

quest5\Tudo.java

```
1 // 1. Dada uma coleção de dados = [12, -2, 4, 8, 29, 45, 78, 36, -17, 2, 12, 8, 3,3,-52]
  faça um programa que:
2
3 package quest5;
4
5 public class ArrayTeste {
6     public static void main(String[] args) {
7         int[] nums = {12, -2, 4, 8, 29, 45, 78, 36, -17, 2, 12, 8, 3, 3, -52};
8
9         // a. imprima o maior elemento;
10        int maior = 0;
11        for(int i = 0; i < nums.length; i++) {
12            if(i == 0) {maior = nums[i];}
13            if(nums[i] > maior) {maior = nums[i];}
14        }
15        System.out.println("Maior elemento: " + maior);
16        // b. imprima o menor elemento;
17        int menor = 0;
18        for(int i = 0; i < nums.length; i++) {
19            if(i == 0) {menor = nums[i];}
20            if(nums[i] < menor) {menor = nums[i];}
21        }
22        System.out.println("Menor elemento: " + menor);
23        // c. imprima os números pares;
24        System.out.printf("Números pares: ");
25        for(int i = 0; i < nums.length; i++) {
26            if(nums[i] % 2 == 0) {
27                if(i != nums.length - 1) {
28                    System.out.print(nums[i] + ", ");
29                } else {
30                    System.out.print(nums[i] + ";\n");
31                }
32            }
33        }
34        // d. imprima o número de ocorrências do primeiro elemento da lista;
35        int elemento = nums[0];
36        int contaEl = 0;
37        for(int i = 0; i < nums.length; i++) {
38            if(nums[i] == elemento) {
39                contaEl++;
40            }
41        }
42        System.out.println("O número de vezes que \"" + elemento + "\" aparece no conjunto
  é: " + contaEl + " vezes!");
43        // e. imprima a média dos elementos;
44        int soma = 0;
45        double media;
46        for(int i = 0; i < nums.length; i++) {
47            soma += nums[i];
48        }
49        media = soma/nums.length;
50        System.out.println("A média dos elementos: " + media);
51        // f. imprima a soma dos elementos de valor negativo.
52        System.out.printf("Números negativos: ");
53        for(int i = 0; i < nums.length; i++) {
54            if(nums[i] < 0) {
55                if(i != nums.length - 1) {
```

```
56         System.out.print(nums[i] + ", ");
57     } else {
58         System.out.print(nums[i] + ";\n");
59     }
60 }
61 }
62 }
63 }
64
65 // 2. Utilize um array unidimensional para resolver o seguinte problema: escreva
66 // um aplicativo que insere cinco números, cada um entre 10 e 100, inclusive.
67 // Enquanto cada número é lido, exiba-o somente se ele não tiver uma
68 // duplicata de um número já lido. Cuide de tratar o “pior caso”, em que todos
69 // os cinco números são diferentes. Utilize o menor array possível para
70 // resolver esse problema. Exiba o conjunto completo de valores únicos
71 // inseridos depois que o usuário inserir cada valor novo.
72
73 package quest5;
74
75 import java.util.Scanner;
76
77 public class Duplicatas {
78
79     public static void main(String[] args) {
80
81         Scanner entrada = new Scanner(System.in);
82         int[] nums = new int[5];
83         int numAtual;
84         boolean repetido = false;
85
86         for(int i = 0; i < nums.length; i++) {
87             do {
88                 System.out.println("Digite um numero entre 10 e 100: ");
89                 numAtual = entrada.nextInt();
90                 if(numAtual < 10 || numAtual > 100) {
91                     System.out.println("Digite um número válido entre 10 e 100!!!");
92                 } else {
93                     nums[i] = numAtual;
94                     System.out.printf("Números únicos atuais: ");
95                     for(int j = 0; j <= i; j++){
96                         if(j == 0) {
97                             System.out.print(nums[j] + " ");
98                         } else {
99                             for(int k = j - 1; k >= 0; k--) {
100                                 if(nums[j] == nums[k]) {repetido = true;}
101                             }
102                             if(!repetido) {
103                                 System.out.print(nums[j] + " ");
104                             }
105                             repetido = false;
106                         }
107                     }
108                     System.out.println();
109                 }
110             } while(numAtual < 10 || numAtual > 100);
111         }
112         entrada.close();
113     }
114 }
115
```

```
116 // 3. Abaixo está o código que produzimos em sala para o Jogo da Forca.
117 // Precisamos melhorá-lo. Por exemplo, se eu chutar a letra 'B' , ele não vai
118 // considerar um chute certo, pois comparará com 'b' . Estão faltando
119 // também algumas funcionalidades, como desenhar um bonequinho sendo
120 // enforcado. Seu trabalho é melhorar o código.
121
122
123 package quest5;
124
125 import java.util.Scanner;
126 import java.util.ArrayList;
127
128 public class ForcaAprimorada {
129
130     public static void main(String[] args) {
131
132         String[] palavras = {"BANANA", "MELANCIA", "PITAYA", "FEIJAO", "ARROZ"};
133         char[] palavraSorteada = palavras[(int) (Math.random() * (palavras.length + 1))]
134         .toCharArray();
135         ArrayList<Character> palavra_reservada = new ArrayList<>();
136         ArrayList<Character> array_palavra = new ArrayList<>();
137
138         for(char letra:palavraSorteada) {
139             palavra_reservada.add(letra);
140             array_palavra.add('_');
141         }
142
143         Scanner entrada = new Scanner(System.in);
144         boolean acertou = false;
145         boolean enforcou = false;
146         int erros = 0;
147
148         while(!acertou && !enforcou) {
149             System.out.println("Qual é a letra? ");
150             char chute = Character.toUpperCase(entrada.next().charAt(0));
151
152             if(palavra_reservada.contains(chute)) {
153                 for(int i=0; i < palavra_reservada.size(); i++) {
154                     if(chute == palavra_reservada.get(i))
155                         array_palavra.set(i, chute);
156                 }
157             } else {
158                 ++erros;
159                 if(erros != 6) {
160                     System.out.println("Letra errada, você ainda tem " + (6 - erros) + "
161 chances!");
162                 } else {
163                     System.out.println("Morreu enforcado coitado");
164                 }
165             }
166
167             switch (erros) {
168                 case 1:
169                     System.out.println("____");
170                     System.out.println("|  O");
171                     System.out.println("|");
172                     System.out.println("|");
173                     break;
174                 case 2:
175                     System.out.println("____");
176                     System.out.println("|  O");
```

```
174         System.out.println(" | |");
175         System.out.println(" |");
176         break;
177     case 3:
178         System.out.println(" ____");
179         System.out.println(" | 0");
180         System.out.println(" | /|");
181         System.out.println(" |");
182         break;
183     case 4:
184         System.out.println(" ____");
185         System.out.println(" | 0");
186         System.out.println(" | /|\");
187         System.out.println(" |");
188         break;
189     case 5:
190         System.out.println(" ____");
191         System.out.println(" | 0");
192         System.out.println(" | /|\");
193         System.out.println(" | /");
194         break;
195     case 6:
196         System.out.println(" ____");
197         System.out.println(" | 0");
198         System.out.println(" | /|\");
199         System.out.println(" | / \");
200         break;
201     default:
202         break;
203 }
204
205 System.out.println(array_palavra);
206 acertou = !array_palavra.contains('_');
207 enforcou = erros == 6;
208 }
209
210 if(array_palavra.contains('_')){
211     System.out.println("Você perdeu!");
212 } else {
213     System.out.println("Você acertou a palavra!");
214 }
215 entrada.close();
216 }
217
218 }
219
220 // 4. Faça um aplicativo que contenha duas classes DiarioNotas e
221 // DiarioNotasTest. Na primeira classe, é necessário armazenar o nome do
222 // curso e as notas do aluno. Crie métodos para verificar a maior e menor
223 // nota do estudante, a média delas e um gráfico de barras ( "*" ). Na classe
224 // DiarioNotasTest, você vai atribuir as notas para o objeto da classe
225 // DiarioNotas e apresentar um relatório das notas, a maior nota, a menor
226 // nota e a distribuição num gráfico de barras ( "*" ).
227
228 package quest5;
229
230 public class DiarioNotas {
231     private String nomeCurso;
232     private int[] notas;
233 }
```

```
234     public DiarioNotas(String nomeCurso, int[] notas) {
235         this.nomeCurso = nomeCurso;
236         this.notas = notas;
237     }
238
239     public String getNomeCurso() {
240         return nomeCurso;
241     }
242     public void setNomeCurso(String nomeCurso) {
243         this.nomeCurso = nomeCurso;
244     }
245     public int[] getNotas() {
246         return notas;
247     }
248     public void setNotas(int[] notas) {
249         this.notas = notas;
250     }
251
252     public int maiorNota() {
253         int maior = this.notas[0];
254         for(int i = 1; i < this.notas.length; i++) {
255             if(this.notas[i] > maior) {
256                 maior = this.notas[i];
257             }
258         }
259         return maior;
260     }
261
262     public int menorNota() {
263         int menor = this.notas[0];
264         for(int i = 1; i < this.notas.length; i++) {
265             if(this.notas[i] < menor) {
266                 menor = this.notas[i];
267             }
268         }
269         return menor;
270     }
271
272     public double mediaNotas() {
273         int soma = 0;
274         for(int i = 0; i < this.notas.length; i++) {
275             soma += this.notas[i];
276         }
277         return (double) soma/this.notas.length;
278     }
279
280     public String criaGrafico() {
281         String grafico = "";
282         for(int i = 0; i < this.notas.length; i++) {
283             grafico += (i + 1) + "º Nota: ";
284             for(int j = 0; j < this.notas[i]; j++) {
285                 grafico += "*";
286             }
287             if(i != this.notas.length) {
288                 grafico += "\n";
289             }
290         }
291         return grafico;
292     }
293
```

```
294 }
295
296 package quest5;
297
298 public class DiarioNotasTest {
299
300     public static void main(String[] args) {
301         int[] notas = {2, 3};
302         DiarioNotas diario = new DiarioNotas("Bases da computação", notas);
303
304         System.out.println("RELATÓRIO DE NOTAS:");
305         System.out.println("Maior Nota: " + diario.maiorNota());
306         System.out.println("Menor Nota: " + diario.menorNota());
307         System.out.println("Média das Notas: " + diario.mediaNotas());
308         System.out.println("Gráfico de barras das notas: \n" + diario.criaGrafico());
309     }
310 }
311
312 // 5 - Desenhe
313 // a. uma espiral com a forma quadrada centralizada no painel, utilizando o
314 // método drawLine . Uma técnica é utilizar um loop que aumenta o
315 // comprimento da linha depois de desenhar cada duas linhas. A direção
316 // na qual desenhar a próxima linha deve seguir um padrão distinto, por
317 // exemplo, para baixo, para a esquerda, para cima, para a direita.
318
319 package quest5;
320
321 import java.awt.Graphics;
322 import javax.swing.JPanel;
323
324 public class DrawPanelLine extends JPanel {
325     public void paint(Graphics g) {
326         super.paintComponent(g);
327
328         int width = getWidth();
329         int height = getHeight();
330         int variacao = 25;
331         int taxa = 25;
332         int posX = width/2;
333         int posY = height/2;
334         int direcao = 0;
335
336         for(int i = 1; i <= 75; i++) {
337             switch (direcao) {
338                 case 0:
339                     g.drawLine(posX,posY,posX,posY + taxa);
340                     posY += taxa;
341                     direcao++;
342                     break;
343                 case 1:
344                     g.drawLine(posX,posY,posX - taxa,posY);
345                     posX -= taxa;
346                     taxa += variacao;
347                     direcao++;
348                     break;
349                 case 2:
350                     g.drawLine(posX,posY,posX,posY - taxa);
351                     posY -= taxa;
352                     direcao++;
353                 case 3:
```

```
354         g.drawLine(posX,posY,posX + taxa,posY);
355         posX += taxa;
356         taxa += variacao;
357         direcao = 0;
358     default:
359         break;
360     }
361 }
362
363 }
364 }
365
366 package quest5;
367
368 import javax.swing.JFrame;
369
370 public class EspiralQuadrada {
371
372     public static void main(String[] args) {
373         DrawPanelLine panel = new DrawPanelLine();
374         JFrame application = new JFrame();
375
376         application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
377
378         application.add(panel);
379         application.setSize(250, 250);
380         application.setVisible(true);
381     }
382 }
383
384 // b. uma espiral circular, utilizando o método drawArc para desenhar um
385 // semicírculo por vez. Cada semicírculo sucessivo deve ter um raio maior
386 // (conforme especificado pela largura do retângulo delimitador) e deve
387 // continuar a desenhar onde o semicírculo anterior concluir.
388
389 package quest5;
390
391 import java.awt.Graphics;
392 import javax.swing.JPanel;
393
394 public class DrawPanelArc extends JPanel {
395     public void paint(Graphics g) {
396         super.paintComponent(g);
397
398         int width = getWidth();
399         int height = getHeight();
400         int posX = width/2;
401         int posY = height/2;
402         int diametro = 25;
403         int variacao = 25;
404
405         for(int i = 0; i < 100; i++) {
406             if(i % 2 == 0) {
407                 g.drawArc(posX, posY, diametro, diametro, 0, 180);
408                 posX -= variacao;
409                 posY -= variacao/2;
410                 diametro += variacao;
411             } else {
412                 g.drawArc(posX, posY, diametro, diametro, 0, -180);
413                 posY -= variacao/2;
```

```
414         diametro += variacao;
415     }
416 }
417 }
418 }
419
420 package quest5;
421
422 import javax.swing.JFrame;
423
424 public class EspiralArco {
425
426     public static void main(String[] args) {
427         DrawPanelArc panel = new DrawPanelArc();
428         JFrame application = new JFrame();
429
430         application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
431
432         application.add(panel);
433         application.setSize(250, 250);
434         application.setVisible(true);
435     }
436 }
437
438 // 6. Crie um código que insira 30 mil números numa ArrayList e pesquise-os.
439 // Usemos um método de System para cronometrar o tempo gasto
440 // ( System.currentTimeMillis(); ). Troque a ArrayList por um HashSet e verifique
441 // o tempo que irá demorar. O que é lento? A inserção de 30 mil elementos ou
442 // as 30 mil buscas? Descubra-o computando o tempo gasto em cada for
443 // separadamente
444
445
446 package quest5;
447
448 import java.util.ArrayList;
449 import java.util.HashSet;
450
451 public class testePerformance {
452
453     public static void main(String[] args) {
454         ArrayList<Integer> nums = new ArrayList<>();
455         HashSet<Integer> nums2 = new HashSet<>();
456
457         long tempoInsercaoArray;
458         long tempoInsercaoHash;
459         long tempoBuscaArray;
460         long tempoBuscaHash;
461         long tempoInicial;
462         long tempoFinal;
463         int qtdElementos = 30000;
464         int elementoBuscado = 10000;
465
466         //Teste tempo Inserção ArrayList
467         tempoInicial = System.currentTimeMillis();
468         for(int i = 1; i <= qtdElementos; i++) {
469             nums.add( (int) (Math.random() * 5000));
470         }
471         nums.add(elementoBuscado);
472         tempoFinal = System.currentTimeMillis();
473         tempoInsercaoArray = tempoFinal - tempoInicial;
```



```
474
475 //Teste tempo Busca ArrayList
476 tempoInicial = System.currentTimeMillis();
477 if(nums.contains(elementoBuscado)) {
478     System.out.println("Achou");
479 }
480 tempoFinal = System.currentTimeMillis();
481 tempoBuscaArray = tempoFinal - tempoInicial;
482
483 //Teste tempo Inserção HashSet
484 tempoInicial = System.currentTimeMillis();
485 for(int i = 1; i <= qtdElementos; i++) {
486     nums2.add( (int) (Math.random() * 5000));
487 }
488 nums2.add(elementoBuscado);
489 tempoFinal = System.currentTimeMillis();
490 tempoInsercaoHash = tempoFinal - tempoInicial;
491
492 //Teste tempo Busca HashSet
493 tempoInicial = System.currentTimeMillis();
494 if(nums2.contains(elementoBuscado)) {
495     System.out.println("Achou");
496 }
497 tempoFinal = System.currentTimeMillis();
498 tempoBuscaHash = tempoFinal - tempoInicial;
499
500 System.out.println("--ARRAY LIST--");
501 System.out.println("Inserção: " + tempoInsercaoArray + "ms");
502 System.out.println("Busca: " + tempoBuscaArray + "ms");
503 System.out.println();
504 System.out.println("--HASH SET--");
505 System.out.println("Inserção: " + tempoInsercaoHash + "ms");
506 System.out.println("Busca: " + tempoBuscaHash + "ms");
507
508 //ArrayList é melhor na inserção e HashSet melhor na Busca
509 }
510 }
511 }
512
513 // 7. Assim como no exercício anterior, crie uma comparação entre ArrayList e
514 // LinkedList a fim de verificar qual é a mais rápida para se adicionar
515 // elementos na primeira posição ( list.add(0, elemento) ). Seguindo o mesmo
516 // raciocínio, você pode ver qual é a mais rápida para se percorrer usando o
517 // get(indice) (sabemos que o correto seria utilizar o for aprimorado ou o
518 // Iterator ). Para isso, insira 30 mil elementos e depois percorra-os usando
519 // cada implementação de List .
520
521 package quest5;
522
523 import java.util.ArrayList;
524 import java.util.LinkedList;
525
526 public class testePerformance2 {
527
528     public static void main(String[] args) {
529         ArrayList<Integer> nums = new ArrayList<>();
530         LinkedList<Integer> nums2 = new LinkedList<>();
531
532         long tempoInsercaoArray;
533         long tempoInsercaoLinked;
```

```
534     long tempoBuscaArray;  
535     long tempoBuscaLinked;  
536     long tempoInicial;  
537     long tempoFinal;  
538     int qtdElementos = 30000;  
539  
540     for(int i = 1; i <= qtdElementos; i++) {  
541         nums.add( (int) (Math.random() * 5000));  
542     }  
543  
544     //Teste tempo Inserção ArrayList  
545     tempoInicial = System.currentTimeMillis();  
546     nums.add(0, 88888);  
547     tempoFinal = System.currentTimeMillis();  
548     tempoInsercaoArray = tempoFinal - tempoInicial;  
549  
550     //Teste tempo Busca ArrayList  
551     tempoInicial = System.currentTimeMillis();  
552     for(int i = 1; i <= qtdElementos; i++) {  
553         nums.get(i);  
554     }  
555     tempoFinal = System.currentTimeMillis();  
556     tempoBuscaArray = tempoFinal - tempoInicial;  
557  
558     for(int i = 1; i <= qtdElementos; i++) {  
559         nums2.add( (int) (Math.random() * 5000));  
560     }  
561  
562     //Teste tempo Inserção LinkedList  
563     tempoInicial = System.currentTimeMillis();  
564     nums2.add(0, 88888);  
565     tempoFinal = System.currentTimeMillis();  
566     tempoInsercaoLinked = tempoFinal - tempoInicial;  
567  
568     //Teste tempo Busca LinkedList  
569     tempoInicial = System.currentTimeMillis();  
570     for(int i = 1; i <= qtdElementos; i++) {  
571         nums2.get(i);  
572     }  
573     tempoFinal = System.currentTimeMillis();  
574     tempoBuscaLinked = tempoFinal - tempoInicial;  
575  
576     System.out.println("--ARRAY LIST--");  
577     System.out.println("Inserção: " + tempoInsercaoArray + "ms");  
578     System.out.println("Busca: " + tempoBuscaArray + "ms");  
579     System.out.println();  
580     System.out.println("--LINKED LIST--");  
581     System.out.println("Inserção: " + tempoInsercaoLinked + "ms");  
582     System.out.println("Busca: " + tempoBuscaLinked + "ms");  
583  
584     //LinkedList é melhor na inserção e ArrayList melhor na Busca  
585 }  
586 }
```