Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação Orientada a Objetos
AP3 2° semestre de 2017.

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) (5.0 pontos)
Suponha o código do programa abaixo:

```
class Tel {
	int pais;
	int cidade;
	int numero;

	public Tel(int pais, int cidade, int numero) {
		this.pais = pais;
		this.cidade = cidade;
		this.numero = numero;
	}

	public boolean igual (Tel t) {
		return (this.pais == t.pais && this.cidade == t.cidade &&
this.numero == t.numero);
	}

	public String toString() {
		return "+" + pais + " " + cidade + " " + numero;
	}
}
public class AP3_2011_2_Q2 {
	public static void main(String[] args) {
		Agenda ag = new Agenda();
		Contato c = new Contato("Fulano", "Petrópolis");
		c.adicionaTelefone(new Tel(55, 24, 99999999));
```

```
                c.adicionaTelefone(new Ramal(55, 24, 22222222, 3500)); // Último
parâmetro é o número do ramal
                ag.adicionaContato(c);
                Contato c2 = new Contato("Ciclano", "Rio das Ostras");
                c2.adicionaTelefone(new Tel(55, 22, 99999999));
                ag.adicionaContato(c2);
                ag.imprimeContatos();
        }
}
```

Implemente as classes Ramal, Contato e Agenda para que a execução do programa tenha a seguinte saída:

```
Fulano
Petrópolis
+55 24 99999999 +55 24 22222222 #3500

Ciclano
Rio das Ostras
+55 22 99999999
```

RESPOSTA:

```
class Ramal extends Tel {
        int ramal;

        public Ramal(int pais, int cidade, int numero, int ramal) {
                super(pais, cidade, numero);
                this.ramal = ramal;
        }

        public boolean igual (Ramal r) {
                return (super.igual(r) && this.ramal == r.ramal);
        }

        public String toString() {
                return super.toString() + " #" + this.ramal;
        }
}

class Contato {
        String nome;
        String endereco;
        List<Tel> telefones;

        public Contato(String nome, String endereco) {
                this.nome = nome;
                this.endereco = endereco;
                this.telefones = new ArrayList<Tel>();
        }

        public void adicionaTelefone (Tel tel) {
                boolean pertence = false;
                for (Tel t : this.telefones)
                        if (t.igual(tel)) {
                                pertence = true;
                                break;
                        }
```

```
                if (!pertence)
                        this.telefones.add(tel);
        }

        public String toString () {
                String tels = "";
                for (Tel t : this.telefones)
                        tels = tels + t.toString() + " ";
                return this.nome + "\n" + this.endereco + "\n" + tels + "\n";
        }
}

class Agenda {
        List<Contato> contatos;
        public Agenda() {
                contatos = new ArrayList<Contato>();
        }

        public void adicionaContato (Contato c) {
                contatos.add(c);
        }

        public void imprimeContatos() {
                for (Contato c: contatos) {
                        System.out.println(c);
                }
        }
}
```

Questão 2) (5.0 pontos)

Considere a existência de um arquivo texto, passado como parâmetro de entrada para o seu programa. A estrutura interna deste arquivo é composta por um valor inteiro N que indica a quantidade de registros contidos no arquivo, seguido pelos N registros propriamente ditos. Cada um dos N registros é composto por um valor inteiro K e um texto T. O valor inteiro K de um registro indica a quantidade máxima de parênteses e colchetes aninhados que se espera encontrar no texto do mesmo registro. Por exemplo, em:

```
4 Mantenha (a calma, [concentre-se] e escreva) programas (que
           atendam (aos [[enunciados]]) propostos)
```

a quantidade máxima de parênteses e colchetes aninhados é $K = 4$.

O problema é que os textos contidos no arquivo podem ter a quantidade máxima de parênteses e colchetes aninhados diferente da indicada em seu registro respectivo. Além disso, nada garante que o padrão de abertura e fechamento esteja correta. Abaixo é apresentado um exemplo de aplicação incorreta de padrão de abertura e fechamento de parênteses e colchetes:

```
3 Este (texto não (respeita [o padrão) de] abertura) e fechamento
                  [de parênteses] e colchetes
```

Dada a classe:

```
class PilhaDeCaracteres {
  private char[] elems = new char[1];
  private int n = 0;
```

```
  public void empilhar(char x) {
    if (this.n == this.elems.length) {
      char[] temp = new char[this.elems.length * 2];
        System.arraycopy(this.elems, 0, temp, 0, this.elems.length);
        this.elems = temp;
      }
      this.elems[this.n++] = x;
  }

  public void desempilhar() {
    verificaSeVazia();
    this.n--;
  }

  public char topo() {
    verificaSeVazia();
    return this.elems[this.n - 1];
  }

  public int tamanho() { return this.n; }

  private void verificaSeVazia() {
    if (this.n == 0) {
      throw new RuntimeException("A pilha está vazia.");
    }
  }
}
```

Seu programa deve abrir o arquivo de entrada, analisar se o padrão de abertura e fechamento de parênteses está correto e, caso esteja, se a quantidade máxima de parênteses e colchetes aninhados é igual à quantidade K indicada no par.

O resultado deverá ser escrito no arquivo texto de nome **"saida-"**, acrescido do nome do arquivo de entrada. Cada linha deste arquivo está relacionada a um registro do arquivo de entrada. As linhas deverão ser iguais a um dos três casos:

`Padrão não respeitado`

`Padrão respeitado, mas quantidade K incorreta`

`Padrão respeitado e quantidade K correta`

Dica: os métodos **`char charAt(int index)`** e **`int length()`** da classe **`java.lang.String`** podem ser utilizados.

RESPOSTA:

```
import java.io.*;

//UNICA CLASSE QUE DEVERIA SER IMPLEMENTADA PELO ALUNO...
public class Q2_AP3_2017_2 {
  public static void main(String[] args) throws IOException{
    BufferedReader in = new BufferedReader(new FileReader(args[0]));
    BufferedWriter out;


    try{
      int n = Integer.parseInt(in.readLine());
      String str = "";
```

```
      for (int i = 0; i < n; i++){
        int k = Integer.parseInt(in.readLine());
        String t = in.readLine();
        str += verificaPar(k, t);
      }
      in.close();
      out = new BufferedWriter(new FileWriter("saida-" + args[0]));
      out.write(str);
      out.close();
    } catch (IOException erro) {
        System.err.println("Houve um erro no manuseio do arquivo de
entrada (" + erro.getMessage() + ").");
        return;
    }
  }

  private static String verificaPar(int n, String t) {
    int k = 0, maiorK = 0;
    PilhaDeCaracteres pilha = new PilhaDeCaracteres();
    for (int i = 0; i < t.length(); i++){
      char c = t.charAt(i);
      if (c == '(' || c == '[') { pilha.empilhar(c); k++; }
      else if (c == ')' || c == ']'){
        if ((pilha.tamanho() == 0) || ((c == ')') && (pilha.topo() !
= '(')) || ((c == ']') && (pilha.topo() != '['))) {
          return "Padrão não respeitado\n";
        }
        maiorK = Math.max(maiorK, k--);
        pilha.desempilhar();
      }
    }

    if (pilha.tamanho() != 0) { return "Padrão não respeitado\n"; }

    else if (n != maiorK) {
      return "Padrão respeitado, mas quantidade K incorreta\n";
    }

    else { return "Padrão respeitado e quantidade K correta\n"; }
  }
}
```