Exercício do Hotel

Objetivo

O objetivo do sistema é gerenciar um hotel. Andares e quartos serão adicionados e hóspedes poderão adquirir diárias nestes quartos. Também teremos uma recepção que ficará responsável por faturar dos nossos hóspedes.

No sistema teremos:

- **Hotel**: suas informações consistirá em um nome e uma lista de andares;
- **Andar**: suas informações consistirá em um número (do andar) e um mapa de quartos, na qual a chave será o número do quarto;
- **Quarto**: terá nome. uma breve descrição do que o quarto oferece, um hóspede caso esteja ocupado e o valor de sua diária quando aplicável;
- **Hospede**: será identificado pelo nome e RG.

Detalhamento do código-fonte

➔ Organizar em pacotes:

- **br.com.cast.exceptions** : *para as exceções;*
- **br.com.cast.interfaces** : *para as interfaces;*
- **br.com.cast.main** : *para a classe Principal (com o método main);*
- **br.com.cast.models.abstracts** : *para as classes abstratas;*
- **br.com.cast.models** : *para as classes concretas.*

➔ Classe concreta **Hospede** *(preencher atributos através do construtor e sobrescrever o toString)*

- **Nome** : *texto privado;*
- **RG** : *texto privado;*

➔ Superclasse abstrata **Quarto** *(preencher atributos através do construtor (exceto o **Hospede**) e sobrescrever o toString)*

- **Nome** : *texto protegido;*
- **Descricao** : *texto protegido;*
- **Hospede** : *Hospede protegido;*
- **adicionarHospede(Hospede h)** : *público, não retorna nada e pode lançar uma exceção do tipo QuartoOcupadoException;*
- **removerHospede()** : *público, não retorna nada e pode lançar uma exceção do tipo QuartoVazioException;*
- **isOcupado()** : *público, retorna **true** caso o quarto esteja ocupado;*
- **isVazio()** : *público, retorna **true** caso o quarto esteja vazio;*

➔ Interface **Faturavel**

- **faturar(int qtdDeDiarias)** : *publico e retorna o valor da fatura (valorDiaria * qtdDeDiarias) do hóspede;*

➔ Subclasse concreta **QuartoCortesia** *(quarto grátis para convidados do **Hotel**, preencher atributos fixos através do construtor da superclasse)*

➔ Subclasse concreta **QuartoLuxo** *(quarto faturável, preencher atributos fixos através do construtor da superclasse)*

- **VALOR_DIARIA = 450.0** : *número, privado, estático e constante;*

➔ Subclasse concreta **QuartoSimples** *(quarto faturável, preencher atributos fixos através do construtor da superclasse)*

- **VALOR_DIARIA = 87.90** : *número, privado, estático e constante;*

➔ Classe concreta **Andar** *(preencher/instanciar atributos através do construtor e sobrescrever o toString)*

- **Andar** : *número privado;*
- **mapaDeQuartos** : *Mapa de quartos, privado, na qual a chave é número do quarto e o valor é o próprio quarto;*
- **adicionarQuarto(Integer numero, Quarto quarto)** : *público, não retorna nada, deve adicionar um quarto no mapa e pode lançar uma exceção do tipo NumeroDoQuartoRepetidoException;*
- **entregarChaveDoQuarto(Integer numero)** : *público, retorna o quarto solicitado e pode lançar uma exceção do tipo QuartoNaoEncontradoException (caso o número do quarto não existir) e também uma exceção do tipo QuartoOcupadoException (caso o quarto esteja ocupado);*

➔ Classe concreta **Hotel** *(preencher/instanciar atributos através do construtor e sobrescrever o toString)*

- **Nome** : *texto privado;*
- **Andares** : *lista de Andar privado;*
- **adicionarAndar(Andar a)** : *público, não retorna nada e deve adicionar um andar na lista;*
- **imprimeDados()** : *público, não retorna nada e deve imprimir **this** no console;*

➔ Classe concreta **Recepcao** (alterar o construtor padrão para privado)

- **pagar(Faturavel f, int qtdDeDiarias)** : *público, estático, não retorna nada e deve imprimir no console o valor a receber para o Hospede.*

➔ Classe concreta **Principal** e método **main** *(para testar nosso sistema)*

- Criar um objeto do tipo Hotel;

- Criar três objetos do tipo Andar;
- Criar oito objetos do tipo Quarto, sendo:
    - Três objetos para QuartoSimples;
    - Três objetos para QuartoLuxo;
    - Dois objetos para QuartoCortesia;
- Criar cinco objetos do tipo Hospede;
- Adicionar dois hóspedes em um quarto simples cada;
- Adicionar dois hóspedes em um quarto de luxo cada;
- Adicionar o último hóspede em um quarto já ocupado acima;
- Adicionar os dois quartos simples no primeiro andar;
- Adicionar um quarto simples e dois quartos cortesias no segundo andar;
- Adicionar os três quartos de luxo no terceiro andar;
- Adicionar os três andares no hotel;
- Invocar o método imprimeDados do hotel;
- Invocar o método pagar da Recepção para:
    - Um quarto simples com 8 diárias;
    - Um quarto de luxo com 4 diárias.