

## TÉCNICAS DE PROGRAMAÇÃO II

### EXERCÍCIOS LISTA 02: **CLASSES**

Profª Lucília Ribeiro

**01)** Quais dos identificadores abaixo podem ser usados como nomes de classes, atributos, métodos e variáveis? Quais não podem, e por quê?

A. four  
B. for  
C. from  
D. 4  
E. FOR

A. dia&noite  
B. diaENoite  
C. dia & noite  
D. dia E noite  
E. dia\_e\_noite

A. contador  
B. 1contador  
C. contador de linhas  
D. Contador  
E. count

**02)**

Considerando os tipos de dados existentes na Linguagem, escolha o tipo de dado ou classe mais adequada para representar:

- \_ O número de municípios de um estado do Brasil.
- \_ O nome de um estado do Brasil.
- \_ A população de um estado do Brasil.
- \_ A área do Brasil em quilômetros quadrados.
- \_ A população total do mundo.
- \_ O CEP de um endereço no Brasil.
- \_ O nome de uma rua em um endereço no Brasil.
- \_ A altura de uma pessoa em metros.
- \_ O peso de uma pessoa em quilos.
- \_ A temperatura corporal de uma pessoa.
- \_ O sexo de uma pessoa.
- \_ A altura de uma pessoa em milímetros.

**03)** Responda verdadeiro ou falso para cada uma das afirmações abaixo, explicando ou justificando a sua resposta.

- A. Um valor do tipo boolean pode receber o valor numérico zero.
- B. Um valor do tipo float pode armazenar valores maiores do que os que podem ser armazenados por um valor do tipo long.
- C. Podemos ter caracteres cujos valores sejam negativos.
- D. O número de bytes ocupados por uma variável do tipo float depende do computador e do sistema operacional sendo usado.
- E. O tipo char pode ser usado para representar pares de caracteres, uma vez que variáveis desse tipo ocupam dois bytes na memória.
- F. Os tipos de dados double e long não são equivalentes, apesar de variáveis desses tipos ocuparem o mesmo espaço na memória.

**04)** Escreva a classe `Lampada` correspondente ao modelo da listagem abaixo. Que tipo de dado pode ser usado para representar o atributo estado?

```
1 modelo Lampada // representa uma lâmpada em uso
2 início do modelo
3   dado estadoDaLampada; // indica se está ligada ou não
4
5   operação acende() // acende a lâmpada
6   início
7     estadoDaLampada = aceso;
8   fim
9
10  operação apaga() // apaga a lâmpada
11  início
12    estadoDaLampada = apagado;
13  fim
14
15  operação mostraEstado() // mostra o estado da lâmpada
16  início
17    se (estadoDaLampada == aceso)
18      imprime "A lâmpada está acesa";
19    senão
20      imprime "A lâmpada está apagada";
21  fim
22
23 fim do modelo
```

05) Inclua, no modelo acima, uma operação `estahLigada` que retorne verdadeiro se a lâmpada estiver ligada e falso, caso contrário.

06) Modifique a resposta do exercício anterior para que a classe represente também o número de watts da lâmpada. Escreva um método `Economica` que retorne o valor booleano `true` se a lâmpada consumir menos de 40 watts e `false` caso contrário.

07) Identifique e explique o(s) erro(s) nas classes a seguir:

```
1 class Registro De Eleitor
2 {
3     /**
4      * Declaração dos atributos desta classe
5      */
6     int títuloDeEleitor; // número do título do eleitor
7     String nome; // nome do eleitor
8     short zonaEleitoral; // número da zona eleitoral
9 } // fim da classe
```

```
1 class DoisValores
2 {
3     /**
4      * Declaração dos atributos desta classe
5      */
6     int valor1, valor2;
7     /**
8      * Declaração dos métodos desta classe
9      */
10    int maior()
11    {
12        if (valor1 > valor2)
13            return true;
14        else return false;
15    }
16    void menor()
17    {
18        if (valor1 < valor2)
19            return valor1;
20        else return valor2;
21    }
22 } // fim da classe
```

```
1 class NumeroComplexo
2 {
3     /**
4      * Declaração dos atributos desta classe
5      */
6     float real, imaginário;
7     /**
8      * Declaração dos métodos desta classe
9      */
10    float valor()
11    {
12        return real, imaginário;
13    }
14 } // fim da classe
```

```

1 class FaceDoDado
2 {
3     /**
4      * Declaração dos atributos desta classe
5      */
6     int 1,2,3,4,5,6;
7     /**
8      * Declaração dos métodos desta classe
9      */
10    void 1()
11    {
12        System.out.println(1);
13    }
14    void 2()
15    {
16        System.out.println(2);
17    }
18    void 3()
19    {
20        System.out.println(3);
21    }
22    void 4()
23    {
24        System.out.println(4);
25    }
26    void 5()
27    {
28        System.out.println(5);
29    }
30    void 6()
31    {
32        System.out.println(6);
33    }
34 } // fim da classe

```

```

1 class Registro De Eleitor
2 {
3     /**
4      * Declaração dos atributos desta classe
5      */
6     int títuloDeEleitor; // número do título do eleitor
7     String nome; // nome do eleitor
8     short zonaEleitoral; // número da zona eleitoral
9 } // fim da classe

```

```

1 class Amplitude
2 {
3     /**
4      * Declaração dos atributos desta classe
5      */
6     double val1,val2,val3;
7     /**
8      * Declaração dos métodos desta classe
9      */
10    double amplitude()
11    {
12        double amplitude2()
13        {
14            return val1-val2;
15        }
16        return amplitude2()-val3;
17    }
18 } // fim da classe

```

**08)** Baseado no modelo Data abaixo, crie a classe HoraAproximada que represente uma hora qualquer (usando valores para representar horas e minutos)

```

1 modelo Data
2 início do modelo
3     dado dia,mês,ano; // componentes da data
4
5     // Inicializa simultaneamente todos os dados do modelo
6     operação inicializaData(umDia,umMês,umAno) // argumentos para esta operação
7     início
8         // Somente muda os valores do dia, mês e ano se a data passada for válida
9         se dataÉVálida(umDia,umMês,umAno) // Repassa os argumentos para a operação
10            início
11                dia = umDia;
12                mês = umMês;
13                ano = umAno;
14            fim
15            // Se a data passada não for válida, considera os valores sendo zero
16            senão
17                início
18                    dia = 0;
19                    mês = 0;
20                    ano = 0;
21                fim
22            fim
23
24     operação dataÉVálida(umDia,umMês,umAno) // argumentos para esta operação
25     início
26         // Se a data passada for válida, retorna verdadeiro
27         se ((dia >= 1) e (dia <= 31) e (mês >= 1) e (mês <= 12))
28             retorna verdadeiro;
29
30         // Senão, retorna falso
31         senão
32             retorna falso;
33         fim
34     operação mostraData() // mostra a data
35     início
36         imprime dia;
37         imprime "/";
38         imprime mês;
39         imprime "/";
40         imprime ano;
41     fim
42 fim do modelo
43

```

**09)** Baseado no modelo Data e considerando a classe criada no exercício anterior, crie a classe `HoraPrecisa` que represente uma hora qualquer (usando valores para representar horas, minutos, segundos e centésimos de segundos).

**10)** Escreva uma classe `CadernoDeEnderecos` que represente os dados de uma pessoa, como nome, telefone, e-mail e endereço. Que atributos e métodos essa classe deve ter?

**11)** Crie uma classe `Ponto2D` para representar um ponto no espaço cartesiano de duas dimensões. Que dados e operações esse modelo deve ter? Dica: Imagine um gráfico no qual você tenha que desenhar pontos, baseados nesse modelo.

**12)** Crie uma classe `Livro` que represente os dados básicos de um livro, sem se preocupar com a sua finalidade.

**13)** Usando a classe do exercício anterior como base, crie uma classe `LivroDeLivraria` que represente os dados básicos de um livro que está à venda em uma livraria.

**14)** Usando a classe do exercício anterior como base, crie uma classe `LivroDeBiblioteca` que represente os dados básicos de um livro de uma biblioteca, que pode ser emprestado a leitores.

**15)** Escreva a classe `Contador` que encapsule um valor usado para contagem de itens ou eventos. Essa classe deve esconder o valor encapsulado de programadores-usuários, fazendo com que o acesso ao valor seja feito somente através de métodos que devem zerar, incrementar e imprimir o valor do contador

**16)** Escreva, na classe Data, um método `duplicaData` que receba como argumento uma outra instância da classe Data, e duplique os valores dos campos da instância passada como argumento para os atributos encapsulados.

17) Escreva a classe `PoligonoRegular`, que represente um polígono de até dez lados. Descreva uma operação que retorne o nome do polígono baseado no seu número de lados.

18) Crie uma classe `Linha2D` para representar uma linha, unida por dois pontos no espaço cartesiano de duas dimensões, usando duas instâncias da classe `Ponto2D`, criada no exercício 11.

19) Crie uma classe `Retangulo` para representar um retângulo cujos pontos opostos sejam duas instâncias da classe `Ponto2D`.

20) Modifique a classe `Lampada` para que esta contenha também um campo que indique quantas vezes a lâmpada foi acesa. Tente usar uma instância da classe `Contador` (exercício 15). Em que método esse atributo deve ser modificado, e como?

21) Escreva uma classe `ContaBancariaSimplificada` que corresponda ao modelo na listagem a seguir. Considere que modificadores de acesso devam ser usados para os métodos e campos da classe.

```
1 modelo ContaBancariaSimplificada
2 início do modelo
3     dado nomeDoCorrentista,saldo,contaÉEspecial; // dados da conta
4
5     // Inicializa simultaneamente todos os dados do modelo
6     operação abreConta(nome,depósito,especial) // argumentos para esta operação
7     início
8         // Usa os argumentos passados para inicializar os dados do modelo
9         nomeDoCorrentista = nome;
10
11         saldo = depósito;
12         contaÉEspecial = especial;
13     fim
14
15     // Inicializa simultaneamente todos os dados do modelo, usando o nome
16     // passado como argumento e os outros valores com valores default
17     operação abreContaSimples(nome) // argumento para esta operação
18     início
19         nomeDoCorrentista = nome;
20         saldo = 0.00;
21         contaÉEspecial = falso;
22     fim
23
24     // Deposita um valor na conta
25     operação deposita(valor)
26     início
27         saldo = saldo + valor;
28     fim
29
30     // Retira um valor da conta
31     operação retira(valor)
32     início
33         se (contaÉEspecial == falso) // A conta não é especial !
34         início
35             se (valor <= saldo) // se existe saldo suficiente...
36                 saldo = saldo - valor; // faz a retirada.
37             fim
38         senão // A conta é especial, pode retirar à vontade !
39             saldo = saldo - valor;
40         fim
41     fim
42
43     operação mostraDados() // mostra os dados da conta, imprimindo os seus valores
44     início
45         imprime "O nome do correntista é ";
46         imprime nomeDoCorrentista;
47         imprime "O saldo é ";
48         imprime saldo;
49         se (contaÉEspecial) imprime "A conta é especial.";
50         senão imprime "A conta é comum.";
51     fim
52 fim do modelo
```

22) A operação `inicializaData` do modelo `Data` tem uma abordagem simplista demais para verificar se o dia sendo usado é válido ou não: nessa operação ainda seria possível passar a data 31/02/2023 e a operação iria considerar os valores passados como sendo válidos. Modifique a operação `dataEhValida` para que esta considere o valor máximo que pode ser aceito como válido, dependendo do mês, de forma que, para meses com 30 dias, o valor 31 para o dia seja considerado incorreto, e que para fevereiro o valor máximo seja calculado em função de o ano ser bissexto ou não. Dica: Anos bissextos (tendo 29 dias em fevereiro) são divisíveis por quatro, a não ser

que sejam divisíveis por 100. Anos que podem ser divididos por 400 também são bissextos. Dessa forma, 1964 e 2000 são bissextos, mas 1900 não é bissexto. A operação de divisibilidade pode ser implementada pela função módulo, representada pelo sinal %, e comparada com zero: a expressão  $(1966 \% 4) == 0$  é verdadeira, enquanto a expressão  $(1967 \% 4) == 0$  é falsa

**23)** Uma das operações que podemos efetuar com datas é a comparação para ver se uma data ocorre antes de outra. O algoritmo para comparação é muito simples, e seus passos estão abaixo. Nesse algoritmo, consideramos que dia1, mês1 e ano1 são os dados da primeira data, e que dia2, mês2 e ano2 são os dados da segunda data.

1. Se  $\text{ano1} < \text{ano2}$  a primeira data vem antes da segunda.
2. Se  $\text{ano1} > \text{ano2}$  a primeira data vem depois da segunda.
3. Se  $\text{ano1} == \text{ano2}$  e  $\text{mês1} < \text{mês2}$  a primeira data vem antes da segunda.
4. Se  $\text{ano1} == \text{ano2}$  e  $\text{mês1} > \text{mês2}$  a primeira data vem depois da segunda.
5. Se  $\text{ano1} == \text{ano2}$  e  $\text{mês1} == \text{mês2}$  e  $\text{dia1} < \text{dia2}$  a primeira data vem antes da segunda.
6. Se  $\text{ano1} == \text{ano2}$  e  $\text{mês1} == \text{mês2}$  e  $\text{dia1} > \text{dia2}$  a primeira data vem depois da segunda.
7. Se nenhum desses casos ocorrer, as datas são exatamente iguais.

Escreva um método `vemAntes` na classe `Data` que receba como argumento outra instância da classe `Data` e implemente o algoritmo acima, retornando `true` se a data encapsulada vier antes da passada como argumento e `false` caso contrário. Se as datas forem exatamente iguais, o método deve retornar `true`.

**24)** Se os métodos `abreConta`, `deposita` e `retira` forem criados conforme o exercício anterior sugere, alguns erros poderão ocorrer, como abrir uma conta com valor negativo, ou depositar ou retirar valores negativos. Modifique os métodos citados para que somente valores positivos sejam considerados pelos métodos.

**25)** Escreva uma classe `RestauranteCaseiro` que implemente o modelo descrito abaixo. Para isso, crie também uma classe `MesaDeRestaurante` que represente uma mesa de restaurante conforme mostrado. Algumas sugestões sobre a criação dessas classes são:

\_ A classe `MesaDeRestaurante` deve ter atributos para representar a quantidade de cada pedido feito, um método `adicionaAoPedido` que incrementa a quantidade de pedidos feitos, o método `zeraPedidos` que cancela todos os pedidos feitos, isto é, faz com que a quantidade de pedidos seja zero para cada item, e o método `calculaTotal`, que calcula o total a ser pago por aquela mesa. Como modelar cada item da comanda separadamente?

\_ A classe `RestauranteCaseiro` deve ter vários atributos que são instâncias da classe `MesaDeRestaurante`, para representar suas mesas separadamente.

\_ A classe `RestauranteCaseiro` também deve ter um método `adicionaAoPedido` que adicionará uma quantidade a um item de uma mesa. Esse método deverá chamar o método `adicionaAoPedido` da mesa à qual o pedido está sendo adicionado.

A solução deste exercício requer a criação de um número predeterminado e imutável de instâncias de `MesaDeRestaurante` em `RestauranteCaseiro`. Comente sobre as vantagens e desvantagens de criar classes desta forma.

Restaurante Caseiro Hipotético		
<b>Mesa 1</b> <input type="text"/> kg refeição <input type="text"/> sobremesa <input type="text"/> refriger.2 L. <input type="text"/> refriger.600mL. <input type="text"/> refriger.lata <input type="text"/> cerveja	<b>Mesa 2</b> <input type="text"/> kg refeição <input type="text"/> sobremesa <input type="text"/> refriger.2 L. <input type="text"/> refriger.600mL. <input type="text"/> refriger.lata <input type="text"/> cerveja	<b>Mesa 3</b> <input type="text"/> kg refeição <input type="text"/> sobremesa <input type="text"/> refriger.2 L. <input type="text"/> refriger.600mL. <input type="text"/> refriger.lata <input type="text"/> cerveja
<b>Mesa 4</b> <input type="text"/> kg refeição <input type="text"/> sobremesa <input type="text"/> refriger.2 L. <input type="text"/> refriger.600mL. <input type="text"/> refriger.lata <input type="text"/> cerveja	<b>Mesa 5</b> <input type="text"/> kg refeição <input type="text"/> sobremesa <input type="text"/> refriger.2 L. <input type="text"/> refriger.600mL. <input type="text"/> refriger.lata <input type="text"/> cerveja	<b>Mesa 6</b> <input type="text"/> kg refeição <input type="text"/> sobremesa <input type="text"/> refriger.2 L. <input type="text"/> refriger.600mL. <input type="text"/> refriger.lata <input type="text"/> cerveja