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 06 – Herança e Polimorfismo I

Atenção

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

Exercício 01
Considere as classes **Pessoa** e **Atividade**. Observe que o construtor de um objeto do tipo **Atividade** recebe como parâmetros uma *string* "**nome**" e dois *int*: "**horasNecessarias**" e "**maximoPessoas**".

Uma **AtividadeComSupervisor** é uma classe filha de **Atividade**. Nessa classe, além de possuir pessoas designadas a realizá-la, ela possui uma Pessoa que a supervisiona. Implemente a classe **AtividadeComSupervisor,** dada nos arquivos fornecidos. Fique atento às alterações necessárias para realizar a herança!

```
/* Inclua os arquivos necessários
 * A classe é filha de Atividade
 */

class AtividadeComSupervisor {
  public:
    AtividadeComSupervisor(string nome, int horasNecessarias,
                           int maximoPessoas, Pessoa* supervisor);
    ~AtividadeComSupervisor();

    Pessoa* getSupervisor();
};
```

- O supervisor não deve ser adicionado ao vetor pessoas.
- Por causa da atividade de supervisão, adiciona-se uma hora necessária para a atividade, além de se adicionar uma hora a mais para cada quatro horas de **horasNecessarias**;
  Por exemplo, se uma **AtividadeComSupervisor** for inicializada com 5 **horasNecessarias**, o método **getHorasNecessarias** deve retornar 7. Se for inicializada com 2 horas, deve retornar 3.
- Altere a visibilidade dos atributos necessários da classe **Atividade** (private ou protected).

Cuidado: não submeta os arquivos `Projeto.cpp` e `Projeto.h` neste exercício.

Exercício 02
Implemente a classe **Projeto** (importe-a ao seu projeto do Code::Blocks). Ela contém um vetor de ponteiros do tipo **Atividade**, alocado dinamicamente. Adicione os atributos necessários para o funcionamento da classe e implemente seus métodos.

```
class Projeto {
  public:
    Projeto(string nome, int maximoAtividades);
    ~Projeto();

    int getDuracao();
    int getQuantidade();
    bool adicionar(Atividade* a);

  private:
    /** Adicionar atributos necessários **/
};
```

- O método **getDuracao** retorna o somatório das durações de cada Atividade adicionada;
- O método **getQuantidade** retorna a quantidade de atividades adicionadas;
- O método **adicionar** não deve adicionar a mesma atividade mais de uma vez, nem adicionar uma quantidade acima de **maximoAtividades**. Se for possível adicioná-la, retorna **true**; Caso contrário, retorna **false**.
    - Compare atividades usando ==. Ou seja, não adicione uma atividade caso o objeto já esteja no vetor.

Dica: repare na necessidade do uso do princípio da substituição de Liskov!

Atenção: é recomendado que se crie uma `main` para testar a criação de um projeto que armazene objetos do tipo **Atividade** e **AtividadeComSupervisor**, além de testar seus métodos.

Testes do Judge
Exercício 1
- AtividadeComSupervisor é classe filha de Atividade;
- getDuracao com menos de 4 horas;
- getDuracao com mais de 4 horas;
- getSupervisor correto.

Exercício 2
- Projeto com objetos Atividade: getters
- Projeto com objetos AtividadeComSupervisor: getters;
- Projeto com objetos Atividades e AtividadeComSupervisor: getters;
- Adicionar atividades iguais;
- Adicionar com vetor cheio.