quest5\Tudo.java

```java
// 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:

package quest5;

public class ArrayTeste {
    public static void main(String[] args) {
        int[] nums =  {12, -2, 4, 8, 29, 45, 78, 36, -17, 2, 12, 8, 3, 3, -52};

        // a. imprima o maior elemento;
            int maior = 0;
            for(int i = 0; i < nums.length; i++) {
                if(i == 0) {maior = nums[i];}
                if(nums[i] > maior) {maior = nums[i];}
            }
            System.out.println("Maior elemento: " + maior);
        // b. imprima o menor elemento;
            int menor = 0;
            for(int i = 0; i < nums.length; i++) {
                if(i == 0) {menor = nums[i];}
                if(nums[i] < menor) {menor = nums[i];}
            }
            System.out.println("Menor elemento: " + menor);
        // c. imprima os números pares;
            System.out.printf("Números pares: ");
            for(int i = 0; i < nums.length; i++) {
                if(nums[i] % 2 == 0) {
                    if(i != nums.length - 1) {
                        System.out.print(nums[i] + ", ");
                    } else {
                        System.out.print(nums[i] + ";\n");
                    }
                }
            }
        // d. imprima o número de ocorrências do primeiro elemento da lista;
        int elemento = nums[0];
        int contaEl = 0;
        for(int i = 0; i < nums.length; i++) {
            if(nums[i] == elemento) {
                contaEl++;
            }
        }
        System.out.println("O número de vezes que \"" + elemento + "\" aparece no conjunto é: " + contaEl + " vezes!");
        // e. imprima a média dos elementos;
        int soma = 0;
        double media;
        for(int i = 0; i < nums.length; i++) {
            soma += nums[i];
        }
        media  = soma/nums.length;
        System.out.println("A média dos elementos: " + media);
        // f. imprima a soma dos elementos de valor negativo.
        System.out.printf("Números negativos: ");
        for(int i = 0; i < nums.length; i++) {
            if(nums[i] < 0) {
                if(i != nums.length - 1) {
```

```
                    System.out.print(nums[i] + ", ");
                } else {
                    System.out.print(nums[i] + ";\n");
                }
            }
        }
    }
}

// 2. Utilize um array unidimensional para resolver o seguinte problema: escreva
// um aplicativo que insere cinco números, cada um entre 10 e 100, inclusive.
// Enquanto cada número é lido, exiba-o somente se ele não tiver uma
// duplicata de um número já lido. Cuide de tratar o “pior caso”, em que todos
// os cinco números são diferentes. Utilize o menor array possível para
// resolver esse problema. Exiba o conjunto completo de valores únicos
// inseridos depois que o usuário inserir cada valor novo.

package quest5;

import java.util.Scanner;

public class Duplicatas {

    public static void main(String[] args) {

        Scanner entrada = new Scanner(System.in);
        int[] nums = new int[5];
        int numAtual;
        boolean repetido = false;

        for(int i = 0; i < nums.length; i++) {
            do {
                System.out.println("Digite um numero entre 10 e 100: ");
                numAtual = entrada.nextInt();
                if(numAtual < 10 || numAtual > 100) {
                    System.out.println("Digite um número válido entre 10 e 100!!!");
                } else {
                    nums[i] = numAtual;
                    System.out.printf("Números únicos atuais: ");
                    for(int j = 0; j <= i; j++){
                        if(j == 0) {
                            System.out.print(nums[j] + " ");
                        } else {
                            for(int k = j - 1; k >= 0; k--) {
                                if(nums[j] == nums[k]) {repetido = true;}
                            }
                            if(!repetido) {
                                System.out.print(nums[j] + " ");
                            }
                            repetido = false;
                        }
                    }
                    System.out.println();
                }
            } while(numAtual < 10 || numAtual > 100);
        }
        entrada.close();
    }
}

```

```java
// 3. Abaixo está o código que produzimos em sala para o Jogo da Forca.
// Precisamos melhorá-lo. Por exemplo, se eu chutar a letra 'B' , ele não vai
// considerar um chute certo, pois comparará com 'b' . Estão faltando
// também algumas funcionalidades, como desenhar um bonequinho sendo
// enforcado. Seu trabalho é melhorar o código.


package quest5;

import java.util.Scanner;
import java.util.ArrayList;

public class ForcaAprimorada {

    public static void main(String[] args) {

        String[] palavras = {"BANANA", "MELANCIA", "PITAYA", "FEIJAO", "ARROZ"};
        char[] palavraSorteada = palavras[(int) (Math.random() * (palavras.length + 1))]
.toCharArray();
        ArrayList<Character> palavra_reservada = new ArrayList<>();
        ArrayList<Character> array_palavra = new ArrayList<>();

        for(char letra:palavraSorteada) {
            palavra_reservada.add(letra);
            array_palavra.add('_');
        }

        Scanner entrada = new Scanner(System.in);
        boolean acertou = false;
        boolean enforcou = false;
        int erros = 0;

        while(!acertou && !enforcou) {
            System.out.println("Qual é a letra? ");
            char chute = Character.toUpperCase(entrada.next().charAt(0));

            if(palavra_reservada.contains(chute)) {
                for(int i=0; i < palavra_reservada.size(); i++) {
                    if(chute == palavra_reservada.get(i))
                    array_palavra.set(i, chute);
                }
            } else {
                ++erros;
                if(erros != 6) {
                    System.out.println("Letra errada, você ainda tem " + (6 - erros) + "
chances!");
                } else {
                    System.out.println("Morreu enforcado coitado");
                }
            }
            switch (erros) {
                case 1:
                    System.out.println("____");
                    System.out.println("|   O");
                    System.out.println("|");
                    System.out.println("|");
                    break;
                case 2:
                    System.out.println("____");
                    System.out.println("|   O");
```

```
                    System.out.println("|   |");
                    System.out.println("|");
                    break;
                case 3:
                    System.out.println("____");
                    System.out.println("|   O");
                    System.out.println("|  /|");
                    System.out.println("|");
                    break;
                case 4:
                    System.out.println("____");
                    System.out.println("|   O");
                    System.out.println("|  /|\\");
                    System.out.println("|");
                    break;
                case 5:
                    System.out.println("____");
                    System.out.println("|   O");
                    System.out.println("|  /|\\");
                    System.out.println("|  /");
                    break;
                case 6:
                    System.out.println("____");
                    System.out.println("|   O");
                    System.out.println("|  /|\\");
                    System.out.println("|  / \\");
                    break;
                default:
                    break;
            }

            System.out.println(array_palavra);
            acertou = !array_palavra.contains('_');
            enforcou = erros == 6;
        }

        if(array_palavra.contains('_')){
            System.out.println("Você perdeu!");
        } else {
            System.out.println("Você acertou a palavra!");
        }
        entrada.close();
    }

}

// 4. Faça um aplicativo que contenha duas classes DiarioNotas e
// DiarioNotasTest. Na primeira classe, é necessário armazenar o nome do
// curso e as notas do aluno. Crie métodos para verificar a maior e menor
// nota do estudante, a média delas e um gráfico de barras ( ”*” ). Na classe
// DiarioNotasTest, você vai atribuir as notas para o objeto da classe
// DiarioNotas e apresentar um relatório das notas, a maior nota, a menor
// nota e a distribuição num gráfico de barras ( ”*” ).

package quest5;

public class DiarioNotas {
    private String nomeCurso;
    private int[] notas;

```

```java
    public DiarioNotas(String nomeCurso, int[] notas) {
        this.nomeCurso = nomeCurso;
        this.notas = notas;
    }

    public String getNomeCurso() {
        return nomeCurso;
    }
    public void setNomeCurso(String nomeCurso) {
        this.nomeCurso = nomeCurso;
    }
    public int[] getNotas() {
        return notas;
    }
    public void setNotas(int[] notas) {
        this.notas = notas;
    }

    public int maiorNota() {
        int maior = this.notas[0];
        for(int i = 1; i < this.notas.length; i++) {
            if(this.notas[i] > maior) {
                maior = this.notas[i];
            }
        }
        return maior;
    }

    public int menorNota() {
        int menor = this.notas[0];
        for(int i = 1; i < this.notas.length; i++) {
            if(this.notas[i] < menor) {
                menor = this.notas[i];
            }
        }
        return menor;
    }

    public double mediaNotas() {
        int soma = 0;
        for(int i = 0; i < this.notas.length; i++) {
            soma += this.notas[i];
        }
        return (double) soma/this.notas.length;
    }

    public String criaGrafico() {
        String grafico = "";
        for(int i = 0; i < this.notas.length; i++) {
            grafico += (i + 1) + "º Nota: ";
            for(int j = 0; j < this.notas[i]; j++) {
                grafico += "*";
            }
            if(i != this.notas.length) {
                grafico += "\n";
            }
        }
        return grafico;
    }

```

```
}

package quest5;

public class DiarioNotasTest {

    public static void main(String[] args) {
        int[] notas = {2, 3};
        DiarioNotas diario = new DiarioNotas("Bases da computação", notas);

        System.out.println("RELATÓRIO DE NOTAS:");
        System.out.println("Maior Nota: " + diario.maiorNota());
        System.out.println("Menor Nota: " + diario.menorNota());
        System.out.println("Média das Notas: " + diario.mediaNotas());
        System.out.println("Gráfico de barras das notas: \n" + diario.criaGrafico());
    }
}

// 5 - Desenhe
// a. uma espiral com a forma quadrada centralizada no painel, utilizando o
// método drawLine . Uma técnica é utilizar um loop que aumenta o
// comprimento da linha depois de desenhar cada duas linhas. A direção
// na qual desenhar a próxima linha deve seguir um padrão distinto, por
// exemplo, para baixo, para a esquerda, para cima, para a direita.

package quest5;

import java.awt.Graphics;
import javax.swing.JPanel;

public class DrawPanelLine extends JPanel {
    public void paint(Graphics g) {
        super.paintComponent(g);

        int width = getWidth();
        int height = getHeight();
        int variacao = 25;
        int taxa = 25;
        int posX = width/2;
        int posY = height/2;
        int direcao = 0;

        for(int i = 1; i <= 75; i++) {
            switch (direcao) {
                case 0:
                    g.drawLine(posX,posY,posX,posY + taxa);
                    posY += taxa;
                    direcao++;
                    break;
                case 1:
                    g.drawLine(posX,posY,posX - taxa,posY);
                    posX -= taxa;
                    taxa += variacao;
                    direcao++;
                    break;
                case 2:
                    g.drawLine(posX,posY,posX,posY - taxa);
                    posY -= taxa;
                    direcao++;
                case 3:
```

```
                    g.drawLine(posX,posY,posX + taxa,posY);
                    posX += taxa;
                    taxa += variacao;
                    direcao = 0;
                default:
                    break;
            }
        }

    }
}

package quest5;

import javax.swing.JFrame;

public class EspiralQuadrada {

    public static void main(String[] args) {
        DrawPanelLine panel = new DrawPanelLine();
        JFrame application = new JFrame();

        application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        application.add(panel);
        application.setSize(250, 250);
        application.setVisible(true);
    }
}

// b. uma espiral circular, utilizando o método drawArc para desenhar um
// semicírculo por vez. Cada semicírculo sucessivo deve ter um raio maior
// (conforme especificado pela largura do retângulo delimitador) e deve
// continuar a desenhar onde o semicírculo anterior concluir.

package quest5;

import java.awt.Graphics;
import javax.swing.JPanel;

public class DrawPanelArc extends JPanel {
    public void paint(Graphics g) {
        super.paintComponent(g);

        int width = getWidth();
        int height = getHeight();
        int posX = width/2;
        int posY = height/2;
        int diametro = 25;
        int variacao = 25;

        for(int i = 0; i < 100; i++) {
            if(i % 2 == 0) {
                g.drawArc(posX, posY, diametro, diametro, 0, 180);
                posX -= variacao;
                posY -= variacao/2;
                diametro += variacao;
            } else {
                g.drawArc(posX, posY, diametro, diametro, 0, -180);
                posY -= variacao/2;
```

```
                diametro += variacao;
            }
        }
    }
}

package quest5;

import javax.swing.JFrame;

public class EspiralArco {

    public static void main(String[] args) {
        DrawPanelArc panel = new DrawPanelArc();
        JFrame application = new JFrame();

        application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        application.add(panel);
        application.setSize(250, 250);
        application.setVisible(true);
    }
}

// 6. Crie um código que insira 30 mil números numa ArrayList e pesquise-os.
// Usemos um método de System para cronometrar o tempo gasto
// ( System.currentTimeMillis(); ). Troque a ArrayList por um HashSet e verifique
// o tempo que irá demorar. O que é lento? A inserção de 30 mil elementos ou
// as 30 mil buscas? Descubra-o computando o tempo gasto em cada for
// separadamente


package quest5;

import java.util.ArrayList;
import java.util.HashSet;

public class testePerformance {

    public static void main(String[] args) {
        ArrayList<Integer> nums = new ArrayList<>();
        HashSet<Integer> nums2 = new HashSet<>();

        long tempoInsercaoArray;
        long tempoInsercaoHash;
        long tempoBuscaArray;
        long tempoBuscaHash;
        long tempoInicial;
        long tempoFinal;
        int qtdElementos = 30000;
        int elementoBuscado = 10000;

        //Teste tempo Inserção ArrayList
        tempoInicial = System.currentTimeMillis();
        for(int i = 1; i <= qtdElementos; i++) {
            nums.add( (int) (Math.random() * 5000));
        }
        nums.add(elementoBuscado);
        tempoFinal = System.currentTimeMillis();
        tempoInsercaoArray = tempoFinal - tempoInicial;
```

```java

        //Teste tempo Busca ArrayList
        tempoInicial = System.currentTimeMillis();
        if(nums.contains(elementoBuscado)) {
            System.out.println("Achou");
        }
        tempoFinal = System.currentTimeMillis();
        tempoBuscaArray = tempoFinal - tempoInicial;

        //Teste tempo Inserção HashSet
        tempoInicial = System.currentTimeMillis();
        for(int i = 1; i <= qtdElementos; i++) {
            nums2.add( (int) (Math.random() * 5000));
        }
        nums2.add(elementoBuscado);
        tempoFinal = System.currentTimeMillis();
        tempoInsercaoHash = tempoFinal - tempoInicial;

        //Teste tempo Busca HashSet
        tempoInicial = System.currentTimeMillis();
        if(nums2.contains(elementoBuscado)) {
            System.out.println("Achou");
        }
        tempoFinal = System.currentTimeMillis();
        tempoBuscaHash = tempoFinal - tempoInicial;

        System.out.println("--ARRAY LIST--");
        System.out.println("Inserção: " + tempoInsercaoArray + "ms");
        System.out.println("Busca: " + tempoBuscaArray + "ms");
        System.out.println();
        System.out.println("--HASH SET--");
        System.out.println("Inserção: " + tempoInsercaoHash + "ms");
        System.out.println("Busca: " + tempoBuscaHash + "ms");

        //ArrayList é melhor na inserção e HashSet melhor na Busca
    }

}

// 7. Assim como no exercício anterior, crie uma comparação entre ArrayList e
// LinkedList a fim de verificar qual é a mais rápida para se adicionar
// elementos na primeira posição ( list.add(0, elemento) ). Seguindo o mesmo
// raciocínio, você pode ver qual é a mais rápida para se percorrer usando o
// get(indice) (sabemos que o correto seria utilizar o for aprimorado ou o
// Iterator ). Para isso, insira 30 mil elementos e depois percorra-os usando
// cada implementação de List .

package quest5;

import java.util.ArrayList;
import java.util.LinkedList;

public class testePerformance2 {

    public static void main(String[] args) {
        ArrayList<Integer> nums = new ArrayList<>();
        LinkedList<Integer> nums2 = new LinkedList<>();

        long tempoInsercaoArray;
        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 }
```