ELE078 - Programação Orientada a Objetos

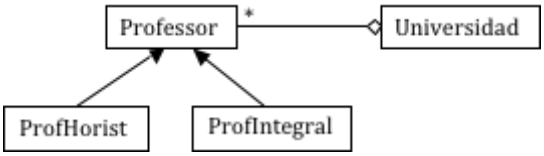
Atividade Prática 09 - Polimorfismo, Funções Virtuais e Classes Abstratas

Exercício 1:

Complemente o código a seguir, para criar uma hierarquia de classes de professores de uma Universidade de acordo com a descrição abaixo:

> Os professores ou são horistas ou são de tempo integral, não existe outro tipo de professor. As únicas informações necessárias no sistema são o nome do professor e os dados para cálculo de seu salário. O salário mensal de um professor horista é o valor da sua hora de trabalho vezes o número de horas trabalhadas por ele no mês. O salário mensal de um professor de tempo integral é um valor fixo por mês.

O diagrama de classes é mostrado abaixo.



In [ ]:

```
class Professor{
        string nome;
    public:
        Professor(string n);
        string getName() const;
        virtual double getSalario() = 0;
        virtual ~Professor();
};

class ProfHorista: public Professor {
        double nrNorasTrabalhadas;
        double valorHora;
    public:
        ProfHorista(string n, double nht, double vh);
        double getSalario();
        virtual ~ProfHorista();
};

class ProfIntegral: public Professor {
        double salarioMensal;
    public:
       ProfIntegral(string n, double sm)
       double getSalario();
       virtual ~ProfIntegral(){}
};
```

Exercício 1: Continuação

A universidade mantém uma lista de todos os seus professores (*vector*) e necessita de operações para adicionar professores nessa lista e para retornar o valor total pago mensalmente para todos os seus professores. Mostre o código da classe Universidade fazendo um código genérico utilizando o conceito de Polimorfismo.

Sugestão de nomes de métodos da classe **Universidade** :

- addProfessor(...): método para adicionar professor à lista de professores;
- totalPago(...): método para retornar o valor total pago mensalmente para todos os professores da Universidade;

Uma forma de verificar se você usou corretamente o conceito de **Polimorfismo** é fazendo o seguinte teste: se um novo tipo de professor for adicionando ao sistema com uma forma diferente de calcular salário, por exemplo professor de tempo parcial, o código da classe Universidade não deve sofrer nenhuma alteração. Faça este teste apenas mentalmente, você não tem que implementá-lo!

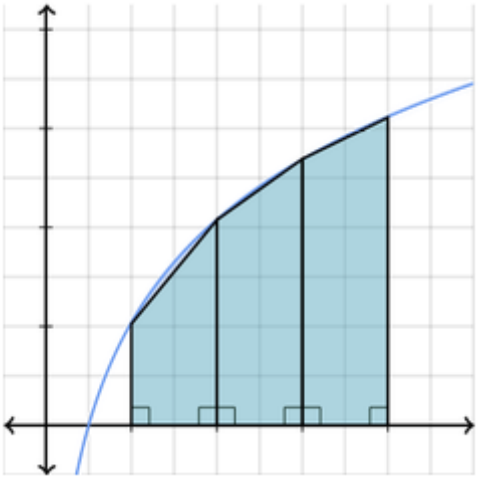
Escreva um código para testar a classe Universidade. O código deve inserir três professores na lista, sendo dois professores horistas e um professor de tempo integral. Uma vez inseridos na lista, deve-se listar (imprimir na tela) as informações desses professores e, logo em seguida, imprimir o valor total mensal que é pago para todos os professores da Universidade.

Exercício 2:

Crie uma método *getIntegral()* que é capaz de obter a integral de uma função $y = f(x)$ via regra do trapézio. Essa função deve ser polimórfica podendo então mudar seu comportamento dependendo se o objeto apontado é do tipo função quadrática, senoide, linear, entre outros. Para relembrar, o método do Trapézio (ou soma Trapezoidal) é um técnica de integração numérica que busca aproximar o valor da integral a partir do somatório da área dos trapézios que dividem a área total da função.

Como mostrado na Figura a seguir, a integral da função:

$$\int_a^b 3ln(x)\,dx$$



Essa divisão pode ser arbitrária e quanto maior, mais próximo será do valor real da integral. A implementação desse cálculo de integral deve considerar no mínimo três tipos de funções:

- **Quadratica**: $ax^2 + bx + c$
- **Senoidal**: $\frac{sin(x)}{x}$
- **Linear**: $ax + b$

In [ ]:

```
// codigo exemplo

#include <iostream>
#include <cmath>
using namespace std;

class Funcao
{
    public:

    //funcao que obtem a integral da funcao pela regra do trapezio
    double getIntegral(double limiteInferior, double limiteSuperior, double inte
rvalos);

    // funcao virtual representando a funcao cuja integral deve ser calculada
    virtual double func(double input) = 0;

    // destrutor
    //virtual ~Funcao(){}

};

...

int main()
{
    double resultado;

    //cria um container de ponteiros do tipo Funcao
    Funcao *f[3];

    f[0] = new Quadratica(1,2,4);
    f[1] = new Senoide();
    f[2] = new Linear(1,4);

    cout << "*** Calculo de integrais usando a regra do trapezio: ***"<<endl<<en
dl;
    cout << "*** Funcoes ***" << endl;
    cout << "(1) x^2 + 2x + 4" << endl;
    cout << "(2) sen(x) / x" << endl;
    cout << "(3) x + 4" << endl;
    cout << endl;

    for (int i=0; i<3; i++)
    {
        resultado = f[i]->getIntegral(1,4,1000);
        cout << "Integral da Funcao (" << i+1 << "): " << resultado;
        cout << endl;
    }

    for (int i=0; i<3; i++)
    {
        delete f[i];
    }

    return 0;
}
```