 <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Trabalho prático 1	Ano letivo 2019/2020	Data
	Curso LEI	Hora	
	Unidade Curricular Engenharia de Software 2	Duração	

Enunciado

Considere a existência de uma plataforma de criação de gestão de testes para suportar atividades letivas de autoavaliação numa escola. É uma ferramenta importante para diagnosticar a compreensão de vários conteúdos de aprendizagem associados a uma determinada unidade curricular.

A aplicação tem como objetivo carregar a informação sobre os testes de avaliação a partir de um ficheiro criado previamente e expor aos alunos um conjunto de questões de resposta variada. Além da resposta fornecida pelo aluno, é anotado o tempo de início o tempo de fim da resposta fornecida. No final do teste, será apresentado o número de respostas certas e erradas, bem como o tempo médio e desvio padrão do tempo médio necessário para cada pergunta.

Descrição Técnica

O desenvolvimento da aplicação irá obrigar ao cumprimento de contratos pré-estabelecidos para a definição de questões através de interfaces específicos, nomeadamente:

- `IQuestion`, define o contrato relacionado com uma questão genérica;
- `IQuestionMetadata`, define o contrato com os metadados associados a uma questão;
- `IQuestionYesNo`, define o contrato para uma questão do tipo: sim ou não;
- `IQuestionNumeric`, define o contrato de uma questão do tipo: numérico;
- `IQuestionMultipleChoice`, define o contrato para uma questão do tipo: escolha múltipla.

A figura 1 apresenta uma visão geral das principais interfaces existentes no programa.

P.PORTO <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Trabalho prático 1	Ano letivo 2019/2020	Data
	Curso LEI	Hora	
	Unidade Curricular Engenharia de Software 2	Duração	

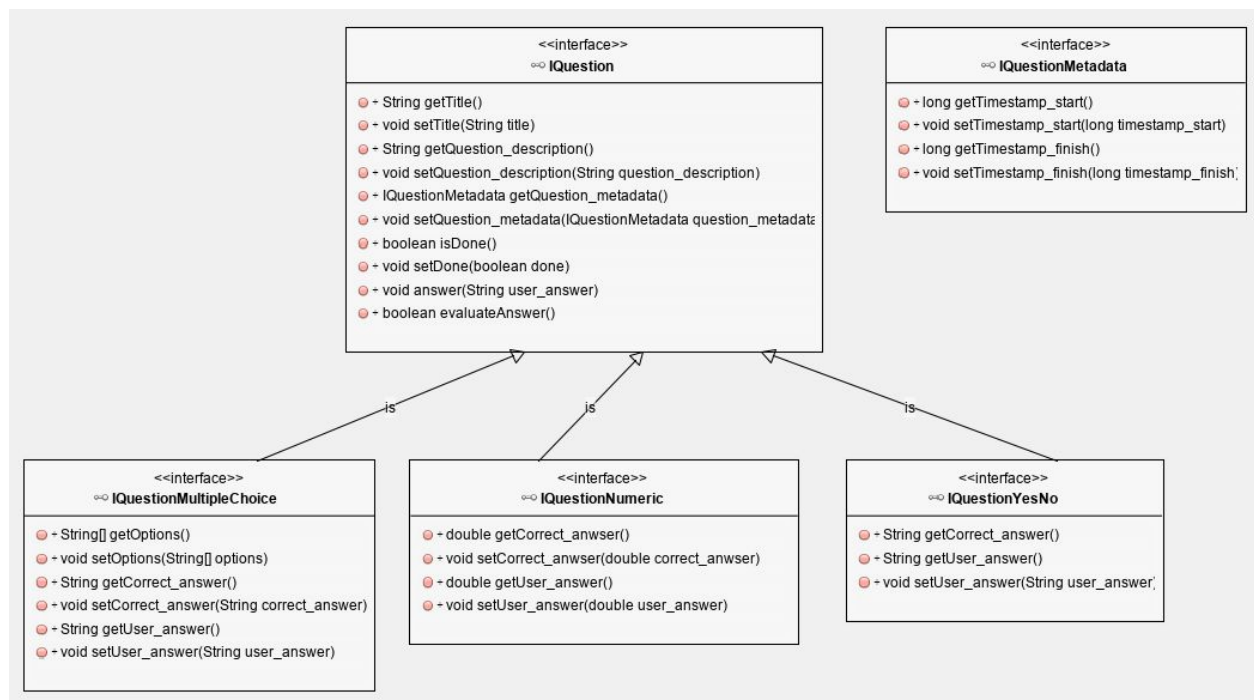


Figura 1 – Principais interfaces do programa

O contrato `ITest` é utilizado para guardar as questões de um determinado teste numa coleção. Esta interface utiliza um objeto que implementa o contrato `ITestStatistics`, utilizado para o cálculo das estatísticas sobre as respostas a um determinado teste de avaliação de conhecimentos.

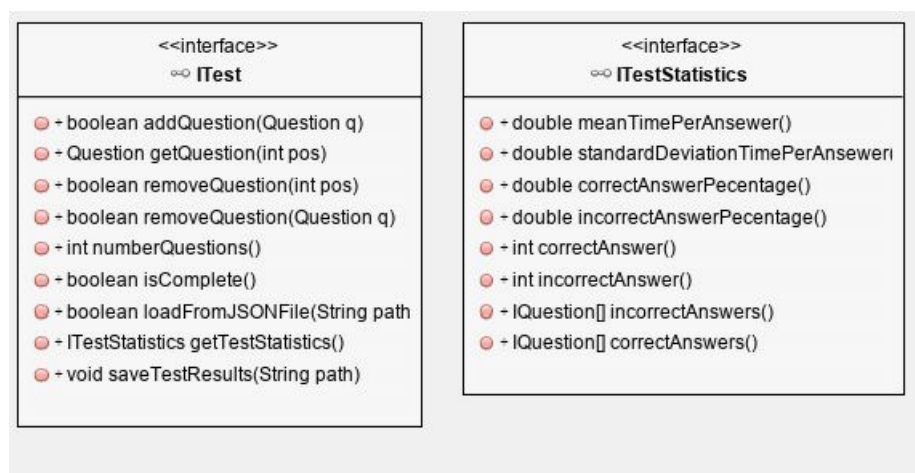


Figura 2 - Contratos relacionados com o teste e as estatísticas do teste

P.PORTO <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Trabalho prático 1	Ano letivo 2019/2020	Data
	Curso LEI	Hora	
	Unidade Curricular Engenharia de Software 2	Duração	

O seguinte excerto exemplifica um exemplo de procedimento utilizado para criar e usar um objeto teste.

```
public static void main(String[] args) throws TestException {

    System.out.println("Inicio de Teste!");

    //Carregar o test
    ITest demoTest = new Test();
    demoTest.loadFromJSONFile("data/teste_A.json");

    //Executar o teste na camada gráfica
    TestWindow t = new TestWindow();
    t.startTest(demoTest);

    //Obter os resultados do teste
    System.out.println("Teste Efectuado!");
    System.out.println(demoTest.toString());

}
```

Figura 3 – Exemplo de instanciação e execução de um teste


Os dados do teste encontram-se armazenados no formato *JSON*¹. Na estrutura do ficheiro *JSON* são definidas múltiplas questões (utilizando um array). Um exemplo encontra-se disponível na pasta *data* no ficheiro com o nome *teste_A.json*.

Objetivo

Considerando a aplicação fornecida, pretende-se que planeie um conjunto de casos de teste, seguindo uma abordagem de funcional numa perspectiva de “back-box testing por forma a verificar o comportamento do programa e dos componentes que definem o conjunto de Use Cases disponíveis para o utilizador, nomeadamente: *Test*, *TestStatistics* e *QuestionMultipleChoice*, *QuestionNumeric* e *QuestionYesNo*.

Para o efeito deverá especificar um conjunto de casos de teste utilizando as técnicas “Equivalence Class Partitioning” e “Boundary Value Analysis”. Para além do exercício de análise, deverá indicar os *test inputs*, *execution conditions* e *expected outputs*, assegurando

¹ <https://www.json.org/>

 <div> ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO </div>	Tipo de Prova Trabalho prático 1	Ano letivo 2019/2020	Data
	Curso LEI	Hora	
	Unidade Curricular Engenharia de Software 2	Duração	

que os casos de teste cobrem *Valid equivalence classes* e *Invalid equivalence classes*. Implemente os casos de teste e execute-os utilizando a framework: JUnit 5².

Deve elaborar um relatório de testes. Registe os resultados dos testes adequadamente. Lembre-se de especificar o test case ID, objeto testado, descrição, entradas, saídas esperadas e outras informações que considere úteis. Para além do relatório de testes, o código usado para implementar cada um dos testes deverá ser devidamente documentado de forma a identificar cada caso de teste.

Para a análise da aplicação e especificação dos casos de teste mais adequado, tenha em consideração as restrições específicas ao comportamento de cada componente.

Especificação do componente *Test*

Método **addQuestion**

```

requer:  existência de uma coleção de perguntas previamente
instanciada
garante:
    Se o parâmetro q for nulo
        uma exceção específica é emitida
    Senão
        Se a questão for adicionada
            retorna true e a questão é adicionada ao teste
        Senão
            retorna falso porque o teste não pode acomodar mais questões
(não existem posições livres)

```


Método **removeQuestion**

```

requer:  position >= 0 e position < tamanho da lista de perguntas e
existência de uma coleção de perguntas previamente instanciada

```

² <https://junit.org/junit5/>

 <div> ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO </div>	Tipo de Prova Trabalho prático 1	Ano letivo 2019/2020	Data
	Curso LEI	Hora	
	Unidade Curricular Engenharia de Software 2	Duração	

garante:

```

    Se o elemento a remover = null
        retorna false
    Senão
        remove a questão na posição indicada, colocando o valor null
na referida posição e retorna true

```

Método **numberQuestions**

requer: existência de uma coleção de perguntas previamente instanciada

garante:

o número de questões não nulas (null) existentes no teste

Método **isComplete**

requer: existência de uma coleção de perguntas previamente instanciada

garante:

```


    Se todas as perguntas do teste estão respondidas (done)
        retorna true
    Senão
        retorna false

```

Especificação do componente *TestStatistics*

Método **correctAnswers**

requer: existência de teste com uma coleção de perguntas previamente instanciada

 <div> ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO </div>	Tipo de Prova Trabalho prático 1	Ano letivo 2019/2020	Data
	Curso LEI	Hora	
	Unidade Curricular Engenharia de Software 2	Duração	

garante:

o retorno de uma coleção instanciada com o número de posições igual ao número de questões incorretas existentes

Método **incorrectAnswers**

requer: existência de teste com uma coleção de perguntas previamente instanciada

garante:

o retorno de uma coleção instanciada com o número de posições igual ao número de questões incorretas existentes

Método **correctAnswerPercentage**

requer: existência de teste com uma coleção de perguntas previamente instanciada

garante:


Se o teste tiver perguntas não respondidas
uma exceção específica é emitida
Senão
retorna percentagem de perguntas corretas

Método **incorrectAnswerPercentage**

requer: existência de teste com uma coleção de perguntas previamente instanciada

garante:

Se o teste tiver perguntas não respondidas
uma exceção específica é emitida
Senão

 <div> ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO </div>	Tipo de Prova Trabalho prático 1	Ano letivo 2019/2020	Data
	Curso LEI	Hora	
	Unidade Curricular Engenharia de Software 2	Duração	

retorna percentagem de perguntas incorretas

Especificação dos componentes *QuestionMultipleChoice*, *QuestionNumeric* e *QuestionYesNo*

Método `evaluateAnswer`

```

requer:    correct_answer != null e número de caracteres de
correct_answer > 0

garante:

    Se user_answer = null
        uma exceção específica é emitida
    Senão
        Se user_answer = correct_answer
            retorna true
        Senão
            retorna false

```

Método `answer`:


```

requer:    user_answer != null e número de caracteres de
correct_answer > 0

garante:

    Para as questões (QuestionNumeric), se a string contiver algum
    caracter != [0-9]
        uma exceção específica é emitida
    Para qualquer tipo de questão:
        atribui a user_answer à variável de instância
        define o tempo de fim de resposta

```

 <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Trabalho prático 1	Ano letivo 2019/2020	Data
	Curso LEI	Hora	
	Unidade Curricular Engenharia de Software 2	Duração	

```
(através de QuestionMetaData)
define a pergunta como respondida (setDone)
```

Documentação

A documentação de testes contempla vários documentos de acordo com o processo de teste. Existe documentação relacionada com especificações organizacionais, gestão de testes e execução dinâmica de testes. Para este trabalho prático em concreto deverão focar-se apenas no sub-processo relacionado com dinâmica dos testes em particular nos documentos de *Test Design Specification* e *Test Case Specification*, respetivamente. Para cada um destes documentos será disponibilizado um template no moodle. A par com cada um destes documentos, será disponibilizada a norma IEEE 829 que define a documentação relacionada com software testing. Tipicamente o documento refere-se a um documento por seção. O template relacionado com Test Case Specification, contém uma proposta de tabela que vos permite efetuar o *log* dos casos de teste.

Submissão dos resultados do Trabalho Prático

Através do moodle, deverão submeter os documentos definidos anteriormente, enquanto que o código desenvolvido deverá ser disponibilizado através de um repositório GIT da ESTG em <https://git.estg.ipp.pt>³ e que deverão identificar ser devidamente no 1º documento. O período de entrega estará disponível até dia **24 de Novembro pelas 23:59**. No repositório GIT deverá estar refletido as contribuições de cada membro do grupo, assim como a evolução do projeto. O relatório poderá estar integrado no repositório GIT, onde poderá ser estruturado recorrendo à linguagem de marcação Markdown⁴, ou via documento pdf incluído no repositório.

O trabalho deverá ser realizado por grupos de **3 elementos** (máximo). A identificação dos grupos deverá ser indicada na plataforma moodle até dia **15 de Novembro**.

³ O acesso estará condicionado ao uso de VPN

⁴ <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>