CEFET-MG

Programação de Computadores II 12/07/2022

Prova 5

Professor: Gustavo Henrique Borges Martins

Aluno: _____ Substituir: _____

Instruções para a prova:

- Preencha o cabeçalho e todas as folhas desta prova com seu nome.
- Leia atentamente a todas as questões antes de resolvê-las.
- As questões desta prova foram planejadas para serem resolvidas em linguagem Java.
- Não deixe de responder nenhuma questão.
- Deixe comentários sobre as questões, eles podem ser considerados para a resolução da questão.

1. (10 pontos) Escreva uma classe **Calendar** que contenha:

   - um atributo privado *long* **dia**,
   - um construtor padrão, que inicialize o atributo com o número de dias do ano zero ao dia 01/01/2000[1],
   - um construtor que receba três valores *long*: o primeiro representando o dia, o segundo representando o mês e o terceiro representando o ano e guarde o número de dias passados do ano zero à data passada,
   - se a data informada no construtor padrão não for válida[2], levante uma exceção do tipo **DateException**,
   - um método **addDate**, que receba uma instância de **Calendar** e que some o número de dias, meses e anos das duas instâncias, e retorne este valor em uma nova instância da classe.
   - um método **toString** que não receba parâmetro e que retorne uma *String* no formato "dd/mm/aaaa", onde dd é equivalente ao dia (com dois algarismos), mm o mês (com dois algarismos), e aaaa o ano (com quatro algarismos).

   Para a realização do Calendário, crie a classe de exceção **DateException** como filha da classe **Exception**.

[1] Não se esqueça dos anos bissextos: anos múltiplos de 4 (quatro) podem ser bissextos, sendo que os anos múltiplos de 100 (cem) não são bissextos, amenos que sejam múltiplos de 400 (quatrocentos).

[2] Uma data válida é uma data que pertence ao calendário Gregoriano. Isso significa que a validação do dia, do mês e do ano devem ser feitas. Exemplo de data inválida: 32/13/-0010.

2. (10 pontos) Analise o seguinte trecho de código:

```
1  public class Numeros {
2      public static void main (String[] args){
3          final int tam = 5;
4          Real VR[] = new Real[tam];
5          VR[0] = new Real(8);
6          VR[1] = new Real(3);
7          VR[2] = new Real(7);
8          VR[3] = new Real(2);
9          VR[4] = new Real(5);
10         Ordena.bolha(VR);
11         Ordena.contagem();
12         VR[0].r = 8;
13         VR[1].r = 3;
14         VR[2].r = 7;
15         VR[3].r = 2;
16         VR[4].r = 5;
17         Ordena.quicksort(VR);
18         Ordena.contagem();
19     }
20 }
21 public interface Comparavel {
22     public int comparacao (Comparavel valor);
23 }
24 public class Ordena {
25     private static long trocas, comparacoes;
26     private static void trocar (Comparavel[] vetor, int a, int b) {
27         trocas++;
28         Comparavel temp = vetor[a];
29         vetor[a] = vetor[b];
30         vetor[b] = temp;
31     }
32     public static void bolha (Comparavel[] vetor) {
33         trocas = 0;
34         comparacoes = 0;
35         for (int i = 0; i < vetor.length; i++) {
36             for (int j = i+1; j < vetor.length; j++) {
37                 comparacoes++;
38                 if (vetor[i].comparacao(vetor[j]) == -1) {
39                     trocar (vetor, i, j);
40                 }
41             }
42         }
43     }
44     private static int QSparticao(Comparavel[] vetor, int e, int d) {
45         Comparavel valor = vetor[d];
46         int i = (e - 1);
47         for (int j = e; j < d; j++) {
48             comparacoes++;
49             if (valor.comparacao(vetor[j]) == -1) {
50                 i++;
51                 trocar(vetor, i, j);
52             }
53         }
```

```
54          i++;
55          trocar(vetor, i, d);
56          return i;
57      }
58      public static void quicksort (Comparavel[] vetor) {
59          int esquerda = 0, direita = vetor.length -1;
60          int[] pilha = new int[vetor.length];
61          int top = -1;
62          pilha[++top] = esquerda;
63          pilha[++top] = direita;
64          while (top >= 0) {
65              direita = pilha[top--];
66              esquerda = pilha[top--];
67              int p = QSparticao(vetor, esquerda, direita);
68              if (p - 1 > esquerda) {
69                  pilha[++top] = esquerda;
70                  pilha[++top] = p - 1;
71              }
72              if (p + 1 < direita) {
73                  pilha[++top] = p + 1;
74                  pilha[++top] = direita;
75              }
76          }
77      }
78      public static void contagem () {
79          System.out.println ("Última ordenação: Trocas: " + trocas + " Comparações: "
        + comparacoes);
80      }
81  }
82  public class Real implements Comparavel {
83      public double r;
84      public Real (double a) {
85          r = a;
86      }
87      public int comparacao (Comparavel valor) {
88          Real comparado = (Real) valor;
89          if (r == comparado.r)
90              return 0;
91          else if (r < comparado.r)
92              return 1;
93          else
94              return -1;
95      }
96  }
97
```

Executando o método **main**, responda:

(a) Quantas comparações o método **bolha** realiza?

(b) Quantas trocas o método método **bolha** realiza?

(c) Quantas comparações o método **quicksort** realiza?

(d) Quantas trocas o método **quicksort** realiza?

(e) Se fosse necessário organizar um vetor de forma ascendente, usando o método bolha, quais alterações no código são necessárias?

(f) Se fosse necessário organizar um vetor de forma descendente, usando o método quicksort, quais alterações no código são necessárias?

Questões	1	2	Total
Total de pontos	10	10	20
Pontos obtidos			