1. Write a client/server application, server provides information about a product and client receives that information.

- Step 1: create interface of server component: Product.java

```
import java.rmi.*;

public interface Product extends Remote
{  String getDescription()
      throws RemoteException;
}
```

- Step 2 : create a server component that implements above interface: ProductImpl.java

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

public class ProductImpl
   extends UnicastRemoteObject
   implements Product
{  public ProductImpl(String n)
      throws RemoteException
   {  name = n;
   }

   public String getDescription()
      throws RemoteException
   {  return "I am a " + name + ". Buy me!";
   }

   private String name;
}
```

- Step 3: create an application at server side that loads server component into memory: ProductServer.java

```
import java.rmi.*;
import java.rmi.server.*;
import sun.applet.*;

public class ProductServer
{  public static void main(String args[])
   {  try
      {  System.out.println
            ("Constructing server implementations...");

         ProductImpl p1
            = new ProductImpl("Blackwell Toaster");
         ProductImpl p2
            = new ProductImpl("ZapXpress Microwave Oven");
```

```
            System.out.println
               ("Binding server implementations to registry...");

            Naming.rebind("toaster", p1);
            Naming.rebind("microwave", p2);

            System.out.println
               ("Waiting for invocations from clients...");
         }
         catch(Exception e)
         {  System.out.println("Error: " + e);
         }
      }
}
```

- Step 4: Create an application at client side: ProductClient.java

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

public class ProductClient
{  public static void main(String[] args)
   {  System.setSecurityManager(new RMISecurityManager());
      String url = "rmi://localhost/";
         // change to "rmi://yourserver.com/"
         // when server runs on remote machine
         // yourserver.com
      try
      {  Product c1 = (Product)Naming.lookup(url + "toaster");
         Product c2 = (Product)Naming.lookup(url + "microwave");
         System.out.println(c1.getDescription());
         System.out.println(c2.getDescription());
      }
      catch(Exception e)
      {  System.out.println("Error " + e);
      }
      System.exit(0);
   }
}
```

Try these steps CAREFULLY to run the client/server application:

1. 1. Compile classes in sequence: Product.java, ProductImpl.java, ProductServer.java, ProductClient.java
2. Create Product stub by using command (the bin folder of JDK is specified in the environment variable Path)
   rmic ProductImpl
3. Put file ProductImpl_Stub.class at client side, in the same folder with ProductClient.class or put in a folder specified in the environment variable ClassPath
4. At server side, turn on rmiregistry by using coomand:
   start rmiregistry

5. Run ProductServer at server side
   java ProductServer
6. Run ProductClient at client side
   java ProductClient

2. Create a small online test application

Details.java

```
import java.sql.*;
import java.io.Serializable;
import java.util.Vector;
public class Details implements Serializable
{
   int qno;
   String question;
   String ans1;
   String ans2;
   String ans3;
   String ans4;
   String correctans;
   public Details(int mqno, String mquestion, String m_ans1,
String m_ans2, String m_ans3, String m_ans4, String mcorrectans)
   {
      qno = mqno;
      question = mquestion;
      ans1 = m_ans1;
      ans2 = m_ans2;
      ans3 = m_ans3;
      ans4 = m_ans4;
      correctans = mcorrectans;
   }
public static Details[] getInstance(Connection con,String sql)
throws SQLException
{
   Vector vect = new Vector();
   Statement stmt = con.createStatement( );
      ResultSet rs  = stmt.executeQuery(sql);
   while(rs.next())
   {
         int mqno = rs.getInt(1);
      String mquestion = rs.getString(2);
      String m_ans1 = rs.getString(3);
      String m_ans2 = rs.getString(4);
      String m_ans3 = rs.getString(5);
      String m_ans4 = rs.getString(6);
      String mcorrectans = rs.getString(7);
         Details data = new Details(mqno, mquestion, m_ans1,
m_ans2,  m_ans3, m_ans4, mcorrectans);
        vect.addElement(data);
```

```
   }
      rs.close();
   stmt.close();
      int num = vect.size();
      Details[] data = new Details[num];
      for(int i = 0; i < num; i++)
      {
         data[i] = (Details) vect.elementAt(i);
   }
   return data;
  }}
```

ServerInterface.java

```
//Remote Interface
import java.rmi.*;
import java.sql.SQLException;
import java.sql.*;
public interface ServerInterface extends Remote
{
   public     Details[]     getDetails(String     sql)     throws
RemoteException, SQLException ;
}
```

Server.java

```
// Server
import java.rmi.*;
import java.net.*;
import java.rmi.server.*;
import java.sql.*;
public   class   Server   extends   UnicastRemoteObject   implements
ServerInterface
{
   public static String url, sql;
   public static Connection con;
   public static Statement stmt;
   public Server() throws RemoteException
   {
         try
         {
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      }
      catch(ClassNotFoundException ce)
      {
            System.out.println(ce);
         }
      try
         {
```

```
         url = "jdbc:odbc:quiz";
         con = DriverManager.getConnection(url);
            stmt = con.createStatement();
      }
   catch(SQLException ce)
   {
         System.out.println(ce);
      }
   }
   public static void main(String [] args)
   {
         System.setSecurityManager(new RMISecurityManager());
         try
          {
            Server server = new Server();
            Naming.rebind("QuizServer", server);
               System.out.println("Server     is     ready     and
running.....");
         }
         catch (Exception ex)
         {
            ex.printStackTrace();
         }
   }

   public Details[] getDetails(String sql) throws SQLException
   {
      Details[]data = Details.getInstance(con,sql);
      System.out.println(data[0].question);
      return data;
   }
}
```

RemoteClient.java

```
//RMI Client
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.*;
import java.rmi.*;
public    class    RemoteClient    extends    Frame    implements
ActionListener, ItemListener
{
   public static final String URL = "rmi://127.0.0.1/QuizServer";
   String sql;
   Details[] data1;
   Details[] data2;
   int score=0;
   ServerInterface theServer;
```

```
   CheckboxGroup cg = new CheckboxGroup();
   Checkbox chkName1 = new Checkbox("",cg,false);
   Checkbox chkName2 = new Checkbox("",cg,false);
   Checkbox chkName3 = new Checkbox("",cg,false);
   Checkbox chkName4 = new Checkbox("",cg,false);
   Button nextbtn = new Button("Next");
   Button exitbtn = new Button("Exit");
         Panel panel1 = new Panel();
   Panel panel2 = new Panel();
   Label lblName1 = new Label("");
   Label lblName2 = new Label("");
   Label lblName3 = new Label("");
   Label lblName4 = new Label("");
   String qn, ans1, ans2, ans3, ans4, selectedans, correctans;
   int num = 1, qno, questions, index = 0;
   ResultSet rs;

   public RemoteClient(String title)
   {
      super(title);
      setLayout(new BorderLayout());
         panel1.setLayout(new GridLayout(7,1));
         lblName1.setFont(new Font("Helvetica",Font.BOLD,16));
      nextbtn.setFont (new Font("Lucida Regular",Font.BOLD,16));
      exitbtn.setFont(new Font("Lucida Regular",Font.BOLD,16));
      chkName1.addItemListener(this);
      chkName2.addItemListener(this);
      chkName3.addItemListener(this);
      chkName4.addItemListener(this);
      chkName1.setFont(new
Font("Helvetica",Font.BOLD|Font.ITALIC,14));
      chkName2.setFont(new
Font("Helvetica",Font.BOLD|Font.ITALIC,14));
      chkName3.setFont(new
Font("Helvetica",Font.BOLD|Font.ITALIC,14));
      chkName4.setFont(new
Font("Helvetica",Font.BOLD|Font.ITALIC,14));
      lblName2.setFont(new Font("Lucida",Font.BOLD,12));
         lblName3.setFont(new Font("Lucida",Font.BOLD,12));
      lblName3.setForeground(Color.blue);
      panel1.add(lblName2);
      panel1.add(lblName3);
      panel1.add(chkName1);
      panel1.add(chkName2);
      panel1.add(chkName3);
      panel1.add(chkName4);
      panel2.add(nextbtn);
      panel2.add(exitbtn);
      panel2.setBackground(Color.black);
      panel2.setForeground(Color.blue);
      panel1.setSize(500,200);
```

```
      panel2.setSize(500,300);
      add("Center",panel1);
      add("South",panel2);
      pack();
      nextbtn.addActionListener(this);
         exitbtn.addActionListener(this);
         try
      {
         theServer =(ServerInterface) Naming.lookup(URL);
      }
      catch(Exception e)
      {
       }

      try
         {
         sql = "Select * from question order by Qno";
         data2 = theServer.getDetails(sql);
            questions = data2.length;
      display();
      }
      catch(Exception ce)
         {
            System.out.println(ce);
         }
   }

   public void actionPerformed(ActionEvent e)
   {
      if (index==(questions-2))
      //Last but one question
         nextbtn.setEnabled(false);
      if (e.getSource().equals(nextbtn))
      {
         index++;
         display();
      }
         else
         {
         if(e.getSource().equals(exitbtn))
         {
            System.out.println("You scored : " + score);
            System.exit(0);
         }
      }
   }
   public void itemStateChanged(ItemEvent e)
   {
      String answer = ((Checkbox)e.getSource()).getLabel();

      if(answer.trim().equals(data2[index].correctans.trim()))
```

```
        score+=25;
   }

   public static void main(String args[])
   {
      RemoteClient quizclient = new RemoteClient("Online Quiz");
      quizclient.setSize(500,300);
      quizclient.setVisible(true);
   }
   public void display()
   {
      qno = data2[index].qno;
      qn = data2[index].question;
      ans1 = data2[index].ans1;
      ans2 = data2[index].ans2;
      ans3 = data2[index].ans3;
      ans4 = data2[index].ans4;
      correctans = data2[index].correctans;
      lblName2.setText("Question "+qno);
      lblName3.setText(qn);
      chkName1.setLabel(ans1);
      chkName2.setLabel(ans2);
      chkName3.setLabel(ans3);
         chkName4.setLabel(ans4);
      chkName1.setState(false);
         chkName2.setState(false);
      chkName3.setState(false);
      chkName4.setState(false);
   }
}
```

3. Write a program to send email to an SMTP server

Steps:

1. Open a socket to your host.

```
Socket s = new Socket("mail.yourserver.com", 25); // 25 is SMTP
hPrintWriter out = new PrintWriter(s.getOutputStream());
```

2. Send the following information to the print stream:

```
HELO sending host
MAIL FROM: <sender e-mail address>
RCPT TO: <>recipient e-mail address>
DATA
mail message
(any number of lines)
.
QUIT
```

The SMTP specification (RFC 821) states that lines must be terminated with `\r` followed by `\n`.

Do It Yourself

5.1. Do workshop 8, 9

5.2. Write a program to send file between two machine by using RMI

Self-study Samples

1. Available samples
   - RMI broadcast
   - WareHouse
   - WareHouseApplet

2. JavaPassion.com

3. Java2s.com