Fundação CECIERJ – Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação Orientada a Objetos
AP1X 1° semestre de 2020.

Nome –

Assinatura –

Observações:

1. **As respostas precisam ser entregues de forma manuscrita e a caneta.**
2. **Utilize qualquer folha em branco para transcrever suas respostas.** Dê preferência a folhas pautadas (com linhas). Assine em cada folha que utilizar.
3. **Estas folhas devem ser escaneadas e enviadas pela plataforma**. Certifique-se de que estejam legíveis. Utilize algum aplicativo para escanear, como o CamScanner, de forma que o(s) arquivo(s) fique(m) pequeno(s).
4. **Não seja o plágio do que ouve, vê, lê por aí. Estilo é plagiar a si mesmo** (Última frase é atribuída a Alfred Hitchcock).

Questão 1) (5.0 pontos)

Imagine que queiramos criar um programa para gerenciar notas de texto. Notas possuem o texto, uma lista de labels (tags, palavras chave que descrevem o que contém a mensagem) e sua data de criação. Considere a implementação que utiliza o sistema supostamente existente:

```
public class AP1X_2020_2_Q1 {
	public static void main(String[] args) {
		Nota n1 = new Nota("CPF: 999999999-99", new String[] {"documento",
"identificação"}); // O segundo argumento é uma forma compacta de criar um vetor de
strings
		Nota n2 = new Nota("Matricula UFF: 99999999-9", new String[] {"documento",
"universidade"});
		Bloco anotacoes = new Bloco();
		anotacoes.adicionarNota(n1);
		anotacoes.adicionarNota(n2);
		System.out.println(anotacoes); // Imprime todas as notas
		Bloco faculdade = anotacoes.buscaNotas("UFF");
		System.out.println(faculdade); // Imprime apenas notas com a palavra UFF
incluída
	}
}
```

Observe atentamente o código no método main() e implemente as classes de forma que o código acima funcione.

RESPOSTA:

```
import java.time.LocalDate;

class Nota {
	String mensagem;
	String labels[];
	LocalDate horaCriacao;

	public Nota(String mensagem, String[] labels) {
		this.mensagem = mensagem;
		this.labels = labels;
		this.horaCriacao = LocalDate.now();
	}

	public String obtemMensagem() {
		return this.mensagem;
	}

	public String toStringLabels() {
		String resultado = "";
		for (int i=0; i<this.labels.length; i++)
			resultado = resultado + labels[i] + " ";
		return resultado;
	}

	public String toString() {
		return this.mensagem + "\n" + this.toStringLabels() + "\n" + horaCriacao
+ "\n";
	}
}

class Bloco {
	Nota notas[];
	private int qtdNotas;

	public Bloco() {
		notas = new Nota[1000];
		this.qtdNotas = 0;
	}

	public void adicionarNota (Nota n) {
		notas[qtdNotas] = n;
		qtdNotas++;
	}

	public Bloco buscaNotas (String palavra) {
		Bloco resultado = new Bloco();
		for (int i=0; i<qtdNotas; i++)
			if (notas[i].obtemMensagem().contains(palavra))
				resultado.adicionarNota(notas[i]);
		return resultado;
	}

	public String toString() {
		String resultado = "";
		for (int i=0; i<qtdNotas; i++)
			resultado = resultado + notas[i];
		return resultado;
	}
}
```

Questão 2) (5.0 pontos)
Escreva um programa em Java que leia uma matriz quadrada de dimensão 9, com valores de 1 a 9 em suas posições, e verifique se esta matriz é uma solução válida para o ***Sudoku*** (isto é, uma solução é válida no ***Sudoku*** implica que cada linha, cada coluna e cada bloco contém os números de 1 a 9 somente uma vez).

```
5 3 4 | 6 7 8 | 9 1 2
6 7 2 | 1 9 5 | 3 4 8
1 9 8 | 3 4 2 | 5 6 7
------+-------+------
8 5 9 | 7 6 1 | 4 2 3
4 2 6 | 8 5 3 | 7 9 1
7 1 3 | 9 2 4 | 8 5 6
------+-------+------
9 6 1 | 5 3 7 | 2 8 4
2 8 7 | 4 1 9 | 6 3 5
3 4 5 | 2 8 6 | 1 7 9
```

Dica: a matriz deve ser lida utilizando-se a classe Scanner.

RESPOSTA:

```
import java.util.*;

public class AP1X_Q2_2020_1 {
  public static void main (String[] args){
    Scanner sc = new Scanner(System.in);
    int i, j, aux, sudoku[][] = new int[9][9];
    for (i = 0; i < 9; i++)
      for (j = 0; j < 9; j++){
        aux = sc.nextInt();
        sudoku[i][j] = aux;
      }
    sc.close();

    boolean resp = testaLinhas(sudoku);
    if(!resp){
      System.out.println(resp);
      return;
    }
    resp = testaColunas(sudoku);
    if(!resp){
      System.out.println(resp);
      return;
    }
    resp = testaBlocos(sudoku);
    System.out.println(resp);
  }

    static boolean testaLinhas (int[][] mat){
    int i, j;
    for(i = 0; i < 9; i++){
```

```
      int vet[] = new int[9];
      for(j = 0; j < 9; j++) vet[mat[i][j] - 1]++;
      for(j = 0; j < 9; j++)
        if(vet[j] != 1)
          return false;
    }
    return true;
  }

  static boolean testaColunas (int[][] mat){
    int i, j;
    for(j = 0; j < 9; j++){
      int vet[] = new int[9];
      for(i = 0; i < 9; i++) vet[mat[i][j] - 1]++;
      for(i = 0; i < 9; i++)
        if(vet[i] != 1)
          return false;
    }
    return true;
  }

  static boolean testaBlocos (int[][] mat){
    int i, j;
    for(i = 0; i < 9; i = i + 3)
        for(j = 0; j < 9; j = j + 3){
          boolean resp = testaBlocos(mat, i, i + 3, j, j + 3);
          if(!resp) return true;
        }
    return true;
  }

   static boolean testaBlocos (int[][] mat, int lini, int lfim,
int cini, int cfim){
    int i, j, vet[] = new int[9];
    for(i = lini; i < lfim; i++)
      for(j = cini; j < cfim; j++)
         vet[mat[i][j] - 1]++;
    for(int k = 0; k < 9; k++)
      if(vet[k] != 1) return false;
    return true;
  }
}
```