PRONTUÁRIO: SC3005071 Início: 08/12/2021 Término: 10/07/2021 (08h00)

✓ 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: PRONTUARIO_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):

Um clube de viciados em aposta pretende guardar o histórico de todas 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 como prêmio. De tão viciados, apostadores podem jogar também sozinhos, contra o clube. Nesse caso, eles apostam uma quantia, escolhem sua a mão e é sorteada aleatoriamente uma mão para o clube (Pedra|Papel|Tesoura). O sistema também deverá permitir visualizar, além das partidas já realizadas no clube, o histórico individual de jogos de um apostador. Caso o apostador seja do tipo "FimDeCarreira" deve ser exibido também o número de divórcios em virtude das intermináveis jogatinas. Se um apostador acumular dez derrotas a mais do que vitórias ele é interditado pela família e não pode mais jogar, mas isso só acontece se ele não for do tipo ApostadorFimDeCarreira.

Classes e Atributos:

Apostador: nome (String), cpf (String), telefone (String), idade (int), partidas (List<Partida>), interditado (Boolean). Método polimórfico => isInterditado(): boolean
ApostadorFimDeCarreira: numDivorcios(int).
Aposta: momento (LocalDate), nomeJuiz (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.
A classe ApostadorFimDeCarreira herda Apostador.
Cada objeto da classe Aposta tem três objetos da classe Apostador

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 gráficas 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 pertinente, 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.	-2,0 pt.

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