CST em Análise e Desenvolvimento de Sistemas
Programação Orientada a Objetos

Prof. Sergio Prolo
sergio.prolo@ifsc.edu.br

Exercícios 2: Introdução à Programação Orientada a Objetos

5 de setembro de 2025

Informações iniciais

- Cada tarefa dessa lista deve ser implementada em **uma** (1) classe Java.
- Em cada classe, adicione um método principal `main(String[] args)`. Use esse método para testar as funcionalidades implementadas na classe. **Apenas** esse método pode interagir com entradas (`Scanner`) e saídas (`prints`) de dados.
- É aconselhado usar um projeto Gradle para gerenciar essa lista de exercícios. A classe principal da aplicação (App.java) deve apenas invocar o método principal de cada classe implementada.
- Todas as classes devem obedecer os princípios de encapsulamento (como bom uso dos modificadores `private` e `public`) e abstração (como comportamentos úteis em métodos e características importantes representadas por atributos).
- Quando aplicável, todas as classes devem ter métodos de acesso (`getter`/`setter`) para seus atributos, um método de comparação semântica (`equals`) e um método de representação (`toString()`).

1. Um contador é um aparelho usado para contar items ou eventos, comumente utilizado em competições esportivas (voltas, pontos, etc..) ou em estudos de tráfego (número de veículos em congestionamento ou interseções). Implemente uma classe Java para modelar um `Contador` que encapsule o comportamento do aparelho. Além dos atributos por você identificados, a classe deve oferecer métodos para:

   - Zerar a contagem;
   - Incrementar o contador;
   - Retornar o valor atual.

   Ao testar sua implementação:

   - Crie um objeto contador e conte até 10;
   - Crie um segundo objeto contador e conte até ele ter o valor do primeiro contador;
   - Zere ambos os contadores.

2. Implemente uma classe Java para modelar um `Relogio`. O objeto deve representar as horas, minutos e segundos de um dia e ser representado no formato 'HH:MM:SS'. Além dos atributos por você identificados, a classe deve oferecer métodos para:

   - Incrementar a hora;
   - Incrementar o minuto;
   - Incrementar o segundo;
   - Ajustar a hora, o minuto e o segundo para valores específicos;
   - Retornar a quantidade total de segundos desde o começo do dia;
   - Retorne a diferença, em segundos, entre o `Relogio` que recebeu a mensagem e outro `Relogio` qualquer;
   - Sincronizar o `Relogio` que recebeu a mensagem com outro `Relogio` qualquer.

   Ao testar sua implementação:

   - Crie um relógio e ajuste o seu valor para 14:58:32;
   - Incremente o valor do relógio em dois minutos. Que horas o relógio marca agora?
   - Crie um segundo relógio e ajuste seu valor para 23:59:59;

- Incremente o valor do relógio em um segundo. Que horas o relógio marca agora?
- Sincronize o segundo relógio com o primeiro. Que horas o relógio marca agora?

3. Na geometria, um ponto do plano cartesiano é uma noção primitiva que determina uma posição no espaço representado como (x, y). Implemente uma classe Java para modelar o `Ponto2D`, representando um ponto do plano cartesiano. Além dos atributos por você identificados, a classe deve prover os seguintes métodos:

   - Método para calcular a distância entre o `Ponto2D` que recebeu a mensagem e outro `Ponto2D` qualquer;
   - Construtores sobrecarregados que permitam a inicialização de um objeto `Ponto2D`:
     - Por padrão (sem argumentos), na origem do plano 2D;
     - Em um local indicado por dois argumentos do tipo `double` (os valores da abcissa e ordenada do ponto);
     - No mesmo lugar de outro `Ponto2D`.

   Ao testar sua implementação:

   - Crie dois pontos, um na origem do plano e outro em $(3, 4)$;
   - Mostre que a distância entre os pontos é 5;
   - Mostre que esses pontos são diferentes.

4. Na matemática, um número complexo $n \in \mathbb{C}$ é aquele representado como $(a + bi)$, onde a e b são números reais que representam suas partes real e imaginária, e i é a constante complexa ($i^2 = -1$). Implemente uma classe Java para modelar um `NumeroComplexo`. Além dos atributos por você identificados, a classe deve prover os seguintes métodos:

   - Método somar, que recebe outro `NumeroComplexo` e o adiciona ao `NumeroComplexo` que recebeu a mensagem;
   $$(a + bi) + (c + di) = (a + c) + (b + d)i$$
   - Método subtrair, que recebe outro `NumeroComplexo` e o subtrai do `NumeroComplexo` que recebeu a mensagem;
   $$(a + bi) - (c + di) = (a - c) + (b - d)i$$
   - Método multiplicar, que recebe outro `NumeroComplexo` e o multiplica ao `NumeroComplexo` que recebeu a mensagem;
   $$(a + bi) * (c + di) = (ac - bd) + (ad + bc)i$$
   - Método dividir, que recebe outro `NumeroComplexo` e divide o `NumeroComplexo` que recebeu a mensagem pelo outro;
   $$\frac{(a + bi)}{(c + di)} = \frac{(ac + bd)}{(c^2 + d^2)} + \frac{(bc - ad)i}{(c^2 + d^2)}$$
   - Método que retorna o conjugado do `NumeroComplexo`;
   $$\overline{(a + bi)} = (a - bi)$$
   - Método que retorna o módulo do `NumeroComplexo`;
   $$|(a + bi)| = \sqrt{a^2 + b^2}$$

   Ao estar sua implementação:

   - Crie dois números complexos, $n = (5 + 4i)$ e $m = (7 - 3i)$;
   - Adicione m a n e mostre que o resultado é $(12 + i)$;
   - Divida n por m e mostre que o resultado é $(0.39 + 0.74i)$
   - Mostre que o módulo de n é 6.4

5. Na geometria, uma reta é uma linha infinita composta por todos os pontos (x, y) que obedecem a equação $y = ax + b$, onde a e b representam o coeficiente angular e linear da reta, respectivamente. Implemente uma classe Java para modelar uma `Reta`. Além dos atributos por você identificados, a classe deve prover os seguintes métodos:

- Método que recebe um `double` representando uma abcissa (x) e retorna a ordenada (y) do ponto que faz parte da reta na abcissa informada;
- Método que verifica se um `Ponto2D` (da questão 3) pertence ou não à reta;
- Método que recebe outra `Reta` e retorna o `Ponto2D` de interseção das retas ou `null` se as retas são paralelas.
- Construtores sobrecarregados que permitam a inicialização de um objeto `Reta` a partir de:
  - Dois valores `double`, representando os coeficientes angular e linear;
  - Dois `Ponto2D`.

Ao testar sua implementação:

- Prove que o ponto $(-6, 0)$ faz parte da reta $y = 0.5x + 3$
- Prove que o ponto $(2.8, 4.4)$ é a interseção das retas $y_1 = 0.5x + 3$ e $y_2 = 10 - 2x$
- Prove que os pontos $(0, 3)$ e $(1, 5)$ formam a reta $y = 2x + 3$

6. Um círculo é uma figura geométrica composta por todos os pontos (x, y) que estão a uma distância menor que $r \in \mathbb{R}$, chamada raio, de um ponto (x_0, y_0), chamado centro. Escreva em Java uma classe que represente um `Circulo` no plano cartesiano. Além dos atributos por você identificados, a classe deve prover os seguintes métodos:
   - Método que retorna a área do círculo
   - Método que recebe um `Ponto2D` (da questão 3) e retorna se o ponto pertence ou não ao *Circulo*;
   - Construtores sobrecarregados que permitam a inicialização de um objeto `Circulo` a partir de:
     - Um `Ponto2D` que representa seu centro e um valor `double` que representa seu raio;
     - Um `Ponto2D` que representa seu centro, considerando que o raio é unitário;
     - Um valor `double` que representa seu raio, considerando que o centro é na origem do plano;
     - Sem argumentos, considerando que o centro é na origem do plano cartesiano e o raio é unitário.
   - Métodos sobrecarregados inflar e desinflar, que, respectivamente, aumentam e diminuem o raio do círculo em:
     - um valor positivo qualquer parametrizado;
     - exatamente uma unidade, quando sem parâmetro.
   - Métodos sobrecarregados para mover, que:
     - por padrão (sem parâmetros) levam o círculo para a origem;
     - movem o círculo adicionando dois parâmetros do tipo `double` ao seu centro (translado);
     - movem o círculo para o local indicado por um `Ponto2D`.
   - Método **estático** que retorne o número total de círculos criados;
7. Em um sistema de registro, um país é representado através dos atributos: código ISO 3166-1 (ex.: BRA), nome (ex.: Brasil), população (ex.: 213.421.037) e a sua dimensão em Km^2 (ex.: 8.515.767,049). Além disso, cada país mantém um arranjo com os códigos dos outros países com os quais ele faz fronteira. Considere que um país tem no máximo 15 outros países com os quais ele faz fronteira. Implemente uma classe Java que represente um `Pais`. Além dos atributos, a classe deve prover os seguintes métodos:
   - Construtor que inicialize um objeto `Pais` com código ISO, nome, população e dimensão;
   - Método que adiciona o código de outro `Pais` à lista de fronteiras;
   - Método que recebe outro `Pais` e retorna se eles fazem fronteira ou não;
   - Método que retorna a densidade populacional do `Pais`;
   - Método que recebe outro `Pais` e retorna um arranjo com os códigos de todos os vizinhos comuns aos dois países;
   - Na comparação semântica, considere que dois países são iguais se seus códigos ISO são iguais;

8. Uma loja requer um sistema para registro dos seus produtos. Cada produto possui um nome, um preço, uma taxa de desconto e um código identificador no formato `COD-XXX-XXX`. O código deve ser único, não pode ser alterado e deve ser gerado automaticamente no momento de criação do objeto, contando os produtos criados a partir de `COD-000-000` (por exemplo, o terceiro produto terá código `COD-000-003`, o milésimo `COD-001-000`, e assim por diante). Implemente uma classe Java que represente um `Produto` para esse sistema. Além dos atributos por você identificados, a classe deve prover os seguintes métodos:

   - Construtor que inicialize objetos `Produto` com nome e preço definidos, sempre com taxa de desconto em 0%;
   - Método para retornar o preço real (sem desconto) e aplicado (com desconto) do `Produto`;
   - Método **estático** que retorne um arranjo com todos os `Produtos` já criados; Considere que esse registro possui, no máximo, 50 produtos. Quando um novo produto é criado, o mais antigo é removido para dar espaço ao mais novo;
   - Método **estático** para expor o registro de produtos em formato `.csv`
     - Esse método **não** deve utilizar System.out ou acesso à arquivos
     - Ele deve retornar um **vetor de Strings** para a aplicação principal imprimir
     - A primeira linha deve ser um cabeçalho com os nomes das colunas do `.csv`
     - As próximas linhas devem conter os atributos dos produtos registrados
     - Assuma que o `.csv` é separado por ‘ ; ’ (para não confundir com o decimal dos números)
     - Abaixo está um exemplo do conjunto de Strings gerada por esse método

```
Código;Nome;Preço;Desconto
CD:000-001;Geladeira;782,08;6
CD:000-002;Micro-ondas;439,12;12
CD:000-003;Fogão 4 bocas Eletrolux KL4003;677,00;0
```

   - Na comparação semântica, considere que dois produtos são iguais se seus nomes e preços são iguais;