INSTITUTO FEDERAL
Paraíba

BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO
PROGRAMAÇÃO ORIENTADA A OBJETOS

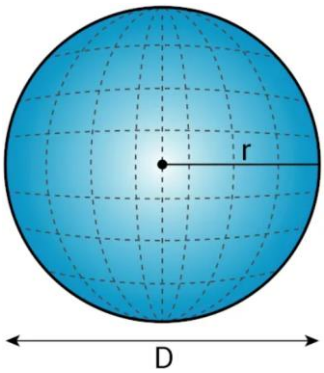
Turma: Terceiro Período (P3) **Semestre**: 2022.1 **Campus**: Cajazeiras - PB
Professor: Leandro Luttiane da S. Linhares
Nome: _____ **Data**: ___/___/_____

Lista de Exercícios 02:
Criação de Classes - Introdução

1. Desenvolva uma classe em linguagem C++ denominada `Ponto` que abstrai um ponto de coordenadas `x` e `y` no plano cartesiano. Sabendo disso, defina adequadamente os atributos dos objetos desta classe e inclua os seguintes métodos:

- Métodos acessores (`get` e `set`) necessários para obter os valores e atribuir valores aos atributos privados definidos para os objetos da classe `Ponto`;
- Método que exibe em tela o estado do objeto da classe `Ponto`;
- Método que retorna em qual quadrante o objeto da classe `Ponto` se encontra no plano cartesiano. O método deve retornar:
  - `0`: caso o ponto esteja localizado na origem do plano cartesiano;
  - `1`: caso o ponto esteja situado no primeiro quadrante do plano cartesiano;
  - `2`: se o ponto estiver no segundo quadrante do plano cartesiano;
  - `3`: se o ponto estiver no terceiro quadrante do plano cartesiano;
  - `4`: se o ponto está localizado no quarto quadrante do plano cartesiano
  - `5`: caso o ponto esteja posicionado no eixo das abscissas, mas não na origem;
  - `6`: caso o ponto esteja posicionado no eixo das ordeandas, mas não na origem;
- Método que retorna a distância do objeto da classe `Ponto` à origem do plano cartesiano (`x = 0 e y = 0`). **Observação**. Use a equação de cálculo de distância entre dois pontos.
- Método que retorna a distância do objeto da classe `Ponto` a um outro objeto `Ponto`. Este método deve possuir como parâmetro um objeto da classe `Ponto`.

2. Implemente uma classe em linguagem C++ com o objetivo de representar uma `Esfera`. Os objetos desta classe devem possuir o atributo privado `raio` em centímetros.



Sabendo disso, crie os seguintes métodos da classe:

- Os métodos acessores necessários para obter o valor deste atributo e definir valores para o mesmo (métodos `get` e `set`, respectivamente). O método `get` deve impedir que o atributo assuma valores inválidos;
- Método que calcula e retorna o volume da esfera em centímetros cúbicos;
- Método que calcula a área da esfera em centímetros quadrados;
- Método que recebe como argumento um determinado valor de altura e calcula o volume de preenchimento da esfera até esta altura (volume do segmento esférico de uma base) em centímetros cúbicos. O valor da altura informada deve ser validada antes do cálculo requisitado pelo método. Por exemplo, se o raio da esfera for de 10 cm, a altura informada deve estar compreendida entre 0 cm e 20 cm. Caso o valor de altura seja inválido, uma mensagem de erro deve ser exibida e o método deve retornar um valor negativo como -1;
- Método que imprime em tela o estado do objeto da classe `Esfera`, o seu volume e sua área.

Para relembrar como efetuar os cálculos solicitados de volume e área de uma `Esfera`, os seguintes links podem ser consultados: link1 e link2.

3. Elabore uma classe em linguagem C++ para representar um `Relogio` digital. O relógio deve ter informações de `horas`, `minutos` e `segundos`. Além disso, deve conter os seguintes métodos:

- Métodos acessores necessários para obter e definir os valores dos atributos da classe. Garanta que não seja possível atribuir valores inválidos para os atributos dos objetos da classe `Relogio`;
- Método que exibe em tela o horário do objeto da classe no seguinte formato: `hh:mm:ss`;
- Método `incrementaSegundos`, que incrementa os `segundos` do relógio em uma unidade. O incremento efetuado por este método não deve afetar nos valores de `horas` e `minutos` do `Relogio`;
- Método `incrementaMinutos`, que incrementa os `minutos` do relógio em uma unidade. O incremento efetuado por este método não deve afetar nos valores de `horas` e `segundos` do `Relogio`;
- Método `incrementaHoras`, que incrementa as `horas` do relógio em uma unidade. O incremento efetuado por este método não deve afetar nos valores de `minutos` e `segundos` do `Relogio`;
- Método `incrementaHorario`, que incrementa o horário do relógio em um segundo (simula o funcionamento de um relógio). O incremento realizado por este método pode ocasionar alteração nos valores de `minutos` e `horas` do `Relogio`.

4. Crie uma classe denominada `Turma` que possui uma determinada quantidade de alunos. Para a representação dos alunos, a classe `Aluno` desenvolvida em sala de aula pode ser utilizada. A classe `Turma` deve possuir as seguintes características e funcionalidades:

- Atributo `codigo`, que representa o código da turma. Este atributo deve ser um número inteiro positivo. Para este atributo devem existir os respectivos métodos acessores;
- Atributo `alunos`, que corresponde a um vetor alocado dinamicamente de objetos da classe `Aluno`. O método `get` para este atributo deve retornar uma cópia de um determinado objeto `Aluno` do vetor de `Alunos`. Para isso, deve ser informado o índice desejado do vetor. O método `set` deve atribuir valores aos atributos de um determinado objeto `Aluno` que se encontra em um determinado índice do vetor de `Alunos`;

- Atributo `nAlunos`, que indica quantos alunos há na `Turma`. Deve ser criado apenas o método `get` para este atributo;
- Método **construtor** da classe que recebe como argumento o número de alunos da turma. Este método deve alocar dinamicamente o vetor de `Alunos` e definir o valor de `nAlunos`;
- Método que imprime em tela o código da `Turma` e os `nomes`, `notas` e `media` de cada um dos `Alunos` da turma;
- Método `mediaGeral`, que calcula a média das médias dos `Alunos` da `Turma`;
- Método **destrutor** que desaloca a memória alocada dinamicamente pelo objeto `Turma`.