Thibault Langlois

Programação Centrada em Objectos

Projecto (fase 1)

2016.10.24.23.08.45

1 TeleTasca

O senhor António é dono de um restaurante (a famosa Tasca do Bairro). Não está satisfeito com o seu funcionamento, por um lado há um grande desperdício de comida porque tem de preparar pratos sem saber exatamente a quandidade que será servida e por outro lado os clientes ficam a espera mais do que necessário. Resolveu alterar completamente o modo de funcionamento. Vai disponibilizar uma aplicação web que permite fazer encomendas que poderão ser consumidas no local na hora marcada. Deste modo resolve os dois problemas: não há desperdicios porque sabe o número de pratos a preparar e os clientes não ficam a espera que o prato seja confecionado desde que chegam a hora marcada.
A sua missão é realizar um protótipo para esta aplicação. Nesta primeira abordagem, o prototipo não terá uma interface web. A interação será feita usando o teclado ou atravez de ficheiros.

2 Especificação

A aplicação tem dois tipos de utilizadores: o gerente, e os clientes. O gerente pode :

- Adicionar pratos a lista de pratos disponíveis,
- Remover pratos da lista,
- Consultar as encomendas efetuadas.

Os clientes podem:

- Criar uma conta (necessário para poder fazer login na aplicação),
- Fazer login (necessário para poder fazer encomendas),
- Encomendar pratos,
- Consultar a lista das encomendas feitas.

A ação “Criar uma conta” consiste em criar um registo para o utilizador. O utilizador deve fornecer o seu nome e endereço de mail. Se não existir nenhum utilizador já registado com este endereço, uma nova conta é criada. A ação de “Login” consiste em pedir ao utilizador um endereço de email caso o email corresponder a algum utilizador registado, este utilizador passa a ser o utilizador corrente. Neste prototipo não é pedido uma password. Caso não existir nenhum utilizador com o email indicado, o programa deve informar o utilizador e voltar a apresentar o menu do cliente.

3 Classes a desenvolver

Os objectos essenciais para esta aplicação são:

- O cliente: é representado no sistema pelo seu nome e o seu endereço de email. Não pode haver dois utilizadores com o mesmo endereço de mail.

- O prato: é representado por uma String que o descreve, e um preço.
- Uma encomenda: é composta por um cliente, uma data (incluindo a hora) e um conjunto de pratos.

Neste protótipo existe apenas um gerente pelo que não é necessário representa-lo por uma classe. A aplicação tem de gerir *vários* clientes, *vários* pratos, *várias* encomendas. Por essa razão vai ser necessário haver objectos que representam esses conjuntos. Tradicionalmente essas classes são chamadas "catalogos". Haverá portanto um catalogo de pratos, um catalogo de clientes e um catalogo de encomendas. As classes descritas até agora são específicas ao domínio da applicação serão portanto agrupadas no pacote `domain`.
A aplicação deve conservar o seu estado entre duas utilizações. Isto é, deve guardar toda a informação relativa aos clientes, pratos e encomendas.
Numa versão profissional da aplicação seria feito usando uma base de dados. As classes responsáveis pela interação com a base de dados costumam ser agrupadas num pacote separado (que vamos chamar `persistence`). No caso deste protótipo os dados vão estar guardados em ficheiros texto usando um formato "CSV" (Comma Separated Values) que tem a vantagem de poder ser lido com uma folha de calculo. Neste formato, cada entidade (prato, cliente, encomenda) é representada por uma linha, os vários atributos (colunas) sendo separados com virgulas. Vejam, na secção seguinte, mais pormenores sobre a representação dos dados em ficheiros.
Finalmente, existe um pacote `main` que agrupa as classes responsáveis por arrancar a aplicação e tratar da interação com os utilizadores. Essas classes são fornecidas mas incompletas.

4 Representação dos dados em ficheiros

Os dados relativos ao clientes, pratos e encomendas vão estar guardados em três ficheiros separados. A informação não deve ser repetida nos vários ficheiros. Por exemplo, associado a uma encomenda temos um cliente e uma lista de pratos. Não podemos repetir toda informação relativa ao cliente ou aos pratos em cada registo de uma encomenda. Para além de ser um enorme desperdício de memória, coloca o problema da consistência dos dados. Se o preço de um prato for alterado teria que se alterar em todas as encomendas que incluem o tal prato.
Para resolver este problema, acrescenta-se uma referência única a cada instância de objecto. No caso do cliente, a referência já existe dado que está especificado que o endereço de email deve ser único na aplicação. No caso do prato ou da encomenda a solução consiste em acrescentar um atributo de tipo inteiro (que vamos chamar `id`) que representa de forma única a instância. Isto é, o programa deve tratar de atribuir os `id` de forma a que sejam únicos em cada classe.
Uma vez que cada instância tem associado um Id único, pode-se (por exemplo) representar no ficheiro das encomendas, a lista de pratos por uma lista de inteiros evitando assim a duplicação dos dados.
Por exemplo, o ficheiro que contém os pratos poderá ser representado assim:

```
1,bacalhau à braz,5.99
2,sopa de legumes,1.5
3,sopa da pedra,2.5
```

5 Casos de Uso

Para testar rapidamente o programa, é conveniente não ter que entrar os dados no teclado para cada teste. A classe `App` está preparada para puder usar o programa em modo interactivo ou não interactivo. No segundo caso os dados de entrada estão lidos de um ficheiro enquanto no primeiro, são lidos do teclado.
O modo interactivo é iniciado pelo método

```
    /**
     * This method may be called to use the application in default mode i.e.
     * interacting with the keyboard.
     *
     * @throws IOException
```

```
     */
    private static void interactiveMode() throws IOException {
        try (Scanner in = new Scanner(System.in)) {
            in.useLocale(Locale.US);
            Menu.mainMenu(in);
        }
    }
```

que crie uma instância de `Scanner`, aberta para o teclado e passa-a ao método que trate do menu principal da aplicação.
O modo não interactivo lê as entradas de um ficheiro. Tipicamente, esses ficheiros de texto representam uma sequência de *inputs* curta que tem como objectivo de testar um número muito limitado de funcionalidades (idealmente apenas uma). Chamamos esses ficheiros "casos de uso" porque representam casos típicos na utilização da aplicação.

```
    /**
     * This method allows the application to work in non interactive mode i.e.
     * the input is read from a file. It should be used for testing. A file,
     * called a use-case contains a sequence of inputs that allows testing some
     * functionalities of the application. A use-case file may contain comments
     * (useful to make it easy to understand). Comments begin with the #
     * character and extend until the end of the line. Here is an example of
     * use-case file:
     * -----
     * # Caso de uso 1: o gerente adiciona um prato.
     * # user = gerente
     * 2
     * 1 # adicionar um prato
     * bacalhau à braz # descrição do prato
     * 5.99 # preço
     * 5 # terminar
     * 3 # terminar
     * -----
     *
     *
     * @param useCaseFileName A String that represents the name of a file that
     * contains a use-case.
     * @throws IOException
     * @requires the contents of the file must be correct with respect of the
     * menus (see class Menu) and the input data expected by the application,
     * unless the objective of the test is to verify an illegal situation.
     */
    private static void executeUseCase(String useCaseFileName) throws IOException {
        System.out.println("Test:  " + useCaseFileName);
        Scanner in = new Scanner(new File(useCaseFileName));
        in.useLocale(Locale.US);
        in.nextLine();
        Menu.mainMenu(in);
        in.close();
    }
```

O método recebe um nome de ficheiro e cria uma instância de `Scanner` para ler o ficheiro. Passa a instância ao método que trate do menu principal da aplicação. Existe outro método (incompleto) que executa todos os testes :

```
    public static void executeAllUseCases() throws IOException {
        executeUseCase("data/usecase1.dat");
        executeUseCase("data/usecase2.dat");
        // ...
        // colocar aqui todos os use-case
    }
```

O ficheiros de casos de uso podem conter comentários. Os comentários começam com o caracter `#` e terminam no fim da linha. Por exemplo, o ficheiro seguinte :

```
# Caso de uso 1: o gerente adiciona um prato.
# user = gerente
2
1 # adicionar um prato
bacalhau à braz # descrição do prato
5.99 # preço
5 # terminar
3 # terminar
```

contém as entradas que permitem introduzir um prato no sistema.
A classe `Menu` contém os menus e a interação com utilizador. Deve ser completada (falta por exemplo a leitura dos dados) mas **não deve alterar os menus !** Caso contrário não será possível avaliar o seu projecto. Para poder ler ficheiros de caso de uso com comentários deverá usar os métodos (fornecidos) `nextInt(Scanner in)`, `nextDouble(Scanner in)`, `nextLine(Scanner in)`, `nextDate(Scanner in)` e `nextTime(Scanner in)`.

6 Exemplo de interação

```
Você é:
Cliente...................1
Gerente...................2
Terminar..................3
>

2


Adicionar um prato.........1
Remover um prato...........2
Consultar as encomendas....4
Terminar...................5
>

2
Descrição do prato:  bacalhau à braz
Preço:  5.99


Adicionar um prato.........1
Remover um prato...........2
Consultar as encomendas....4
Terminar...................5
>

5


Você é:
Cliente...................1
Gerente...................2
Terminar..................3
>

3
```

---

7 Código fornecido

Pode importar o projecto fornecido (que contém um esqueleto das principais classes) seguindo as instruções seguintes no eclipse:

1. Fazer unzip do ficheiro fornecido,
2. `File > Import...  > Maven > Existing Maven Project > Next > Browse` (escolher a pasta onde está o projecto) `> Finish`.

As assinaturas dos métodos fornecidos (mesmo que sejam apenas esboços) não devem ser alteradas.

8 Entrega

O projecto deve ser realizado por grupos de dois alunos. Não há registo prévio dos grupos, basta indicar a composição do grupo na altura da entrega. O trabalho deve ser entregue no dia 20 de Novembro. As modalidades de entega serão divulgadas em breve.