Chapter1 : MULTITHREADING

Objectives

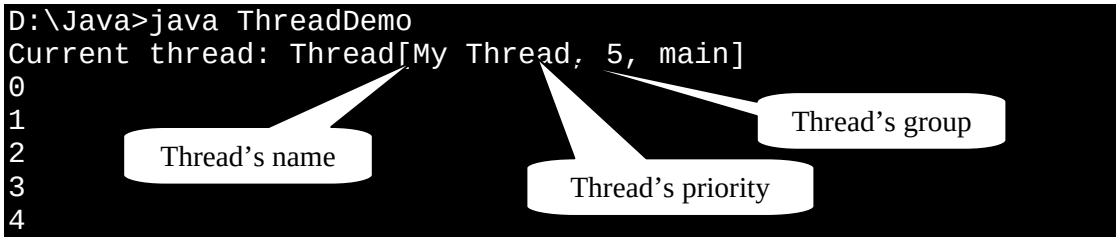
- Thread properties: thread's name, priority and group
- Create a thread by subclassing the **Thread** class
- Synchronization
- Animation

Section 1: Sample Programs

1. **Thread properties**. Write program to display the properties of current thread, and display numbers in one-second interval.

```
class ThreadDemo {
      public static void main(String args[]) {
         Thread t = Thread.currentThread();
         t.setName("My Thread");
         System.out.println("Current thread: " + t);
         try {
            for (int i=0; i <5; i++) {
               System.out.println(i);
               Thread.sleep(1000);
            }
         } catch (InterruptedException e) {}
      }
}
```

Here is the output of running this program:



2. **Two threads**. The following program demonstrates 2 threads which are concurrently running.

```
class A extends Thread {
   public void run() {
      for (int i=0; i<10; i++) {
         System.out.print(i + " ");
         try {
         Thread.sleep(1000);
         } catch (InterruptedException e) {}
      }
   }
```

```
}
class B extends Thread {
   public void run() {
      for (int i=0; i<10; i++) {
         System.out.print((char) (i+65) + " ");
         try {
         Thread.sleep(2500);
         }
         catch (InterruptedException e) {}
      }
   }
}
class C {
   public static void main(String args[]) {
      new A().start();
      new B().start();
   }
}
```

Here is the output of running this program:

```
D:\Java>java C
0 A 1 2 B 3 4 C 5 6 7 D 8 9 E F G H I J
D:\Java>
```

3. **Synchronization**. The following program demonstrates the use of synchronized keyword. This program displays several strings in the form:

   <string1>
   <string2>
   <string3>

The program will print "<string1", then it sleeps for a second and then prints ">".

```
class A {
   synchronized void display (String s) {
      try {
         System.out.print("<" + s);
         Thread.sleep(1000);
      } catch (InterruptedException e) {}
         System.out.println(">");
   }
}
class B extends Thread {
   A a;
   String s;
   B (A a, String s) {
      this.a = a;
      this.s = s;
   }
   public void run() {
      a.display(s);
   }
}
```

```
class C {
   public static void main(String args[]) {
      A a = new A();
      new B(a, “Hello”).start();
      new B(a, “Java”).start();
      new B(a, “World”).start();
   }
}
```

Here is the output of running this program:

```
C:\java>java C
<Hello>
<Java>
<World>
C:\java>javac C.java
```

Note that without the synchronized keyword, the output will be something like this:

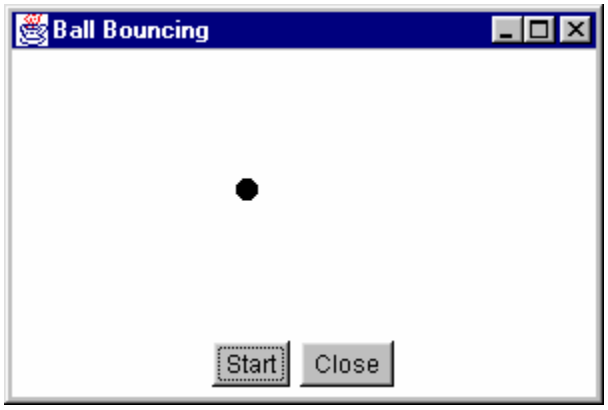
```
C:\java>java C
<Hello<Java<World>
>
>
C:\java>
```

4. **AnimationDemo**. This program demonstrates animation of two images T1.gif, T2.gif (or you can take other two images of the same size).

```
import java.awt.*;
class MyThread extends Thread {
   int i = 0;
   AnimationFrame af;
   MyThread (AnimationFrame af) {
      this.af = af;
   }
   public void run() {
      while (true) {
      i = (i+1) % 2;
      af.repaint();
      try {
         Thread.sleep(400);
      } catch (InterruptedException e)
            {System.out.println("Interrupted”);}
      }
   }
}
class AnimationFrame extends Frame {
   Image[] img;
   MyThread v;
   AnimationFrame (String s) {
      super(s);
```

```
      img = new Image[2];
      img[0] = getToolkit().getImage("T1.gif");
      img[1] = getToolkit().getImage("T2.gif");
      setBounds(300, 200, 200, 200);
      v = new MyThread(this);
      v.start();
      show();
   }
   public static void main(String args[]) {
      AnimationFrame af = new AnimationFrame ("Animation
Demo");
   }
   public void paint(Graphics g) {
      g.drawImage(img[v.i], 0, 20, this);
   }
}
```

5. **Ball bouncing**. Write program to demonstrate a ball bouncing inside a rectangle as in figure below:




```
// BallBouce.java
import java.awt.*;
import java.awt.event.*;
public class BallBounce {
   public static void main(String s[]) {
      Frame f = new BallBounceFrame();
      f.setBounds(200, 200, 300, 200);
      f.show();
   }
}
class BallBounceFrame extends Frame {
   Panel canvas;
   public BallBounceFrame() {
      setTitle("Ball Bouncing");
      addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent e) {
            System.exit(0);
         }
```

```
      });
      canvas = new Panel();
      add(canvas, "Center");
      Panel p = new Panel();
      Button start = new Button("Start");
      p.add(start);
      start.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
          Ball b = new Ball(canvas);
          b.start();
        }
      });
      Button close = new Button("Close");
      p.add(close);
      close.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
          canvas.setVisible(false);
          System.exit(0);
        }
      });
      add(p, "South");
   }
}
class Ball extends Thread {
   Panel box;
   int x = 0, y = 0;
   int dx = 2, dy = 2;
   public Ball(Panel p) { box = p;}
   public void draw() {
      Graphics g = box.getGraphics();
      g.fillOval(x, y, 10, 10);
      g.dispose();
   }
   public void move() {
      if (!box.isVisible()) return;
      Graphics g = box.getGraphics();
      g.setXORMode(box.getBackground());
      g.fillOval(x, y, 10, 10);
      x += dx; y += dy;
      Dimension d = box.getSize();
      if (x < 0) { x = 0; dx = -dx;   }
      if (x + 10 >= d.width) {x = d.width - 10; dx = -dx;}
      if (y < 0) {y = 0; dy = -dy;}
      if (y + 10 >= d.height) {y = d.height - 10; dy = -
dy;}      g.fillOval(x, y, 10, 10);
      g.dispose();
   }
```

```
   public void run() {
      try {
         draw();
         for (int i=1; i <= 1000; i++) {
            move();
            sleep(5);
         }
      } catch (InterruptedException e) {}
   }
}
```

Section 2: Do More

1. Modify sample program 2 so that it has another third thread that displays random strings.
2. Modify sample program 4 so that it has 2 buttons Start and Stop. When you click on Start button, it begins to animate, and when you click on Stop button, it will stop the animation.
3. Modify sample program 5 BallBouce.java to add two more buttons Stop and Continue. When you click on button Stop, the ball will stop moving. When you click on button Continue, the ball continues to move.

Section 3: Do Yourself

1. Write application to display simultaneously random lines on two areas using multithreading.
2. Write application containing three buttons (namely Circle, Line and Both) and one drawing area. Clicking on the Circle button will make the applet to draw random circles, clicking on the Line button will make the applet to draw random lines, and clicking on the Both button will make the applet to draw random circles and lines.
3. Write program to demonstrate transfer of bank accounts. Use thread and synchronization so that there is no account to be unsafe during transactions.

Chapter 2: Networking

Exercise

1. Write a program that accepts the input string, uses it to construct a URL object and returns the properties.

```
// program that accepts input string and uses it to construct a URL object

import java.net.*;
public class net1
{
    public static void main(String arg[ ])
    {
            try
            {
                    URL u = new URL(arg[0]);
                    System.out.println("Name of the file is: " + u.getFile( ));
                    System.out.println("Host Name is: " + u.getHost( ));
                    System.out.println("Port number is: " + u.getPort( ));
                    System.out.println("Protocol type is: " + u.getProtocol( ));
            }
            catch(MalformedURLException e)
            {
                    System.out.println(e);
            }
    }
}
```

2. Write a program that lists the contents i.e., entire text of any web page on the net.

```
import java.net.*;

public class net2
{
    public static void main(String arg[ ])
    {
            int i;
            BufferedInputStream bis;
            try
            {
                    URL u = new URL(arg[0]);
                    bis = (BufferedInputStream)u.getContent( );
                    while((i = bis.read( )) > 0)
                            System.out.print((char)i);
```

```
                    System.out.println( );
            }
            catch(MalformedURLException e)
            {
                    System.out.println(e);
            }
            catch (IOException e)
            {
                    System.out.println(e);
            }
    }
}
```

3. Write a program that when given a URL string as input, opens a connection to the URL and displays the header information.

```
public class net3
{
    public static void main(String arg[ ])
    {
            int i = 1;
            BufferedInputStream bis;
            try
            {
                    URL ul = new URL(arg[0]);
                    URLConnection u = ul.openConnection( );
                    String s = u.getHeaderField(i);
                    String sg = u.getHeaderFieldKey(i);
                    while(s != null)
                    {
                            System.out.println("Header" + i + ":" + sg + "=" +
s);
                            i++;
                            s = u.getHeaderField(i);
                            sg = u.getHeaderFieldKey(i);
                    }
            }
            catch(MalformedURLException e)
            {
                    System.out.println(e);
            }
            catch(IOException e)
            {
                    System.out.println(e);
            }
    }
```

```
}
```

4. Write a program that retrieves the address of the local host, Null host and host of the site.

```
public class net4
{
    public static void main(String arg[ ])
    {
            InetAddress i;
            BufferedInputStream bis;
            try
            {
                    i = InetAddress.getLocalHost( );
                    System.out.println("The local host is: " + i);
                    i = InetAddress.getByName(null);
                    System.out.println("The Null host is: " + i);
                    i = InetAddress.getByName("www.yahoo.com");
                    System.out.println("Yahoo Host address is: " + i);
            }
            catch(IOException e)
            {
                    System.out.println(e);
            }
    }
}
```

5. Write a program that opens a connection using the URL object and then examines the document's properties and content.

```
public class net5
{
    public static void main(String arg[ ]) throws IOException
    {
            int i;
            URL ul = new URL ("http://www.yahoo.com");
            URLConnection url = ul.openConnection( );

            System.out.println("Date: " + new Date(url.getDate( )));
            System.out.println("Content-type:" + url.getContentType( ));
            System.out.println("Expires: " + url.getExpiration( ));
            System.out.println("Last Modified: " + url.getLastModified( ));
            int l = url.getContentLength( );
            System.out.println("Content-Length:" + l);
            if(l>0)
```

```
                {
                        System.out.println("Content");
                        InputStream is = url.getInputStream( );
                        int a = l;
                        while(((i = is.read( )) != -1) && (--a > 0))
                        {
                                System.out.print((char)i);
                        }
                        is.close();
                } else
                {
                        System.out.println("Content is not available");
                }
        }
}
```

6. Write an application to show how to write a server and the client that goes with it.

```
//code for KnockKnock-server
import java.net.*;
import java.io.*;

public class KnockKnockServer {
   public static void main(String[] args) throws IOException {

      ServerSocket serverSocket = null;
      try {
         serverSocket = new ServerSocket(4444);
      } catch (IOException e) {
         System.err.println("Could not listen on port: 4444.");
         System.exit(1);
      }

      Socket clientSocket = null;
      try {
         clientSocket = serverSocket.accept();
      } catch (IOException e) {
         System.err.println("Accept failed.");
         System.exit(1);
      }

      PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
      BufferedReader in = new BufferedReader(
                                 new InputStreamReader(
                                 clientSocket.getInputStream()));
      String inputLine, outputLine;
      KnockKnockProtocol kkp = new KnockKnockProtocol();

      outputLine = kkp.processInput(null);
      out.println(outputLine);

      while ((inputLine = in.readLine()) != null) {
          outputLine = kkp.processInput(inputLine);
          out.println(outputLine);
          if (outputLine.equals("Bye."))
             break;
      }
      out.close();
      in.close();
      clientSocket.close();
```

```
        serverSocket.close();
    }
}
```

****

```
//code for KnockKnock-client

import java.io.*;
import java.net.*;

public class KnockKnockClient {
    public static void main(String[] args) throws IOException {

        Socket kkSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;

        try {
            kkSocket = new Socket("taranis", 4444);
            out = new PrintWriter(kkSocket.getOutputStream(), true);
            in = new BufferedReader(new
InputStreamReader(kkSocket.getInputStream()));
        } catch (UnknownHostException e) {
            System.err.println("Don't know about host: taranis.");
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for the connection to: taranis.");
            System.exit(1);
        }

        BufferedReader stdIn = new BufferedReader(new
InputStreamReader(System.in));
        String fromServer;
        String fromUser;

        while ((fromServer = in.readLine()) != null) {
            System.out.println("Server: " + fromServer);
            if (fromServer.equals("Bye."))
                break;

            fromUser = stdIn.readLine();
                if (fromUser != null) {
                System.out.println("Client: " + fromUser);
                out.println(fromUser);
                }
```

```
        }

        out.close();
        in.close();
        stdIn.close();
        kkSocket.close();
    }
}
```

****

```
// code for KnockKnockProtocol

import java.net.*;
import java.io.*;

public class KnockKnockProtocol {
    private static final int WAITING = 0;
    private static final int SENTKNOCKKNOCK = 1;
    private static final int SENTCLUE = 2;
    private static final int ANOTHER = 3;

    private static final int NUMJOKES = 5;

    private int state = WAITING;
    private int currentJoke = 0;

    private String[] clues = { "Turnip", "Little Old Lady", "Atch", "Who", "Who" };
    private String[] answers = { "Turnip the heat, it's cold in here!",
                                 "I didn't know you could yodel!",
                                 "Bless you!",
                                 "Is there an owl in here?",
                                 "Is there an echo in here?" };

    public String processInput(String theInput) {
        String theOutput = null;

        if (state == WAITING) {
            theOutput = "Knock! Knock!";
            state = SENTKNOCKKNOCK;
        } else if (state == SENTKNOCKKNOCK) {
            if (theInput.equalsIgnoreCase("Who's there?")) {
                theOutput = clues[currentJoke];
                state = SENTCLUE;
            } else {
                theOutput = "You're supposed to say \"Who's there?\"! " +
```

```
                        "Try again. Knock! Knock!";
            }
        } else if (state == SENTCLUE) {
            if (theInput.equalsIgnoreCase(clues[currentJoke] + " who?")) {
                theOutput = answers[currentJoke] + " Want another? (y/n)";
                state = ANOTHER;
            } else {
                theOutput = "You're supposed to say \"" +
                            clues[currentJoke] +
                            " who?\"" +
                            "! Try again. Knock! Knock!";
                state = SENTKNOCKKNOCK;
            }
        } else if (state == ANOTHER) {
            if (theInput.equalsIgnoreCase("y")) {
                theOutput = "Knock! Knock!";
                if (currentJoke == (NUMJOKES - 1))
                    currentJoke = 0;
                else
                    currentJoke++;
                state = SENTKNOCKKNOCK;
            } else {
                theOutput = "Bye.";
                state = WAITING;
            }
        }
        return theOutput;
    }
}
```

Running the Programs

You must start the server program first. To do this, run the server program using the Java interpreter, just as you would any other Java application. Remember to run the server on the machine that the client program specifies when it creates the socket.

Next, run the client program. Note that you can run the client on any machine on your network; it does not have to run on the same machine as the server.

If you are too quick, you might start the client before the server has a chance to initialize itself and begin listening on the port. If this happens, you will see a stack trace from the client. If this happens, just restart the client.

If you forget to change the host name in the source code for the `KnockKnockClient` program, you will see the following error message:

```
Don't know about host: taranis
```

To fix this, modify the `KnockKnockClient` program and provide a valid host name(If you are running the application on the same machine provide the host name of your machine) for your network. Recompile the client program and try again.

If you try to start a second client while the first client is connected to the server, the second client just hangs.

When you successfully get a connection between the client and server, you will see the following text displayed on your screen:

Server: Knock! Knock!

Now, you must respond with:

Who's there?

The client echoes what you type and sends the text to the server. The server responds with the first line of one of the many Knock Knock jokes in its repertoire. Now your screen should contain this (the text you typed is in bold):

Server: Knock! Knock!
Who's there?
Client: Who's there?
Server: Turnip

Now, you respond with:

Turnip who?"

Again, the client echoes what you type and sends the text to the server. The server responds with the punch line. Now your screen should contain this:

Server: Knock! Knock!
Who's there?
Client: Who's there?
Server: Turnip
Turnip who?
Client: Turnip who?
Server: Turnip the heat, it's cold in here! Want another? (y/n)

If you want to hear another joke, type **y**; if not, type **n**. If you type **y**, the server begins again with "Knock! Knock!" If you type **n**, the server says "Bye." thus causing both the client and the server to exit.

If at any point you make a typing mistake, the `KnockKnockServer` object catches it and the server responds with a message similar to this:

Server: You're supposed to say "Who's there?"!

The server then starts the joke over again:

Server: Try again. Knock! Knock!

Note that the `KnockKnockProtocol` object is particular about spelling and punctuation but not about capitalization.

Extra Exercises

1. **Write the FTP server and FTP client Application**
2. **Change the exercise 1 with Applet**
3. **Write the Mail Server application**
4. **Write the Tictactoe Game that two person can play over network**
5. **Write the Applet that perform the tasks**
    a. **Client sent to server string**
    b. **Server rerutn to client Reverse String**
6. **Write the program that alert when hacker violate server by IP address and port.**

Chapter 3: Remote Method Invocation

Exercise

1. Write a program depicting remote interface.

```
package myrem.host;

import java.rmi.*;

public interface MyRem extends Remote
{
    int gnum( ) throws RemoteException;
    String gchar(int n) throws RemoteException;
}
```

2. Write a program that displays random number.

```
package ju.exam.random.server;

import java.rmi.*;

public interface RandomServer extends Remote {
 double getRandom() throws RemoteException;
}

package ju.exam.random.client;

import ju.exam.random.server.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.rmi.*;
import java.rmi.server.*;

public class RandomClient extends Frame implements Runnable {
 String host="192.160.90.233";
 TextField text = new TextField(50);
 Button newRandom = new Button("Random");
 RandomServer server;
 int screenWidth = 500;
 int screenHeight = 100;
 public void run(){
 }
 public RandomClient() {
  super("RandomClient");
  setup();
```

```
 setSize(screenWidth,screenHeight);
 addWindowListener(new WindowEventHandler());
 show();
}
void setup() {
 setLayout(new FlowLayout(FlowLayout.LEFT));
 newRandom.addActionListener(new ButtonHandler());
 add(newRandom);
 add(text);
 connectToServer();
}
void connectToServer() {
 try {
  server = (RandomServer) Naming.lookup("//"+host+"/RandomServer");
 } catch (Exception ex) {
  text.setText(ex.toString());
 }
}
class ButtonHandler implements ActionListener {
 public void actionPerformed(ActionEvent ev){
  String s=ev.getActionCommand();
  if("Random".equals(s)){
   try {
    double rand=server.getRandom();
    text.setText(new Double(rand).toString());
   } catch (Exception ex){
    text.setText(ex.toString());
   }
  }
 }
}
```

```
package ju.exam.random.server;

import java.rmi.*;
import java.rmi.server.*;

public class RandomServerImpl extends UnicastRemoteObject
 implements RandomServer {
 public RandomServerImpl() throws RemoteException {
  super();
 }
 public double getRandom() throws RemoteException {
  return Math.random();
 }
```

```
 public static void main(String args[]){
  System.setSecurityManager(new RMISecurityManager());
  try {
   RandomServerImpl instance = new RandomServerImpl();
   Naming.rebind("///RandomServer", instance);
   System.out.println("RandomServer is registered.");
  } catch (Exception ex) {
   System.out.println(ex.toString());
  }
 }
}
```

3. Write an applet that accesses the remote server. This applet is used as a local client.

```
package ju.ab.info.client;

import ju.ab.info.server.*;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.rmi.*;

public class eg1 extends Applet {
 String text = "Click here to Update the server.";
 TextArea textArea = new TextArea(15,50);
 Button update = new Button("Update");
 InfoServer server;
 public void init() {
  setLayout(new BorderLayout());
  add("Center",textArea);
  update.addActionListener(new ButtonHandler());
  add("South",update);
  try {
   URL hostURL = getCodeBase();
   String host = hostURL.getHost();
   server = (InfoServer) Naming.lookup("//"+host+"/InfoServer");
   textArea.setText(text);
  } catch (Exception ex) {
   textArea.setText(ex.toString());
  }
 }
 class ButtonHandler implements ActionListener {
  public void actionPerformed(ActionEvent ev){
```

```
  String s=ev.getActionCommand();
  if("Update".equals(s)){
   try {
    String newText=server.getInfo();
    text=newText+"\n"+text;
    textArea.setText(text);
   } catch (Exception ex){
    textArea.setText(ex.toString());
   }
  }
 }
}
```

The file to display the above applet.

```
<HTML>
<HEAD>
<TITLE>The applet representing the client</TITLE>
</HEAD>
<BODY>
<APPLET CODEBASE="http://204.115.182.233/codebase/"
CODE=" ju.ab.info.client.eg1.class"
WIDTH=800 HEIGHT=400>
</APPLET>
</BODY>
</HTML>
```

4. Write an applet that helps in developing a client application. This client application enables to browse through a host's registry so as to see which remote objects it supports.

Ans:

```
import java.awt.*;
import java.awt.event.*;
import java.rmi.registry.*;

public class clnteg extends Frame {
 TextField host = new TextField(30);
 TextArea objectNames = new TextArea(15,50);
 Button browse = new Button("Browse host");
 int screenWidth = 500;
 int screenHeight = 400;
 public static void main(String args[]){
```

```
 Browser app = new clnteg();
}
public clnteg() {
 super("Browser");
 setup();
 setSize(screenWidth,screenHeight);
 addWindowListener(new WindowEventHandler());
 show();
}
void setup() {
 browse.addActionListener(new ButtonHandler());
 Panel panel = new Panel();
 panel.add(new Label("Host name: "));
 panel.add(host);
 panel.add(browse);
 add("North",panel);
 add("Center",objectNames);
}
class ButtonHandler implements ActionListener {
 public void actionPerformed(ActionEvent ev){
  String s=ev.getActionCommand();
  if("Browse host".equals(s)){
   try {
    Registry registry = LocateRegistry.getRegistry(host.getText());
    String objectList[] = registry.list();
    String objects="";
    for(int i=0;i<objectList.length;++i){
     objects+=objectList[i]+"\n";
    }
    objectNames.setText(objects);
   } catch (Exception ex){
    objectNames.setText(ex.toString());
   }
  }
 } }   }
```

Extra Exercises

1. **Write the Program that performing Add, sub, Multi, division by RMI**
2. **Write the program that Matrix Add calculating by RMI**
3. **Write the Program that performs Reverse Number by clients**

Chapter 4: Enterprise JavaBean

Exercise

1. Develop a bean that has a juggler on the screen. There should be two buttons Start and Stop that starts and stops the juggling.

Modify the above program and increase the juggling speed and then reduce the juggling speed. Save the bean and make an applet.

Write a bean demonstrating a label.

```
// program demonstrating a label bean

package event;
import java.awt.*;
import java.awt.event.*;
import java.io.Serializable;

public class bean1 extends java.applet.Applet implements ActionListener,
Serializable
{
        private int a = 0;
        public bean1()
        {
        }
        public synchronized void paint(Graphics g)
        {
                String valueText = " " + a;
                FontMetrics fmt = g.getFontMetrics();
                g.setColor(getBackground());
                g.setColor(getForeground());
                g.drawString(valueText, 0,0);
        }

        public synchronized void update(Graphics g)
        {
                Image hidden = createImage(getSize().width, getSize().height);
                paint(hidden.getGraphics());
                g.drawImage(hidden, 0, 0, null);
        }
        public void actionDone(ActionEvent aevt)
        {
                a = a + 1;
                repaint();
        }
}
```

2. Write a code for a bean representing a simple gauge bean.

```
import java.io.Serializable;
import java.beans.*;
import java.awt.*;
import java.awt.event.*;

public class bean2 extends Canvas implements Serializable {
 public static int h = 1;
 public static int v = 2;
 public static int w = 100;
 public static int h = 20;
 public int orientation = h;
 public int width = w;
 public int height = h;
 public double minval = 0.0;
 public double maxval = 1.0;
 public double curval = 0.0;
 public Color gaugeColor = Color.lightGray;
 public Color valueColor = Color.blue;

 public bean2()
 {
  super();
 }

 public Dimension getPreferredSize()
 {
  return new Dimension(width,height);
 }

 public synchronized void paint(Graphics g)
 {
  g.setColor(gaugeColor);
  g.fill3DRect(0,0,width-1,height-1,false);
  int border=3;
  int innerHeight=height-2*border;
  int innerWidth=width-2*border;
  double scale=(double)(curval-minval)/
   (double)(maxval-minval);
  int gaugeValue;
  g.setColor(valueColor);
  if(orientation==h){
   gaugeValue=(int)((double)innerWidth*scale);
```

```
  g.fillRect(border,border,gaugeValue,innerHeight);
 }else{
  gaugeValue=(int)((double)innerHeight*scale);
  g.fillRect(border,border+(innerHeight-gaugeValue),innerWidth,gaugeValue);
 }
}

public double getCurrentValue()
{
 return currentValue;
}

public void setCurrentValue(double newCurrentValue)
{
 if(newCurrentValue>=minval && newCurrentValue<=maxval)
  curval=newCurrentValue;
}

public double getMinValue()
{
 return minval;
}
public void setMinValue(double newMinValue)
{
 if(newMinValue<=curval)
  minval=newMinValue;
}

public double getMaxValue()
{
 return maxval;
}

public void setMaxValue(double newMaxValue)
{
 if(newMaxValue >= curval)
  maxval=newMaxValue;
}

public int getWidth()
{
 return width;
}

public void setWidth(int newWidth)
{
```

```
 if(newWidth > 0)
 {
  width=newWidth;
  updateSize();
 }
}

public int getHeight()
{
 return h;
}

public void setHeight(int newHeight)
{
 if(newHeight > 0)
 {
  h=newHeight;
  updateSize();
 }
}

public Color getGaugeColor()
{
 return gaugeColor;
}

public void setGaugeColor(Color newGaugeColor)
{
 gaugeColor=newGaugeColor;
}

public Color getValueColor()
{
 return valueColor;
}

public void setValueColor(Color newValueColor)
{
 valueColor=newValueColor;
}

public boolean isHorizontal()
{
 if(orientation==h) return true;
 else return false;
}
```

```
public void setHorizontal(boolean newOrientation)
{
 if(newOrientation)
 {
  if(orientation==v) switchDimensions();
 }
 else
 {
  if(orientation==h) switchDimensions();
  orientation=v;
 }
 updateSize();
}
void switchDimensions(){
 int temp=w;
 w=ht;
 h=temp;
}
void updateSize(){
 setSize(w,h);
 Container cnt=getParent();
 if(cnt!=null){
  cnt.invalidate();
  cnt.doLayout();
 }
}
}
```

3. Write a bean representing the time generator. This bean displays the time zone wise.

```
// program representing the source code of the bean generating the bean

import java.util.*;
import java.io.*;
import javax.infobus.*;
import java.beans.*;

public class tmgen extends InfoBusMemberSupport
  implements InfoBusDataProducer, Serializable, Runnable {

 public tmgen()
 {
  this(null);
 }
```

```
 public tmgen(InfoBusMember infoBusMember)
 {
  super(infoBusMember);
  try {
   joinInfoBus("Time Bus");
   getInfoBus().addDataProducer(this);
  } catch (Exception e) {
   System.out.println(e.toString());
  }
 }
 public void run() {
  generateTime();
 }
 void generateTime() {
  while(true) {
   // Notify consumers
   getInfoBus().fireItemAvailable("Time",null,this);
   // Wait a second
   try{
    Thread.currentThread().sleep(1000);
   }catch(Exception ex) {
   }
  }
 }
 public void dataItemRequested(InfoBusItemRequestedEvent e) {
  // Get GMT time zone IDs
  String[] tzID = TimeZone.getAvailableIDs(0);
  SimpleTimeZone tz = new SimpleTimeZone(0,tzID[0]);
  GregorianCalendar calendar = new GregorianCalendar(tz);
  calendar.setTime(new Date());
  e.setDataItem(calendar);
 }
 public void propertyChange(PropertyChangeEvent e) {
 }
}

// program representing a bean that indicates the time zone list

import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.io.*;
import javax.infobus.*;
import java.beans.*;
```

```
public class tmznlst extends java.awt.List
  implements InfoBusMember, InfoBusDataProducer,
   ItemListener, Serializable {
 InfoBusMemberSupport ims;
 TimeZone tz = TimeZone.getDefault();
 String[] tzIDs = TimeZone.getAvailableIDs();

 public tmznlst(int rows) {
  super(rows);
  ims = new InfoBusMemberSupport(null);
  try {
   ims.joinInfoBus("Time Bus");
   ims.getInfoBus().addDataProducer(this);
  }catch (Exception e) {
   System.out.println(e.toString());
  }
  for(int i=0;i<tzIDs.length;++i) add(tzIDs[i]);
  addItemListener(this);
 }
 public void itemStateChanged(ItemEvent e) {
  int index = ((Integer) e.getItem()).intValue();
  String tzID = tzIDs[index];
  tz = TimeZone.getTimeZone(tzID);
  getInfoBus().fireItemAvailable("Zone",null,this);
 }
 public void setInfoBus(InfoBus newInfoBus)
   throws PropertyVetoException {
  ims.setInfoBus(newInfoBus);
 }
 public InfoBus getInfoBus() {
  return ims.getInfoBus();
 }
 public void addInfoBusVetoableListener(VetoableChangeListener vcl) {
  ims.addInfoBusVetoableListener(vcl);
 }
 public void removeInfoBusVetoableListener(VetoableChangeListener vcl) {
  ims.removeInfoBusVetoableListener(vcl);
 }
 public void addInfoBusPropertyListener(PropertyChangeListener pcl) {
  ims.addInfoBusPropertyListener(pcl);
 }
 public void removeInfoBusPropertyListener(PropertyChangeListener pcl) {
  ims.removeInfoBusPropertyListener(pcl);
 }
 public void propertyChange(PropertyChangeEvent e) {
 }
```

```
 public void dataItemRequested(InfoBusItemRequestedEvent e) {
  e.setDataItem(tz);
 }
}

// program representing a bean indicating time display

import java.awt.*;
import java.util.*;
import java.io.*;
import javax.infobus.*;
import java.beans.*;

public class tmdsp extends Canvas
  implements InfoBusMember, InfoBusDataConsumer, Serializable {
 InfoBusMemberSupport ims;
 String time1 = "";
 String time2 = "";
 TimeZone tz = TimeZone.getDefault();

 public tmdsp() {
  ims = new InfoBusMemberSupport(null);
  try {
   ims.joinInfoBus("Time Bus");
   ims.getInfoBus().addDataConsumer(this);
  }catch (Exception e) {
   System.out.println(e.toString());
  }
 }
 public Dimension getPreferredSize() {
  return new Dimension(120,100);
 }
 public void paint(Graphics g) {
  g.drawString(time1,10,25);
  g.drawString(time2,10,75);
 }
 public void setInfoBus(InfoBus newInfoBus)
   throws PropertyVetoException {
  ims.setInfoBus(newInfoBus);
 }
 public InfoBus getInfoBus() {
  return ims.getInfoBus();
 }
 public void addInfoBusVetoableListener(VetoableChangeListener vcl) {
  ims.addInfoBusVetoableListener(vcl);
 }
```

```
public void removeInfoBusVetoableListener(VetoableChangeListener vcl) {
 ims.removeInfoBusVetoableListener(vcl);
}
public void addInfoBusPropertyListener(PropertyChangeListener pcl) {
 ims.addInfoBusPropertyListener(pcl);
}
public void removeInfoBusPropertyListener(PropertyChangeListener pcl) {
 ims.removeInfoBusPropertyListener(pcl);
}
public void propertyChange(PropertyChangeEvent e) {
}
public void dataItemAvailable(InfoBusItemAvailableEvent e) {
 String name = e.getDataItemName();
 if(name.equals("Time")) {
  GregorianCalendar calendar1 =
   (GregorianCalendar) e.requestDataItem(this,null);
  GregorianCalendar calendar2 = new GregorianCalendar(tz);
  time1 = getTimeString(calendar1);
  time2 = getTimeString(calendar2);
  repaint();
 }else if(name.equals("Zone")) {
  tz = (TimeZone) e.requestDataItem(this,null);
 }
}
String getTimeString(GregorianCalendar calendar) {
  String hour = String.valueOf(calendar.get(Calendar.HOUR_OF_DAY));
  if(hour.length()==1) hour = "0"+hour;
  String minute = String.valueOf(calendar.get(Calendar.MINUTE));
  if(minute.length()==1) minute = "0"+minute;
  String second = String.valueOf(calendar.get(Calendar.SECOND));
  if(second.length()==1) second = "0"+second;
  TimeZone timeZone = calendar.getTimeZone();
  return hour+":"+minute+":"+second+" "+timeZone.getID();
}
public void dataItemRevoked(InfoBusItemRevokedEvent e) {
 }
}
```

// program representing an applet that combines the time generator, time display and time zone list

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import javax.infobus.*;
```

```
public class tmapplet extends Applet {
 Thread generator = new Thread(new tmgen());
 TimeDisplay display = new tmdsp();
 TimeZoneList tzl = new tmznlst(5);

 public void init() {
  add(tzl);
  add(display);
  generator.start();
 }
}
```

Extra Exercises

1. **Write the bean that manages hour working of 10 employee**
2. **Write the bean that calculates salary total in the month of 10 employees by formula**

   Salary/day = hour_total * 12 + OverTimer * 24

Chapter 4 : Database with JDBC

Exercise

1. Write a program that loads its database driver with the JDBC-ODBC bridge, and creates an initial connection

```
// program using JDBC ODBC bridge and creates an initial connection

Connection c = null;

try
{
    // loading JDBC ODBC Bridge
    Class.forName("sun.jbdc.odbc.JdbcOdbcDriver");

    // getting a connection
    c = DriverManager.getConnection("jdbc:odbc:empdb", "user", "passwd");

}

catch(IOException e)
{
    System.out.println(e);
}

finally
{
    //closing the connection
    try
    {
            if(c != null)
               c.close( );
    }
    catch(SQLException ignored)
    { }
}
```

2. Write a program that uses basic SQL statement to create table, insert record, query record, and drop table.

```
class examp1 implements RunExample
{
    boolean localMode = false;
      void examp1(String url, String user, String password)
```

```
    {
            Connection conn = null;
            Statement stmt = null;
            try
            {
               Class.forName("jet.bridge.JetDriver").newInstance();
               new jet.bridge.JetDriver();         // Connect to the data source
            if (localMode)
                  conn = DriverManager.getConnection("jdbc:jet:" + url, user,
                  password);
            else
                  conn = DriverManager.getConnection("jdbc:jet.jettp:" + url,
                  user, password);
                     stmt = conn.createStatement();
                   try
                     {
                        stmt.executeUpdate("Drop Table nameid");
                     conn.commit();
                     }
                     catch (SQLException e)
                   {
                     System.out.println(e.getMessage());           }
                     stmt.executeUpdate("Create table nameid (id integer, name
char(50))");
                        conn.commit();

          PreparedStatement pre = conn.prepareStatement("Insert into nameid
values (?,?)");
          for (int i = 1; i <= 20; i++)
        {
            pre.setInt(1, i);
            pre.setString(2, "John " + Integer.toString(i,10));
            pre.executeUpdate();
          }

          ResultSet resultSet = stmt.executeQuery("Select id, name from
nameid");
        while (resultSet.next())
          {   int id = resultSet.getInt(1);
            String name = resultSet.getString(2);
            System.out.println(String.valueOf(id) + " " + name);
          }

            resultSet.close();
            stmt.close();
            conn.close();
```

```
    }

     catch(Exception e)       {
        System.out.println(e.getMessage());       }       finally       {
        try          {             Close c1;
           if (stmt != null)                stmt.close();
           if (conn != null && !conn.isClosed())             {
              conn.close();             }          }
        catch(SQLException e)          {
           System.out.println("Error closing connection");
           System.out.println(e.getMessage());          }       }    }

    public static void main(String args[])
    {
           if (4 > args.length)
           {          new examp1( args );
           }
         else
           {
              new examp1( args[0], args[1], args[2], args[3] );       }    }
           public examp1(String args[])
           {
         GetConnectInfo info = new GetConnectInfo(this, args);    }

       public examp1(String server, String uid, String pwd, String host)
       {
          if (host.equals(""))       {          localMode = true;
                host = server;       }       else
        host = host + '/' + server;       examp1(host, uid, pwd);    }

   public void run(GetConnectInfo info)    {
      if (info.sURLSpec.equals(""))       {          localMode = true;
         info.sURLSpec = info.sDSN;       }       else
         info.sURLSpec = info.sURLSpec + '/' + info.sDSN;
         examp1(info.sURLSpec, info.sUser, info.sPassword );
         System.exit(0);
      }
} //end    class examp1
```

3. Write an application in Java, which will execute a SELECT statement to return a result set of the employee name, age, and birthday, sorted by birthday.

```
import net.connect.*;

class BindCol implements RunExample
{
```

```
   final static int NAME_LEN = 30;
   DbEnv    dbenv   = null;
   DbDbc    dbcon   = null;
   DbStmt   dbstmt  = null;

   public void initialize(String dsn, String user, String password, String url )
   {
      try
      {
         dbenv  = DbEnv.SQLAllocEnv(url);
         dbcon  = dbenv.SQLAllocConnect();

         dbcon.SQLConnect(dsn, user, password);

         dbstmt = dbcon.SQLAllocStmt();

         dbstmt.SQLExecDirect("CREATE TABLE EMPLOYEE (NAME
CHAR(30), AGE INTEGER, BIRTHDAY DATE)");
         dbstmt.SQLExecDirect("INSERT INTO EMPLOYEE VALUES('BILL
SMITH', 25, '1970-11-24 00:00:00')");
         dbstmt.SQLExecDirect("INSERT INTO EMPLOYEE
VALUES('MARK FORD', 20, '1975-04-02 00:00:00')");
         dbstmt.SQLExecDirect("INSERT INTO EMPLOYEE
VALUES('JASON MONK', 22, '1974-01-27 00:00:00')");
         dbstmt.SQLExecDirect("INSERT INTO EMPLOYEE
VALUES('SCOTT SANDER', 25, '1971-10-05 00:00:00')");
         dbstmt.SQLExecDirect("INSERT INTO EMPLOYEE
VALUES('MARY KELLOG', 21, '1974-09-30 00:00:00')");

         bindcolExample();

      }
      catch (Exception e)
      {
         System.out.println(e.toString());
      }
      finally
      {
         try
         {
            if (dbstmt != null)
            {
               dbstmt.SQLFreeStmt(Db.SQL_CLOSE);
               dbstmt.SQLExecDirect("DROP TABLE EMPLOYEE");
               dbstmt.SQLFreeStmt(Db.SQL_DROP);
            }
```

```
            if (dbcon != null)
            {
               dbcon.SQLDisconnect();
               dbcon.SQLFreeConnect();
            }
         }
         catch (DbException e) {}

         try
         {
            if (dbenv != null)
               dbenv.SQLFreeEnv();
         }
         catch (DbException e) {}
      }

   }

   void bindcolExample() throws DbException
   {
      DbValue   szName     = new DbChar(NAME_LEN);
      DbValue   szBirthday = new DbDate();
      DbValue   sAge       = new DbInteger();

      if (dbstmt.SQLExecDirect("SELECT NAME, AGE, BIRTHDAY FROM
EMPLOYEE ORDER BY 3,2,1"))
      {
         /* Bind columns 1, 2, 3
          */

         dbstmt.SQLBindCol(1, szName);
         dbstmt.SQLBindCol(2, sAge);
         dbstmt.SQLBindCol(3, szBirthday);

         /* Fetch and print each row of data. On an
          * error, display a message and exit
          */
         while (true)
         {
            if (dbstmt.SQLFetch())
            {
               System.out.println(szName.toString() + " " + szBirthday.toString()
+
                        " " + sAge.toString());
            }
```

```
            else break;
          }
       }
    }

    /* The follwing series of methods are not relevant to the main objective of
     * this example.  They are included here only to complete a runnable
program.
     */

    /* The main method will call one of the two constructors, depending on
number of
     * arguments provided by the user at the DOS prompt.  If insufficient
arguments
     * are provided, a dialog box will be displayed for input of individual
arguments.
     */

    public static void main(String args[])
    {

       if (4 > args.length)
       {
          new BindCol( args );
       }
       else
       {
          new BindCol( args[0], args[1], args[2], args[3] );
       }
    }

    /* This method displays a dialog for inputing Database Connection
information
     */

    public BindCol(String args[])
    {
       GetConnectInfo info = new GetConnectInfo(this, args);
    }

    /* This method contains call to the core method of this example.
     */

    public BindCol(String server, String uid, String pwd, String host)
    {
       initialize(server, uid, pwd, host);
```

```
        }

        /* This method is an interface method called from the dialog box.
         * This method contains call to the core method of this example.
         */

        public void run(GetConnectInfo info)
        {
              initialize( info.sDSN, info.sUser, info.sPassword, info.sURLSpec );
              System.exit(0);
        }

}
```

4. Write a program in Java that illustrates the usage of the following JDBC functions:

- **Create a Table**
- **Insert into a Table**
- **Select from a Table**
- **Update a Table**
- **Delete from a Table**

Ans:

```
// program to illustrate the use of the JDBC.

import java.net.URL;
import java.sql.*;
import java.lang.Runtime;

class demo
 {

// instance variables

   public static String query;
   public static Connection con;
   public static Statement  stmt;
   public static ResultSet  results;

   public static void CreateTable(  ) {

   try
```

```
  {
      stmt = con.createStatement();
      int ResultCode;

//this is done to cleanly format listing only

      String SQLText = "create table worker";
           SQLText += "( workerID int, workerName CHAR(15),";
           SQLText += " hourlyRate float4,skillType CHAR(8), SupvID int )";
//process create table
      ResultCode = stmt.executeUpdate( SQLText );
//setup index sql
      SQLText = "create unique index workerindex on worker(workerID)";
      ResultCode = stmt.executeUpdate( SQLText );
  }
   catch (java.lang.Exception ex)
  {

// Print description of the exception.
     System.out.println( "** Error on create table. ** " );
     ex.printStackTrace ();

  }
}
// ------------------------End create

public static void InsertWorkerData(  ) {

  try {
      stmt = con.createStatement();
      int ResultCode;
      String SQLText = "insert into worker values ";
           SQLText += "(1235,'M. Faraday',12.50,'Electric',1311)";
      ResultCode = stmt.executeUpdate( SQLText );
      System.out.println( "Inserted " + ResultCode + " rows." );

      SQLText = "insert into worker values";
      SQLText += "(1412,'C. Nemo'     ,13.75,'Plumbing',1520)";
      ResultCode = stmt.executeUpdate( SQLText );
      System.out.println( "Inserted " + ResultCode + " rows." );

      SQLText = "insert into worker values";
      SQLText +="(2920,'R. Garret'   ,10.00,'Roofing' ,2920)";
      ResultCode = stmt.executeUpdate( SQLText );
      System.out.println( "Inserted " + ResultCode + " rows." );
```

```
        SQLText = "insert into worker values ";
        SQLText +="(3231,'F. Mason'    ,17.40,'Framing' ,3231)";
        ResultCode = stmt.executeUpdate( SQLText );
        System.out.println( "Inserted " + ResultCode + " rows." );

        SQLText = "insert into worker values ";
        SQLText +="(1520,'H. Rickover' ,11.75,'Plumbing',1520)";
        ResultCode = stmt.executeUpdate( SQLText );
        System.out.println( "Inserted " + ResultCode + " rows." );

        SQLText = "insert into worker values ";
        SQLText +="(1311,'C. Coulomb'  ,15.50,'Electric',1311)";
        ResultCode = stmt.executeUpdate( SQLText );
        System.out.println( "Inserted " + ResultCode + " rows." );

        SQLText = "insert into worker values ";
        SQLText +="(3001,'J. Barrister',8.20 ,'Framing' ,3231)";
        ResultCode = stmt.executeUpdate( SQLText );
        System.out.println( "Inserted " + ResultCode + " rows." );
       }
      catch (java.lang.Exception ex)
      {
      // Print description of the exception.
      System.out.println( "** Error on data insert. ** " );
      ex.printStackTrace ();
      }
}
// ----------------------End Insert

public static void SelectData(  )
{
  try {
       stmt = con.createStatement();
       String SQLText = "select * from worker";
       results = stmt.executeQuery( SQLText );
       DisplayResults( results );
      }
  catch (java.lang.Exception ex)
     {
// Print description of the exception.
      System.out.println( "** Error on data select. ** " );
      ex.printStackTrace ();
      }
}

// ----------------------End Select
```

```
public static void UpdateData(  )
 {
  try
 {
      stmt = con.createStatement();
      int ResultCode;

      String SQLText = "update worker set hourlyRate = ";
      SQLText = SQLText + " 1.05 * hourlyrate" ;
      SQLText = SQLText + " where workerID = 3001";
      ResultCode = stmt.executeUpdate( SQLText );
// print the number of rows updated
      System.out.println( "Updated " + ResultCode + " rows." );
// show the updated row
      demo.SelectData();
     }
  catch (java.lang.Exception ex)
     {
// Print description of the exception.
     System.out.println( "** Error on data update. ** " );
     ex.printStackTrace ();

     }
}
// ----------------------End Update

public static void DeleteData(  ) {

  try {
      stmt = con.createStatement();
      int ResultCode;
      String SQLText = "delete from worker where workerID = 3001";
      ResultCode = stmt.executeUpdate( SQLText );
 // print the number of rows deleted
      System.out.println( "Deleted " + ResultCode + " rows." );
 // show the data
      demo.SelectData();

     }
  catch (java.lang.Exception ex)
    {
// Print description of the exception.
      System.out.println( "** Error on data delete. ** " );
      ex.printStackTrace ();
     }
```

```
}

// ----------------------------End Delete


  public static void DisplayResults( ResultSet results )
        throws SQLException
          {
                int i;

             // Get the ResultSetMetaData and use this for
                // the column headings

             ResultSetMetaData rsmd = results.getMetaData ();

                // Get the number of columns in the result set

                int numCols = rsmd.getColumnCount ();

                // Display column headings

                for (i=1; i<=numCols; i++)
             {
                if (i > 1) System.out.print(" - ");
                        System.out.print(rsmd.getColumnLabel(i));
                }
                System.out.println("");

                // Display data, fetching until end of the result set

             boolean more = results.next ();
                while (more) {

 // Loop through each column, getting the
 // column data and displaying it.

                        for (i=1; i<=numCols; i++)
                {
                     if (i > 1) System.out.print(" - ");
                     System.out.print(results.getString(i));
                        }
                        System.out.println("");
// Fetch the next result set row
                more = results.next ();
             }
  }
```

```
}

/* ----------------------END of SET ---------------------- */

class JDBCx1
 {

       public static void main (String argv[])
 {
//Example format for url for access and postgress
//    String url   = "jdbc:odbc:msaccessdb";
//    String url   = "jdbc:postgresql:yourdbname
// Program execution is java JDBCx1 url

              String url = new String(argv[0]);
                    try
              {
// Load the jdbc-odbc bridge driver
//for access
//         Class.forName ("jdbc.odbc.JdbcOdbcDriver");
//for postgres
                    Class.forName("postgresql.Driver");
// Attempt to connect to a driver. I use my own username and no password
                    demo.con = DriverManager.getConnection (
                                url, "fred", "");
// If we get here, we are successfully
// connected to the URL
// Let's Create the table
                    demo.CreateTable();
// Let's Insert Data
                    demo.InsertWorkerData();
// Let's Select Data
                    demo.SelectData();
// Lets's Update Data
                    demo.UpdateData();
// Let's Delete data
                    demo.DeleteData();
// Let's Close the statement
                    demo.stmt.close();
// Let's Close the connection
                    demo.con.close();
                    }
                    catch (SQLException ex)
              {
// An SQLException was generated.  We Catch it and
```

```
// display the error information.  There may be multiple
// error objects chained together. Let's get them all ok.
System.out.println ("\n*** SQLException caught ***\n");
while (ex != null)
                    {
System.out.println ("SQLState: " +
                                        ex.getSQLState ());
                                System.out.println ("Message:  " +
                                        ex.getMessage ());
                                System.out.println ("Vendor:   " +
                                        ex.getErrorCode ());
                                ex = ex.getNextException ();
                                System.out.println ("");
                        }
                }
                catch (java.lang.Exception ex)
            {
// Got some other type of exception.  Lets look it over.
                        ex.printStackTrace ();
                }
        }

        //---------------------------------------------------------------
        // checkForWarning
        // Checks for and displays warnings.  Returns true if a warning
        // existed
        //---------------------------------------------------------------

        private static boolean checkForWarning (SQLWarning warn)
                        throws SQLException
        {
                boolean rc = false;

                // If a SQLWarning object was given, display the
                // warning messages.  Note that there could be
                // multiple warnings chained together

                if (warn != null) {
                        System.out.println ("\n *** Warning ***\n");
                        rc = true;
                        while (warn != null) {
                                System.out.println ("SQLState: " +
                                        warn.getSQLState ());
                                System.out.println ("Message:  " +
                                        warn.getMessage ());
                                System.out.println ("Vendor:   " +
```

```
                                warn.getErrorCode ());
                        System.out.println ("");
                        warn = warn.getNextWarning ();
                }
        }
        return rc;
}

//---------------------------------------------------------------------
// dispResultSet
// Displays all columns and rows in the given result set
//---------------------------------------------------------------------

private static void dispResultSet (ResultSet rs)
        throws SQLException
{
        int i;

        // Get the ResultSetMetaData.  This will be used for
        // the column headings

        ResultSetMetaData rsmd = rs.getMetaData ();

        // Get the number of columns in the result set

        int numCols = rsmd.getColumnCount ();

        // Display column headings

        for (i=1; i<=numCols; i++) {
        if (i > 1) System.out.print(" - ");
                System.out.print(rsmd.getColumnLabel(i));
        }
        System.out.println("");

        // Display data, fetching until end of the result set

        boolean more = rs.next ();
        while (more) {

                // Loop through each column, getting the
                // column data and displaying

                for (i=1; i<=numCols; i++) {
              if (i > 1) System.out.print(" - ");
                        System.out.print(rs.getString(i));
```

```
                    }
                    System.out.println("");
                    // Fetch the next result set row

                    more = rs.next ();
                }
            }
        }
```