Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP2 1° semestre de 2007.

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) (2.5 pontos)

Escreva um programa que receba como parâmetro de entrada, o nome de um arquivo texto, cujo conteúdo são três notas dos alunos do curso, uma em cada linha, e que calcule e exiba a média dos alunos e a média da turma. Um exemplo de uso desse programa seria java calculaMedia notas.txt.

```
import java.io.*;
public class Calc {
 public static void main(String[] args) throws IOException {
   BufferedReader in = new BufferedReader(new FileReader(args[0]));
   String s;
   float x, aluno = 0f, media = 0f;
   int n = 0;
   try{
      while (true){
        n++;
        s = in.readLine();
        x = Float.parseFloat(s);
        aluno += x;
        media += x;
        if(n % 3 == 0){
          System.out.println(aluno/3 + "\n");
          aluno = 0f;
        }
```

```
      }
    }
    catch (Exception e){
      System.out.println("Excecao\n");
    }
    finally{
      System.out.println("media " + media/(n - 1) + "\n");
      in.close();
    }
  }
}
```

Questão 2) (2.5 pontos)

Em uma aplicação de agência de encontros, cada pessoa possui um nome e um perfil. Uma pessoa "casa" com outra se os seus perfis combinam. A combinação de perfis é a seguinte:

uma pessoa com perfil intelectual apenas aceita se encontrar com outra pessoa de perfil intelectual.

uma pessoa com perfil esportivo aceita se encontrar com outra pessoa também esportiva ou uma pessoa com perfil romântico.

uma pessoa com perfil romântico aceita se encontrar com pessoas de qualquer perfil.

Projete e construa uma solução que verifica se uma pessoa aceita se encontrar com uma outra. Para testar sua solução, crie um programa principal que instancie várias pessoas, com nomes e perfis diferentes, e exiba na console se o encontro entre elas é possível ou não.

```
abstract class Perfil {
  public boolean combina (Perfil p) {
    return false;
  }
}

class Intelectual extends Perfil {
  public boolean combina (Perfil p) {
    if (p instanceof Intelectual) return true;
    return false;
  }
}

class Esportivo extends Perfil {

  public boolean combina (Perfil p) {
    if (p instanceof Esportivo) return true;
    return false;
  }
}

class Romantico extends Perfil {
```

```
  public boolean combina (Perfil p) {
    if (p instanceof Romantico) return true;
    if (p instanceof Intelectual) return true;
    if (p instanceof Esportivo) return true;
    return false;
  }
}

class Pessoa {

  private Perfil perfil;

  public Pessoa (Perfil perfil) {
    this.perfil = perfil;
  }

  public boolean casa (Pessoa p) {
    return this.perfil.combina (p.perfil);
  }

  public static void main (String[] args) {
    Pessoa maria = new Pessoa(new Esportivo());
    Pessoa joao = new Pessoa (new Esportivo());

    Pessoa ana = new Pessoa (new Intelectual());
    Pessoa jose = new Pessoa (new Intelectual());

    Pessoa romeu = new Pessoa (new Romantico());
    Pessoa julieta = new Pessoa (new Romantico());

    System.out.println ("Maria e Joao: " + maria.casa(joao));
    System.out.println ("Maria e Ana: " + maria.casa(ana));
    System.out.println ("Maria e Jose: " + maria.casa(jose));

    System.out.println ("Joao e Maria: " + joao.casa(maria));
    System.out.println ("Joao e Ana: " + joao.casa(ana));
    System.out.println ("Joao e Jose: " + joao.casa(jose));

    System.out.println ("Ana e Maria: " + ana.casa(maria));
    System.out.println ("Ana e Joao: " + ana.casa(joao));
    System.out.println ("Ana e Jose: " + ana.casa(jose));

    System.out.println ("Jose e Maria: " + jose.casa(maria));
    System.out.println ("Jose e Joao: " + jose.casa(joao));
    System.out.println ("Jose e Ana: " + jose.casa(ana));

    System.out.println ("Romeu e Julieta: " + romeu.casa(julieta));
    System.out.println ("Julieta e Romeu: " + julieta.casa(romeu));
    System.out.println ("Romeu e Maria: " + romeu.casa(maria));
    System.out.println ("Maria e Romeu: " + maria.casa(romeu));
  }
}
```

Questão 3) (2.5 pontos)

Faça um programa para, dado um arquivo texto composto por linhas e uma palavra a ser procurada, imprima na tela os números das linhas no arquivo que contém a palavra buscada. Se a primeira linha contém a palavra, o valor a ser impresso deve ser "1". Separe os valores impressos com espaços.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class Questao3AP2_20071 {
	public static void buscaPalavra (String arquivo, String palavra) {
		BufferedReader in = null;
		try {
			int quant_linhas = 1;
		in = new BufferedReader(new FileReader(arquivo));
		String str;
		while ((str = in.readLine()) != null) {
				if (str.contains(palavra)) {
					System.out.print(quant_linhas + " ");
			}
				quant_linhas++;
		}
		in.close();
		} catch (IOException e) {
			System.out.print("Problemas na manipulação do arquivo!");
		}
	}
	public static void main(String[] args) {
		buscaPalavra("c:\\teste.txt", "teste");
	}
}
```

Nesta questão, o objetivo foi avaliar o entendimento do aluno quanto à manipulação de arquivos em Java. Como era um arquivo texto, bastava ler linha a linha e testar se a palavra buscada se encontrava na linha. Além disso, precisávamos de um contador para contar o número de linhas, que é incrementado independente de termos ou não a palavra na linha. O método main() é apenas para exemplificação do uso da função.

Questão 4) (2.5 pontos)

Define um método para, dada uma LinkedList e um valor do tipo inteiro, este método retorne uma nova LinkedList com todos os valores menores que o valor passado.

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Iterator;
import java.util.List;

public class Questao4AP2_20071 {
	public static List filtraLista (List l, int valor) {
		List nova = new ArrayList();
```

```
            Iterator it = l.iterator();
            while (it.hasNext()) {
                  int proximo = (Integer) it.next();
                  if (proximo < valor)
                        nova.add(proximo);
            }
            if (nova.size() == 0)
                  return null;
            return nova;
      }
      public static void main(String[] args) {
            List valores = Arrays.asList(0, 1, 3, 4, 7, 5, 2, 8);
            List nova = filtraLista(valores, 5);
            if (nova != null)
                  System.out.println(nova.toString());
      }
}
```

Nesta questão, o objetivo foi avaliar o entendimento do aluno quanto à manipulação de coleções em Java. Na função apresentada, utilizamos um iterador, o qual é utilizado para percorrer a lista passada. Observe que o parâmetro *l* é do tipo *List*, o que permite que nossa função seja utilizada com qualquer exemplo de implementação da interface *List* (não apenas *ArrayList*). O método main(), a exemplo da questão anterior, é apresentado apenas para exemplificação do uso da função.