Teste e Validação de Software 2023/24
Instituto Superior Técnico

Enunciado do Projecto

Data de Entrega: 10 de Maio de 2024

1 Introdução

O desenvolvimento de grandes sistemas de informação é um processo complexo e requer diversos níveis de abstracção. Um primeiro passo é a especificação do problema de uma forma rigorosa. Após a especificação rigorosa do problema, é possível efectuar provas de correcção do mesmo ou, caso este esteja incorrecto, descobrir falhas que de outra forma seriam difíceis de detectar. Depois, dever-se-á começar a codificar a solução e a testá-la simultaneamente por forma a detectar os erros o mais cedo possível.

O projecto da disciplina Teste e Validação de Software consiste no desenho de casos de teste a partir de uma especificação disponibilizada. Esta especificação diz respeito a algumas entidades de uma dada aplicação. Este projecto tem como objectivo principal de aprendizagem que os alunos ganhem experiência no desenho de casos de testes ao aplicarem as técnicas de desenho de casos de teste leccionadas nas aulas teóricas a um conjunto de classes e métodos cujo comportamento está descrito neste documento.

Este documento está organizado da seguinte forma. A secção 2 descreve as entidades que participam no sistema a testar assim como alguns detalhes de concretização destas entidades e que serão importantes para os casos de teste a desenhar. A secção 3 identifica os testes a realizar e a forma como a resolução do projecto vai ser avaliada.

2 Descrição do sistema

O sistema a testar é responsável por gerar exames a partir de um modelo e depois fazer a gestão dos exames realizados por alunos. As questões de cada exame são questões de escolha múltipla. De seguida descrevem-se as principais entidades deste sistema.

2.1 A entidade Question

Cada questão é composta pelo corpo da questão e um conjunto de escolhas. A questão sabe ainda qual é a escolha correcta. Cada questão pertence ainda a um ou mais tópicos e tem uma dada cotação. A cotação é um valor inteiro maior do que 0 e inferior ou igual a 15. Cada tópico é representado simplesmente por uma cadeia de caracteres e tem que ter pelo menos 6 caracteres. Os tópicos de uma questão são distintos. Uma questão pode estar associada no máximo a 5 tópicos e pode ter entre 2 e 8 opções. A interface desta entidade está representada na listagem 1.

Invocações a métodos desta classe que coloquem a questão num estado inválido devem ser abortadas, lançando a excepção `InvalidOperationException`, e o estado da questão não deve ser alterado.

2.2 A entidade ExamModel

Cada exame pertence a um dado modelo. Um modelo especifica o número de grupos do exame, o tema de cada grupo e as questões que devem fazer parte de cada grupo. A listagem 2 descreve a interface da classe *ExamModel*. Cada grupo tem um identificador único atribuído quando o grupo é adicionado ao modelo. O identificador de

Listing 1: A interface da classe *Question*.

```
public class Question {
    public Question(String body, List<String> choices, int correctChoice, String topic, int weight) throws
         InvalidOperationException { /* .... */ }

    // Removes a topic
    public void remove(String topic) throws InvalidOperationException { /* .... */ }

    // Adds a new topic
    public void add(String topic) throws InvalidOperationException { /* .... */ }

    // Returns all topics of this question
    public List<String> getTopics() { /* .... */ }

    // Computes the grade considering the weight of this question and the selected choice
    public float grade(int selectedChoice) { /* .... */ }

    // Changes the weight of this question
    public void setWeight(int weight) throws InvalidOperationException { /* .... */ }

    // Returns the weight of this question
    public int getWeight() { /* .... */ }

    // Returns the choices of this question
    public List<String> getChoices() { /* .... */ }
}
```

cada grupo é igual ao número de grupos existentes no modelo no momento em que o grupo foi adicionado. No momento da criação de um grupo é indicado o número máximo de questões que o grupo pode ter e o tópico do grupo. Um grupo só poderá ter questões que incluam o tópico do grupo. O método `addQuestion` adiciona uma nova questão ao grupo indicado caso seja possível, e o seu valor de retorno indica se a questão foi adicionada ou não.

De seguida descreve-se o comportamento da classe *ExamModel*. Um modelo pode estar aberto ou fechado. Enquanto estiver aberto, o seu estado pode ser alterado e podem ser adicionados novos grupos e questões a um grupo, por exemplo. Uma vez que seja fechado, o seu conteúdo já não pode ser alterado. Um modelo só pode ser fechado caso seja válido. O método `validate()` é responsável por validar um modelo de exame, alterando o seu estado para fechado caso seja válido. Este método devolve uma cadeia de caracteres que descreve porque é que o modelo é inválido. Se o modelo for válido, o método devolve simplesmente a referência nula (`null`). Assim, este método funciona de acordo com as seguintes regras (por ordem decrescente de prioridade):

- Se o modelo já estiver fechado, simplesmente devolve a referência nula (`null`);
- Se o número de grupos do modelo for menor do que 2 ou maior do que 8, deve devolver "Número inválido de grupos";
- Caso o número de grupos esteja entre 2 e 8, então é necessário considerar o seguinte:
  - Se a cotação do modelo fôr diferente de 100, deve devolver "Cotação total do exame inválida";
  - Se a cotação do modelo for 100 e se o número médio de questões por grupo for menor ou igual a 4, então o modelo do exame é válido e nesse caso o método retorna a referência nula e passa o estado deste modelo de exame para *fechado*. Caso o número médio de questões por grupo seja maior do que 4, então deve devolver "Número médio de questões inválido".

O método *addQuestion* adiciona uma nova questão ao grupo especificado. A adição da nova questão só é realizada caso o modelo ainda esteja aberto, o grupo indicado seja válido e o tema do grupo seja um dos temas da questão. Caso alguma destas condições não seja válida, então o estado do modelo deve permanecer inalterado. O método devolve um valor boleano que indica se a questão foi adicionada ou não.

A cotação de um modelo de exame é igual ao somatório da cotação dos grupos do modelo. A cotação de um grupo é igual ao somatório das cotações das questões pertencentes ao grupo.

Listing 2: A interface da classe *ExamModel*.

```
public enum ModelState {OPEN, CLOSED;}

public class ExamModel {
  public ExamModel() { /* .... */ }

  // Adds a new group and returns the number of the created group
  public int addGroup((int maxNumberOfQuestions, String topic) { /* .... */ }

  // Adds a question to the specified group of this exam model. groupId can range
  // between 1 and the number of groups of this exam model.
  public boolean addQuestion(Question q, int groupId) { /* .... */ }

  // Removes the question from the specified group of this exam model
  public void removeQuestion(Question q, int groupId) { /* .... */ }

  // Validates this exam model
  public String validate() { /* .... */ }

  // ...
}
```

2.3 A entidade ExamManager

A realização de um exame é gerido pela entidade `ExamManager`. Cada gestor de exames está associado a um modelo de exame e é responsável por disponibilizar um exame aos alunos inscritos, recolher os exames realizados pelos alunos e disponibilizar os resultados dos exames entregues. A listagem 3 apresenta a interface da classe `ExamManager`. Esta entidade mantém os alunos inscritos, os alunos que acederam ao exame a realizar e os alunos que entregaram o exame resolvido.

Listing 3: A interface da classe *ExamManager*.

```
public enum ManagerMode { OPEN, PUBLISHED, TERMINATED, EVALUATION; }

public class ExamManager {
  public ExamManager(ExamModel model) { /* ... */ }

  // Returns the mode of this exam manager
  public final ManagerMode getMode() { /* ... */ }

  // Enrolls a student for this exam
  public void enroll(Student t) { /* ... */ }

  // Cancels start and close operations
  public void cancel() { /* ... */ }

  // Returns an exam for the (enrolled) student
  public Exam getExam(Student student) { /* ... */ }

  // Submits an exam
  public void submit(Exam exam) { /* ... */ }

  // Finish current state (
  public void close() { /* ... */ }

  // Returns the evaluation of the exam made by the specified student
  public float evaluate(Student student) { /* ... */ }
}
```

Relativamente à entidade `ExamManager` existem as seguintes restrições relativamente à invocação dos seus métodos:

- Quando se cria um gestor de exames, ele fica *aberto* (`OPEN`). Neste modo, um aluno pode inscrever-se para realizar o exame através do método `enrollStudent()`. Quando o número de alunos inscritos é maior ou igual a 10, a invocação do método `close()` termina o período de inscrição e o gestor de exames passa para o modo *avaliação* (`EVALUATION`).

- Quando um gestor está em modo avaliação, é possível pedir o exame a realizar para um dado aluno inscrito previamente (através de `getExam()`), e entregar o exame depois de ter respondido às questões do exame (através de `submit()`). Caso seja invocado o método `close()` sobre um gestor em modo avaliação, o gestor passa para o modo *finalizado* (`TERMINATED`). Caso nenhum aluno tenha ainda pedido o seu exame , é possível voltar a colocar um gestor em modo avaliação no modo *aberto* outra vez invocando o método *cancel()*.
- Num gestor *finalizado*, pode-se voltar a colocá-lo no modo avaliação caso se invoque o método `cancel()` ou colocá-lo em modo *publicado* (`PUBLISHED`) caso se invoque o método `close()`.
- Finalmente, um gestor de exame que esteja *publicado* permite avaliar o exame de cada aluno que realizou o exame através do método `evaluate()`.

Caso a invocação dos métodos indicados não ocorra numa das situações descritas, então o método não deverá ter qualquer efeito e deverá lançar a excepção *InvalidInvocationException*. O método `getMode()` pode ser invocado em qualquer situação e devolve o modo do gestor.

3 Avaliação do projecto

Todos os casos de teste a desenhar devem ser determinados aplicando os padrões de desenho de testes mais apropriados.

O projecto consiste em especificar os casos de teste para testar alguns métodos e algumas classes ao nível de classe. Será ianda necessário concretizar alguns dos casos de teste especificados. Os casos de teste a especificar são os seguintes:

- Casos de teste ao nível de classe da classe *Question*. Valem entre 0 e 3,5 valores.
- Casos de teste ao nível de classe da classe *ExamManager*. Valem entre 0 e 6,5 valores.
- Casos de teste correspondentes ao método *validate* da classe *ExamModel*. Valem entre 0 e 3,5 valores.
- Casos de teste correspondentes ao método *addQuestion* da classe *ExamModel*. Valem entre 0 e 3,5 valores.

A especificação dos casos de teste deve ser realizada aplicando a estratégia de desenho dos testes mais apropriada para cada situação. **Adicionalmente**, é necessário concretizar 8 casos de teste (4 casos de teste de *sucesso* e 4 casos de teste de *insucesso*) da bateria de testes desenhada para exercitar a classe *Question* ao nível de classe. Esta concretização deve ser feita utilizando a framework de testes *TestNG* e vale entre 0 e 3 valores.

Caso aplique a estratégia de desenho de teste *Category Partition*, se o número de combinações a apresentar for maior do 30, apenas é necessário apresentar as primeiras 30 combinações e indicar o número total de combinações. Para cada método ou classe a testar é necessário indicar o seguinte:

- O nome da estratégia de desenho de testes aplicado.
- Caso seja aplicável, indicar o resultado dos vários passos da estratégia aplicada, utilizando o formato apresentado nas aulas teóricas.
- A especificação dos casos de teste resultantes da aplicação da estratégia de desenho de tested escolhida. Normalmente, a especificação corresponde ao resultado obtido no último passo da estratégia.

Caso algum dos pontos mencionados acima apenas seja satisfeito parcialmente a nota será dada na proporção realizada.

3.1 Detecção de cópias

A submissão de um projecto pressupõe o **compromisso de honra** que o trabalho incluso foi realizado pelos alunos referenciados nos ficheiros/documentos submetidos para avaliação. A quebra deste compromisso, ou seja a tentativa de um grupo se apropriar de trabalho realizado por colegas, tem como consequência a reprovação nesta disciplina de todos os alunos envolvidos (incluindo os que possibilitaram a ocorrência).

3.2 Notas Finais

Poderá existir uma discussão que avalie a capacidade dos alunos em realizar testes semelhantes aos realizados no projecto. Esta decisão cabe exclusivamente ao corpo docente da cadeira. Alunos cuja nota no exame seja inferior em mais de 5 valores à nota do projecto poderão ter de realizar uma discussão. Caso haja lugar a uma discussão, a nota final do projecto poderá ser individualizada e será a nota obtida na discussão.

Toda a informação referente ao projecto estará disponível na página da disciplina na secção *Projecto*. O protocolo de entrega do projecto está disponível nesta secção da página da disciplina.

Bom Trabalho!