Proramação Centrada em Objectos

Projecto (fase 2)

2016.11.27

A segunda fase do projecto consiste em implementar alguns melhoramentos tanto do ponto de vista da qualidade da progrmação como em termos de novas funcionalidades.

1 Padrão Singleton

O correcto funcionamento do programa implica que as intâncias que representam os catalogos sejam únicas. Deve alterar essas classes de forma a implementar o padrão singleton.

2 Ordenação das encomendas

Quando o gerente consulta as encomendas feitas, essas devem aparecer por ordem cronologica inversa (a mais recente primeiro). Altere o seu programa usando o mecanismo de ordenação visto nas aulas i.e. atraves da implementação das interfaces adequadas.

3 Processamento das exceções

De maneira a tormar a aplicação mais robusta, na segunda fase será implementado o tratamento de erros. O objectivo é avisar o utilizador dos eventuais problemas com o máximo de detalhes. A maioria dos problemas podem ocorrer na leitura dos dados (tanto ao teclado como a partir de um ficheiro).
Os erros cometidos na escolha de uma opção de um menu (número que não consta nas opções do menu, entrada de um valor não númerico) devem ser ignorados, isto é: o programa deve voltar a apresentar o mesmo menu caso o utilizador entre um valor inválido.
O programa deve detectar os erros seguintes, lançando as exceções indicadas:

- Detecção dos ID duplicados. Exceção: **`DuplicatedIdException`**.
- Datas inválidas. Exceção: **`InvalidDateException`**. Esta verificação é simples de detectar desde que use a instância de `Calendar` em modo não "*lenient*" (indulgente). Basta alterar a instância com : `c.setLenient(false);` para que seja lançada a exceção `IllegalArgumentException` quando o método `set` é usado com valores errados.
- Quando durante a leitura de um ficheiro é feito referência a um ID (correspondente a classe `Dish` ou `Client`) que não existe no respectivo catalogo, a exceção **`InvalidIdException`** deverá ser lançada.
- Outros erros de formatação (diferente dos anteriores). A exceção **`BadDataFormatException`** será lançada quando um erro no formato de um ficheiro será detectado.

Note que a classe **`InvalidDateException`** deve extender a classe **`BadDataFormatException`** e a classe **`DuplicatedIdException`** deve extender a classe **`InvalidIdException`**.
As classes de exceções que não existem na API devem ser criadas. Devem possuir um construtor que permite associar uma mensagem à exceção. **As exceções devem ser apanhadas (bloco `catch` na classe `App`)**. O tratamento da exceção deve imprimir a informação relevante e terminar a execução do programa (usando o método `System.exit()`).
Existem situações de erro que podem ser detectadas procurando apanhar uma exceção da API (por exemplo a `InputMismatchException`. Neste caso a exceção pretendida (por exemplo `BadDataFormatException`) deve ser lançada no bloco catch que apanhou a exceção da API. .

4 Remoção dos pratos

Tal como foi concebida na primeira fase, a operação de remoção dos pratos (por parte do gerente) pode causar problemas. Por exemplo quando um cliente (ou o próprio gerente) quer visualizar uma encomenda composta de pratos que já foram removidos da base de dados.
Nesta fase do projecto o problema será corrigido. Deve alterar o seu programa de forma a que, em vez de remover o prato da lista de pratos disponíveis, seja apenas marcado como "não disponível". Como é natural, os pratos assim marcados não serão propostos aos clientes mas permanecerão na aplicação evitando assim os eventuais erros. Nas listagens de encomendas, os partos não disponiveis serão assinalados.

5 Novas funcionalidades

Na segunda fase do projecto, estendemos o conceito de prato ao conceito de alimento. Cada alimento traz informação sobre alguns factos nutricionais.
Exemplo: *servings* (número de porções), calorias (gramas), lípidos (gramas), etc.
Os alimentos que consideramos na segunda fase do projecto são:

- Dish
- Drink

Uma instância de `Dish` traz todas as informações e funcionalidades da primeira versão como também as informações nutricionais. Uma instância de `Drink` traz só as informações nutricionais.
O objetivo da aplicação *TeleTasca-Version 2* é oferecer menus diferenciados que encontram as exigências do cliente, sendo as características suportadas: menu normal (*standard*), menu dietético (*light*), menu para dois pessoas (*for two*). A nova política da Tasca é atirar mais clientes: além dos menu diferenciados, a Tasca oferece uma bebida em todas as encomendas de valor igual ou maior a 6 euros (10 euros no caso do menu para dois pessoas). A bebida é fixa para cada menu (não há escolha de bebida): por isso, a aplicação não precisa de gravar alguma informação relativa á bebida nas ordens (Assumimos que a bebida é oferta automaticamente na Tasca).

5.1 Especificação

A aplicação da segunda fase adiciona as seguintes funcionalidades:
Clientes. Um cliente pode ordenar pratos a partir de três tipologias de menus:

- Standard (idem à primeira fase)
- Light (contém pratos com poucas calorias)
- ForTwo (contém pratos para dois ou mais pessoas)

No caso do Menu Light, a aplicação mostra todas as informações nutricionais.
Exemplo:

```
sopa de legumes...1.5 EUR...1 servings...290 kcal...0.1g of glucides...
sopa da pedra...2.5 EUR...1 servings...380 kcal...0.2g of glucides...
```

Para cada menu, a TeleTasca oferece uma bebida quando o valor total da encomenda é maior o igual a 6 euros (10 euros no caso do menu ForTwo). Assumimos que cada menu tem uma bebida associada (não há escolha).
Exemplo:

- Cerveja (menu Standard)
- CocaZero (menu Light)
- Vinho (menu ForTwo)

Quando uma bebida é oferta, a aplicação imprime uma mensagem da forma:

```
Parabens! A TeleTasca oferece-lhe uma bebida!
Cerveja..1 servings...155 kcal...13.0g of glucides...
```

5.2 Classes a desenvolver

NutritionFacts Cada alimento traz informação sobre alguns factos nutricionais. Essa informação não pode ser mudada depois da inicialização.

- servingSize (int)
- servings (int)
- calories (int)
- fat (double)
- sodium (double)
- carbohydrate (double)

Invariantes: `servingSize` e `servings` têm de ser definidos (maior que 0). A aplicação lança uma exceção de tipo `InvalidParameterException` (presente na API) se tentar de inicializar um objecto com `servingSize` ou `servings` inferiores ou iguais a zero. As restantes variáveis são inicializadas com valor ≥ 0: se tentar inicializar um objecto com calories, fat, sodium ou carbohydrate ≤ 0, a aplicação imprime um aviso no e põe o valor por omissão 0.
Métodos: A classe disponibiliza métodos *getters* para as suas variáveis, o método `@Override String toString()`, e o método `String quickFacts()` que restitui uma `String` da forma:

```
...1 servings...290 kcal...0.1g of glucides...
```

A classe utiliza métodos estáticos privados para garantir as invariantes: `static int mustBePositive(int x, String msg)`, e `static double mayBePositive(double x, String msg)`. Em cada método, o parâmetro de tipo `String` é utilizado em caso de erro para imprimir a origem do problema.
Dish extends NutritionFacts A classe `Dish` estende `NutritionFacts` e tem um atributo `DishType dishType`, onde `DishType` é o `enum` :

```
public enum DishType{
    STANDARD, LIGHT, FORTWO
}
```

O campo é inicializado automaticamente no momento da criação de uma instância de Dish a partir dos valores dos campos `int servings` e `double calorias`: `servings > 1` implica `dishType = FORTWO`, e `calorias <= 500` implica `dishType = LIGHT`; nos restantes casos `dishType = STANDARD`.A inicialização é feita seguindo essa ordem: se há um prato com `servings > 1` e `calorias <= 500`, o prato terá `dishType = FORTWO`.
Métodos A classe disponibiliza novos métodos: o getter pelo field dishType, e o metódo `@Override String quickFacts()` que restitui um texto da forma:

```
sopa de legumes...1.5 EUR...1 servings...290 kcal...0.1g of glucides...2
```

A classe utiliza o método privado `void setDishType()` para o efeito.
Drink extends NutritionFacts A classe `Drink` estende `Nutritionfacts` e contém as constantes `static final Drink BEER`, `static final Drink COCAZERO}`, `static final Drink WINE`.
Metodos. O construtor é privado (não há outras bebidas). A classe disponibiliza o método `@Override String toString()`.
Menu A classe menu tem métodos adicionais para implementar as novas funcionalidades da aplicação.
Métodos. A classe utiliza os métodos estáticos privados : `Dish selectLightDish(Scanner in)`, `Dish selectDish(Scanner in, boolean menuTwoPersons)`, e `Drink offerDrink(double preço, int menu)`.
O metodo `selectDish` da segunda versão é obtido a partir do método homónimo da primeira versão, mas tem um parâmetro `flag` adicional: quando `menuTwoPersons = false`, o método da segunda versão comporta-se como o método da primeira versão. O método `offerDrink` restitui a bebida oferta, ou `nil`.
O método estático privado `void makeOrder(Scanner in)` recebe a escolha de menu do utilizador e invoca o método apropriado. Se o utilizador selecionar o menu `Light` ou o menu `ForTwo` e não há pratos no menu, o método imprime uma mensagem de aviso e mostra automaticamente o menu `Standard` **sem pedir** uma ulterior escolha ao utilizador.

5.3 Representação dos dados em ficheiros

Dish. A representação de um prato em ficheiro pode ser ampliada da seguinte forma:

`1,bacalhau à braz,5.99,true,350,1,800,5.1,20.6,0.42`

A partir da quinta posição, temos a representação dos valores dos factos nutricionais: e.g. `servingSize = 350` (gramas), `servings = 1` (pessoa),..., `carbohydrate=0.42`(gramas).

5.4 Casos de Uso

Para testar a aplicação, modificar os casos de uso da primeira versão de uma forma. Exemplo:

```
# Caso de uso 2: o gerente adiciona um prato.
# user = gerente
2
1 # adicionar um prato
sopa de legumes # descrição do prato
1.50 # preço
250 #servingSize
1 #servings
290 #calories
0.8 #fat
0.1 #sodium
5.2 #carbohydrate
1
```

5.5 Exemplos de interação

Exemplo 1: o gerente adiciona um prato.

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

Gerente
Adicionar um prato.....1
Remover um prato.....2
Consultar as encomendas....4
Terminar.....5
> 1
Descrição do prato:  Lasagne
Preço:  7.5
Peso:  620
Número de pessoas:  1
Calorias:  900
Lípidos:  29.85
Sal:  2.4
Hidratos de carbono:  99.5

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

```
> 5


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

> 3
```

Exemplo 2: o cliente regista-se, faz a login, seleciona o menu Light, ordena três pratos e recebe uma bebida em oferta.

```
**********************TELETASCA***********************
Você é:
Cliente...................1
Gerente...................2
Terminar..................3

> 1


Cliente
Criar conta...............1
Log in....................2
Encomendar pratos.........3
Lista de encomendas.......4
Terminar..................5

> 1
O seu nome:  Marco Goa
O seu email:  mgoa@inyour.dreams.pt


Criar conta...............1
Log in....................2
Encomendar pratos.........3
Lista de encomendas.......4
Terminar..................5

> 3


Deve fazer log in antes de fazer uma encomenda.
Criar conta...............1
Log in....................2
Encomendar pratos.........3
Lista de encomendas.......4
Terminar..................5

> 2
Email:  mgoa@inyour.dreams.pt


Criar conta...............1
Log in....................2
Encomendar pratos.........3
Lista de encomendas.......4
Terminar..................5
```

```
> 3
Data da entrega [dd/mm/yyy] :  30/12/2016
Hora da entega [hh:mm] :  23:00


Menu normal .............1
Menu light ..............2
Menu para dois ......... 3

> 2


sopa de legumes...1.5 EUR...1 servings...290 kcal...0.1g of glucides...2
sopa da pedra...2.5 EUR...1 servings...380 kcal...0.2g of glucides...3
Escolhe um prato (0 para terminar):

3


sopa de legumes...1.5 EUR...1 servings...290 kcal...0.1g of glucides...2
sopa da pedra...2.5 EUR...1 servings...380 kcal...0.2g of glucides...3
Escolhe um prato (0 para terminar):

3
sopa de legumes...1.5 EUR...1 servings...290 kcal...0.1g of glucides...2 sopa da pedra...2.5 EUR...1 se
Escolhe um prato (0 para terminar):
2
sopa de legumes...1.5 EUR...1 servings...290 kcal...0.1g of glucides...2 sopa da pedra...2.5 EUR...1 se
Escolhe um prato (0 para terminar):
0


Parabens! A TeleTasca oferece-ilhe uma bebida!
CocaZero...1 servings...0 kcal...0.1g of glucides...
Criar conta.................1
Log in......................2
Encomendar pratos..........3
Lista de encomendas........4
Terminar...................5

> 4


Listar encomendas.
5,mgoa@inyour.dreams.pt,2016/12/30/0/23,3,3,2

Criar conta.................1
Log in......................2
Encomendar pratos..........3
Lista de encomendas........4
Terminar...................5

> 5


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

> 3
```

6 Entrega

O projecto deve ser realizado por grupos de dois alunos. A composição dos grupos deve ser igal à entrega da primeira fase. O trabalho deve ser entregue no dia 20 de Dezembro. As modalidades de entrega serão divulgadas atempadamente.