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

## Aula 4 – Encapsulamento

**Atenção**

1. Código inicial a ser usado na resolução dos exercícios encontra-se **disponível no e-Disciplinas**.
2. Ao enviar para o Judge mantenha o `#define NUMERO_MAXIMO_VALORES 10` com o valor 10, por motivos de correção.

Utilize o código fornecido no e-Disciplinas para implementar as classes **Pessoa** e **Atividade,** aplicando os conceitos de encapsulamento vistos em aula.

**Exercício 1**
No arquivo main.cpp é fornecida a definição e a implementação da classe `Pessoa`. Separe-a em dois arquivos, "**Pessoa.h**" e "**Pessoa.cpp**". O arquivo .h deve conter apenas a definição. O arquivo .cpp deve conter apenas a implementação. Use adequadamente as diretivas de compilação.

Defina a visibilidade dos atributos e dos métodos de modo que os atributos sejam acessíveis apenas no escopo da classe e os métodos sejam acessíveis externamente.

**Exercício 2**
De acordo com a definição a seguir, implemente a classe `Atividade`. Note que essa classe utiliza a classe `Pessoa`, do exercício 1. Novamente, é necessário separar a classe em dois arquivos, "**Atividade.h**" e "**Atividade.cpp**", usando adequadamente as diretivas de compilação. Defina corretamente a **visibilidade** de seus métodos e atributos.

```
#define NUMERO_MAXIMO_VALORES 10

class Atividade {
  public:
  void setNome(string nome);
  void setHorasNecessarias(int horasNecessarias);
  int getHorasNecessarias();
  int getQuantidade();

  bool adicionar(Pessoa* p);
  int getDuracao();
  void imprimir();
};
```

A implementação da classe Atividade deve atender aos seguintes requisitos:

- O método `getQuantidade` retorna o número de pessoas que foram adicionadas ao vetor **pessoas**.
- O método `adicionar` deve adicionar um objeto do tipo `Pessoa` ao vetor **pessoas,** se possível. Caso o vetor já esteja completamente preenchido ou a pessoa não esteja disponível (o método `isDisponível()` informa isso), o método não modifica o vetor e retorna **false**. Caso seja bem sucedido, deve retornar **true,** além de mudar a disponibilidade da pessoa para **false** (evitando que outra atividade use a pessoa). Utilize a constante `NUMERO_MAXIMO_VALORES` como o número máximo de pessoas que a atividade comporta.
- O método `getDuracao` deve retornar a duração em dias que a atividade levará para ser finalizada, seguindo a fórmula:

$$\boldsymbol{duracao} = \left\lceil \frac{\textbf{horas necessárias para terminar a atividade}}{\sum \boldsymbol{horas\ por\ dia\ de\ cada\ pessoa\ adicionada}} \right\rceil$$

  Caso não existam pessoas adicionadas à tarefa, esse método deve retornar -1.

  **Dica:** Use a função `ceil` para calcular o teto (arredondamento para cima). Para isso faça include da biblioteca `cmath` e faça "`using namespace std;`".
- O método `imprimir` deve mostrar na tela as informações de cada atividade, seguindo o formato:

  ```
  <nome> - <duração> dias estimados
  ```

  Além de imprimir as informações de todas as pessoas que foram adicionadas à tarefa. Por exemplo:

  ```
  EP1 - 3 dias estimados
  Fulano: indisponível - 2 hora(s) por dia
  Ciclano: indisponível - 1 hora(s) por dia
  ```

  **Dica:** a classe `Pessoa` já possui um método `imprimir` – utilize-o para facilitar a implementação deste método!
  **Observação:** esse método não será corrigido pelo Judge. Utilize-o para testar a sua classe!

É recomendado criar um `main` que utilize as classes `Pessoa` e `Atividade`, a fim de testar e de validar as implementações.

Lembre-se de utilizar a diretiva *#ifndef* nos arquivos de cabeçalhos ("**.h**") para evitar problemas de conflitos de definição de classes causados por múltiplas inclusões de um cabeçalho.

Dicas importantes
1. Os nomes, os atributos, os métodos, e as respectivas assinaturas das classes dadas **devem seguir o especificado** para fins de correção automática.

2. A função main não deve ser submetida. Caso contrário, a correção automática retornará um *Compilation Error.*

**Testes do Judge**

Exercício 1

- Utilização dos *setters* e dos *getters*;
- `imprimir` com pessoa disponível;
- `imprimir` com pessoa indisponível;

Exercício 2

- Criação de um objeto `Atividade` e utilização dos *setters* e dos *getters*;
- `adicionar` com vetor vazio;
- `adicionar` com vetor parcialmente preenchido;
- `adicionar` com vetor cheio;
- `getDuracao` com vetor vazio;
- `getDuracao` com vetor parcialmente preenchido;
- `getDuracao` com vetor cheio;