PRONTUÁRIOS: SC300533X|SC3006905 Início: 10/08/2020 Término: 17/08/2020 (23h59)

✓ A prova deve ser realizada em dupla. ✓ Não é permitido compartilhar qualquer código entre as duplas. ✓ Atribui-se nota zero à prova em desacordo com os itens acima. ✓ Os membros da dupla devem participar de todas as partes do desenvolvimento.	✓ A nota final da prova será dada apenas após a arguição pelo professor. ✓ O projeto deve ser nomeado da seguinte forma: PRONTUARIOS_P3, 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.

Essa prova é um self-service! Cada dupla deve propor o contexto no qual demonstrará suas habilidades de Orientação a Objetos usando a linguagem Java e um banco de dados SQLite. O problema proposto pela dupla deverá conter ao menos três classes no modelo, sendo que entre elas deverá haver um relacionamento do tipo "um para um" e outro do tipo "um para muitos". Vocês podem 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:

Um clube de viciados em aposta pretende guardar o histórico de atoas as partidas de pedra-papel-tesoura de seus participantes. Cada partida deve ficar registrada no histórico de seus dois participantes, pois envolve um bem colocado com prêmio. De tão viciados, apostadores podem jogar também sozinhos, contra o clube. Nesse caso, ele aposta uma quantia, escolhe sua mão e é sorteada aleatoriamente uma mão para o clube (Pedra | Papel | Tesoura).

Classes e Atributos:

Apostador: nome(String), cpf(String), telefone(String), idade(int), partidas(List<Partida>), numDivorcios(int).

Aposta: momento(LocalDate), juiz(String), jogada1(Enum{PEDRA, PAPEL, TESOURA}), jogada2(Enum{PEDRA, PAPEL, TESOURA}), jogador1(Apostador), jogador2(Apostador), ganhador(Apostador), premio(Premio);

Premio: nome(String), valorDeclarado(double), itemDeFamilia(boolean).

Relacionamentos:

Cada objeto da classe Aposta tem um objeto da classe Premio.

Cada objeto da classe Apostador tem muitos objetos da classe Aposta.

Cada objeto da classe Aposta tem três objetos da classe Apostador.

A proposta de contexto feita pela dupla deverá ser previamente aprovada pelo professor. Caso uma proposta seja muito similar à de outra dupla ou esteja aquém do esperado, o contexto será proposto pelo próprio professor. A partir do contexto proposto, 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.	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 uma interface com Generics para padronizar a criação de classes DAO, garantindo as operações CRUD e listAll (ler todas as entradas).	0,5 pt.
9	Utilize ao menos uma vez Tratamento de Exceção com a funcionalidade try-with-resources.	0,5 pt.
10	Não seguir as orientações sobre a criação e envio do projeto descritas no preâmbulo da prova.	-1,0 pt.

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