Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP1 1° semestre de 2013.

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

Questão 1) (4.0 pontos)
Uma matriz quadrada composta somente de zeros e uns é considerada booleana se ela possui a seguinte propriedade de paridade: a soma de cada linha e de cada coluna é par. Esta matriz tem sido amplamente aplicada na área de transmissão de dados em redes de computadores pelo fato de ser uma forma simples de correção de erros. Se um bit de transmissão está corrompido (ou seja, um único bit está invertido), ele pode ser facilmente detectado e reparado.

Por exemplo, a matriz 4x4 abaixo segue a propriedade de paridade:

```
1 0 1 0
0 0 0 0
1 1 1 1
0 1 0 1
```

Já a matriz abaixo não segue esta propriedade porque o primeiro bit está invertido. Se o primeiro bit for alterado para 1, esta matriz passa a ser de paridade.

```
0 0 1 0
0 0 0 0
1 1 1 1
0 1 0 1
```

Dado o seguinte programa teste, que lê os dados a partir da entrada padrão (NÃO ALTERE O MÉTODO MAIN E NEM A CLASSE INDICE), escreva os métodos que estão faltando:

a) (1.0 ponto) **`matrizParidade`**: indica se a matriz tem a propriedade de paridade. Este método retorna **`true`** se a matriz segue a propriedade e **`false`**, caso contrário;

b) (1.5 pontos) **`soUmBit`**: indica se a matriz não segue a propriedade de paridade por um único bit. Este método retorna **`true`** se a matriz não segue a propriedade de paridade por um único bit e **`false`**, caso contrário; e

c) (1.5 pontos) **`qualBit`**: indica o único bit corrompido (i, j) que pode ser invertido para restaurar a propriedade de paridade. Se não existir nenhum bit a ser invertido, este método deve retornar **`null`**.

```
import java.util.Scanner;
class Indice{
  int i, j;
  Indice(int i, int j){
    this.i = i;
    this.j = j;
  }
  public String toString(){return "(" + i + "," + j + ")";}
}

public class Q1_AP1_2013_1{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    int N = sc.nextInt(), i, j;
    int mat[][] = new int[N][N];
    for(i = 0; i < N; i++)
      for(j = 0; j < N; j++)
        mat[i][j] = sc.nextInt();
    sc.close();
    boolean resp = matrizParidade(mat);
    if(resp){
      System.out.println("Matriz paridade");
      return;
    }
    resp = soUmBit(mat);
    if(resp){
      Indice par = qualBit(mat);
      System.out.println(par);
      return;
    }
  }

  static boolean matrizParidade(int mat[][]){...}

  static boolean soUmBit(int mat[][]){...}

  static Indice qualBit(int mat[][]){...}
}
```

RESPOSTA:

```
import java.util.Scanner;
class Indice{...}
```

```
public class Q1_AP1_2013_1{
  public static void main(String[] args){...}

  static boolean matrizParidade(int mat[][]){
    int i, j, aux;
    for(i = 0; i < mat.length; i++){
      aux = 0;
      for(j = 0; j < mat.length; j++) aux += mat[i][j];
      if(aux % 2 == 1) return false;
    }
    return true;
  }

  static boolean soUmBit(int mat[][]){
    int i, j, linha[];
    linha = new int[mat.length];
    for(i = 0; i < mat.length; i++){
      linha[i] = 0;
      for(j = 0; j < mat.length; j++) linha[i] += mat[i][j];
    }
    int cont = 0;
    for(i = 0; i < linha.length; i++) if(linha[i] % 2 == 1) cont++;
    if (cont % 2 == 1) return true;
    return false;
  }

  static Indice qualBit(int mat[][]){
    int i, j, linha[], coluna[];
    linha = new int[mat.length];
    coluna = new int[mat.length];
    for(i = 0; i < mat.length; i++){
      linha[i] = 0;
      for(j = 0; j < mat.length; j++) linha[i] += mat[i][j];
    }

    for(i = 0; i < mat.length; i++){
      coluna[i] = 0;
      for(j = 0; j < mat.length; j++) coluna[i] += mat[j][i];
    }

    for(i = 0; i < mat.length; i++)
      for(j = 0; j < mat.length; j++)
        if((linha[i] % 2 == 1) && (coluna[j] % 2 == 1))
          return new Indice(i, j);
    return null;
  }
}
```

Questão 2) (3.0 pontos)

Suponha que precisamos implementar um sistema para a gerência de embarcações. Uma embarcação possui como dados básicos a quantidade de passageiros e suas dimensões (comprimento e largura). Como tipos de embarcações, para este nosso sistema, podemos ter as voltadas para esporte e lazer (por ex., caiaques, barcos à vela, remo, etc) e as para transporte (barcas, navios que transportam petróleo, balsas, etc). Nas de esporte e lazer, uma característica comum é a existência ou não de vela. Para as de transporte,

normalmente nos interessa saber a capacidade de carga e a potência da embarcação. Em todas também é interessante registrar a velocidade máxima.

a) Crie as classes que representam os tipos de embarcações do sistema com seus atributos.
b) Defina os construtores dessas classes. Para o caso das embarcações de transporte, há uma restrição de que o valor da potência tenha que ser, no mínimo, o valor da sua carga. Na construção de um objeto, caso os valores passados não respeitem esta regra, altere-os de forma a respeitá-la.
c) Crie 2 objetos para exemplificar a instanciação das classes: i) 1 caiaque para 1 pessoa, com dimensão (2 x 0.3), velocidade máxima de 10 (unidades de velocidade), lembrando que caiaques não possuem vela; ii) 1 barca para 2000 pessoas, com dimensão (40 x 20), velocidade máxima de 100, potência de 500 e carga de 1000.

RESPOSTA:

```
class Embarcacao {
	int capacidade;
	double velocMax;
	double comprimento;
	double largura;

	public Embarcacao(int capacidade, double velocMax, double comprimento,
			double largura) {
		this.capacidade = capacidade;
		this.velocMax = velocMax;
		this.comprimento = comprimento;
		this.largura = largura;
	}
}

class Esporte extends Embarcacao {
	boolean vela;
	public Esporte(int capacidade, double velocMax, double comprimento,
			double largura, boolean vela) {
		super(capacidade, velocMax, comprimento, largura);
		this.vela = vela;
	}
}

class Transporte extends Embarcacao {
	double potencia;
	double carga;
	public Transporte(int capacidade, double velocMax, double comprimento,
			double largura, double potencia, double carga) {
		super(capacidade, velocMax, comprimento, largura);
		if (potencia < carga)
			potencia = carga;
		this.potencia = potencia;
		this.carga = carga;
	}
}

public class AP1_2013_1_Q2 {
	public static void main(String[] args) {
		Embarcacao caiaque = new Esporte(1, 10, 2, 0.3, false);
		Embarcacao barca = new Transporte(2000, 100, 40, 20, 500, 1000);
	}
}
```

Questão 3) (3.0 pontos)
Defina uma classe de objetos que representam uma data (dia, mês e ano), contendo os seguintes métodos:

a) um método toString, que retorna uma cadeia de caracteres correspondente à data representada pelo objeto alvo da chamada, no formato "dd/mm/aaaa", onde dd, mm e aaaa correspondem, respectivamente, ao dia, mês e ano desta data;
b) um método compara, que compara a data representada pelo objeto alvo da chamada com uma data passada como argumento para o método; o valor retornado deve ser 0 se essas datas são iguais, ou um número negativo se a primeira data é anterior à última, ou um número positivo se a primeira é posterior à última.

RESPOSTA:

```
class Data {
	int dia;
	int mes;
	int ano;

	public Data(int dia, int mes, int ano) {
		this.dia = dia;
		this.mes = mes;
		this.ano = ano;
	}

	public String toString() {
		return dia + "/" + mes + "/" + ano;
	}

	public int compara (Data d) {
		if (dia == d.dia && mes == d.mes && ano == d.ano)
			return 0;
		if (ano < d.ano ||
				(ano == d.ano && mes < d.mes) ||
					(ano == d.ano && mes == d.mes && dia < d.dia))
			return -1;
		return 1;
	}
}
```