ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia de Computação e Sistemas Digitais

PCS 3111 - Laboratório de Programação Orientada a Objetos para Engenharia Elétrica

2024

Aula 03 – Conceitos Básicos de OO

Atenção

- As definições das classes usadas nos exercícios encontram-se **disponíveis no e-Disciplinas**. Use o código fornecido.
- Os nomes, os atributos, os métodos, e as respectivas assinaturas das classes dadas **devem seguir o especificado** em cada exercício para fins de correção automática.
- A **ordem de declaração** de atributos e métodos fornecidos **não deve ser alterada**. Caso contrário, poderá haver redução automática da nota.
- A função `main` **não deve ser submetida**. Caso contrário, a correção automática retornará um *Compilation Error.*

Exercício 01
Usando a definição fornecida e a especificação a seguir, implemente a classe `Quarto`:

```
class Quarto {
public:
    int numeroDoQuarto = 0;
    int numeroDePessoas = 0;
    int numeroDeCamas = 0;
    double getPrecoDiaria();
    void imprimir();
};
```

- Um `Quarto` possui, como atributos, o número de pessoas, o número do quarto e o número de camas.
- O número de pessoas será usado para o cálculo do preço da diária, como será explicado a seguir.
- O método `getPrecoDiaria()` deve retornar o valor da diária de acordo com o número de pessoas. O preço da diária por pessoa é de 100 reais. Se o número de pessoas for 2, a segunda pessoa tem 50% de desconto na sua diária. Caso contrário, o preço base por pessoa é cobrado.

Exemplos:

Quarto com 3 pessoas:
getPrecoDiaria() deve retornar o valor 300.0 (em double).
Quarto com 2 pessoas:
getPrecoDiaria() deve retornar o valor 150.0 (em double).

O método `imprimir` deve exibir na tela (usando o `cout`) os dados de cada `Quarto` no seguinte formato:

```
Quarto <numeroDoQuarto>: <numeroDePessoas> pessoas,
<numeroDeCamas> camas - Diaria custa <valor da diaria>
```

- Sendo <...> o valor armazenado em cada variável. Por exemplo, para um quarto de número 25, com 4 pessoas e 2 camas, a saída seria (pule uma linha ao final):

```
Quarto 25: 4 pessoas, 2 camas - Diaria custa 400
```

- Implemente a função **teste1** declarada no código fornecido seguindo os próximos passos:
  - Crie o quarto de número 31.
  - Guarde o número de pessoas igual a **3** e o número de camas igual a **2**.
  - Chame o método **imprimir** para esse quarto.
  - Crie o quarto de número 32.
  - Guarde o número de pessoas igual a 2 e o número de camas igual a 1;
  - Chame o método **imprimir** para esse quarto.

  Com esses parâmetros, a saída deve ser:

```
Quarto 31: 3 pessoas, 2 camas - Diaria custa 300
Quarto 32: 2 pessoas, 1 camas - Diaria custa 150
```

- Teste a classe antes de enviar, utilizando a main!
- Não se esqueça de comentar a main ao submeter.

Exercício 02
Usando a definição fornecida e a especificação a seguir, implemente a classe `Reserva`:

```
class Reserva {
public:
    Quarto *quarto1 = nullptr;
    Quarto *quarto2 = nullptr;
    int inicio = 0;
    int fim = 0;

    bool adicionarQuarto(Quarto *q);
    double calcularPreco();
    void imprimir();
};
```

- Uma `Reserva` é constituída por, no máximo, dois quartos. Os atributos `inicio` e `fim` representam as datas de início e fim da reserva, que devem ser utilizadas para calcular o número de diárias (n° de diárias = fim - inicio).
- O método `calcularPreco()` retorna o valor total da reserva. Considere o número de diárias e o preço da diária de cada quarto (utilize o método getPrecoDiaria). Caso não existam quartos na reserva, este método deve retornar 0.
- O método `adicionarQuarto` armazena um quarto em `quarto1` ou `quarto2` (nesta ordem) e retorna `true` caso tenha adicionado. Confira se o quarto é nullptr para saber em qual das variáveis deve-se armazenar o produto (se `quarto1 == nullptr`, é em `quarto1`; senão se `quarto2 == nullptr`, é em `quarto2`; senão a reserva está cheia). Caso o quarto já tenha sido adicionado ou não seja possível adicionar mais quartos (já existem dois quartos na `Reserva`), deve-se retornar `false` e não adicione.
    - Ao testar seu programa, **não** atribua quartos diretamente pelos atributos `quarto1` e `quarto2`. Use sempre o método `adicionarQuarto`.
- O método `imprimir` deve exibir na tela (usando o `cout`) os dados da reserva no seguinte formato (pulando uma linha no final):

```
Reserva:
<Chame o imprimir do Quarto 1, se ele existir>
<Chame o imprimir do Quarto 2, se ele existir>
Preco total: <Preco Total>
```

- Implemente a função **`teste2()`** declarada no código fornecido seguindo os próximos passos:
    - Copie o que foi feito em **`teste1()`**
    - Crie uma reserva com inicio = 2 e fim = 13
    - Adicione, seguindo a ordem de criação, os dois quartos à reserva
    - Chame o método imprimir para Reserva.

A saída para o **`teste2`** deve ser:

```
Quarto 31: 3 pessoas, 2 camas - Diaria custa 300
Quarto 32: 2 pessoas, 1 camas - Diaria custa 150
Reserva:
Quarto 31: 3 pessoas, 2 camas - Diaria custa 300
Quarto 32: 2 pessoas, 1 camas - Diaria custa 150
Preco Total: 4950
```

- Teste as classes antes de enviar, utilizando a `main`!
- Não se esqueça de comentar o `main` ao submeter no *judge*.

Testes do Judge

Exercício 1

- Preço da diária com desconto (duas pessoas).
- Preço da diária sem desconto.
- Teste função teste1

Exercício 2

- Adicionar um quarto com reserva vazia (sem desconto).
- Adicionar um quarto com reserva vazia (com desconto).
- Adicionar mais de um quarto com reserva vazia.
- Adicionar o mesmo quarto duas vezes.
- Adicionar mais de dois quartos.
- Teste função teste2