PRONTUÁRIO: INDICAR NO ZIP Dia: 27/03/2025 Horário: 08h00 – 11h30

✓ A prova é individual e sem consulta. ✓ Não é permitido utilizar qualquer código implementado por você anteriormente. ✓ Não coloque seu nome no projeto. ✓ Atribui-se nota zero à prova em desacordo com os itens acima.	✓ A prova deve ser nomeada da seguinte forma: PRONTUARIO_P1, com o "SC". ✓ Não envie apenas as classes, mas todo o projeto. ✓ Envie o projeto como zip no Moodle.

Considere o contexto a seguir:

Estudantes de Computação do IFSP São Carlos sofrem semestralmente para organizar as disciplinas durante a rematrícula. Relatos de estudantes que encontram conflitos de horários entre as disciplinas ou se matriculam com uma carga horária excessiva são relativamente comuns. Por isso, o CAADES encomendou um protótipo de software para simular um processo de rematrícula, no qual um estudante escolhe as disciplinas que vai cursar e monta o seu horário, sem conflitos ou uma carga horária incompatível com o bom desempenho acadêmico. O modelo criado para o domínio é representado no diagrama UML a seguir:

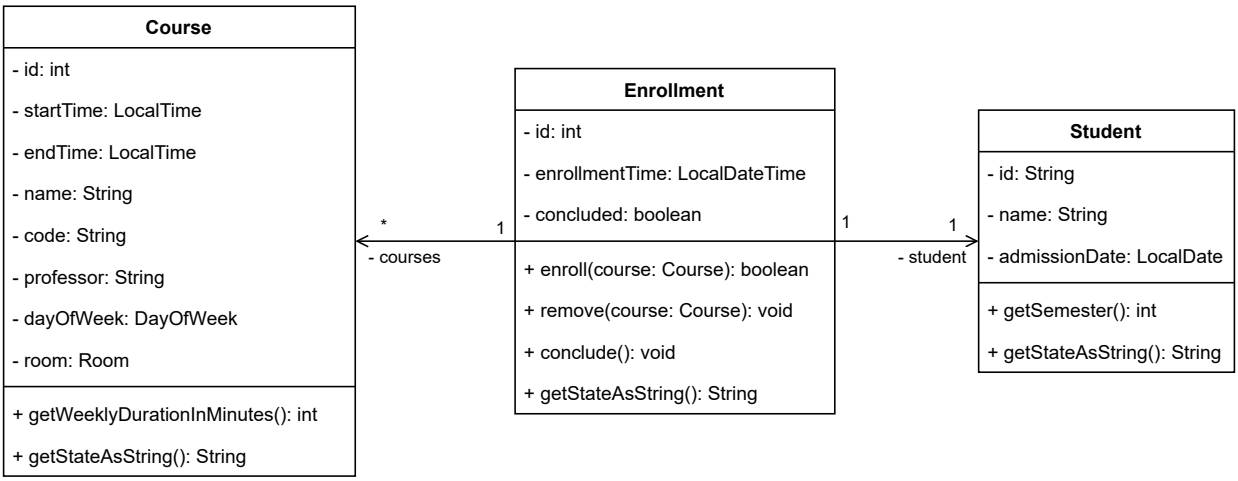

Figura 1 - Modelo de domínio

Execute as atividades a seguir para a implementação do exercício em Java. Você pode adicionar métodos privados, mas não deve modificar o que está previsto no diagrama. Para a atribuição da nota será levada em conta não apenas a funcionalidade, mas a qualidade, adequação e pertinência de cada solução. Bom senso faz parte da prova.

Dica 1: Utilize arrays em relacionamentos um para muitos. O uso de outra estrutura de dados invalidará as questões relacionadas.
Dica 2: Busque implementar as questões na ordem. Caso tenha dificuldade com alguma implementação, emule os dados, siga para a próxima questão e depois volte para tentar novamente.

#	Descrição	Pont.
1	Crie o projeto, os arquivos representando as classes do modelo e um arquivo adicional chamado Main, que contenha o método main do projeto.	0,5pt
2	Implemente a classe Student, que representa um estudante do curso. Siga o princípio do encapsulamento. Objetos da classe Student devem ser imutáveis. Valide os parâmetros do	1,0pt

	construtor, garantindo que os valores passados são válidos. Um ID deve sempre ter o formato SCDDDDDDW, onde D é um dígito e W pode ser a letra 'X' ou um dígito. Se os parâmetros forem inválidos, produza um objeto com valores padrão nos atributos. No método getStateAsString(), retorne o estado do objeto como uma String. No método getSemester(), retorne o período que o aluno se encontra no curso.	
3	Implemente a classe Course, que representa um oferecimento de disciplina. Siga o princípio do encapsulamento. Objetos da classe Course devem ser imutáveis. Valide os parametros do construtor, garantindo que os valores passados são válidos. Se os parâmetros forem inválidos, produza um objeto com valores padrão nos atributos. Crie um enum chamado Room para representar uma sala, com os seguintes valores possíveis (C102, C104, C105, C106, C107, C209). O ID de cada objeto da classe Course deve ser único e gerado de maneira incremental, começando do 1 (um). No método getStateAsString(), retorne o estado do objeto como uma String (em uma única linha).	2,0pts
4	Implemente o construtor da classe Enrollment. O ID de cada objeto deve ser único e gerado de maneira incremental, começando do 1 (um). Caso o objeto Student seja inválido ou possua estado padrão, faça com que os atributos do objeto Enrollment recebam valores padrão e indique a matrícula como concluída.	0,5pt
5	Na classe Enrollment, crie o método enroll(), que adiciona uma disciplina a lista de disciplinas da matrícula, caso isso seja possível. Não deve ser possível adicionar uma disciplina nos seguintes casos: i) a matrícula já estiver concluída; ii) houver colisão de horário com outra matrícula inserida anteriormente; iii) a carga horária total em disciplinas ultrapassar 22 horas por semana. O método deve retornar verdadeiro se a disciplina for inserida e falso caso ela não tenha sido inserida.	2,5pt
6	Na classe Enrollment, implemente o método remove(), que remove uma disciplina da lista de disciplinas da matrícula. Não deve ser permitido remover disciplinas após a matrícula ser concluída.	1,0pt
7	Na classe Enrollment, crie o método getStateAsString, que retorna o estado do objeto como uma String, de acordo com o formato do Quadro A. Caso a matrícula não tenha sido concluída, retorne "Enrollment not concluded!". Se o objeto tiver sido criado com erro, imprima "Invalid enrollment!". Para obter a pontuação completa, você deve utilizar a classe StringBuilder na implementação.	1,5pt
8	Na classe Enrollment, implemente o método conclude(), que marca a matrícula como concluída e registra o horário de conclusão. Implemente também métodos de acesso para a classe, de acordo com o princípio do encapsulamento. O estudante da matrícula não pode ser alterado. Embora a classe seja mutável, ela não pode ser alterada após a conclusão da matrícula.	0,5pt
9	No método main, crie disciplinas com diferentes dados, um estudante e realize diversas tentativas de adição e remoção de disciplinas, sempre imprimindo o resultado obtido. Apenas a classe Main deve interagir com o console.	0,5pt

Quadro A – Modelo de texto para representar uma reserva

```
================================================================================

SC203203 | Lucas Oliveira | Admission date = 2024-09-17

Enrollment Time: 17/09/2024 11:10:52

--------------------------------------------------------------------------------------------------------------------------

Courses:

| id = 1 | Object-orientation (POO) | FRIDAY | Start = 19:00 | End = 22:30 | Lucas Bueno | Room = C106 |

| id = 2 | Databases I (BD1)  | MONDAY | Start = 8:00 | End = 11:30 | Carlão | Room = C102  |

================================================================================
```

Prêmio Usain Bolt: O aluno que terminar todas as atividades corretamente primeiro ganha 1,0pt adicional para usar em outra prova. Você está voando?

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