POO (MiEI/LCC)

2018/2019

Ficha Prática #07

Hierarquia, herança, polimorfismo, interfaces e serialização de informação.

Conteúdo

1 Objectivos

Durante três aulas, o projecto HoteisInc (Hoteis Incorporated) servirá para abordar vários conceitos leccionados em POO. Do mesmo modo, servirá como auto-avaliação para que os alunos possam fazer um ponto da situação sobre a disciplina.

O objectivo deste projecto é o desenvolvimento, de uma forma incremental, de uma aplicação que combina todos os conceitos leccionados ao longo do semestre. Na primeira fase serão abordados os conceitos de hierarquias de classes, assim como mapeamentos (c.f. API `Map<K,V>` já praticada anteriormente na Ficha 6). A segunda fase abordará os conceitos de Interface, nomeadamente as interfaces `Comparable<T>` e `Comparator<T>`. Na terceira e última fase será abordado o tema de *streams* de dados, assim como o tratamento de excepções.

2 Projeto HoteisInc

Considere que se pretende desenvolver um sistema para a gestão de uma cadeia de hotéis (sistema HoteisInc). Pretende-se ter uma classe que agrupe a informação sobre os diversos hotéis da cadeia, sendo que existem diferentes hotéis (com diferentes características). O sistema a desenvolver deverá ser capaz de representar essa informação, e fornecer um conjunto de funcionalidades que permita consultá-la e manipulá-la.

2.1 Fase 1

Um `Hotel` é a entidade base da aplicação, que contém um `codigo` de identificação, um `nome`, uma `localidade`, uma `categoria` (normalmente expressa em estrelas), um `número de quartos disponíveis` e um `preço` por quarto. Actualmente a cadeia de hotéis faz equipamentos homogéneos a nível de quarto, pelo que todos os quartos de um hotel são do mesmo tipo (previsivelmente esta situação mudará num futuro próximo).

1. Crie a classe `Hotel` e a classe `HoteisInc`, e implemente nesta os seguintes métodos:

   (a) Verificar a existência de um hotel, dado o seu código.
   ```
   public boolean existeHotel(String cod)
   ```

   (b) Devolver a quantidade de hotéis existentes na cadeia.
   ```
   public int quantos()
   ```

   (c) Devolver o número total de hotéis de uma dada localidade.
   ```
   public int quantos(String loc)
   ```

   (d) Devolver a ficha de um hotel, dado o seu código.
   ```
   public Hotel getHotel(String cod)
   ```

(e) Adicionar a informação de um novo hotel.
```
public void adiciona(Hotel h)
```

(f) Devolver uma lista contendo a cópia de todos os hotéis no sistema.
```
public List<Hotel> getHoteis()
```

(g) Adicionar a informação de um conjunto de hotéis.
```
public void adiciona(Set<Hotel> hs)
```

2. Crie agora uma nova classe de hotéis, o `HotelStandard` em que o preço dos quartos é ainda igual em todo o hotel, mas varia com a época (que pode ser alta ou baixa), sendo que na época alta acrescem 20 euros ao preço. Refaça a classe `HoteisInc` para que possa passar a gerir além dos hotéis normais este novo tipo de equipamentos.

3. Considere agora que aos tipos de hotéis já especificados se acrescentam os hotéis *Premium*, nos quais existe uma `taxa` de luxo (variável de hotel para hotel), que acresce ao valor do quarto.

   Utilizando as implementações de `HoteisInc`, `Hotel`, `HotelStandard` e `HotelPremium` já desenvolvidas, implemente novamente as seguintes funcionalidades na classe `HoteisInc`:

   (a) Verificar a existência de um hotel, dado o seu código.
   ```
   public boolean existeHotel(String cod)
   ```

   (b) Devolver a quantidade de hotéis existentes na cadeia.
   ```
   public int quantos()
   ```

   (c) Devolver o número total de hotéis de uma dada localidade.
   ```
   public int quantos(String loc)
   ```

   (d) Devolver a quantidade de hotéis de um dado tipo (Standard, Premium).
   ```
   public int quantosT(String tipo)
   ```

   (e) Devolver a ficha de um hotel, dado o seu código.
   ```
   public Hotel getHotel(String cod)
   ```

   (f) Adicionar a informação de um novo hotel.
   ```
   public void adiciona(Hotel h)
   ```

   (g) Devolver uma lista contendo a cópia de todos os hotéis no sistema.
   ```
   public List<Hotel> getHoteisAsList()
   ```

   (h) Adicionar a informação de um conjunto de hotéis.
   ```
   public void adiciona(Set<Hotel> hs)
   ```

   (i) Alterar o estado da época de todos os `HotelStandard`.
   ```
   public void mudaPara(String epoca)
   ```

Pretende-se agora criar um novo tipo de hotéis, os hotéis `Discount`. Estes têm um preço variável de acordo com a sua ocupação. O preço por quarto destes hotéis varia linearmente desde 50% do preço (face aos hotéis `Standard`), para uma ocupação de 0%, até 90% do preço, para uma ocupação de 100%.

10. Adicione a classe `Discount` à hierarquia de hotéis já definida.

11. Acrescente, na classe `HoteisInc`, um método para calcular o valor total diário recebido pela cadeia, considerando uma ocupação dos hotéis de 100%.

Auto avaliação 1 Como forma de auto avaliação, propõe-se a implementação da classee `HoteisIncList`, com os mesmos requisitos que `HoteisInc`, mas utilizando como representação dos hotéis um `List<Hotel>`.

Auto avaliação 2 Propõe-se também a implementação da classe `HoteisIncSet`, com os mesmos requisitos que `HoteisInc`, mas utilizando como representação dos hotéis um `TreeSet<Hotel>`.

2.2 Fase 2

Considere que esta cadeia de hotéis também criou o seu cartão de fidelidade que atribui aos clientes. Ao fazer uma reserva nos hotéis `Standard` e `Premium`, é possível obter pontos de bonificação (X pontos por cada euro, sendo X um valor variável em função de cada hotel).

1. Desenvolva a interface `CartãoHoteis`, que deverá ser implementada por estes dois tipos de hotéis. Esta interface deverá garantir as funcionalidades a seguir descritas.
   - Definir o valor de pontos a atribuir por cada euro.
   - Obter o valor de pontos a atribuir por cada euro.
2. Actualize as classes `HotelStandard` e `HotelPremium` de modo a que implementem a interface.
3. Acrescente, na classe `HoteisInc`, um método para obter a lista dos hotéis que atribuem pontos.

   ```
   public List<CartaoHoteis> daoPontos()
   ```

Pretende-se agora poder consultar a informação de forma ordenada por diferentes critérios. Assim, altere a classe `HoteisInc` para suportar as seguintes funcionalidades.

4. Obter um `TreeSet<Hotel>`, ordenado de acordo com a ordem natural dos hotéis

   ```
   public TreeSet<Hotel> ordenarHoteis()
   ```

5. Obter um `TreeSet<Hotel>`, ordenado de acordo com o comparador fornecido

   ```
   public TreeSet<Hotel> ordenarHoteis(Comparator<Hotel> c)
   ```

6. Guardar (de forma a que sejam identificáveis por nome) comparadores na classe `HoteisInc`, permitindo assim ter disponíveis diferentes critérios de ordenação.
7. Obter um iterador de `Hotel`, ordenado de acordo com o critério indicado:
   `public Iterator<Hotel> ordenarHoteis(String criterio)`

2.3 Fase 3

Deverá ser possível persistir a informação dos hotéis, representada em memória por `HoteisInc`. Assim, implemente em `HoteisInc` funcionalidades descritas a seguir.

1. Exportar a informação de todos os hotéis como ficheiro CSV.
2. Gravar e carregar a instância de `HoteisInc` num ficheiro de objeto em disco.

De forma a melhorar a implementação de `HoteisInc`, implemente agora excepções para suportar o tratamento dos erros a seguir descritos:

3. Hotel inexistente numa consulta.
4. Adicionar hotel repetido.
5. Adicionar hotel inválido.
6. Definição de valores inválidos (e.x. valores negativos).

Finalmente, para completar a implementação do sistema...

7. Desenvolva a classe `HoteisIncMenu` que deverá dar accesso às funcionalidades da classe `HoteisInc` através de um menu em modo texto.