Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP2 2° semestre de 2008.

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)

Projete uma hierarquia de classes (classes ou interfaces) para dar suporte à um programa que manipula objetos geográficos. Esta hierarquia deve dar suporte às seguintes classes:

- Países
- Estados
- Cidades
- Fronteiras
- Rios

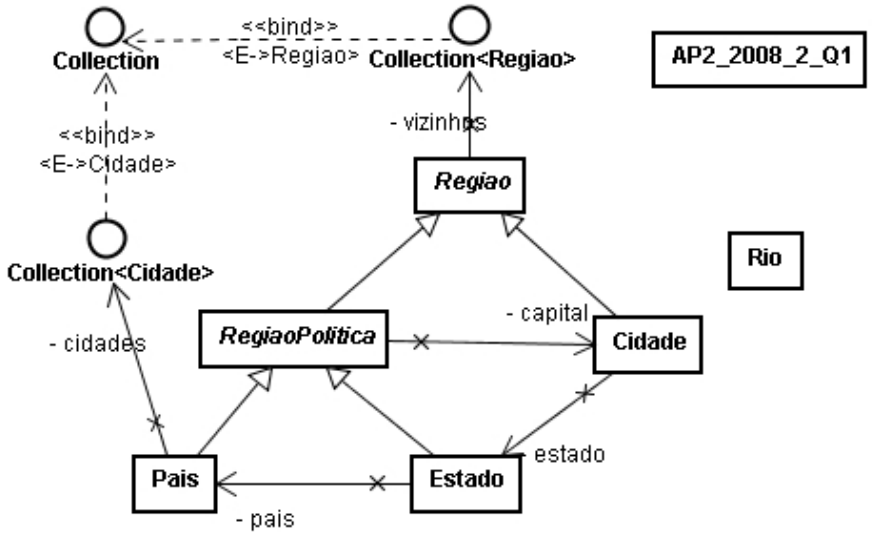
Ofereça as seguintes operações, quando aplicável (os argumentos foram omitidos):

- area()
- capital()
- obtemCidades()
- obtemPais()
- distancia(), distância entre cidades, países
- tamanhoDaFronteira()
- vizinhos(), elementos que compartilham a mesma fronteira

Desenha uma figura abstrata de suas classes (sem se preocupar com nenhuma sintaxe mais rígida). Após, apresente a definição de cada classe, variáveis de instância e definição de métodos. Crie interfaces e superclasses quando apropriado.

Resposta:

O objetivo deste exercício é exercitar a tarefa de modelagem OO. O desenho da modelagem sugerida foi feita utilizando o programa Jude (versão Community – gratuita): https://jude.change-vision.com/jude-web/index.html



Nesta modelagem, as setas terminando com triângulo representam as relações de herança (por exemplo, de Cidade para Regiao). As setas simples indicam a declaração de atributos da classe apontada (por exemplo, a seta de Cidade para Estado indica que a classe possui um atributo chamado *estado*, o qual indica qual estado a cidade pertence).

Abaixo apresentamos a listagem com a estrutura básica para essas classes. Nesta listagem, os valores de área e distância foram colocados com valor 0 apenas para deixar a listagem completa (não gerar erro de compilação). Naturalmente, estes valores poderiam ser recuperados de alguma estrutura que os armazenassem.

```
import java.util.Collection;

/**
 * Classe principal que inicia o programa
 * @author bazilio
 */
public class AP2_2008_2_Q1 {
      public static void main(String[] args) {
      }
}
```

```
abstract class Regiao {
	Collection<Regiao> vizinhos;
	double tamanho;
	public abstract double area();
	public Collection<Regiao> vizinhos() {
		return vizinhos;
	}
	public double tamanhoDaFronteira () {
		return tamanho;
	}
}

abstract class RegiaoPolitica extends Regiao {
	Cidade capital;
	public abstract double area();
	public Cidade capital() {
		return capital;
	}
	public double distancia (RegiaoPolitica e) {
		return this.capital().distancia(e.capital());
	}
}

class Pais extends RegiaoPolitica {
	Collection<Cidade> cidades;
	public Collection<Cidade> obtemCidades() {
		return cidades;
	}
	public double area() {
		return 0;
	}
}

class Estado extends RegiaoPolitica {
	Pais pais;
	public Pais obtemPais() {
		return pais;
	}
	public double area() {
		return 0;
	}
}

class Cidade extends Regiao {
	Estado estado;
	public Pais obtemPais() {
		return estado.obtemPais();
	}
	public double distancia (Cidade destino) {
		return 0;
	}
	public double area() {
		return 0;
	}
}

class Rio {
```

```
}
```

Questão 2) (2.5 pontos)

Podemos representar um polinômio como uma lista ordenada de termos, onde os termos são ordenados por seus expoentes.

Para adição de 2 (dois) polinômios, suas respectivas listas são percorridas e os termos nas posições correntes dos iteradores são examinadas. Se o expoente de um é menor que o de outro, insere o primeiro na lista resultante e avança o iterador do primeiro. Se os expoentes são iguais, então criamos um novo termo com este expoente, somamos os coeficientes e avançamos os 2 (dois) iteradores. Por exemplo: $3x^4 + 2x^2 + 3x + 7$ somado com $2x^3 + 4x + 5$ resulta em $3x^4 + 2x^3 + 2x^2 + 7x + 12$.

Se encararmos um polinômio como uma função f(x), o valor obtido por este é dado pela substituição de x no valor fornecido em f. Por exemplo, f(2), para $f(x) = 2x^3 + 4x + 5$ é igual a 29

Escreva um programa para ler, adicionar e realizar o cálculo de polinômios. Além da classe polinômio, você deve definir uma classe Termo que contém atributos expoente e coeficiente.

Resposta:

A listagem abaixo contém alguns métodos extras que tornam o programa mais completo, como por exemplo, os métodos *imprimePolinomio* (exibe os termos de um polinômio) e *impriveValorPolinomio* (exibe o valor produzido por uma f(x), onde f é definida pelo polinômio). Entretanto, só será cobrado o que foi pedido na questão.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * Classe principal que inicia o programa
 * @author bazilio
 */
public class AP2_2008_2_Q2 {
      public static void main(String[] args) {
            Polinomio polinomio1 = new Polinomio();
            Polinomio polinomio2 = new Polinomio();

            polinomio1.lePolinomio();
            polinomio2.lePolinomio();

            Polinomio polinomio3 = polinomio1.adicao(polinomio2);

            polinomio3.imprimePolinomio();
```

```
            polinomio3.imprimeValorPolinomio();
      }
}

/**
 * Classe que modela um polinômio
 * @author bazilio
 */
class Polinomio {
      private List<Termo> termos;

      public Polinomio() {
            termos = new ArrayList<Termo>();
      }

      public List<Termo> obtemTermos() {
            return termos;
      }

      public void imprimePolinomio() {
            Iterator<Termo> it = termos.iterator();
            while (it.hasNext()) {
                  System.out.print(((Termo)it.next()).toString() + " ");
            }
            System.out.println();
      }

      public void lePolinomio() {
            try {
              BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
              String str = "";
              int exp = -1, coef = -1;
              System.out.println("Digite os expoentes e coeficientes do
polinômio ");
            while (exp != 0 && coef != 0) {
                  System.out.println("Expoente: ");
                  str = in.readLine();
                  exp = Integer.parseInt(str);
                  System.out.println("Coeficiente: ");
                  str = in.readLine();
                  coef = Integer.parseInt(str);
                  if (exp != 0 && coef != 0) {
                        Termo t = new Termo(coef, exp);
                        termos.add(t);
                  }
              }
          } catch (IOException e) {
          }
      }

      public Polinomio adicao(Polinomio p) {
            Termo t1 = null;
            Termo t2 = null;

            Iterator<Termo> it1 = termos.iterator();
            Iterator<Termo> it2 = p.obtemTermos().iterator();
```

```
            Polinomio pRes = new Polinomio();
            List<Termo> res = pRes.obtemTermos();

            while (it1.hasNext() && it2.hasNext()) {
                  if (t1 == null)
                        t1 = (Termo)(it1.next());
                  if (t2 == null)
                        t2 = (Termo)(it2.next());
                  if (t1.getExpoente() == t2.getExpoente())
                        res.add(new
Termo(t1.getCoeficiente()+t2.getCoeficiente(), t1.getExpoente()));
                  else
                        if (t1.getExpoente() < t2.getExpoente())
                              res.add(new Termo(t1.getCoeficiente(),
t1.getExpoente()));
                        else
                              res.add(new Termo(t2.getCoeficiente(),
t2.getExpoente()));

                  if (it1.hasNext())
                        t1 = (Termo)(it1.next());
                  else
                  {
                        res.add(new Termo(t2.getCoeficiente(),
t2.getExpoente()));
                        while (it2.hasNext()) {
                              t2 = (Termo)(it2.next());
                              res.add(new Termo(t2.getCoeficiente(),
t2.getExpoente()));
                        }
                        break;
                  }
                  if (it2.hasNext())
                        t2 = (Termo)(it2.next());
                  else
                  {
                        res.add(new Termo(t1.getCoeficiente(),
t1.getExpoente()));
                        while (it1.hasNext()) {
                              t1 = (Termo)(it1.next());
                              res.add(new Termo(t1.getCoeficiente(),
t1.getExpoente()));
                        }
                        break;
                  }
            }

            while (it1.hasNext()) {
                  t1 = (Termo)(it1.next());
                  res.add(new Termo(t1.getCoeficiente(),
t1.getExpoente()));
            }

            while (it2.hasNext()) {
                  t2 = (Termo)(it2.next());
```

```
                  res.add(new Termo(t2.getCoeficiente(),
t2.getExpoente()));
            }

            return pRes;
      }

      public void imprimeValorPolinomio() {
            BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
        String str = "";
        int exp = 0;
        System.out.println("Digite o valor de x: ");
        try {
                  str = in.readLine();
            } catch (IOException e) {
                  e.printStackTrace();
            }
        exp = Integer.parseInt(str);
        System.out.println("O valor do polinomio(" + str + ") é: " +
this.calculaPolinomio(exp));
      }

      public double calculaPolinomio(int x) {
            double valor = 0;
            Termo t1 = null;
            Iterator<Termo> it1 = termos.iterator();

            while (it1.hasNext()) {
                  t1 = (Termo)(it1.next());
                  valor = valor + t1.getCoeficiente() * Math.pow(x,
t1.getExpoente());
            }

            return valor;
      }
}

/**
 * Classe que modela 1 único termo
 * @author bazilio
 */
class Termo {
      int coeficiente;
      int expoente;

      public Termo(int c, int e) {
            coeficiente = c;
            expoente = e;
      }

      public int getCoeficiente() {
            return coeficiente;
      }

      public int getExpoente() {
            return expoente;
```

```
	}

	public void setCoeficiente(int coeficiente) {
		this.coeficiente = coeficiente;
	}

	public void setExpoente(int expoente) {
		this.expoente = expoente;
	}

	public String toString() {
		return coeficiente + "x" + expoente;
	}
}
```

Questão 3) (2.5 pontos)
Suponha que em várias aplicações diferentes você precisa de uma funcionalidade genérica que calcula a soma dos valores de cada objeto de um conjunto. Para facilitar, você resolve implementar uma classe **Totalizador** que seja capaz de retornar essa informação para diferentes conjuntos de objetos.

Implemente seu componente reusável **Totalizador** e um programa de teste do seu componente aplicado a dois conjuntos de objetos diferentes: um conjunto de produtos e um conjunto de pessoas. O valor total obtido sobre o conjunto de produtos é a soma dos preços dos produtos. O valor total obtido sobre o conjunto de pessoas é a soma das idades das pessoas.

RESPOSTA:

```
import java.util.*;
class Totalizador {
  private Iterator all;
  public Totalizador (Iterator all) {
   this.all = all;
  }

  public float getTotal() {
    float total=0;
    while (all.hasNext()) {
      ItemValoravel item = (ItemValoravel)all.next();
       total+=item.getValor();
    }
    return total;
  }

  public static void main(String[] args) {
    Vector produtos = new Vector();
    produtos.add (new Produto (3));
    produtos.add (new Produto (10));
    produtos.add (new Produto (8));
    produtos.add (new Produto (4.5F));

    Totalizador s1 = new Totalizador(produtos.iterator());
```

```
    System.out.println ("Total de preco de produtos = " +
s1.getTotal());

    Vector pessoas = new Vector();
    pessoas.add (new Pessoa (30));
    pessoas.add (new Pessoa (10));

    Totalizador s2 = new Totalizador(pessoas.iterator());
    System.out.println ("Total de idade das pessoas = " +
s2.getTotal());
  }
}

interface ItemValoravel {
  float getValor();
}

class Produto implements ItemValoravel {
  private float preco;

  public Produto (float preco) {
    this.preco = preco;
  }

  public float getValor() {
    return preco;
  }
}

class Pessoa implements ItemValoravel {
  private int idade;

  public Pessoa (int idade) {
    this.idade = idade;
  }
  public float getValor() {
    return idade;
  }
}
```

Questão 4) (2.5 pontos)

Escreva um programa que receba como parâmetro de entrada, o nome de um arquivo texto, cujo conteúdo são o nome do aluno e as duas notas dos alunos do curso, uma em cada linha, e que ordene o arquivo de saída em ordem crescente pela média do aluno. Isto é, se eu tiver como entrada o arquivo:

Paulo	10.0	10.0
José	3.0	4.0
João	7.0	7.0
Alex	0.5	1.5
Ítalo	5.0	6.0

A saída será:

Alex 1.0
José 3.5

```
Ítalo  5.5
João   7.0
Paulo  10.0
```

Um exemplo de uso desse programa seria java calculaMedia notas.txt.

RESPOSTA:

```
import java.io.*;
public class calculaMedia {
 public static void main(String[] args) throws IOException {
   BufferedReader in = new BufferedReader(new FileReader(args[0]));
   int n = 0;
   String s;

   try{
     while ((s = in.readLine()) != null) n++;
   }
   catch (Exception e) {
     System.out.println("Excecao1\n");
   }

   try {
      String vet_nomes[] = new String[n];
      float vet[] = new float[n];
      int cont = 0;
      String vs[];
      float x, media;

      in.close();
      in = new BufferedReader(new FileReader(args[0]));

      while((s = in.readLine()) != null) {
        media = 0f;
        vs = s.split("\t");
        vet_nomes[cont] = vs[0];
          x = Float.parseFloat(vs[1]);
          media += x;
          x = Float.parseFloat(vs[2]);
          media += x;
        media /= 2;
          vet[cont++] = media;
      }
      BufferedWriter out = new BufferedWriter(new FileWriter("saida-
"+args[0]));
      Ordena(vet, vet_nomes);
      for (int i = 0; i < n; i++) {
          out.write(vet_nomes[i]+"\t");
          out.write(vet[i]+"\n");
      }
      out.close();
   }
   catch (Exception e){
     System.out.println("Excecao2\n");
   }
   finally{
     in.close();
```

```
    }
  }

  public static void Ordena (float[] vet, String[] vet_nomes){
    int i, j, menor;
    float temp;
    String aux;

    for (i = 0; i < vet.length; i++){
       menor = i;

       for (j = i + 1; j < vet.length; j++)
          if (vet[j] < vet [menor]) menor = j;

       temp = vet[menor];
       vet[menor] = vet[i];
       vet[i] = temp;

       aux = vet_nomes[menor];
       vet_nomes[menor] = vet_nomes[i];
       vet_nomes[i] = aux;
    }
  }
}
```