Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP2 2° semestre de 2010.

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) (3 pontos)

Escreva a classe **Polinomio** com os métodos soma, subtração, derivada e integral. Use a seguinte classe Teste para dar suporte a sua implementação:

```
import java.io.*;
public class Teste{
 public static void main(String[] args) throws IOException {
    BufferedReader in = new BufferedReader(new FileReader(args[0]));
    int grau1, grau2, i;
    try{
      grau1 = Integer.parseInt(in.readLine());
      Polinomio p = new Polinomio(grau1);
      for(i = 0; i <= grau1; i++)
        p.modificaCoeficiente(i, Float.parseFloat(in.readLine()));
      System.out.println(p);
      grau2 = Integer.parseInt(in.readLine());
      Polinomio q = new Polinomio(grau2);
      for(i = 0; i <= grau2; i++)
        q.modificaCoeficiente(i, Float.parseFloat(in.readLine()));
      System.out.println(q);
      in.close();
      Polinomio soma = p.soma(q);
      System.out.println(soma);
      Polinomio sub1 = p.subtrai(q); //sub1 = p - q
      System.out.println(sub1);
      Polinomio sub2 = q.subtrai(p); //sub2 = q - p
      System.out.println(sub2);
```

```
      Polinomio derp = p.derivada();
      System.out.println(derp);
      Polinomio derq = q.derivada();
      System.out.println(derq);
      Polinomio intp = p.integral();
      System.out.println(intp);
      Polinomio intq = q.integral();
      System.out.println(intq);
    } catch(Exception e){
      System.out.println(e);
    }
  }
}
```

RESPOSTA:

```
class Polinomio{
  int grau;
  float coef[];

  Polinomio(int g){
    grau = g;
    coef = new float[g + 1];
  }

  void modificaCoeficiente(int g, float valor){
    if(((g == grau) && (valor == 0.0)) || (g > grau)){
      System.out.println("Seu polinomio esta errado!");
      return;
    }
    if(g <= grau) coef[g] = valor;
  }

  public String toString(){
    String resp = "";
    for(int i = grau; i >= 0; i--)
      if(coef[i] >= 0)
        resp += "+ " + coef[i] + " X^" + i + " ";
      else
        resp += coef[i] + " X^" + i + " ";
    resp += "\n";
    return resp;
  }

  Polinomio derivada(){
    Polinomio resp = new Polinomio(grau - 1);
    for(int i = grau; i > 0; i--) resp.coef[i - 1] = i * coef[i];
    return resp;
  }


  Polinomio integral(){
    Polinomio resp = new Polinomio(grau + 1);
    resp.coef[0] = 0;
     for(int i = grau; i >= 0; i--) resp.coef[i + 1] = coef[i] / (i +
1);
    return resp;
  }
```

```
  Polinomio soma(Polinomio p){
    Polinomio resp;
    if((grau == p.grau) && (coef[grau] == (-1 * p.coef[p.grau])))
      resp = new Polinomio(grau - 1);
    else if(grau >= p.grau)
      resp = new Polinomio(grau);
    else
      resp = new Polinomio(p.grau);
    int i = 0;
    while(i <= resp.grau){
      if((i <= grau) && (i <= p.grau))
        resp.modificaCoeficiente(i, (coef[i] + p.coef[i]));
      else if(i <= grau)
        resp.modificaCoeficiente(i, coef[i]);
      else
        resp.modificaCoeficiente(i, p.coef[i]);
      i++;
    }
    return resp;
  }

  Polinomio subtrai(Polinomio p){
    Polinomio resp;
    if((grau == p.grau) && (coef[grau] == p.coef[p.grau]))
      resp = new Polinomio(grau - 1);
    else if(grau >= p.grau)
      resp = new Polinomio(grau);
    else
      resp = new Polinomio(p.grau);
    int i = 0;
    while(i <= resp.grau){
      if((i <= grau) && (i <= p.grau))
        resp.modificaCoeficiente(i, (coef[i] - p.coef[i]));
      else if(i <= grau)
        resp.modificaCoeficiente(i, coef[i]);
      else
        resp.modificaCoeficiente(i, (-1 * p.coef[i]));
      i++;
    }
    return resp;
  }
}
```

Questão 2) (3 pontos)

Considere dois arquivos texto **"itens1.txt"** e **itens2.txt"**, contendo registros sobre itens de estoque de um supermercado. Cada registro contém o nome do produto, preço, marca e data de validade. Em ambos os arquivos, os registros estão ordenados pelo nome do produto. Escreva um programa que leia os dois arquivos e gere um terceiro arquivo, cujo nome é **"tudo.txt"**, formado pela combinação dos dois anteriores de modo que os registros continuem ordenados pela chave nome. Os arquivos não devem ser lidos para memória.

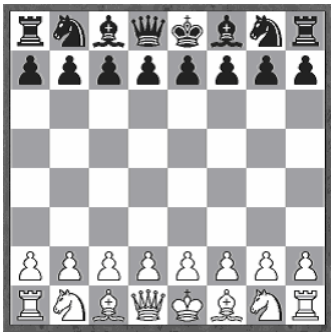
RESPOSTA:

```
import java.io.*;
public class CEDERJ_q2{
 public static void main(String[] args) throws IOException {
   BufferedReader in1 = new
BufferedReader(newFileReader("itens1.txt"));
   BufferedReader in2 = new
BufferedReader(newFileReader("itens2.txt"));
   BufferedWriter out = new BufferedWriter(new FileWriter("tudo.txt"));
   String s1, s2;
   String[] p1, p2;
   try {
      s1 = in1.readLine();
      s2 = in2.readLine();
      while((s1 != null) && (s2 != null)){
        p1 = s1.split(" ");
        p2 = s2.split(" ");
        if(p1[0].compareTo(p2[0]) < 0){
          out.write(s1 + "\n");
          s1 = in1.readLine();
        }
        else{
          out.write(s2 + "\n");
          s2 = in2.readLine();
        }
      }
      if(s1 == null){
        in1.close();
        while(s2 != null){
          out.write(s2 + "\n");
          s2 = in2.readLine();
        }
      }
      else{
        in2.close();
        while(s1 != null){
          out.write(s1 + "\n");
          s1 = in1.readLine();
        }
      }
      out.close();
   }
   catch (Exception e){
     System.out.println("Excecao\n");
   }
 }
}
```

Questão 3) (4 pontos)

O Xadrez é um jogo para dois oponentes em um tabuleiro com 32 peças (16 para cada jogador) de seis tipos: Peões (linhas 2 e 7 do tabuleiro), Torres (cantos do tabuleiros), Cavalos, Bispos (peças com uma cruz), Dama (peças nas linha 1 e coluna 5, e nas linha 8 e coluna 5) e Rei (linha 1 e coluna 4, e linha 8 e coluna 4). Cada peça move-se de forma

distinta. O objetivo do jogo é dar o xeque-mate, isto é, ameaçar o Rei do oponente. Os jogos não terminam necessariamente com o xeque-mate – os jogadores, com certa frequência, desistem se acreditam que irão perder. Além disso, existem várias formas de um jogo terminar em um empate.



Imagine que queiramos iniciar a implementação este jogo.

a) Inicie criando a estrutura básica (declaração da classe, relação de herança, se existir, e construtores) para as classes de cada tipo de peça. Cada peça é distinguida pela sua imagem própria (pode ser representada pela classe java.awt.Image). Além disso, para que não haja confusão entre as peças dos oponentes, temos que distinguir as peças pretas das brancas.
b) Crie uma classe Tabuleiro, a qual manterá informação sobre o conteúdo (peça ou vazio) de cada posição do tabuleiro (matriz 8x8).
c) Crie uma classe JogoXadrez, a qual deve conter como atributos o tabuleiro e 2 listas de peças, referentes às peças capturadas por cada um dos jogadores. Esta classe deve ter um construtor para inicialização destes atributos e posicionamento inicial das peças do jogo, segundo a figura ilustrada na questão.
d) Das peças listadas o Cavalo é quem possui o comportamento um tanto diferente: além de poder pular qualquer peça durante o seu movimento, este sempre se move formando um L (2 posições na horizontal e 1 para a vertical, ou vice-versa, ou seja, 1 para a vertical e 2 posições para a horizontal, em qualquer sentido). Resguardando os limites do tabuleiro, crie um método na classe Cavalo que, dada 2 posições no tabuleiro (a posição corrente e a posição para a qual se deseja ir), indique se esta pode ser uma nova posição para o Cavalo em questão (ATENÇÃO: Observe os limites do tabuleiro!).

Obviamente, utilize os conceitos de OO sempre que possível.

RESPOSTA:

```
package br.cederj.comp.ano2010;

import java.awt.Image;
import java.awt.Point;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

import javax.imageio.ImageIO;
```

```
abstract class Peca {
	static enum Cor {preta, branca};
	Cor tipo;
	Image imagem;

	public Peca(Cor c) {
		tipo = c;
		imagem = null;
	}

	public Image getImagem() { return imagem; };
}

class Peao extends Peca {
	public Peao(Cor c) {
		super(c);
	}
}

class Torre extends Peca {
	public Torre(Cor c) {
		super(c);
	}
}

class Cavalo extends Peca {
	public Cavalo(Cor c) {
		super(c);
		try {
			// A imagem pode ser carregada de qualquer local.
			// Neste caso, apenas como exemplo, é uma URL
			URL url = new URL("http://xadrez.com//cavalo.gif");
			imagem = ImageIO.read(url);
		} catch (IOException e) { }
	}

	public boolean movimentoValido (int x, int y, int x2, int y2) {
		if ((((x+1 == x2) && (y-2 == y2)) ||
			((x+2 == x2) && (y-1 == y2)) ||
			((x+2 == x2) && (y+1 == y2)) ||
			((x+1 == x2) && (y+2 == y2)) ||
			((x-1 == x2) && (y+2 == y2)) ||
			((x-2 == x2) && (y+1 == y2)) ||
			((x-2 == x2) && (y-1 == y2)) ||
			((x-1 == x2) && (y-2 == y2)))
			&& Tabuleiro.posicaoValida(x2, y2))
			return true;
		return false;
	}
}

class Bispo extends Peca {
	public Bispo(Cor c) {
		super(c);
	}
}
```

```
class Dama extends Peca {
	public Dama(Cor c) {
		super(c);
	}
}

class Rei extends Peca {
	public Rei(Cor c) {
		super(c);
	}
}

class Tabuleiro {
	Peca tabuleiro [][];

	public Tabuleiro() {
		tabuleiro = new Peca[8][8];
	}

	public static boolean posicaoValida(int x, int y) {
		return (x >= 1 && x <= 8 && y >= 1 && y <= 8) ? true : false;
	}

	public static boolean posicaoValida(Point p) {
		return Tabuleiro.posicaoValida(p.x, p.y);
	}

	public void atribui(int i, int j, Peca peca) {
		tabuleiro[i][j] = peca;
	}
}

class JogoXadrez {
	Tabuleiro tabuleiro;
	List<Peca> pecasCapturadasJogadorPreto;
	List<Peca> pecasCapturadasJogadorBranco;

	public JogoXadrez() {
		pecasCapturadasJogadorBranco = new ArrayList<Peca>();
		pecasCapturadasJogadorPreto = new ArrayList<Peca>();
		tabuleiro = new Tabuleiro();

		// PeÂ›Ãµes
		for (int j=0;j<8;j++) {
			tabuleiro.atribui(1, j, new Bispo(Peca.Cor.preta));
			tabuleiro.atribui(6, j, new Bispo(Peca.Cor.branca));
		}

		// Torres
		tabuleiro.atribui(0, 0, new Torre(Peca.Cor.preta));
		tabuleiro.atribui(0, 7, new Torre(Peca.Cor.preta));
		tabuleiro.atribui(7, 0, new Torre(Peca.Cor.branca));
		tabuleiro.atribui(7, 7, new Torre(Peca.Cor.branca));

		// Cavalos
		tabuleiro.atribui(0, 1, new Cavalo(Peca.Cor.preta));
```

```
                tabuleiro.atribui(0, 6, new Cavalo(Peca.Cor.preta));
                tabuleiro.atribui(7, 1, new Cavalo(Peca.Cor.branca));
                tabuleiro.atribui(7, 6, new Cavalo(Peca.Cor.branca));

                // Bispos
                tabuleiro.atribui(0, 2, new Bispo(Peca.Cor.preta));
                tabuleiro.atribui(0, 5, new Bispo(Peca.Cor.preta));
                tabuleiro.atribui(7, 2, new Bispo(Peca.Cor.branca));
                tabuleiro.atribui(7, 5, new Bispo(Peca.Cor.branca));

                // Reis
                tabuleiro.atribui(0, 3, new Rei(Peca.Cor.preta));
                tabuleiro.atribui(7, 3, new Rei(Peca.Cor.branca));

                // Bispos
                tabuleiro.atribui(0, 4, new Dama(Peca.Cor.preta));
                tabuleiro.atribui(7, 4, new Dama(Peca.Cor.branca));
        }
}
```