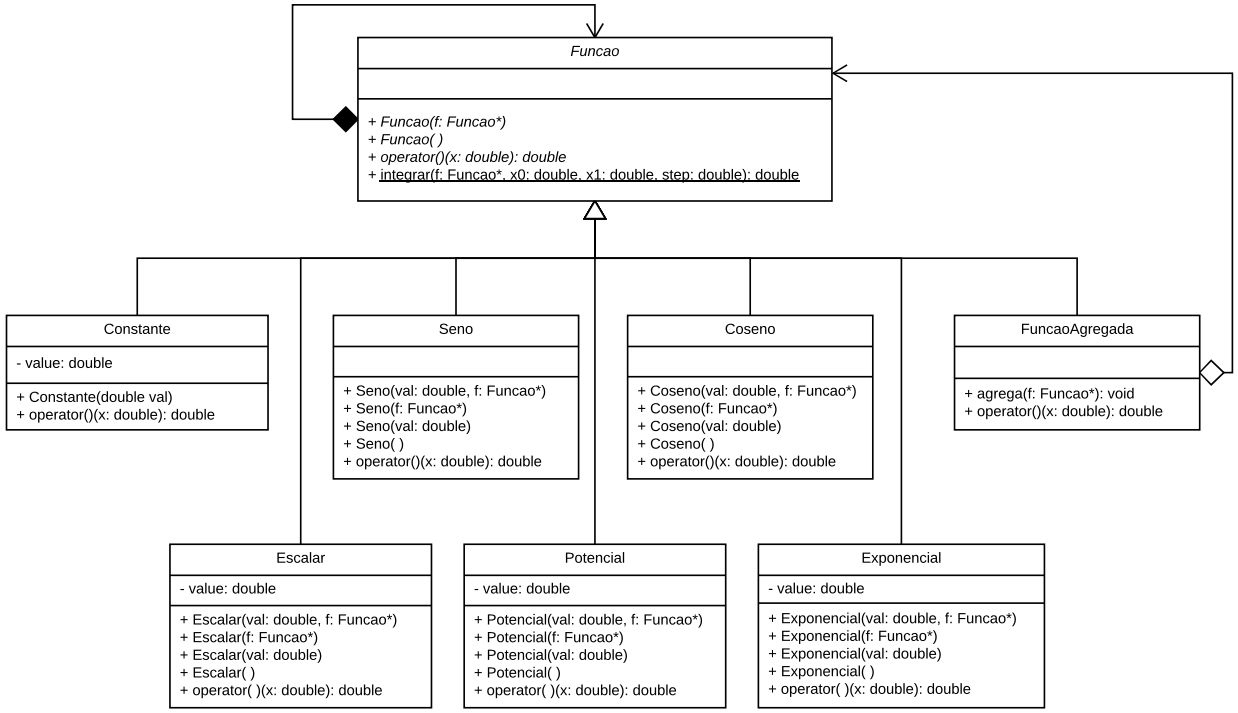
Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina
Departamento Acadêmico de Eletrônica
Curso de Engenharia Eletrônica
PRG22107 – Programação Orientada a Objetos

Avaliação 02

Nome: _____ Data: _____

A integral de uma função para um determinado intervalo é a soma de sua área naquele intervalo. Ela pode ser calculada numericamente a partir de aproximações de retângulos. O diagrama UML apresentado abaixo modela um conjunto de classes que implementam funções fundamentais, que quando compostas, calculam os valores de funções mais complexas.



1. Implemente o conjunto de classes apresentado no diagrama UML, de acordo com a modelagem apresentada. Considere as observações abaixo:

   - Implemente todas as classes em um único arquivo .h nomeado de acordo com o modelo <prenome>_p2.h
   - A classe ***Funcao*** retorna o valor da função através do operador de chamada de função, tendo como parâmetro o valor de x. Este operador deve ser polimórfico e redefinido nas especializações da classe ***Funcao***
   - A classe ***Funcao*** deve ser **abstrata**

- A classe ***Funcao*** pode ter uma sub função permitindo a criação de funções compostas ($f(g(x))$). Para isso, a sub função deve ser passada para o construtor da classe.
- A classe ***Funcao*** possui um método estático (não membro) chamado integrar que deve realizar a integração da função através do algoritmo de aproximações de retângulos. Seus parâmetros são o ponteiro para a função a ser integrada (f), o intervalo no domínio da função (x0 e x1) e o intervalo de discretização da aproximação (step)
- A classe ***Constante*** deve implementar uma função constante (ex: $f(x) = 10$). O seu valor constante é inicializado pelo construtor do objeto.
- A classe ***Escalar*** deve implementar uma função do tipo $f(x) = ax$. O valor de a é inicializado pelo construtor do objeto. A classe também deve prover construtores para inicializar subfunções da mesma, conforme o diagrama UML.
- A classe ***Potencial*** deve implementar uma função do tipo $f(x) = x^a$. O valor de a é inicializado pelo construtor do objeto. A classe também deve prover construtores para inicializar subfunções da mesma, conforme o diagrama UML.
- A classe ***Exponencial*** deve implementar uma função do tipo $f(x) = a^x$. O valor de a é inicializado pelo construtor do objeto. A classe também deve prover construtores para inicializar subfunções da mesma, conforme o diagrama UML.
- A classe ***Seno*** deve implementar uma função do tipo $f(x) = sen(x)$. O valor de a é inicializado pelo construtor do objeto. A classe também deve prover construtores para inicializar subfunções da mesma, conforme o diagrama UML.
- A classe ***Coseno*** deve implementar uma função do tipo $f(x) = cos(x)$. O valor de a é inicializado pelo construtor do objeto. A classe também deve prover construtores para inicializar subfunções da mesma, conforme o diagrama UML.
- A classe ***FuncaoAgregada*** deve implementar uma agregação (somas) de funções (ex. $f(x) = g(x) + h(x)$). O número de funções agregadas é variável e indefinido (utilize uma estrutura de dados adequada). As funções agregadas devem ser adicionadas pelo método agrega, conforme apresentado no diagrama UML.

2. Escreva um programa para testar as classes implementadas. A rotina de testes deve ser implementada como uma função no seu arquivo .h, com o protótipo ***void teste()***. A função main em seu arquivo de compilação deve apenas chamar esta função teste(). A sua rotina de teste deve apresentar o a integração da função $f(x) = x^2 + 5sen(2x - 1) + 5$ no intervalo [0,5] e intervalo de discretização de 0,01. Apresente também os valores da f(x) nos pontos da discretização da função.

   Abaixo é apresentado um exemplo para a função $f(x) = 3x^2 + 2x + 5$:

```
    Escalar g(3, new Potencial(2)); // g(x) = 3x^2;
    Escalar h(2);                   // h(x) = 2x;
    Constante i(5);                 // i(x) = 5;
    FuncaoAgregada f;               // f(x) = g(x) + h(x) + i(x)
    f.agrega(&g); f.agrega(&h); f.agrega(&i);

    double x = 0;
    while (x < 5){
        cout << "f(" << x << ") = " << f(x) << endl;
        x += 0.01;
    }
    cout << "A integral de [0,5]: ";
    cout << Funcao::integrar(&f,0,5,0.01) << endl;
```