Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP3 1° semestre de 2016.

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 abaixo:

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

public class AP3_2012_2_Q3 {
	public static void main(String[] args) {
		//Nome e url
		EnderecoWWW e = new EnderecoWWW("Cederj", "www.cederj.edu.br");
		//Nome, url e tamanho da imagem
		Imagem i = new Imagem("Cederj",
"http://www.cederj.edu.br/fundacao/imagenssuperior_arquivos/menu_logo.jpg", 50);
		List<Recurso> pagina = new ArrayList<Recurso>();
		pagina.add(e);
		pagina.add(i);
		for (Recurso r : pagina)
			System.out.println("O recurso " + r.toString() + " eh " +
r.valido());
	}
}
```

Implemente as classes/interfaces necessárias para que o código acima funcione. A chamada ao método *valido()* retorna verdadeiro quando o recurso se tratar de uma imagem e terminar com *jpg* ou *png* ou quando o recurso for um endereço www e iniciar com a string *www*. O método *toString()* deve retornar a url respectiva de cada recurso. Utilize os conceitos de OO vistos sempre que possível para, por exemplo, evitar redundância no código.

Dica: Os métodos startsWith e endsWith testam (retornam booleano) se uma string passada por parâmetro começa e termina, respectivamente, uma dada string.

RESPOSTA:

```
interface IRecurso { // Opcional
	boolean valido();
}

abstract class Recurso implements IRecurso {
	String nome;
	String url;

	public Recurso (String nome, String url) {
		this.nome = nome;
		this.url = url;
	}

	public String toString() {
		return url;
	}
}

class EnderecoWWW extends Recurso {
	public EnderecoWWW (String nome, String url) {
		super(nome, url);
	}

	public boolean valido() {
		return this.url.startsWith("www");
	}
}

class Imagem extends Recurso {
	int tamanho;
	public Imagem (String nome, String url, int tamanho) {
		super(nome, url);
		this.tamanho = tamanho;
	}

	public boolean valido() {
		return this.url.endsWith("jpg") || this.url.endsWith("png");
	}
}
```

Questão 2) (5.0 pontos)
Supondo que você tenha uma empresa de software, e que a Associação Mundial de Tênis contrate sua empresa para escrever um programa que informe, automaticamente, o ranking dos jogadores de tênis. Este ranking segue o sistema de pontos corridos. Por questões de simplificação, seu software deve contabilizar, somente, os quatro grand slams (Roland Garros – RG, Austrália Open – AO, Wimbledom – WI e US Open – US). A pontuação destes torneios é a seguinte:

Colocação	Sigla	Pontos
Vencedor	WIN	2000
Finalista	FIN	1200
Semifinalista	SF	720
Quartas-de-final	QF	360
Oitavas-de-final	OF	180
Terceira rodada	TR	90
Segunda rodada	SR	45
Primeira rodada	FR	10

O dado de entrada é um arquivo texto, em que o nome é passado como parâmetro de entrada, cujo o conteúdo é formado por jogadores e suas posições nestes torneios nos dois últimos anos (neste caso os anos de 2009 e 2010, respectivamente). Para o seguinte exemplo de arquivo:

```
Roger Federer/RG WIN SF/AO FIN WIN/WI WIN SF/US FIN SF
Rafael Nadal/RG SF WIN/AO WIN SF/WI QF WIN/US SF WIN
Novak Djokovic/RG SF SF/AO QF QF/WI QF FIN/US SF FIN
Andy Murray/RG SF OF/AO SF FIN/WI TR TR/US SR SR
```

O seu software deve informar o seguinte ranking, **APÓS LER O ARQUIVO DE ENTRADA UMA ÚNICA VEZ:**

```
1 Rafael Nadal (de 3800 para 6720): subiu 1 posicao
2 Roger Federer (de 6400 para 4160): desceu 1 posicao
3 Novak Djokovic (de 2160 para 3480): inalterada posicao
4 Andy Murray (de 1575 para 1515): inalterada posicao
```

SE SEU PROGRAMA LER MAIS DE UMA VEZ O ARQUIVO DE ENTRADA OU NÃO RESPONDER CORRETAMENTE PARA QUALQUER ARQUIVO QUE SIGA O FORMATO ANTERIORMENTE CITADO, SUA RESPOSTA SERÁ SEVERAMENTE DESCONTADA.

RESPOSTA:

```
import java.io.*;

class Rank{
  int posicao, npontos;

  Rank(int np){ npontos = np; posicao = 0; }

  void modificaPontos(int np){ npontos += np; }
  public String toString(){ return " " + npontos; }
}


class Jogador{
  String nome;
  Rank anterior, atual;
  Jogador prox;

  Jogador(String n){
    nome = n;
```

```
    anterior = new Rank(0);
    atual = new Rank(0);
    prox = null;
  }

  void incluiPontos(int np, boolean rank){
    if(rank) atual.modificaPontos(np);
    else anterior.modificaPontos(np);
  }

  public String toString(){
     String resp = nome + "\t (de" + anterior.toString() + " para" +
atual.toString() + "): ";
    return resp;
  }
}


class ATP{
  Jogador prim;

  ATP(){ prim = null; }

  void inclui(Jogador j){
    if(prim != null) j.prox = prim;
    prim = j;
  }

  void troca(Jogador p, Jogador q){
    String nome_temp = p.nome;
    p.nome = q.nome;
    q.nome = nome_temp;

    Rank temp = p.atual;
    p.atual = q.atual;
    q.atual = temp;

    temp = p.anterior;
    p.anterior = q.anterior;
    q.anterior = temp;
  }

  void ordena(){
    Jogador p;
    for(p = prim; p != null; p = p.prox){
      Jogador maior = p, q;
      for(q = p.prox; q != null; q = q.prox)
        if(p.anterior.npontos < q.anterior.npontos) maior = q;
      if(maior != p) troca (maior, p);
    }
    int i = 1;
    for(p = prim; p != null; p = p.prox) p.anterior.posicao = i++;

    for(p = prim; p != null; p = p.prox){
      Jogador maior = p, q;
      for(q = p.prox; q != null; q = q.prox)
        if(p.atual.npontos < q.atual.npontos) maior = q;
      if(maior != p) troca (maior, p);
    }
    i = 1;
    for(p = prim; p != null; p = p.prox) p.atual.posicao = i++;
```

```
  }

  public String toString(){
    String resp = "";
    int i = 1;
    Jogador p;
    for(p = prim; p != null; p = p.prox){
      resp += i++ + "\t" + p.toString();
      int aux = p.atual.posicao - p.anterior.posicao;
      if(aux == 0) resp += "inalterada posicao\n";
      else if(aux > 0) resp += "desceu " + aux + " posicao\n";
      else resp += "subiu " +  Math.abs(aux) + " posicao\n";
    }
    return resp;
  }
}


public class AP3_2016_1_Q2{
  public static void main(String[] args) throws IOException {
    BufferedReader in = new BufferedReader(new FileReader(args[0]));
    try{
      String linha;
      ATP tenis = new ATP();
      String[] partes;
      int j;
      linha = in.readLine();
      while(linha != null){
        partes = linha.split("/");
        Jogador jog = new Jogador(partes[0]);
        for(j = 1; j <= 4; j++){
          String pos[] = partes[j].split(" ");
          int k, pontos[] = new int [2];


          for(k = 0; k < 2; k++){
            if(pos[k + 1].equals("WIN")) pontos[k] = 2000;
            else if(pos[k + 1].equals("FIN")) pontos[k] = 1200;
            else if(pos[k + 1].equals("SF")) pontos[k] = 720;
            else if(pos[k + 1].equals("QF")) pontos[k] = 360;
            else if(pos[k + 1].equals("OF")) pontos[k] = 180;
            else if(pos[k + 1].equals("TR")) pontos[k] = 90;
            else if(pos[k + 1].equals("SR")) pontos[k] = 45;
            else pontos[k] = 10;
          }
          jog.incluiPontos(pontos[0], false);
          jog.incluiPontos(pontos[1], true);
        }
        tenis.inclui(jog);
        linha = in.readLine();
      }
      in.close();
      tenis.ordena();
      System.out.println(tenis);
    }catch(Exception e){ System.out.println(e); }
  }
}
```