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

2020

Aula 06 – Herança e Polimorfismo I

Atenção

Código inicial a ser usado na resolução dos exercícios encontra-se **disponível no e-Disciplinas**.

Exercício 01

Considere a classe **Equipe**. Construa agora a classe **EquipeComCapitao**, sendo que nela, o construtor da classe recebe o nome do capitão em **nomeCapitao**, além dos parâmetros já existentes no construtor de **Equipe**. Siga as seguintes instruções:

```
/* Inclua os arquivos necessarios
 * A classe eh filha de Equipe
 */
class EquipeComCapitao {
   public:

       EquipeComCapitao(string nome, int membros, string nomeCapitao);
       ~EquipeComCapitao();

       string getNomeCapitao();
} ;
```

- Considere que o capitão já está incluído na quantidade de `membros` enviado no construtor.
- No destrutor da **EquipeComCapitao**, escreva a seguinte mensagem:

  Equipe com Capitao `<nomeCapitao>` destruida

  Ex: Se o nome do capitão for Marcos, a mensagem será:

  Equipe com Capitao Marcos destruida

- Altere a visibilidade dos atributos necessários da classe **Equipe** (private ou protected).
- Implemente o método `getNomeCapitao()`, que retorna o nome do capitão.

Observação: na aula que vem veremos o que significa a palavra reservada *virtual* no destrutor de Equipe. Por enquanto não se preocupe com isso.

Exercício 02

Implemente a classe **Competicao**. Ela contém um vetor de ponteiros do tipo **Equipe**, alocado dinamicamente. Adicione os atributos necessários para o funcionamento da classe e implemente seus métodos.). **Inclua os arquivos ".h" e ".cpp" referentes à classe Competicao ao projeto do Code::Blocks** (clique com o botão direito no projeto e selecione "Add files..").

```
class Competicao {
  public:
    Competicao(string nome, int maximoEquipes);
    ~Competicao();

    Equipe** getEquipes();
    int getQuantidade();
    bool adicionar(Equipe* e);
    void imprimir();
  private:
    /** Adicionar atributos necessarios **/
};
```

- Declare um vetor como atributo, mas apenas o crie no construtor, com tamanho maximoEquipes.
- O método **getEquipes** retorna o vetor de equipes adicionadas.
- O método **getQuantidade** retorna a quantidade de equipes adicionadas (use um atributo adicional para guardar essa informação).
- O método **adicionar** não deve adicionar a mesma equipe mais de uma vez, nem adicionar uma quantidade acima de **maximoEquipes**. Se for possível adicioná-la, retorna **true**; Caso contrário, retorna **false**.
    - Veja se a equipe já foi adicionada ao usar o "==" para comparar se as variáveis apontam para o mesmo objeto na memória.
- O método **imprimir** não será avaliado. Use-o para testar a Competição.
    - No destrutor destrua apenas o vetor de equipes alocado dinamicamente. **Não destrua as equipes**.

Atenção. Veja o princípio da substituição funcionando: crie uma `main` para testar a criação de um projeto que armazene objetos do tipo Equipe e EquipeComCapitao, além de testar seus métodos.

Testes do Judge

Exercício 1

- EquipeComCapitao é classe filha de Equipe;
- Destrutor: Envia a mensagem
- Métodos: getNome e getMembros
- Método: getNomeCapitao

Exercício 2

- Competicao com objetos Equipe: getters
- Competicao com objetos EquipeComCapitao: getters;
- Projeto com objetos Equipe e EquipeComCapitao: getters;
- Adicionar equipes iguais;
- Adicionar com vetor cheio.