Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP2 1° semestre de 2009.

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)
Considere a Questão 4 da AP1, a qual pedia a construção de uma classe para manipulação de endereços www. Uma versão simplificada desta classe se encontra à seguir.

```
class EnderecoWWW {
      private String dominio;
      private int porta;
      private String recurso;
      public EnderecoWWW(String dom, int port, String rec) {
            dominio = dom;
            porta = port;
            recurso = rec;
      }
      public String getDominio() {
            return dominio;
      }
      public void setDominio(String dominio) {
            this.dominio = dominio;
      }
      public int getPorta() {
            return porta;
      }
      public void setPorta(int porta) {
            this.porta = porta;
      }
      public String getRecurso() {
```

```
            return recurso;
      }
      public void setRecurso(String recurso) {
            this.recurso = recurso;
      }
}
```

Suponha que agora desejamos ampliar nosso sistema para manipular favoritos (*bookmark* - conjunto de endereços www de nosso interesse). Cada favorito deverá ser classificado em um ou mais assuntos. Por exemplo, o favorito www.cederj.edu.br poderia estar classificado como "educação", mas também como "ead" e "ensino". Sem alterar a classe EnderecoWWW, crie 1 ou mais classes de forma que isso seja possível, assim como: a) adição de favoritos; b) remoção de favoritos; c) busca e impressão de favoritos que satisfaçam uma dada classificação.

RESPOSTA:

```
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

class Favoritos {
      private Set<Favorito> favs;

      public Favoritos() {
            favs = new HashSet<Favorito>();
      }

      // item a) da questão
      public void adicionaFavorito (Favorito fav) {
            favs.add(fav);
      }

      // item b) da questão
      public void removeFavorito (String nome) {
            Iterator<Favorito> it = favs.iterator();
            while (it.hasNext()) {
                  Favorito fav = (Favorito)it.next();
                  if (fav.getNome().equals(nome)) {
                        favs.remove(fav);
                        break;
                  }
            }
      }

      // item c) da questão
      public Set<Favorito> buscaFavoritosPorAssunto (String assunto) {
            Set<Favorito> fav_assunto = new HashSet<Favorito>();
            Iterator<Favorito> it = favs.iterator();
            while (it.hasNext()) {
                  Favorito fav = (Favorito)it.next();
                  if (fav.getAssuntos().contains(assunto)) {
                        fav_assunto.add(fav);
                  }
```

```
            }
            return fav_assunto;
      }

      // método não solicitado na questão
      public void imprimeFavoritosPorAssunto (String assunto) {
            Set<Favorito> fav_assunto =
this.buscaFavoritosPorAssunto(assunto);
            Iterator<Favorito> it = fav_assunto.iterator();
            while (it.hasNext()) {
                  System.out.println("Nome favorito: " +
((Favorito)it.next()).getNome());
            }
      }
}

// Classe necessária para se modelar um favorito
// Observem que os métodos getXXX() e setXXX() são declarados para
//encapsularmos os campos da classe. Entretanto, uma solução mais
//enxuta poderia ser dada colocando estes campos como públicos.
class Favorito {
      private String nome;
      private EnderecoWWW url;
      private Set<String> assuntos;

      public Favorito (String n, EnderecoWWW ender, Set<String>
assuntos) {
            this.nome = n;
            this.url = ender;
            this.assuntos = assuntos;
      }

      public Favorito (String n, EnderecoWWW ender, String assunto) {
            this.nome = n;
            this.url = ender;
            this.assuntos = new HashSet<String>();
            this.assuntos.add(assunto);
      }

      public void adicionaAssunto (String assunto) {
            this.assuntos.add(assunto);
      }

      public void removeAssunto (String assunto) {
            this.assuntos.remove(assunto);
      }

      public String getNome() {
            return nome;
      }

      public void setNome(String nome) {
            this.nome = nome;
      }

      public EnderecoWWW getUrl() {
            return url;
```

```
	}

	public void setUrl(EnderecoWWW url) {
		this.url = url;
	}

	public Set<String> getAssuntos() {
		return assuntos;
	}

	public void setAssuntos(Set<String> assuntos) {
		this.assuntos = assuntos;
	}
}

// Classe criada apenas para testar as classes apresentadas, não
//solicitada na questão
public class AP2_2009_1_Q1 {
	public static void main(String[] args) {
		String www = "http://www.cederj.edu.br/vest";
		EnderecoWWW cederj = new EnderecoWWW(www);
		cederj.exibe();

		Favoritos bookmark = new Favoritos();
		Favorito fav = new Favorito("cederj", cederj, "ensino");
		fav.adicionaAssunto("ead");
		fav.adicionaAssunto("educação");
		bookmark.adicionaFavorito(fav);
		fav = new Favorito("sitepessoal", new
EnderecoWWW("http://www.ic.uff.br/~bazilio"), "ensino");
		fav.adicionaAssunto("educação");
		bookmark.adicionaFavorito(fav);
		bookmark.imprimeFavoritosPorAssunto("ensino");
}
```

Questão 2) (2.5 pontos)
Considere o código abaixo, o qual cria uma janela com 3 botões:

```
import java.awt.Container;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

class JanelaJogo extends JFrame {
	JButton bt1, bt2, bt3;
	public JanelaJogo() {
		// Define tamanho padrão da janela
		this.setSize(600, 200);
		// Atribui título da janela
		this.setTitle("Escolha o botão correto!");
		// Termina o processo no fechamando da janela
		this.setDefaultCloseOperation(EXIT_ON_CLOSE);
		// Adiciona botões à janela
		Container contentPane = this.getContentPane();
```

```
            this.getContentPane().setLayout(new GridLayout(1,3));
            bt1 = new JButton("?");
            bt2 = new JButton("?");
            bt3 = new JButton("?");
            contentPane.add(bt1);
            contentPane.add(bt2);
            contentPane.add(bt3);
      }
}
public class AP2_2009_1_Q2 {
      public static void main(String[] args) {
            JanelaJogo edt = new JanelaJogo();
            edt.setVisible(true);
      }
}
```

Suponha que queremos criar um jogo de advinhação. Para tal, o programa escolherá aleatoriamente algum botão e o usuário tem que advinhar qual é o botão clicando neste. Caso o usuário clique no botão correto, o sistema deve abrir uma caixa de diálogo informando que o usuário teve sucesso. Caso contrário, o sistema deve exibir outra caixa de diálogo informando que a escolha foi incorreta. Altere o código fornecido para implementar tal sistema.

RESPOSTA:

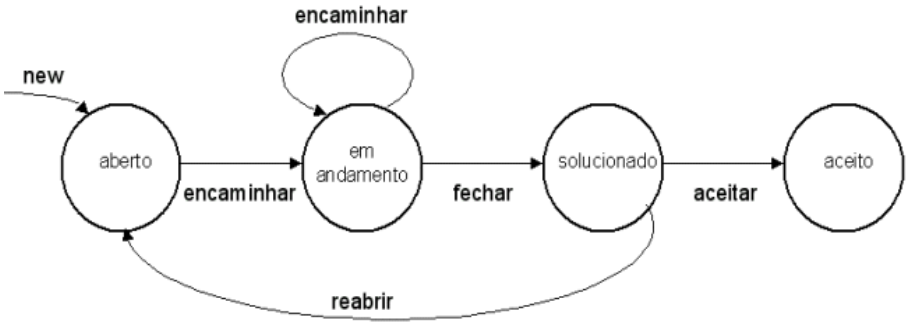
Em *itálico* está o que foi necessário incluir no código fornecido acima. Em essência, foi necessário tratar os eventos de clique nos botões para refletir as regras do jogo solicitado.

```
import java.awt.Container;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

class JanelaJogo extends JFrame implements ActionListener {
      JButton bt1, bt2, bt3;
      public JanelaJogo() {
            // Define tamanho padrão da janela
            this.setSize(600, 200);
            // Atribui título da janela
            this.setTitle("Escolha o botão correto!");
            // Termina o processo no fechamando da janela
            this.setDefaultCloseOperation(EXIT_ON_CLOSE);

            // Adiciona botões à janela
            Container contentPane = this.getContentPane();
            this.getContentPane().setLayout(new GridLayout(1,3));
            bt1 = new JButton("?");
            bt2 = new JButton("?");
            bt3 = new JButton("?");
            contentPane.add(bt1);
```

```
            contentPane.add(bt2);
            contentPane.add(bt3);

            bt1.addActionListener(this);
            bt2.addActionListener(this);
            bt3.addActionListener(this);
      }

      public void actionPerformed(ActionEvent arg0) {
            int botaoSorteado = this.sorteiaBotao();
            if ((((JButton)arg0.getSource()) == bt1 && botaoSorteado ==
1) ||
                ((JButton)arg0.getSource()) == bt2 && botaoSorteado ==
2) ||
                ((JButton)arg0.getSource()) == bt3 && botaoSorteado ==
3)) {
                  setaAcerto((JButton)arg0.getSource(), botaoSorteado);
                  JOptionPane.showMessageDialog(this, "!!!
Acertou !!!");
            }
            else {
                  this.setaValorBotoes("X");
                  JOptionPane.showMessageDialog(this, "Errou! Tente
novamente !!");
            }
            this.setaValorBotoes("?");
      }

      public int sorteiaBotao() {
            return ((int)(Math.random()*100) % 3) + 1;
      }

      public void setaAcerto (JButton botao, int valor) {
            this.setaValorBotoes("X");
            botao.setText(String.valueOf(valor));
      }

      public void setaValorBotoes (String v) {
            bt1.setText(v);
            bt2.setText(v);
            bt3.setText(v);
      }
}

public class AP2_2009_1_Q2 {
      public static void main(String[] args) {
            JanelaJogo edt = new JanelaJogo();
            edt.setVisible(true);
      }
}
```

Questão 3) (2.5 pontos)

Uma aplicação de gerência de chamadas de suporte utiliza tickets que são abertos, encaminhados, verificados e fechados, de acordo com o seguinte diagrama de estados:



Um ticket é aberto na sua criação e, partir daí, as ações podem ocorrer de acordo com os estados acima. A ação de encaminhar tem como parâmetro o nome do responsável pela solução do ticket.

Sempre que uma ação inválida é efetuada em um ticket (por exemplo, um ticket somente pode ser aceito se ele já tiver sido solucionado), uma exceção é lançada indicando qual ação inválida foi efetuada.

Projete e construa as classes para essa questão. Implemente também um programa de teste que crie um ticket e execute diversas ações sobre ele, mostrando como está o ticket entre cada ação solicitada. Um possível caso de teste poderia ser:

```
    try {
      Ticket t = new Ticket();
      System.out.println (t);
      t.encaminhar ("Joao");
      System.out.println (t);
      t.encaminhar ("José");
      System.out.println (t);
      t.fechar ();
      System.out.println (t);
      t.reabrir();
      System.out.println (t);
      t.encaminhar ("Maria");
      System.out.println (t);
      t.fechar();
      System.out.println (t);
      t.aceitar();
      System.out.println (t);
      t.fechar();
      System.out.println (t);
    } catch (AcaoInvalidaException e) {
      System.out.println (e);}
```

RESPOSTA:

```
public class Ticket {

  private State state;
  private String responsavel;

  protected abstract class State {
    void encaminhar(String pessoa)
```

```
  throws AcaoInvalidaException {
    throw new AcaoInvalidaException(state,"encaminhar");
  }

  void fechar()
  throws AcaoInvalidaException {
    throw new AcaoInvalidaException(state,"fechar");
  }

  void aceitar()
  throws AcaoInvalidaException {
    throw new AcaoInvalidaException(state,"aceitar");
  }

  void reabrir()
  throws AcaoInvalidaException {
    throw new AcaoInvalidaException(state,"reabrir");
  }
}

private class Aberto extends State {
  void encaminhar(String pessoa)
  throws AcaoInvalidaException {
    responsavel = pessoa;
    state = new EmAndamento();
  }
  public String toString() {
    return "Aberto";
  }
}

private class EmAndamento extends State {
  void encaminhar(String pessoa)
  throws AcaoInvalidaException {
    responsavel = pessoa;
  }
  void fechar()
  throws AcaoInvalidaException {
    state = new Solucionado();
  }
  public String toString() {
    return "Em andamento, pelo responsável "+responsavel;
  }
}

private class Aceito extends State {
  public String toString() {
    return "Aceito";
  }
}

private class Solucionado extends State {
  void aceitar()
  throws AcaoInvalidaException {
    state = new Aceito();
  }
  void reabrir()
```

```
  throws AcaoInvalidaException {
    responsavel = null;
    state = new Aberto();
  }
  void fechar()
  throws AcaoInvalidaException {
    state = new Solucionado();
  }
  public String toString() {
    return "Solucionado";
  }
}

public Ticket() {
  state = new Aberto();
}

public void encaminhar(String pessoa)
throws AcaoInvalidaException {
  state.encaminhar(pessoa);
}

public void fechar()
throws AcaoInvalidaException {
  state.fechar();
}

public void aceitar()
throws AcaoInvalidaException {
  state.aceitar();
}

public void reabrir()
throws AcaoInvalidaException {
  state.reabrir();
}

public String toString() {
  return state.toString();
}

public static void main (String[] args) {
  try {
    Ticket t = new Ticket();
    System.out.println (t);
    t.encaminhar ("Joao");
    System.out.println (t);
    t.encaminhar ("José");
    System.out.println (t);
    t.fechar ();
    System.out.println (t);
    t.reabrir();
    System.out.println (t);
    t.encaminhar ("Maria");
    System.out.println (t);
    t.fechar();
    System.out.println (t);
```

```
      t.aceitar();
      System.out.println (t);
      t.fechar();
      System.out.println (t);
    } catch (AcaoInvalidaException e) {
      System.out.println (e);
    }
  }
}

class AcaoInvalidaException extends Exception {
  AcaoInvalidaException (Ticket.State from, String acao) {
     super("Transicao invalida: " + from + " nao permite a acao " +
acao);
  }
  AcaoInvalidaException (String msg) {
    super(msg);
  }
}
```

Questão 4) (2.5 pontos)
Escreva um programa que receba, como parâmetro de entrada, o nome de um arquivo texto, cujo conteúdo é composto dos seguintes campos, separados por "**/**":

- nome do cliente,
- código da conta,
- saldo e
- descrição da conta,

que ordene as informações, em ordem decrescente de saldo do cliente, e que grave as informações ordenadas no arquivo **`resultado.txt`**, com o seguinte formato: **`<descrição> = <nome> (<código>): <saldo>`**. Um exemplo de uso desse programa seria **`java ordena contas.txt`**, onde **`contas.txt`** é o nome do arquivo de entrada.

RESPOSTA:

```
import java.io.*;

class Dados{
  private String nome;
  private int codigo;
  private float saldo;
  private String desc;

  Dados (String n, int c, float s, String d){
    nome = n;
    codigo = c;
    saldo = s;
    desc = d;
  }

  float retornaSaldo( ){
    return saldo;
```

```
  }

  String imprimeDados ( ){
    return desc + " = " + nome + " (" + codigo + "): " + saldo + "\n";
  }
}

public class ordena {
  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 {
      Dados vet_dados[] = new Dados[n];
      int cont = 0;
      String vs[], nome, desc;
      float saldo;
      int i, cod;

      in.close();
      in = new BufferedReader(new FileReader(args[0]));
      System.out.println(n);

      while(cont < n) {
          s = in.readLine();
        vs = s.split("/");
          nome = vs[0];
          cod = Integer.parseInt(vs[1]);
          saldo = Float.parseFloat (vs[2]);
          desc = vs[3];

          vet_dados[cont++] = new Dados (nome, cod, saldo, desc);
      }

      BufferedWriter out = new BufferedWriter(new FileWriter
("resultado.txt"));
      ordenaSaldo (vet_dados);

      for (i = 0; i < cont; i++) {
          out.write(vet_dados[i].imprimeDados());
      }
      out.close();
   }
   catch (Exception e){
     System.out.println("Excecao2\n");
   }
   finally{
     in.close();
   }
```

```
 }

 public static void ordenaSaldo (Dados[] vet){
   int i, j, maior;
   Dados temp;

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

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

      temp = vet[maior];
      vet[maior] = vet[i];
      vet[i] = temp;
   }
 }
}
```