INSTITUTO SUPERIOR MANUEL TEIXEIRA GOMES

Trabalho Prático 1 - Programação Orientada a Objetos
Teste de questões múltiplas

Relatório de Programação Orientada a Objetos

Pedro Pinto, n.º 22000888

Gustavo Paulino, nº 22180593

Docente:

Prof. Doutor Tiago Candeias

2021

Conteúdo

1 Introdução

1.1 Descrição do Tema

O problema que pretendemos resolver prende-se na realização de um teste com várias perguntas, por parte de um grupo de alunos. Então cada aluno irá responder ao teste e irá obter uma classificação final. No fim terá de ser entregue uma pauta de todos os alunos, ordenada decrescentemente por nota e caso exista notas iguais teremos de apresentar por ordem alfabética dos nomes dos alunos.

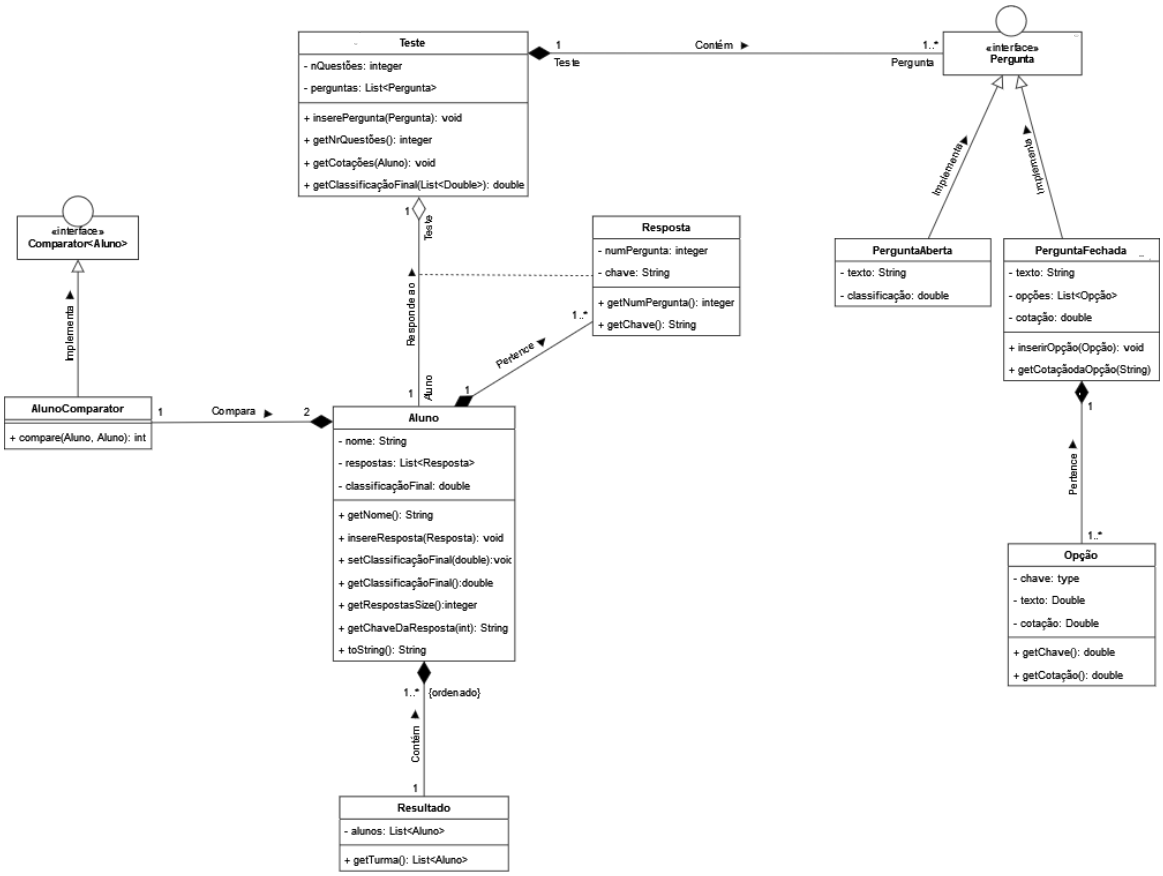
1.2 Abordagem ao Problema

A abordagem do grupo iniciou-se pela leitura do enunciado e rapidamente tentar reconhecer as classes de que iríamos necessitar para resolver este desafio. Primeiramente criámos um diagramas de classes em UML para nos orientar na contrução do código, e de seguida fomos á procura de definir as responsabilidades de cada classe, sendo estas primeiras etapas fundamentais para a resolução tendo em vista a utilização do paradigma de POO. Após discussão do tema e do problema em si chegamos á conclusão de que o ideal seria criar um teste que iria guardar uma coleção de perguntas, perguntas essas que seriam de dois tipos, aberta e fechada. Aqui usámos o poliformismo para criar uma interface pergunta, guardando assim no teste apenas perguntas, sem olhar a tipos. Decidimos ainda tornar uma opção de resposta de uma pergunta fechada também um objeto, que seriam guardadas numa lista de objetos Opção dentro da classe Pergunta Fechada. Quanto ao lado de responder ao teste, decidimos criar uma classe Resultado que teria a responsabilidade de ordenar e apresentar os resultados e para isso criámos então objetos do tipo Aluno. Estes objetos para além de representar cada aluno que iria responder ao teste, também guardaria a sua classificação final. Assim comparando alunos, com uma classe comparadora, conseguiríamos obter uma solução eficaz. No cálculo da classificação atribuímos essa responsabilidade ao teste, invocando cada aluno e calculando a sua nota final através das respostas dadas. Aqui decidimos então que cada aluno guardaria a sua lista de respostas.

Diagrama de classes em UML

2

2.1 Diagrama de classes



2.2 Análise do diagrama

1. **Classe Teste** É a classe que tem a responsabilidade de guardar uma coleção de perguntas e calcular o resultado final de cada aluno. Um teste contém um ou mais perguntas e um ou mais alunos respondem ao teste.

2. **Classe Aluno** É a classe que identifica cada aluno que responde ao teste. Esta guarda as suas respostas e a sua classificação final. Um aluno responde a um teste e cria uma ou mais respostas. A cada aluno esta associado um resultado.

3. **Classe Resposta** É a classe que é a classe associativa que representa uma resposta a uma questão do teste. São guardadas na classe aluno à qual pertencem.

4. **Classe Resultado** É a classe que recebe uma lista de alunos com os resultados já calculados e no seu contrutor ordena essa lista segundo as notas finais e alfabéticamente em caso de empate.

5. **Classe AlunoComparator** É a classe que compara os alunos, e que atua na classe Resultado. Implementa uma interface comparator<T> comparando objetos do tipo Aluno.

6. **Classe PerguntaFechada** É a classe que implementa a interface Pergunta. É a classe que representa uma pergunta fechada à qual está associado 4 opções de resposta.

7. **Classe PerguntaAberta** É a classe que implementa a interface Pergunta. É a classe que representa uma pergunta aberta à qual está associado uma cotação

8. **Classe Opção** É a classe opção de uma pergunta fechada que são as opções de resposta para responder à pergunta

Teste JUNIT e de Cenários 3

3.1 Classe JunitTeste

testInserirPerguntas()

Criação de uma nova instância de um objeto teste com parâmetro de entrada de 3, que indica que o teste terá 3 questões. Testamos se o número de questões indicada ficou guardado na variável nQuestões do teste.

testCalculoNota()

Criação de uma nova instância de um objeto teste com parâmetro de entrada de 2, que indica que o teste terá 2 questões. Criação de 2 perguntas fechadas e inserção de 4 opções para cada. Inserção de cada pergunta no teste. Criação de 3 alunos e as suas respostas às perguntas inseridas no teste e calculamos as respetivas notas finais. Testamos aqui as notas finais e se correspondem ao esperado.

calculaCotacoes()

Criação de uma nova instância de um objeto teste com parâmetro de entrada de 2, que indica que o teste terá 2 questões. Criação de uma arraylist de Doubles que guarda cotações de respostas certas de um aluno. Testamos se a soma de cotações é igual ao resultado final esperado.

3.2 Classe JunitAluno

testName()

Criação de uma instância de um objeto Aluno, com um nome. Testamos se a variável nome não fica a null e se o valor guardado corresponde ao nome introduzido.

testSetClassificacao()

Criação de uma instância de um objeto Aluno, com um nome. Indicamos a classificação do aluno com uma nota x. Testamos se guarda a classificação e se é a expectável.

testGetRespostasSize()

Criação de uma instância de um objeto Aluno, com um nome. Inserção de 2 respostas no arraylist de respostas do aluno. Testamos se as respostas são guardadas.

testToString()

Criação de uma instância de um objeto Aluno, com um nome. Inserção da sua classificação final. Testamos se a função toString printa o aluno como queriamos.

3.3 Classe JunitPerguntaFechada

testInsereOpcao()

Criação de uma instância de um objeto de pergunta fechada. Testamos se o a instância é de facto do tipo criado. Inserção de 2 opções diferentes na pergunta. Testamos de as opções da pergunta guardaram as suas cotações.

3.4 Classe JunitOpcao

testOpcao()

Criação de uma opção. Testamos se a opção guardou a chave e a cotação corretamente.

3.5 Classe JunitResposta

testResposta()

Criação de uma instância de um objeto resposta. Testamos de o número da da pergunta, ao qual pertence a resposta, e a chave foram guardados corretamente.

3.6 Classe JunitResultado

testResultado()

Criação de uma arraylist de alunos. Criação de 4 alunos, cada um com o seu nome e nota final. Inserção dos alunos na arraylist. Testamos a ordem dos alunos inseridos. Criação de uma instância de um objeto Resultado com parâmetro de entrada a arraylist desordenada. Testamos de a arraylist foi ordenada no final

4 Conclusão

Olhando para trás conseguimos identificar várias dificuldades. Primeiro o desenho do diagrama de classes em UML que pela falta de experiência, acabou por nos causar dificuldades. A estruturação do código em java também nos desafiou, e acabou por ser a maior dificuldade que tivemos. Para além da constante aprendizagem sobre a linguagem outros obstáculos como as boas práticas a ter ou a otimização do código atravessaram o nosso caminho. Superando esses desafios conseguimos concluir que ainda muito há a aprender e conseguimos identificar algumas melhorias a fazer no próximo trabalho prático, conseguindo compreender e adquirir competências para o futuro.

Bibliografia

[1] Prof. Doutor Tiago Candeias, 2021/2022, aulas Teórico-práticas da disciplina de Programação Orientada a Objetos, 2º ano, 1º semestre da licenciatura em Engenharia Informática do Instituto Superior Manuel Teixeira Gomes.

[2] K. Berry. Tex live documentation. TeX Live has been developed since 1996 by collaboration between the TeX user groups. [Online]. Available: https://tug.org/texlive/