Lista de Exercícios

Tema 06. Sobrecarga e encadeamento de métodos em POO.

Recomendações:

- Utilizando a linguagem de programação da sua preferência (Java, Python ou ambas) implemente as classes especificadas nos exercícios a seguir.
- Utilize apropriadamente os conceitos associados a sobrecarga de métodos.
- Utilize apropriadamente as recomendações em relação ao princípio DRY e encadeamento de métodos discutidas em sala de aula.
- Crie em cada caso uma classe cliente, ou aplicação simples onde mostre o uso da classe implementada.

1. Usando como base a classe `Time` implementada na aula anterior inclua os seguintes requisitos na classe:
    - Objetos `Time` podem ser construídos para cada uma das seguintes condições:
        - Construtor sem parâmetros neste caso, teremos o horário 00:00:00
        - Construtor com um parâmetro indicando a hora HH, neste caso teremos o horário HH:00:00
        - Construtor com dois parâmetros indicando a hora HH, e os minutos MM respetivamente neste caso teremos o horário HH:MM:00
        - Construtor com três parâmetros HH, MM, SS
        - Construtor que recebe como parâmetro um objeto Time, e retorna um objeto idêntico ao parâmetro
    - Converta o método `void addSeconds(int secs)` desenvolvido anteriormente no método `addTime` com as seguintes assinaturas:
    - `addTime(int s)`, adiciona s segundos ao tempo do objeto
    - `addTime(int m, int s)`, adiciona m minutos e s segundos ao tempo do objeto
    - `addTime(int h, int m, int s)`, adiciona h horas, m minutos e s segundos ao tempo do objeto
    - `addTime(Time obj)`, adiciona o tempo obj ao objeto corrente

2. Usando como base classe Data implementada nas listas anteriores adicione os seguintes requisitos:
    - Inclua construtores da classe para os seguintes contextos
        - Um construtor com três parâmetros dia, mês e ano
        - Um construtor sem parâmetros neste caso, teremos como data o 1/1/1970
        - Um construtor que recebe como parâmetro um objeto Data criado anteriormente
    - Reescreva o método int howManyDays(Data outraData) agora ele deverá funcionar com duas assinaturas
        - int howManyDays(Data outraData), já implementada anteriormente
        - int howManyDays(int day, int month, int year), retornando a diferença em dias com relação à data representada pelos três parâmetros.
    - Reescreva o método setData() para atender as assinaturas e funcionalidades propostas no slide 13.

3. Usando como base a classe Contato desenvolvida nas listas anteriores, análise quais construtores podem ser permitidos no contexto de uma agenda, implemente esses construtores em sua classe.

4. Refatore a classe Ponto2D desenvolvida nas listas anteriores para atender os seguintes requisitos:
    - Inclua construtores da classe para os seguintes contextos
        - Um construtor com dois parâmetros x, y
        - Um construtor sem parâmetros, neste caso o ponto (0,0) deve ser construído
    - Modifique o método distancia para ele atender as seguintes assinaturas:
        - float distance(Ponto2D outroPonto), retorna a distância entre o ponto parâmetro e o objeto
        - float distance(), calcula a distancia em relação à origem de coordenadas
        - float distance(float x, float y), retorna a distância em relação ao ponto representado pelo par ordenado (x, y)

5. Realize modificações na classe círculo para atender os seguintes requisitos:

- Círculos podem ser construídos para cada uma das seguintes condições:
  - A partir de três parâmetros, x, y, e r, estando o círculo centralizado em (x, y) com raio r
  - Sem fornecer parâmetros neste caso o círculo estará localizado na origem de coordenadas e terão raio 1
  - A partir de dois parâmetros x, y, círculo centralizado em (x, y) com raio 1
  - A partir de um parâmetro r, círculo estará localizado na origem de coordenadas e terá radio r

6. Partindo da classe RoboSimples mostrada na sala de aulas, adicione as seguintes funcionalidades

   Métodos:

   validaRobo, o nome de robô não pode ser nulo, e a direção do robô deve estar restrita aos 4 valores possíveis N, S, E, O

   move, o método deve suportar adicionalmente movimentos na diagonal, isto é, nas direções NE, NO, SE, SO. Medite sobre qual a melhor alternativa para armazenar as direções possíveis do robô, e o tipo de parâmetro a ser utilizado.

   Gerenciador de colisões, isto é um robô não pode se movimentar a uma posição final já ocupada por outro robô. Quais as alternativas para implementar esta função? É necessário criarmos alguma nova classe?