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 07 – Herança e Polimorfismo II

Atenção

1. Código inicial a ser usado na resolução dos exercícios encontra-se **disponível no e-Disciplinas**.
2. 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.

Exercício 1

Considere as classes **Equipe**, **Modalidade** e **Competicao**, implementadas nas aulas anteriores.

Na classe **Competicao**:

- Além de armazenar equipes, a **Competicao** deve armazenar **Modalidades**. Com isso, implemente o método `bool adicionar(Modalidade* m)`, o qual possibilita a adição de uma **Modalidade** a **Competicao**. Crie no construtor um vetor alocado dinamicamente para armazenar as **Modalidades**, o qual deve ter tamanho máximo `maximoValor`. Se já houver `maximoValor` de **Modalidade** ou na hipótese de a **Modalidade** ter sido previamente adicionada (ou seja, o mesmo objeto), o método deve retornar false. Outra condição é a de somente adicionar uma **Modalidade** em que todas as equipes dela façam parte da **Competição**. Por exemplo, considere a competição InterUSP na qual participam as equipes Poli, FEA e EACH. Caso a modalidade "Atletismo" tenha as equipes Poli e FFLCH, ela **não** poderá ser adicionada (já que FFLCH não faz parte da competição InterUSP); caso a modalidade "Natacao" tenha as equipes FEA e Poli, ela pode ser adicionada.

  Caso seja possível adicionar a **Modalidade** à **Competicao**, o método deve retornar true.
- Implemente também os métodos `int getQuantidadeDeModalidades()` e `Modalidade** getModalidades()` que devem retornar a quantidade de modalidades já adicionadas ao vetor e o vetor alocado dinamicamente que contém as modalidades da **Competicao**, respectivamente.

Note que o método `adicionar` foi ***sobrecarregado***. Os métodos públicos dessa classe são apresentados a seguir.

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

    int getQuantidadeDeEquipes();
    int getQuantidadeDeModalidades();
    Equipe** getEquipes();
    Modalidade** getModalidades();

    bool adicionar(Equipe* e);
    bool adicionar(Modalidade* m);

    void imprimir();
};
```

Implemente apropriadamente os métodos conforme especificado e adicione atributos conforme necessário. **NÃO** altere as classes **Equipe** e **Modalidade** fornecidas.

Exercício 2
Implemente a classe **CompeticaoImprimivel**, apresentada a seguir.

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

    void imprimir();
};
```

Redefina o método `imprimir` em `CompeticaoImprimivel` de forma que ele imprima o nome da competição (como é feito em **Competicao**) e também o nome todas as equipes participantes. Faça um ***refinamento***.

Altere o arquivo `Competicao.h` para que o método mais especializado seja sempre utilizado, independente do tipo da variável. Mas não modifique a implementação dos métodos em `Competicao.cpp`.

- O método **imprimir** (redefinido) deve escrever a seguinte mensagem:

Competicao <nome>
Equipe <nomeEquipe1>
Equipe <nomeEquipe2>
...
Equipe <nomeEquipeN>

Exemplo: Se tivermos uma competição "InterUSP", com equipes "Poli'', "FEA" e "IFUSP", teremos:

Competicao InterUSP:
Equipe Poli
Equipe FEA
Equipe IFUSP

Utilize \t para fazer a indentação do nome das equipes na impressão. Utilize o método `imprimir` da classe **Equipe**.

Exercício 3

Considere a classe **Historico** apresentada a seguir:

```
class Historico {
public:
    Historico(int maximoValor);
    ~Historico();

    int getQuantidade();
    bool adicionar(Competicao* c);
    Competicao** getCompeticoes();
    Competicao** getCompeticoesImprimiveis(int& tamanho);
private:
    Competicao** competicoes;
    int quantidade;
    int maximoValor;
};
```

Essa classe é um histórico que guarda diversas competições que já ocorreram. O método `getQuantidade` retorna a quantidade de competições (normais ou imprimíveis) que foram adicionadas ao vetor alocado dinamicamente pelo método `adicionar` (ou seja, objetos que sejam da classe **Competicao** ou **CompeticaoImprimivel**). Já o método `getCompeticoes` retorna o vetor alocado dinamicamente `competicoes`.

Implemente o método:

```
Competicao** getCompeticoesImprimiveis(int& tamanho);
```

Esse método deve retornar um vetor alocado dinamicamente contendo todas as **CompeticaoImprimivel** do vetor `competicoes` de um **Historico**. Além disso, deve-se retornar a quantidade de **CompeticaoImprimivel** através do parâmetro `tamanho`, passado por referência. Caso o **Historico** não possua **CompeticaoImprimivel**, deve-se retornar NULL e o `tamanho` deve ser 0.

Dicas:

- Para facilitar, aloque um vetor com tamanho `quantidade`.
- Não se esqueça de inicializar `tamanho` com 0.

Testes do Judge

Exercício 1

- Competicao: adicionar Modalidade com Equipes não participantes da Modalidade no início do vetor
- Competicao: adicionar Modalidade com Equipes não participantes da Modalidade no meio do vetor
- Competicao: adicionar Modalidade com Equipes não participantes da Modalidade no fim do vetor
- Competicao: adicionar Modalidade com Equipes participantes da Modalidade
- Competicao: adicionar Modalidade sem espaço no vetor

Exercício 2

- CompeticaoImprimivel: imprimir
- CompeticaoImprimivel: imprimir via uma variável Competicao

Exercício 3

- getCompeticoesImprimiveis sem CompeticaoImprimivel
- getCompeticoesImprimiveis com uma CompeticaoImprimivel
- getCompeticoesImprimiveis com mais de uma CompeticaoImprimivel