo LocalDate e uma enumeração (Enum). Indique nos campos a seguir o contexto abordado na prova e suas características principais.

**Breve descrição do contexto (até três linhas):**

Uma concessionária de veículos vende tanto carro, quanto motos e necessita de um sistema que realiza a venda desses produtos e também controla o estoque da concessionária. Motos podem ser financiadas em 60x. Carros possuem como atributo adicional o número de portas e podem ser financiados em no máximo 36x. Cada venda negocia um único veículo e é realizada por um único vendedor.

**Classes e Atributos:**

Nome da classe: atributo 1 (tipo do atributo), ... atributo N (tipo do atributo). <<Exemplo>>

Venda: id(Integer), veiculo(Veiculo), tipoPagamento(Enum ), valor(Double), numParcelas(Integer), valorParcela(Double).

Vendedor: id(Integer), nome (String), e-mail (String), veiculosVendidos (List<Veículo>)

Veiculo: Integer id, marca(String),  String modelo, preco(Double), quantidade(Integer), ano(Integer)

Carro: numeroPortas (Integer) = polimórfico (getLimiteParcelas)

Moto:  = polimórfico (getLimiteParcelas)

**Relacionamentos:**

Cada objeto da classe Venda tem um objeto da classe Veiculo.

Cada objeto da classe Vendedor tem muitos objetos da classe Veiculo.

A classe Carro e Moto herdam a classe Veiculo.

A proposta de contexto deverá ser previamente aprovada pelo professor. A partir do contexto aprovado, realize as seguintes atividades:

|  |  |  |
| --- | --- | --- |
| **#** | **Descrição** | **Pontuação** |
| 1 | Crie classes representando um modelo para o problema proposto na especificação. Utilize tipos de dados, relacionamentos e modificadores de acesso adequados em sua solução. | 1 pt. |
| 2 | Crie interfaces gráficas usando arquivos FXML e componentes pertinentes à solução do problema proposto. No local mais adequado, utilize ao menos uma vez os seguintes componentes: TableView, ComboBox e DatePicker. Deve haver também, em ao menos um local, um campo de texto que permita filtrar elementos da TableView por atributos (String) nela contidos. As interfaces gráficas devem permitir a realização de operações CRUD para todas as classes. | 1,5 pt. |
| 3 | Crie classes para carregar os arquivos FXML e seus respectivos controladores. Você pode utilizar códigos adicionais para permitir o carregamento de dados nas interfaces sempre que necessário. | 0,5 pt. |
| 4 | Implemente controladores para cada uma das interfaces gráficas, de forma a integrar os elementos do FXML com objetos do modelo, bem como realizar os CRUDs junto ao banco de dados. | 1,5 pt. |
| 5 | Crie o banco de dados a partir de uma classe Java e insira alguns dados para teste. Essa classe deverá conter um método main que permita reconstruir o banco de dados sempre que necessário. OBS: Não se esqueça de criar chaves estrangeiras para os casos necessários. | 1 pt. |
| 6 | Implemente operações CRUD para cada classe do modelo. Crie também métodos listAll() que permitam ler todas entradas de uma tabela. | 2 pts. |
| 7 | Implemente classes segundo o padrão Data Access Object (DAO) para encapsular as operações CRUD, separando as Regras de Persistência das Regras de Negócio. | 1,5 pts. |
| 8 | Utilize Tratamento de Exceção e crie exceções personalizadas sempre que aplicável, evitando que a aplicação venha a travar. Aplique as exceções corretas para os problemas e, sempre que possível, adote a funcionalidade try-with-resources. | 1,0 pt. |
| 9 | Não seguir as orientações sobre a criação e envio do projeto descritas no preâmbulo da prova. | -1,0 pt. |

**\*\*\* Boa sorte! \*\*\***