Ficha Prática #09

9.1 Objectivos

1. Descrever a camada de apresentação do exemplo apresentado na Secção 1.3, da Ficha Prática #01.

9.2 Camada de apresentação

Esta camada segue o padrão *Model-Delegate*. O *Delegate* é implementado pela classe `TextUI`. O *Model* é qualquer classe que implemente a interface `ITurmasFacade` (ver linha 23 da classe `TextUI`). Neste caso concreto, utiliza-se uma instância da classe `TurmasFacade` (ver construtor de `TextUI`).

A classe `TextUI` recorre à classe `Menu` para implementar uma interface com menus em modo texto. A **classe** `Menu` assegura a execução da interface através de menus em modo texto. Um exemplo de utilização pode ser visto na classe `TextUI`. Note que esta classe não está completa.

Para criar um menu é necessário definir a lista de opções a apresentar e, para cada opção, qual a sua pré-condição e qual o *event handler* a utilizar. A pré-condição (do tipo `PreCondition`) deverá retornar um valor booleano e é calculada sempre que o menu é apresentado. A opção só fica disponível se o valor da pré-condição for verdade.

O *event handler* é o método que será chamado caso a opção seja seleccionada pelo utilizador. Este método tipicamente irá invocar um método do *Model* ou executer um outro menu. O primeiro caso utiliza-se para apresentar informação ao utilizador e/ou para ler dados e executar operações. O método `adicionarAlunoATurma()` é um exemplo de um *event handler* deste tipo. O segundo caso utiliza-se para implementar interfaces com menus e sub-menus (não é o caso deste exemplo).

Constructor Summary

Constructors

Constructor and Description
`Menu()` Constructor vazio para objectos da classe Menu.
`Menu(java.util.List<java.lang.String> opcoes)` Constructor para objectos da classe Menu (sem título e com List de opções).
`Menu(java.lang.String[] opcoes)` Constructor para objectos da classe Menu (sem título e com array de opções).
`Menu(java.lang.String titulo, java.util.List<java.lang.String> opcoes)` Constructor para objectos da classe Menu (com título e List de opções).
`Menu(java.lang.String titulo, java.lang.String[] opcoes)` Constructor para objectos da classe Menu (com título e array de opções).

Figura 9.1: Construtores da classe `Menu`

Assim, a utilização de um menu consiste nos seguintes passos:

1. Criar o menu.

   Como se pode ver pela lista de construtores da classe (ver Figura 9.1) é possível criar menus com ou sem título e com ou sem a lista de opções a apresentar. Um exemplo de criação de um menu pode ser visto no construtor da classe `TextUI`:

```
this.menu = new Menu(new String[]{
        "Adicionar Aluno",
        "Consultar Aluno",
        "Listar Alunos",
        "Adicionar Turma",
        "Mudar Sala a Turma",
        "Listar Turmas",
        "Adicionar Aluno a Turma",
        "Remover Aluno da Turma",
        "Listar Alunos de Turma"
});
```

   Neste caso, está a utilizar-se o construtor que recebe apenas um array de opções, o título não está a ser definido. Assim sendo, é utilizado o título genérico "*** Menu ***".

2. Definir pré-condições.

   O método `setPreCondition(int i, PreCondition b)` permite (re)definir a pré-condição da `i`-ésima opção do menu[1]. A seguinte definição garantiria que a sétima opção do menu (adicionar aluno a turma) só estaria disponível se existissem alunos e turmas na aplicação[2]:

```
// Registar pré-condições das transições
menu.setPreCondition(7,
    ()->this.model.haAlunos() && this.model.haTurmas());
```

   Note a utilização de uma expressão lambda para definir a pré-condição. Note, ainda, a utilização do *Model* para obter informações sobre os dados.

3. Definir *event handlers*.

   De modo análogo às pré-condições, o método `setHandler(int i, Handler h)` permite definir os *event handlers* das opções do menu. Também no construtor da classe `TextUI`, podemos ver a definição de todos os *event handlers* do menu principal:

```
this.menu.setHandler(1, this::trataAdicionarAluno);
this.menu.setHandler(2, this::trataConsultarAluno);
this.menu.setHandler(3, this::trataListarAlunos);
this.menu.setHandler(4, this::trataAdicionarTurma);
this.menu.setHandler(5, this::trataMudarSalaTurma);
this.menu.setHandler(6, this::trataListarTurmas);
this.menu.setHandler(7, this::trataAdicionarAlunoATurma);
this.menu.setHandler(8, this::trataRemoverAlunoDaTurma);
this.menu.setHandler(9, this::trataListarAlunosTurma);
```

   Neste caso, escolher a opção 1 causa a execução do método `trataAdicionarAluno()`, escolher a opção 2, do método `trataConsultarAluno()`, etc.

   Na prática, ao definir os *event handlers*, estamos a definir de que modo o utilizador pode navegar na aplicação.

4. Executar o menu.

   O método `run()` da classe `TextUI` executa o menu, utilizando, para isso, o método `run()` da classe `Menu`:

[1] A primeira opção do menu está na posição 1.
[2] A versão da aplicação que lhe foi fornecida, não define pré-condições. Tal fica como exercício.

```
public void run() {
    this.menu.run();
    System.out.println("Até breve!...");
}
```

O método apresenta, ou não, as opções, dependendo dos valores das respectivas pré-condições, e executa os *event handlers*, em resposta às escolhas do utilizador.

O método `Menu::run` acrescenta a opção 0 (zero) ao menu, correspondente a sair do menu. Deste modo,o menu será executado até que o utilizador opte por sair. Para os casos em que se pretenda executar um menu apenas uma vez, a classe `Menu` disponibiliza o método `runOnce()`.

Vale a pena referir que a definição de pré-condições e *event handlers* pode ser realizada por qualquer ordem.

Finalmente, o método `void option(String name, PreCondition p, Handler h)`, não ilustrado aqui, permite adicionar uma opção ao menu, com a respectiva pré-condição e *event handler*. Este método é útil nos casos em que se pretende criar o menu de forma dinâmica (o menu é criado sem qualquer opção e estas são adicionadas com o método acima).

9.3 Exercícios

1. A camada de interface está incompleta, faltando implementar as pré-condições. Adicione-as ao programa.

   Teste a aplicação para verificar que funciona correctamente.

2. A interface apresenta um único menu com todas as opção. Na Figura 9.2 é apresentada uma lógica de navegação para a interface que é mais estruturada. Segundo o modelo, existirão dois sub-menus: um para gestão de alunos, outro para gestão de turmas. Adapte a solução existente para implementar a interface de acordo com o modelo apresentado na figura.

   Verifique, mais uma vez, que tudo funciona correctamente.

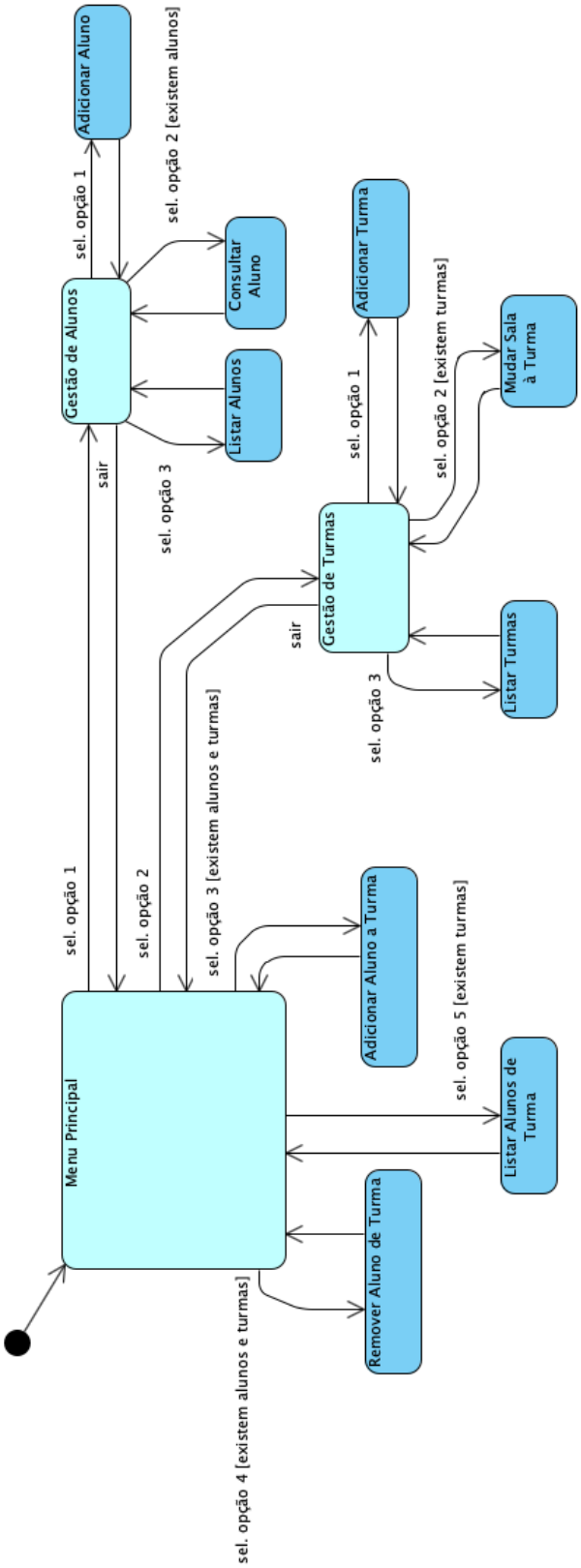

Figura 9.2: Mapa de navegação pretendido para a aplicação