Instituto Politécnico de Setúbal

Escola Superior de Tecnologia de Setúbal

Departamento de Sistemas e Informática

Aulas Práticas
GUI e Applets

Computação na Internet
Ano Lectivo 2002/2003

Realizado por: João Ascenso e António Gonçalves

1. **Escreva uma applet que mostre uma label com a palavra “Applet”. Utilize a classe Console do Apêndice A.**

```
package textArea;
import javax.swing.*;
import java.awt.*;
import com.bruceeckel.swing.*;

public class Applet1d extends JApplet {
  public void init() {
    getContentPane().add(new JLabel("Applet!"));
  }
  public static void main(String[] args) {
    Console.run(new Applet1d(), 100, 50);
  }
} ///:~
```

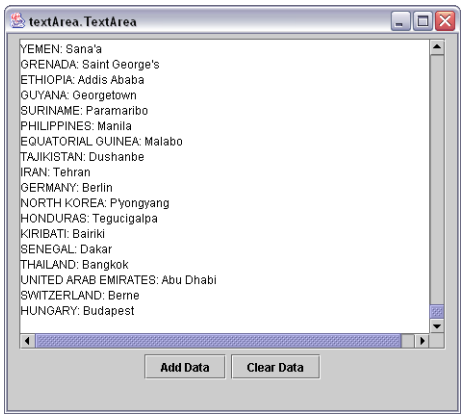
2. **Faça uma applet que mostre dois botões. Utilize o flow layout como layout manager.**

```
package textArea;

import javax.swing.*;
import java.awt.*;
import com.bruceeckel.swing.*;

public class Button1 extends JApplet {
  JButton
    b1 = new JButton("Button 1"),
    b2 = new JButton("Button 2");
  public void init() {
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());
    cp.add(b1);
    cp.add(b2);
  }
  public static void main(String[] args) {
    Console.run(new Button1(), 200, 50);
  }
}
```

3. **Escreva uma applet que mostre dois botões, e quando um deles for pressionado escreva num campo de texto (JTextField) o nome deles.**


```
package textArea;

import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import com.bruceeckel.swing.*;

public class Button2b extends JApplet {
  JButton
    b1 = new JButton("Button 1"),
    b2 = new JButton("Button 2");
  JTextField txt = new JTextField(10);
  ActionListener al = new ActionListener() {
    public void actionPerformed(ActionEvent e){
      String name =
        ((JButton)e.getSource()).getText();
      txt.setText(name);
    }
  };
  public void init() {
    b1.addActionListener(al);
    b2.addActionListener(al);
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());
    cp.add(b1);
    cp.add(b2);
    cp.add(txt);
  }
  public static void main(String[] args) {
    Console.run(new Button2b(), 200, 75);
  }
}
```

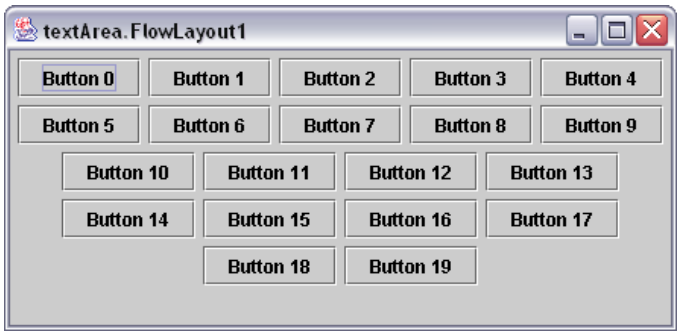
4. **Escreva uma applet que mostre dois botões, um para adicionar dados a uma área de texto (JTextArea) e outro para limpar a mesma área. Utilize a classe Collections2 do Apêndice A para gerar os dados.**




```
package textArea;

import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.util.*;
import com.bruceeckel.swing.*;
import com.bruceeckel.util.*;

public class TextArea extends JApplet {
  JButton
    b = new JButton("Add Data"),
    c = new JButton("Clear Data");
  JTextArea t = new JTextArea(20, 40);
  Map m = new HashMap();
  public void init() {
    // Use up all the data:
    Collections2.fill(m,
      Collections2.geography,
      CountryCapitals.pairs.length);
    b.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e){
        for(Iterator it= m.entrySet().iterator();
            it.hasNext();){
          Map.Entry me = (Map.Entry)(it.next());
          t.append(me.getKey() + ": "
            + me.getValue() + "\n");
        }
      }
    });
    c.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e){
        t.setText("");
      }
    });
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());
    cp.add(new JScrollPane(t));
    cp.add(b);
    cp.add(c);
  }
  public static void main(String[] args) {
    Console.run(new TextArea(), 475, 425);
  }

}
```

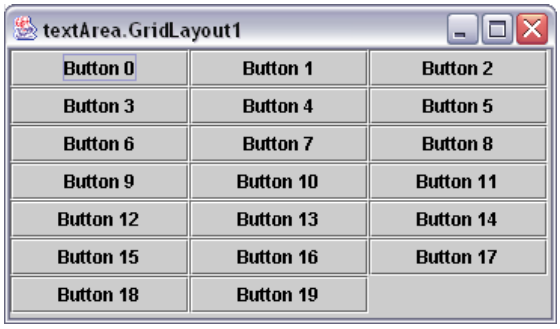
5. **Escreva uma applet que mostre 20 botões, utilizando os seguintes layout managers. Altere a dimensão da janela e explique a forma como os botões são reorganizados:**

a. FlowLayout



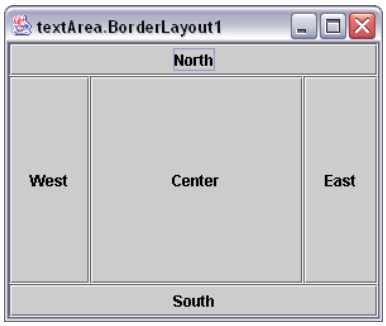

```
package textArea;

import javax.swing.*;
import java.awt.*;
import com.bruceeckel.swing.*;

public class FlowLayout1 extends JApplet {
  public void init() {
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());
    for(int i = 0; i < 20; i++)
      cp.add(new JButton("Button " + i));
  }
  public static void main(String[] args) {
    Console.run(new FlowLayout1(), 300, 250);
  }
} ///:~
```

b. GridLayout




```
package textArea;

import javax.swing.*;
import java.awt.*;
import com.bruceeckel.swing.*;

public class GridLayout1 extends JApplet {
  public void init() {
    Container cp = getContentPane();
    cp.setLayout(new GridLayout(7,3));
    for(int i = 0; i < 20; i++)
      cp.add(new JButton("Button " + i));
  }
  public static void main(String[] args) {
    Console.run(new GridLayout1(), 300, 250);
  }
} ///:~
```

c. BorderLayout (apenas 4 botões)



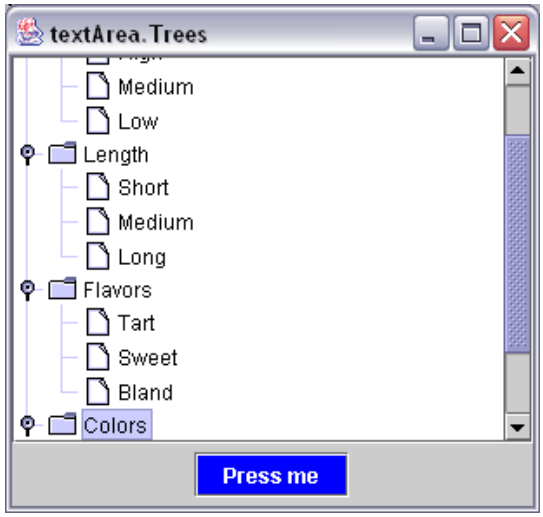

```
package textArea;

import javax.swing.*;
import java.awt.*;
import com.bruceeckel.swing.*;

public class BorderLayout1 extends JApplet {
  public void init() {
    Container cp = getContentPane();
    cp.add(BorderLayout.NORTH,
      new JButton("North"));
    cp.add(BorderLayout.SOUTH,
      new JButton("South"));
    cp.add(BorderLayout.EAST,
      new JButton("East"));
    cp.add(BorderLayout.WEST,
      new JButton("West"));
    cp.add(BorderLayout.CENTER,
      new JButton("Center"));
  }
  public static void main(String[] args) {
    Console.run(new BorderLayout1(), 300, 250);
  }
} ///:~
```

6. Escreva uma applet que mostre um menu com 3 items: Winken, Blinken e Nod. Cada um dos menus deve conter três items. Quando um dos items for seleccionado, a opção deve ser vizualizada numa área de texto (JTextField)


```
package textArea;

import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import com.bruceeckel.swing.*;

public class SimpleMenus extends JApplet {
  JTextField t = new JTextField(15);
  ActionListener al = new ActionListener() {
    public void actionPerformed(ActionEvent e){
      t.setText(
        ((JMenuItem)e.getSource()).getText());
    }
```

```
  };
  JMenu[] menus = { new JMenu("Winken"),
    new JMenu("Blinken"), new JMenu("Nod") };
  JMenuItem[] items = {
    new JMenuItem("Fee"), new JMenuItem("Fi"),
    new JMenuItem("Fo"),  new JMenuItem("Zip"),
    new JMenuItem("Zap"), new JMenuItem("Zot"),
    new JMenuItem("Olly"), new JMenuItem("Oxen"),
    new JMenuItem("Free") };
  public void init() {
    for(int i = 0; i < items.length; i++) {
      items[i].addActionListener(al);
      menus[i%3].add(items[i]);
    }
    JMenuBar mb = new JMenuBar();
    for(int i = 0; i < menus.length; i++)
      mb.add(menus[i]);
    setJMenuBar(mb);
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());
    cp.add(t);
  }
  public static void main(String[] args) {
    Console.run(new SimpleMenus(), 200, 75);
  }
} ///:~
```

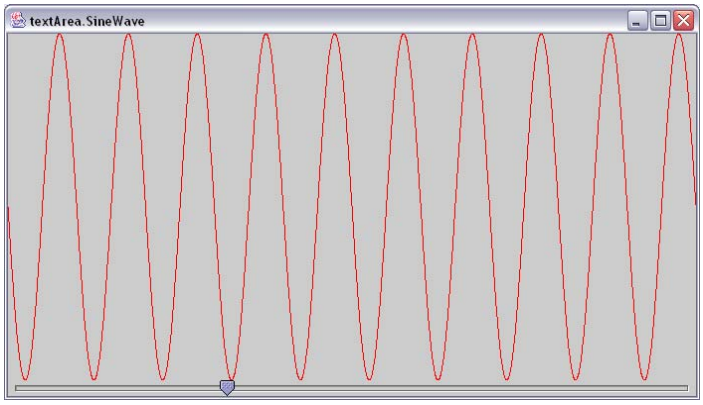
7. **Escreva uma applet que permita uma árvore (JTree) de uma forma dinâmica. Quando se pressionar o botão "Press me" a aplicação deve adicionar um ramo à árvore existente no nó seleccionado.**




```
package textArea;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.tree.*;
import com.bruceeckel.swing.*;

// Takes an array of Strings and makes the first
// element a node and the rest leaves:
class Branch {
  DefaultMutableTreeNode r;
  public Branch(String[] data) {
    r = new DefaultMutableTreeNode(data[0]);
    for(int i = 1; i < data.length; i++)
      r.add(new DefaultMutableTreeNode(data[i]));
  }
  public DefaultMutableTreeNode node() {
    return r;
  }
}

public class Trees extends JApplet {
  String[][] data = {
    { "Colors", "Red", "Blue", "Green" },
    { "Flavors", "Tart", "Sweet", "Bland" },
    { "Length", "Short", "Medium", "Long" },
    { "Volume", "High", "Medium", "Low" },
    { "Temperature", "High", "Medium", "Low" },
    { "Intensity", "High", "Medium", "Low" },
  };
  static int i = 0;
  DefaultMutableTreeNode root, child, chosen;
  JTree tree;
  DefaultTreeModel model;
  public void init() {
    Container cp = getContentPane();
    root = new DefaultMutableTreeNode("root");
    tree = new JTree(root);
    // Add it and make it take care of scrolling:
    cp.add(new JScrollPane(tree),
      BorderLayout.CENTER);
    // Capture the tree's model:
    model =(DefaultTreeModel)tree.getModel();
```

```
    JButton test = new JButton("Press me");
    test.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e){
        if(i < data.length) {
          child = new Branch(data[i++]).node();
          // What's the last one you clicked?
          chosen = (DefaultMutableTreeNode)
            tree.getLastSelectedPathComponent();
          if(chosen == null) chosen = root;
          // The model will create the
          // appropriate event. In response, the
          // tree will update itself:
          model.insertNodeInto(child, chosen, 0);
          // This puts the new node on the
          // currently chosen node.
        }
      }
    });
    // Change the button's colors:
    test.setBackground(Color.blue);
    test.setForeground(Color.white);
    JPanel p = new JPanel();
    p.add(test);
    cp.add(p, BorderLayout.SOUTH);
  }
  public static void main(String[] args) {
    Console.run(new Trees(), 250, 250);
  }
} ///:~
```

8. Escreva uma aplicação que desenhe uma onda sinusoidal num painel (JPane). Também deve conter um slider (JSlider) para que o utilizador escolha o comprimento de onda desejado.




```
package textArea;

import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import com.bruceeckel.swing.*;

class SineDraw extends JPanel {
  static final int SCALEFACTOR = 200;
  int cycles;
  int points;
  double[] sines;
  int[] pts;
  SineDraw() { setCycles(5); }
  public void setCycles(int newCycles) {
    cycles = newCycles;
    points = SCALEFACTOR * cycles * 2;
    sines = new double[points];
    pts = new int[points];
    for(int i = 0; i < points; i++) {
      double radians = (Math.PI/SCALEFACTOR) * i;
```

```
      sines[i] = Math.sin(radians);
    }
    repaint();
  }
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    int maxWidth = getWidth();
    double hstep = (double)maxWidth/(double)points;
    int maxHeight = getHeight();
    for(int i = 0; i < points; i++)
      pts[i] = (int)(sines[i] * maxHeight/2 + maxHeight/2);
    g.setColor(Color.red);
    for(int i = 1; i < points; i++) {
      int x1 = (int)((i - 1) * hstep);
      int x2 = (int)(i * hstep);
      int y1 = pts[i-1];
      int y2 = pts[i];
      g.drawLine(x1, y1, x2, y2);
    }
  }
}

public class SineWave extends JApplet {
  SineDraw sines = new SineDraw();
  JSlider cycles = new JSlider(1, 30, 5);
  public void init() {
    Container cp = getContentPane();
    cp.add(sines);
    cycles.addChangeListener(new ChangeListener(){
      public void stateChanged(ChangeEvent e) {
        sines.setCycles(
          ((JSlider)e.getSource()).getValue());
      }
    });
    cp.add(BorderLayout.SOUTH, cycles);
  }
  public static void main(String[] args) {
    Console.run(new SineWave(), 700, 400);
  }
} ///:~
```

9. Escreva uma aplicação que mediante a selecção de um botão “open” ou “save” e faça o display da correspondente dialog para escolha de ficheiros.

```
package textArea;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import com.bruceeckel.swing.*;

public class FileChooserTest extends JFrame {
  JTextField
    filename = new JTextField(),
    dir = new JTextField();
  JButton
    open = new JButton("Open"),
    save = new JButton("Save");
  public FileChooserTest() {
    JPanel p = new JPanel();
    open.addActionListener(new OpenL());
    p.add(open);
    save.addActionListener(new SaveL());
    p.add(save);
    Container cp = getContentPane();
    cp.add(p, BorderLayout.SOUTH);
    dir.setEditable(false);
    filename.setEditable(false);
```

```
    p = new JPanel();
    p.setLayout(new GridLayout(2,1));
    p.add(filename);
    p.add(dir);
    cp.add(p, BorderLayout.NORTH);
  }
  class OpenL implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      JFileChooser c = new JFileChooser();
      // Demonstrate "Open" dialog:
      int rVal =
        c.showOpenDialog(FileChooserTest.this);
      if(rVal == JFileChooser.APPROVE_OPTION) {
        filename.setText(
          c.getSelectedFile().getName());
          dir.setText(
            c.getCurrentDirectory().toString());
      }
      if(rVal == JFileChooser.CANCEL_OPTION) {
        filename.setText("You pressed cancel");
        dir.setText("");
      }
    }
  }
  class SaveL implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      JFileChooser c = new JFileChooser();
      // Demonstrate "Save" dialog:
      int rVal =
        c.showSaveDialog(FileChooserTest.this);
      if(rVal == JFileChooser.APPROVE_OPTION) {
        filename.setText(
          c.getSelectedFile().getName());
          dir.setText(
            c.getCurrentDirectory().toString());
      }
      if(rVal == JFileChooser.CANCEL_OPTION) {
        filename.setText("You pressed cancel");
        dir.setText("");
      }
    }
  }
  public static void main(String[] args) {
    Console.run(new FileChooserTest(), 250, 110);
  }
} ///:~
```

10. Escreva uma aplicação que implemente as funcionalidades “copy”, “cut” e “paste”. Devem ser criado um menu, e uma área de texto para esse efeito.

```
package textArea;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.datatransfer.*;
import com.bruceeckel.swing.*;

public class CutAndPaste extends JFrame  {
  JMenuBar mb = new JMenuBar();
  JMenu edit = new JMenu("Edit");
  JMenuItem
    cut = new JMenuItem("Cut"),
    copy = new JMenuItem("Copy"),
    paste = new JMenuItem("Paste");
  JTextArea text = new JTextArea(20, 20);
  Clipboard clipbd =
    getToolkit().getSystemClipboard();
```

```
  public CutAndPaste()  {
    cut.addActionListener(new CutL());
    copy.addActionListener(new CopyL());
    paste.addActionListener(new PasteL());
    edit.add(cut);
    edit.add(copy);
    edit.add(paste);
    mb.add(edit);
    setJMenuBar(mb);
    getContentPane().add(text);
  }
  class CopyL implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      String selection = text.getSelectedText();
      if (selection == null)
        return;
      StringSelection clipString =
        new StringSelection(selection);
      clipbd.setContents(clipString,clipString);
    }
  }
  class CutL implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      String selection = text.getSelectedText();
      if (selection == null)
        return;
      StringSelection clipString =
        new StringSelection(selection);
      clipbd.setContents(clipString, clipString);
      text.replaceRange("",
        text.getSelectionStart(),
        text.getSelectionEnd());
    }
  }
  class PasteL implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      Transferable clipData =
        clipbd.getContents(CutAndPaste.this);
      try {
        String clipString =
          (String)clipData.
            getTransferData(
              DataFlavor.stringFlavor);
        text.replaceRange(clipString,
          text.getSelectionStart(),
          text.getSelectionEnd());
      } catch(Exception ex) {
        System.err.println("Not String flavor");
      }
    }
  }
  public static void main(String[] args) {
    Console.run(new CutAndPaste(), 300, 200);
  }
} ///:~
```

Servlets

1. Escreva uma servlet que gere texto.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.println("Hello World");
  }
}
```

2. Escreva uma servlet que gere HTML.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String docType =
      "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
      "Transitional//EN\">\n";
    out.println(docType +
                "<HTML>\n" +
                "<HEAD><TITLE>Hello</TITLE></HEAD>\n" +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1>Hello</H1>\n" +
                "</BODY></HTML>");
  }
}
```

3. Escreva uma servlet que leia dois parâmetro de incialização, do ficheiro de configuração web.xml (Tomcat). O primeiro parâmetro deve conter um nome e o segundo um endereço de e-mail.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class InitServlet extends HttpServlet {
  private String firstName, emailAddress;

  public void init() {
    ServletConfig config = getServletConfig();
    firstName = config.getInitParameter("firstName");
    emailAddress = config.getInitParameter("emailAddress");
  }

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
```

```
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String uri = request.getRequestURI();
    out.println(ServletUtilities.headWithTitle("Init Servlet") +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H2>Init Parameters:</H2>\n" +
                "<UL>\n" +
                "<LI>First name: " + firstName + "\n" +
                "<LI>Email address: " + emailAddress + "\n" +
                "</UL>\n" +
                "</BODY></HTML>");
  }
}
```

WEB.XML

```
<web-app>
  <servlet>
    <servlet-name>InitTest</servlet-name>
    <servlet-class>InitServlet</servlet-class>
    <init-param>
      <param-name>firstName</param-name>
      <param-value>Larry</param-value>
    </init-param>
    <init-param>
      <param-name>emailAddress</param-name>
      <param-value>ellison@microsoft.com</param-value>
    </init-param>
  </servlet>
</web-app>
```

4. Escreva uma servlet que leia três parâmetros introduzidos pelo utilizador de uma página da Web (ver figura).

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ThreeParams extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Reading Three Request Parameters";
    out.println(ServletUtilities.headWithTitle(title) +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
                "<UL>\n" +
                "  <LI><B>param1</B>: "
                + request.getParameter("param1") + "\n" +
                "  <LI><B>param2</B>: "
                + request.getParameter("param2") + "\n" +
                "  <LI><B>param3</B>: "
                + request.getParameter("param3") + "\n" +
                "</UL>\n" +
                "</BODY></HTML>");
  }
}
```

HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Collecting Three Parameters</TITLE>
</HEAD>
<BODY BGCOLOR="#FDF5E6">
<H1 ALIGN="CENTER">Collecting Three Parameters</H1>
```

```
<FORM ACTION="/servlet/moreservlets.ThreeParams">
  First Parameter:  <INPUT TYPE="TEXT" NAME="param1"><BR>
  Second Parameter: <INPUT TYPE="TEXT" NAME="param2"><BR>
  Third Parameter:  <INPUT TYPE="TEXT" NAME="param3"><BR>
  <CENTER><INPUT TYPE="SUBMIT"></CENTER>
</FORM>

</BODY>
</HTML>
```

5. Escreva uma servlet que leia um conjunto indeterminado de parâmetros de uma página Web. Deve permitir ambos os métodos POST e GET por parte do cliente.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class ShowParameters extends HttpServlet {

   public void doGet(HttpServletRequest request,
                                 HttpServletResponse response)
                             throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String title = "Reading All Request Parameters";
      out.println(ServletUtilities.headWithTitle(title) +
                        "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                        "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
                        "<TABLE BORDER=1 ALIGN=CENTER>\n" +
                        "<TR BGCOLOR=\"#FFAD00\">\n" +
                        "<TH>Parameter Name<TH>Parameter Value(s)");
      Enumeration paramNames = request.getParameterNames();
      while(paramNames.hasMoreElements()) {
            String paramName = (String)paramNames.nextElement();
            out.print("<TR><TD>" + paramName + "\n<TD>");
            String[] paramValues =
            request.getParameterValues(paramName);
            if (paramValues.length == 1) {
                  String paramValue = paramValues[0];
                  if (paramValue.length() == 0)
                        out.println("<I>No Value</I>");
                  else
                        out.println(paramValue);
            } else {
                  out.println("<UL>");
                  for(int i=0; i<paramValues.length; i++) {
                        out.println("<LI>" + paramValues[i]);
                  }
                  out.println("</UL>");
            }
      }
      out.println("</TABLE>\n</BODY></HTML>");
      }

      public void doPost(HttpServletRequest request,
                             HttpServletResponse response)
                             throws ServletException, IOException {
            doGet(request, response);
      }

   }
```

6. Escreva uma servlet que leia e imprima os cabeçalhos da informação HTTP que é enviada do cliente para o servidor.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
```

```
public class ShowRequestHeaders extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Servlet Example: Showing Request Headers";
    out.println(ServletUtilities.headWithTitle(title) +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
                "<B>Request Method: </B>" +
                request.getMethod() + "<BR>\n" +
                "<B>Request URI: </B>" +
                request.getRequestURI() + "<BR>\n" +
                "<B>Request Protocol: </B>" +
                request.getProtocol() + "<BR><BR>\n" +
                "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
                "<TR BGCOLOR=\"#FFAD00\">\n" +
                "<TH>Header Name<TH>Header Value");
    Enumeration headerNames = request.getHeaderNames();
    while(headerNames.hasMoreElements()) {
      String headerName = (String)headerNames.nextElement();
      out.println("<TR><TD>" + headerName);
      out.println("    <TD>" + request.getHeader(headerName));
    }
    out.println("</TABLE>\n</BODY></HTML>");
  }

  /** Let the same servlet handle both GET and POST. */

  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
      throws ServletException, IOException {
    doGet(request, response);
  }
}
```

7. Escreva uma servlet que só permite o acesso a uma página mediante a introdução de um login/password válido.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Properties;
import sun.misc.BASE64Decoder;

public class ProtectedPage extends HttpServlet {
      private Properties passwords;
      private String passwordFile;

      public void init(ServletConfig config)
                        throws ServletException {
            super.init(config);
            try {
                  passwordFile = config.getInitParameter("passwordFile");
                  passwords = new Properties();
                  passwords.load(new FileInputStream(passwordFile));
            } catch(IOException ioe) {}
      }

      public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                  throws ServletException, IOException {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            String authorization = request.getHeader("Authorization");
            if (authorization == null) {
                  askForPassword(response);
            } else {
                  String userInfo = authorization.substring(6).trim();
```

```
                    BASE64Decoder decoder = new BASE64Decoder();
                    String nameAndPassword =
                          new String(decoder.decodeBuffer(userInfo));
                    int index = nameAndPassword.indexOf(":");
                    String user = nameAndPassword.substring(0, index);
                    String password = nameAndPassword.substring(index+1);
                    String realPassword = passwords.getProperty(user);
                    if ((realPassword != null)
                          && (realPassword.equals(password))) {
                          String title = "Welcome to the Protected Page";
                          out.println(ServletUtilities.headWithTitle(title) +
                                      "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                                      "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
                                      "Congratulations. You have accessed a\n" +
                                      "highly proprietary company document.\n" +
                                      "Shred or eat all hardcopies before\n" +
                                      "going to bed tonight.\n" +
                                      "</BODY></HTML>");
                    } else {
                          askForPassword(response);
                    }
              }
        }

        private void askForPassword(HttpServletResponse response) {
              response.setStatus(response.SC_UNAUTHORIZED); // Ie 401
              response.setHeader("WWW-Authenticate",
              "BASIC realm=\"privileged-few\"");
        }

        public void doPost(HttpServletRequest request,
              HttpServletResponse response)
              throws ServletException, IOException {
              doGet(request, response);
        }

    }
```

8. Escreva uma servlet que envie seis cookies para o cliente. Três devem persistir apenas durante a sessão actual e três devem persistir durante uma hora, independentemente do estado do cliente Web.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SetCookies extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    for(int i=0; i<3; i++) {
      // Default maxAge is -1, indicating cookie
      // applies only to current browsing session.
      Cookie cookie = new Cookie("Session-Cookie-" + i,
                                 "Cookie-Value-S" + i);
      response.addCookie(cookie);
      cookie = new Cookie("Persistent-Cookie-" + i,
                          "Cookie-Value-P" + i);
      // Cookie is valid for an hour, regardless of whether
      // user quits browser, reboots computer, or whatever.
      cookie.setMaxAge(3600);
      response.addCookie(cookie);
    }
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Setting Cookies";
    out.println
      (ServletUtilities.headWithTitle(title) +
       "<BODY BGCOLOR=\"#FDF5E6\">\n" +
       "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
       "There are six cookies associated with this page.\n" +
```

```
        "To see them, visit the\n" +
        "<A HREF=\"/servlet/moreservlets.ShowCookies\">\n" +
        "<CODE>ShowCookies</CODE> servlet</A>.\n" +
        "<P>\n" +
        "Three of the cookies are associated only with the\n" +
        "current session, while three are persistent.\n" +
        "Quit the browser, restart, and return to the\n" +
        "<CODE>ShowCookies</CODE> servlet to verify that\n" +
        "the three long-lived ones persist across sessions.\n" +
        "</BODY></HTML>");
  }
}
```

9. Escreva uma servlet que leia os seis cookies enviados para o cliente (ver o exercício anterior).

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ShowCookies extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Active Cookies";
    out.println(ServletUtilities.headWithTitle(title) +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
                "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
                "<TR BGCOLOR=\"#FFAD00\">\n" +
                "  <TH>Cookie Name\n" +
                "  <TH>Cookie Value");
    Cookie[] cookies = request.getCookies();
    if (cookies == null) {
      out.println("<TR><TH COLSPAN=2>No cookies");
    } else {
      Cookie cookie;
      for(int i=0; i<cookies.length; i++) {
        cookie = cookies[i];
        out.println("<TR>\n" +
                    "  <TD>" + cookie.getName() + "\n" +
                    "  <TD>" + cookie.getValue());
      }
    }
    out.println("</TABLE></BODY></HTML>");
  }
}
```

10. Escreva uma classe que derive da classe Cookie e que consiste num cookie com a duração de um ano.

```
import javax.servlet.http.*;


public class LongLivedCookie extends Cookie {
  public static final int SECONDS_PER_YEAR = 60*60*24*365;

  public LongLivedCookie(String name, String value) {
    super(name, value);
    setMaxAge(SECONDS_PER_YEAR);
  }

}
```

11. Escreva uma servlet que mostre a seguinte informação sobre uma sessão iniciado pelo cliente a) Identificador da Sessão b) Data/Hora de criação da sessão c) Última vez que acedeu d) Número de acessos.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.net.*;
import java.util.*;

public class ShowSession extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Session Tracking Example";
    HttpSession session = request.getSession(true);
    String heading;
    Integer accessCount =
      (Integer)session.getAttribute("accessCount");
    if (accessCount == null) {
      accessCount = new Integer(0);
      heading = "Welcome, Newcomer";
    } else {
      heading = "Welcome Back";
      accessCount = new Integer(accessCount.intValue() + 1);
    }
    session.setAttribute("accessCount", accessCount);
    out.println(ServletUtilities.headWithTitle(title) +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1 ALIGN=\"CENTER\">" + heading + "</H1>\n" +
                "<H2>Information on Your Session:</H2>\n" +
                "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
                "<TR BGCOLOR=\"#FFAD00\">\n" +
                "  <TH>Info Type<TH>Value\n" +
                "<TR>\n" +
                "  <TD>ID\n" +
                "  <TD>" + session.getId() + "\n" +
                "<TR>\n" +
                "  <TD>Creation Time\n" +
                "  <TD>" +
                new Date(session.getCreationTime()) + "\n" +
                "<TR>\n" +
                "  <TD>Time of Last Access\n" +
                "  <TD>" +
                new Date(session.getLastAccessedTime()) + "\n" +
                "<TR>\n" +
                "  <TD>Number of Previous Accesses\n" +
                "  <TD>" + accessCount + "\n" +
                "</TABLE>\n" +
                "</BODY></HTML>");

  }

  /** Handle GET and POST requests identically. */

  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
      throws ServletException, IOException {
    doGet(request, response);
  }
}
```