Parte 2 - 12.5 valores

Considere que se pretende implementar uma cadeia de hotéis que é assegurada pela classe `HoteisPOO`. Essa classe guarda os hotéis que o grupo hoteleiro possui. Neste momento os hotéis do grupo são todos similares, possuindo quartos de características diferentes e consequentemente com preços por dia também diferenciados.

Cada instância da classe `Hotel` deverá ter a informação respeitante ao seu nome, aos seus quartos e também deverá guardar a informação dos registos de alocação de um determinado quarto. Considere-se que esses registos são instâncias da classe `Registo` e esta deverá conter além da informação do número de registo, que deve ser *sequencial e atribuída automaticamente*, a data de início da ocupação do quarto e a data de término.

Cada hotel tem quartos de tipo diferente, sabendo que estão neste momento disponíveis implementações de `QuartoSingle`, `QuartoDuplo` e `QuartoPremium`. O preço por dia do `QuartoSingle` é função da época do ano, o `QuartoDuplo` se tem vista privilegiada e o `QuartoPremium` é função do número de metros quadrados disponíveis. Como é habitual nestes casos, o valor a pagar por uma estadia pode depender de vários factores, tais como descontos, promoções, etc.

Considere os seguintes excertos de código:

```
public class HoteisPOO implements Serializable {
  private Map<String, Hotel> hoteis;
  ...
}


public class Registo implements Comparable<Registo>, Serializable {

  // variáveis
  // ...

  public int numDiasReserva() {...} // devolve o número de dias deste
                                    // registo de ocupação do quarto

  public double valorAPagar() {...} // determina o valor a pagar pela
                                    // ocupação do quarto, tendo em conta
                                    // os descontos que se decida aplicar
}



public abstract class Quarto implements Serializable {
  private String numeroQuarto;

  ...
  public abstract double precoPorDia(); // determina o valor do preço do quarto
                                        // que é anunciado pelo hotel

}
```

Considere que a estratégia de associação entre `Hotel` e os seus quartos e registos de ocupação dos mesmos é de **composição**, mas tal já não é necessário na relação entre o `Registo` e o `Quarto` a que se refere.

Assuma, para as perguntas seguintes, que os métodos usuais (get, set, equals, clone, hashcode, ...) estão disponíveis, a menos que sejam solicitados, e responda às questões:

Questão 6

1) Efectue a declaração das variáveis de instância de `Hotel` e `Registo` e 2) **justifique brevemente** a escolha das estruturas de dados que faz.

3) Codifique também o método construtor `public Hotel(Iterator<Quarto> quartos)`, da classe `Hotel`, que cria uma instância de hotel com os quartos fornecidos por parâmetro.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1 *Reservado aos docentes*

Questão 7

Codifique o método `public void adicionaRegisto(LocalDate entrada, LocalDate saida, String numQuarto) throws...`, da classe `Hotel`, que adiciona um registo de ocupação de um quarto, caso este exista (não necessita de fazer o código da classe de excepção).

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1 *Reservado aos docentes*

Questão 8

Codifique o método `public List<String> hoteisMaisOcupados()`, da classe `HoteisPOO`, que devolve os três hotéis que registam o maior número de dias de reserva entre todos os hotéis existentes. Caso exista mais do que um hotel com o mesmo número de dias de ocupação então deverá ser devolvido aquele que tiver o nome alfabeticamente maior. Caso não tenha respondido à pergunta 6, considere que cada hotel guarda um Map com os Registos de ocupação de quartos.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1 *Reservado aos docentes*

Questão 9

Considere que a classe `Hotel` possui um método `public List<Quarto> quartosLivres(LocalDate entrada, LocalDate saida) throws SemDisponibilidadeException` que determina para um hotel, em função das datas, quais as hipóteses de quartos disponíveis, sabendo que não será possível o registo de reservas quando todos os quartos estão ocupados.

Pretende-se agora ter uma nova classe `HotelComFilaEspera`, que para as reservas tardias as coloca numa estrutura de dados indexadas por ordem de chegada, de acordo com o excerto de código abaixo.

```
public class HotelComFilaEspera extends Hotel {
   private List<Registo>> reservasEmEspera;
   ...
}
```

Para esta nova classe codifique o método `public void adicionaReserva(LocalDate entrada, LocalDate saida)`, por forma a permitir também registar os pedidos de reserva que não podem ser, de momento, atendidos. Codifique também o método `public void ocupaQuartoComReservaEmEspera()`, que determina se existe, naquele momento, algum quarto livre no hotel e atribui-o à reserva em espera há mais tempo.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1 *Reservado aos docentes*

Questão 10

Considere agora que existe a necessidade de criar um novo tipo de quarto, o `QuartoDuploSenior`, que se destina apenas a hóspedes com idade superior a 60 anos. Codifique o método `public void reservaQuartoSenior(...)`, da classe `Hotel`, que deverá validar se o quarto é compatível com ocupantes séniores, isto é, se é um `QuartoDuploSenior` e se os ocupantes são séniores. Escreva também a implementação que teria de fazer para o método `public double valorAPagar()`, considerando que então o preço por dia deverá ter um desconto igual a 25% da soma das idades dos seus ocupantes.

☐0 ☐.2 ☐.4 ☐.5 ☐.6 ☐.8 ☐1 *Reservado aos docentes*