Licenciatura em Sistemas Informáticos

Linguagens de Programação II

Relatório do Trabalho Prático

André Cardoso & José Cosgrove
18848 & 18826

Barcelos, 22 de maio de 2020

Contents

1 Introdução

Foi proposto para a unidade curricular de Linguagens de Programação II, o desenvolvimento de uma solução em C# para problemas reais de complexidade moderada. Para tal é necessário provar:

- Compreensão dos conceitos básicos de programação segundo o Paradigma Orientado a Objetos.
- Capacidade de Análise de problemas reais.

1.1 Motivação

Pretende-se o desenvolvimento de soluções em C# para problemas reais, trabalhando o paradigma de POO.

Numa primeira fase foi nos requerido uma implementação essencial das classes a utilizar, uma boa documentação do código, a criação do seu diagrama, a utilização de software gestor de versões e ainda a redação de um relatório de qualidade.

Para esta segunda fase, já devemos mostrar um pouco mais com aquilo que o programa final será, tento em conta conceitos como, heranças, encapsulamento, abstração e polimorfismo, interfaces, exeptions, generics, estrutura de código por camadas e tratamento de ficheiros.

2 POO - Programação Orientada a Objetos

A POO é um dos grandes paradigmas da programação.

A principal forma de utilização desta técnica é a determinação de estruturas e operações referentes a um objeto (classes). Desta forma podemos determinar que um objeto obedece a um determinado conjunto de regras, podendo ter atributos (Properties) e comportamentos(Methods).

A POO acenta ainda em 4 pilares fundamentais:

- Heranças
- Encapsulamento
- Abstração
- Polimorfismo

Estes 4 pilares são bem demonstrados no projeto. A herança através da classe-pai Pessoa e as classes-filho Árbitro e Jogador. O encapsulamento e abstração pelos atributos e métodos sendo privates, sendo estes acessíveis através das proprieties e não diretamente à variável.

3 Enumeráveis

Para as classes Arbitro e Jogador é necessário indicar uma ASSOCIACAO e CATEGORIA, e uma POSICAO, respetivamente.

- ASSOCIACAO
  Contém todas as associações de Árbitros de Portugal.
- CATEGORIA
  Discriminação dos níveis profissionais que um árbitro pode alcançar.
- POSICAO
  Enumerável com as posições 'gerais' que um jogador pode ocupar no decorrer do jogo.

4 MVC - Model View Controller

Para uma primeira fase foi-nos requerido uma implementação básica e essencial de classes. No entanto, nesta segunda fase, apresentamos uma estrutura de classes avançada. O Programa encontra-se estruturado segundo o padrão de MVC. De modo a relacionarmos as Classes utilizamos Interfaces para cada membro MVC das classes. Vamos falar do modo que a Interface mantém as relações entre os membros MVC a seguir.

De salientar que cada um dos seguintes continuam a ser classes e devem manter:

- Variáveis de Membro –> Aqui são inicialmente definidas as variáveis como private.
- Construtores –> Os construtores são onde definimos quais os parâmetros que vão corresponder às variáveis, que podem ser definidas pelo utilizador, pelo programa, ou ficarem sem valor
- Propriedades –> São definidos os get e set para cada atributo, podendo alguns sofrer algumas verificações.
- Métodos –> Funções próprias da classe, definidas para serem utilizadas com a classe.

4.1 Model

Os models são as classes onde estão definidos cada tipo de dados e quais as suas variáveis. A sua principal função é ser a estrutura para os tipos de dados definidos.

4.2 View

A view nada mais é do que o Input/Output entre a aplicação e o utilizador, sendo definidas aqui todas as ações a que o utilizador tem acesso na sua tela, fazendo desta forma uma ligação entre o utilizador e os controllers.

4.3 Controller

O controller é onde estão definidos os requests e responses entre toda a aplicação. É feito um request pelo utilizador através da view, o controller processa esse pedido acendendo aos models, enviando um response, de forma a dar resposta à pergunta.

5 Interfaces

Uma interface é implementada para cada classe criada. Em cada interface deve estar definido o método em utilização.Por convenção, é adicionado um 'I' antes do nome da classe que vai ser utilizada e a classe que vai utilizar a interface como herança. Deste modo, para qualquer classe model, view e controller existe uma interface correspondente à classe.

6 Generics

Numa primeira fase do trabalho eram utilizados arrays para fazer conjuntos de objetos, o que iria claramente trazer um grande problema: o espaço de memória, uma vez que ao declarar um array temos também de declarar o número de espaço a reservar na memória.

Os Generics vieram para levar esse problema e ainda dar auxilio através de funções já pré-definidas.

O generic utilizado nesta aplicação foi o List<>, onde podemos utilizar métodos já pré-definidos, tais como:

- .Add() –> Insere um novo objeto à lista.
- .Remove() –> Remove o objeto da lista.
- .Sort() –> Ordena a lista, de forma predefinida.
- .ForEach() –> Percorre toda a lista, elemento a elemento.
- .Contains(Object) –> Verifica se um objeto existe na lista. Retorna true se existir, false no caso de não existir.
- .IndexOf(Object) –> Retorna a posição na lista do objeto.

Alguns exemplos onde são utilizadas listas são ao construir uma equipa, onde existe um conjunto de jogadores.

7 Exceptions

As exceptions trazem auxilio no que toca ao tratamento de erros. Existem 2 tipos de exceptions:

- System Exceptions
- Aplication Exceptions

Tal como o próprio nome indica, um trata de resolver os erros executados pelo sistema, e a outra, de resolver os problemas gerados pela aplicação.

Dentro dos erros de sistemas existem várias formas de apresentar os erros. Os dois mais utilizados nesta aplicação foram:

- FormatException –> Utilizado para precaver erros quando existe a conversão de um tipo de dados para outro tipo.
- OverflowException –> Utilizado quando uma variável excede a sua capacidade máxima.

As exceptions são utilizadas neste programa na sua forma mais simples, uma vez que poderiamos construir classes para dar uma melhor resposta ao tipo de erros. Poderá ser uma melhoria futura neste projeto.

8 Tratamento de Ficheiros

Ainda não implementado (3ª fase).

9 Conclusão

Aquando da entrega desta segunda parte do trabalho, apercebe-mo-nos do quanto este programa tem potencial para crescer e o quanto ainda o podemos melhorar, pois existem alguns pontos que ainda não estão totalmente alinhavados. Será algo a ter em consideração para a entrega final.

Sentimo-nos desafiados cada vez que se ligava o computador e se abria o IDE para dar continuação ao trabalho, ao pensar em como dar a volta às adversidades cada vez que elas surgiam e também ao ter de aprender cada vez mais para puder cada vez mais melhorar o trabalho.

Sabemos que para a entrega final temos muito ainda a melhorar e a acrescentar, pois não nos queremos satisfazer com os mínimos, mas sim lutar pelo melhor que consigamos fazer.

10 Webgrafia

1. Sitio oficial da microsoft, Exceptions
   `https://docs.microsoft.com/en-us/dotnet/api/system.exception`
2. Sitio oficial da microsoft, Generic Lists
   `https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1`
3. Sitio oficial da microsoft, Interfaces
   `https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/interface`