ADVANCED JAVA LAB MANUAL

INDEX

RATIONALE BEHIND ADVANCED JAVA LAB

If technology touches life, chances are, so does Java technology. Invented by Sun Microsystems in 1995, Java technology has become the essential ingredient of the digital experience for hundreds of millions of people in all walks of life, all over the planet.

Java software powers the onboard computers in toys, cars, planes, rockets, and even the NASA Mars Rover. It brings interactivity to the Internet, real-time graphics to television, instant imaging to cameras, and multi-player games to mobile phones and desktop PCs. It connects the largest enterprises and smallest businesses to their employees, customers, and data. And it secures the vast majority of electronic transactions in retail, finance, government, science, and medicine. In short, Java technology goes everywhere you go.

It's no wonder that Java technology has become the most powerful force in software and the most prevalent software in technology. In fact, it is the software of choice for more software engineers than any other brand of software.

Java is the most efficient, fast, secure, animated, compatible, and reliable software.

Java Technology Facts

- Java technology was invented by Sun Microsystems and celebrated its 10-year Birthday in March 2005
- The Java platform specifications and compatibility standards are controlled by the independent industry members of the Java Community Process
- Java software runs on more types of consumer and embedded devices, smart cards, ATMs, thin clients, PCs, servers, and mainframes than any other software
- Java applications are more resistant to viruses than any other popular programming language
- Today's five million Java developers are the largest community of software developers
- The Java economy includes 1.65 billion smart cards, 800 million PCs shipped with Java, 1.2 billion Java Powered phones (source: Ovum), and over 180 telecom providers who deploy Java technology

After completing Advanced Java Lab students will be able to develop program in Java for following applications

Java Data Base Connectivity: A Data Base can be accessed from program

Servlets: For development of web basd components

Java Beans: Java Beans are, quite simply, reusable controls written in Java, for Java application development. They let you visually assemble components and dynamically change properties

Java Server Pages: to dynamically generate HTML, XML or other types of documents in response to a Web client request.

Remote Method Invocation: RMI provides the mechanism by which the server and the client communicate and pass information back and forth.

As we all know there is great demand of Java in industry so definitely student will benefited by advanced java Lab.It can help them to blossom their future

HARDWARE REQUIRED

P-IV/III PROCESSOR

HDD 40GB

RAM 128MB or above

SOFTWARE REQUIRED

Window 98/2000/ME/XP

Java Entrprise Edition 5

SQL Server / MySql

Web Server(Apache Tomcat/JSWDK)

Net Beans IDE

PROGRAM LIST

Advanced Java Lab is newly introduced Lab in 8th Semester.There is no particular Program List specified by University
In advanced java lab we shall have two programs based on each :

- JDBC
- SERVLETS
- JSP
- JAVA BEANS
- RMI

Experiment 1

THEORY AND CONCEPTS

JDBC

Call-level interfaces such as JDBC are programming interfaces allowing external access to SQL database manipulation and update commands. They allow the integration of SQL calls into a general programming environment by providing library routines which interface with the database. In particular, Java based JDBC has a rich collection of routines which make such an interface extremely simple and intuitive.
what happens in a call level interface: You are writing a normal Java program. Somewhere in the program, you need to interact with a database. Using standard library routines, you open a connection to the database. You then use JDBC to send your SQL code to the database, and process the results that are returned. When you are done, you close the connection.

PROGRAMS

Program1

Purpose: A Program to execute select query using JDBC

```
import java.sql.*;

class SelectFromPer {

public static void main(String argv[]) {

try {

// Load the JDBC-ODBC bridge
Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");

// specify the ODBC data source's URL
String url = "jdbc:odbc:SSPer";

// connect
Connection con = DriverManager.getConnection(url,"North","Ken");

// create and execute a SELECT
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery
("SELECT Surname,FirstName,Category FROM Per");

System.out.println("Class is SelectFromPer\n");
// traverse through results
System.out.println("Found row:");

while (rs.next()) {

// get current row values
String Surname = rs.getString(1);
String FirstName = rs.getString(2);
int Category = rs.getInt(3);

// print values
System.out.print (" Surname=" + Surname);
System.out.print (" FirstName=" + FirstName);
System.out.print (" Category=" + Category);
System.out.print(" \n");
}
```

```
// close statement and connection
stmt.close();
con.close();
} catch (java.lang.Exception ex) {

ex.printStackTrace();
}

}

}
```

PROGRAM 2

A Program to Update Customer Information

```
import java.sql.* ;

class JDBCUpdate
{
 public static void main( String args[] )
 {
  try
     {
      // Load the database driver
      Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" ) ;

      // Get a connection to the database
      Connection conn = DriverManager.getConnection( "jdbc:odbc:Database" ) ;

      // Print all warnings
      for( SQLWarning warn = conn.getWarnings(); warn != null; warn =
warn.getNextWarning() )
         {
          System.out.println( "SQL Warning:" ) ;
          System.out.println( "State : " + warn.getSQLState() ) ;
          System.out.println( "Message: " + warn.getMessage()   ) ;
          System.out.println( "Error : " + warn.getErrorCode() ) ;
         }

      // Get a statement from the connection
      Statement stmt = conn.createStatement() ;
```

```
      // Execute the Update
      int rows = stmt.executeUpdate( "UPDATE Cust SET CUST_NO = 9842
WHERE CUST_NO = 9841" ) ;

      // Print how many rows were modified
      System.out.println( rows + " Rows modified" ) ;

      // Close the statement and the connection
      stmt.close() ;
      conn.close() ;
     }
  catch( SQLException se )
     {
      System.out.println( "SQL Exception:" ) ;

      // Loop through the SQL Exceptions
      while( se != null )
         {
          System.out.println( "State : " + se.getSQLState() ) ;
          System.out.println( "Message: " + se.getMessage()   ) ;
          System.out.println( "Error : " + se.getErrorCode() ) ;

          se = se.getNextException() ;
         }
     }
  catch( Exception e )
     {
      System.out.println( e ) ;
     }
 }
}
```

SERVLETS

Servlets are the Java platform technology of choice for extending and enhancing Web servers. Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. And unlike proprietary server extension mechanisms (such as the Netscape Server API or Apache modules), servlets are server- and platform-independent. This leaves you free to select a "best of breed" strategy for your servers, platforms, and tools.

Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. Servlets can also access a library of HTTP-specific calls and receive all the benefits of the mature Java language, including portability, performance, reusability, and crash protection.

Today servlets are a popular choice for building interactive Web applications. Third-party servlet containers are available for Apache Web Server, Microsoft IIS, and others. Servlet containers are usually a component of Web and application servers, such as BEA WebLogic Application Server, IBM WebSphere, Sun Java System Web Server, Sun Java System Application Server, and others.

PROGRAMS

PROGRAM 1

Purpose: A simple servlet that just generates plain text

```
package hall;

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");
  }
}
```

PROGRAM 2

Purpose: A Program which displays cookie id

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

public class CookieExample extends HttpServlet {
```

```
    public void doGet(HttpServletRequest request, HttpServletResponse
response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // print out cookies

        Cookie[] cookies = request.getCookies();
        for (int i = 0; i < cookies.length; i++) {
            Cookie c = cookies[i];
            String name = c.getName();
            String value = c.getValue();
            out.println(name + " = " + value);
        }

        // set a cookie

        String name = request.getParameter("cookieName");
        if (name != null && name.length() > 0) {
            String value = request.getParameter("cookieValue");
            Cookie c = new Cookie(name, value);
            response.addCookie(c);
        }
    }
}
```

JAVA SERVER PAGES

JavaServer Pages (JSP) is a Java technology that allows software developers to dynamically generate HTML, XML or other types of documents in response to a Web client request. The technology allows Java code and certain pre-defined actions to be embedded into static content.

The JSP syntax adds additional XML-like tags, called JSP actions, to be used to invoke built-in functionality. Additionally, the technology allows for the creation of JSP tag libraries that act as extensions to the standard HTML or XML tags. Tag libraries provide a platform independent way of extending the capabilities of a Web server.

JSPs are compiled into Java Servlets by a JSP compiler. A JSP compiler may generate a servlet in Java code that is then compiled by the Java compiler, orit may generate byte code for the servlet directly.

JSP technology enables Web developers and designers to rapidly develop and easily maintain, information-rich, dynamic Web pages that leverage existing business systems. As part of the Java technology family, JSP technology enables rapid development of Web-based applications that are platform independent. JSP technology separates the user interface from content generation, enabling designers to change the overall page layout without altering the underlying dynamic content

PROGRAMS

PROGRAM 1

Purpose A program for basic arithmetic functions

```
<html>
  <head>
    <title>JSP 2.0 Expression Language - Basic Arithmetic</title>
  </head>
  <body>
    <h1>JSP 2.0 Expression Language - Basic Arithmetic</h1>
    <hr>
    This example illustrates basic Expression Language arithmetic.
    Addition (+), subtraction (-), multiplication (*), division (/ or div),
    and modulus (% or mod) are all supported. Error conditions, like
    division by zero, are handled gracefully.
    <br>
    <blockquote>
      <code>
        <table border="1">
          <thead>
              <td><b>EL Expression</b></td>
              <td><b>Result</b></td>
            </thead>
            <tr>
              <td>\${1}</td>
```

```
        <td>${1}</td>
      </tr>
      <tr>
        <td>\${1 + 2}</td>
        <td>${1 + 2}</td>
      </tr>
      <tr>
        <td>\${1.2 + 2.3}</td>
        <td>${1.2 + 2.3}</td>
      </tr>
      <tr>
        <td>\${1.2E4 + 1.4}</td>
        <td>${1.2E4 + 1.4}</td>
      </tr>
      <tr>
        <td>\${-4 - 2}</td>
        <td>${-4 - 2}</td>
      </tr>
      <tr>
        <td>\${21 * 2}</td>
        <td>${21 * 2}</td>
      </tr>
      <tr>
        <td>\${3/4}</td>
        <td>${3/4}</td>
      </tr>
      <tr>
        <td>\${3 div 4}</td>
        <td>${3 div 4}</td>
      </tr>
      <tr>
        <td>\${3/0}</td>
        <td>${3/0}</td>
      </tr>
      <tr>
        <td>\${10%4}</td>
        <td>${10%4}</td>
      </tr>
      <tr>
        <td>\${10 mod 4}</td>
        <td>${10 mod 4}</td>
      </tr>
<tr>
  <td>\${(1==2) ? 3 : 4}</td>
  <td>${(1==2) ? 3 : 4}</td>
```

```
    </tr>
          </table>
      </code>
    </blockquote>
  </body>
</html>
```

PROGRAM 2

Purpose A Program to display a String

```
<html>
  <head>
    <title>JSP 2.0 Examples - Hello World SimpleTag Handler</title>
  </head>
  <body>
    <h1>JSP 2.0 Examples - Hello World SimpleTag Handler</h1>
    <hr>
    <p>This tag handler simply echos "Hello, World!" It's an example of
    a very basic SimpleTag handler with no body.</p>
    <br>
    <b><u>Result:</u></b>
    <mytag:helloWorld/>
  </body>
</html>
```

PROGRAM 3

Purpose :A Program to create check boxes

```
<html>
<body bgcolor="white">
<font size=5 color="red">
<%! String[] fruits; %>
<jsp:useBean id="foo" scope="page" class="checkbox.CheckTest" />

<jsp:setProperty name="foo" property="fruit" param="fruit" />
<hr>
The checked fruits (got using request) are: <br>
<%
        fruits = request.getParameterValues("fruit");
```

```
%>
<ul>
<%
   if (fruits != null) {
          for (int i = 0; i < fruits.length; i++) {
%>
<li>
<%
             out.println (util.HTMLFilter.filter(fruits[i]));
          }
        } else out.println ("none selected");
%>
</ul>
<br>
<hr>

The checked fruits (got using beans) are <br>

<%
                fruits = foo.getFruit();
%>
<ul>
<%
   if (!fruits[0].equals("1")) {
          for (int i = 0; i < fruits.length; i++) {
%>
<li>
<%
                  out.println (util.HTMLFilter.filter(fruits[i]));
          }
        } else out.println ("none selected");
%>
</ul>
</font>
</body>
</html>
```

JAVA BEANS

A Java Bean is the name trademarked by Sun and given to a Java class that adheres to a specific and well-defined set of interface specifications.

According to JavaSoft,

"A Java Bean is a reusable software component that can be manipulated visually in a builder tool."

Beans are "capsules" of code, each designed for a specific purpose. The advantage of Java Beans over standard programming controls is that Beans are independent. They are not specific to operating systems or development environments. A Bean created in one development environment can be easily copied and modified by another. This allows Java Beans greater flexibility in enterprise computing, as components are easily shared between developers

The following five attributes are common to Beans.

- Properties
- Customization
- Persistence
- Events
- Introspection

we are usually referring to the following three attributes of the class:

- Properties
- Methods
- Events

PROGRAMS

PROGRAM 1

Purpose: A program to generates plain text

```
import java.awt.Color;
import java.beans.XMLDecoder;
import javax.swing.JLabel;
import java.io.Serializable;
public class SimpleBean extends JLabel implements Serializable
{
   public SimpleBean()
   {
     setText( "Hello world!" );
     setOpaque( true );
```

```
    setBackground( Color.RED );
    setForeground( Color.YELLOW );
    setVerticalAlignment( CENTER );
    setHorizontalAlignment( CENTER );
  }

}
```

Remote Method Invocation (RMI)

Remote Method Invocation (RMI) is the object equivalent of Remote Procedure Calls (RPC). While RPC allows you to call procedures over a network, RMI invokes an object's methods over a network.

In the RMI model, the server defines objects that the client can use remotely. The clients can now invoke methods of this remote object as if it were a local object running in the same virtual machine as the client. RMI hides the underlying mechanism of transporting method arguments and return values across the network. In Java-RMI, an argument or return value can be of any primitive Java type or any other Serializable Java object.

PROGRAMS

PROGRAM 1

Purpose :A Program on Stock Market

1. Develop your Remote Interface

```
package SimpleStocks;
import java.util.*;
import java.rmi.*;

public interface StockMarket extends java.rmi.Remote {
  float get_price( String symbol ) throws RemoteException;
}
```

2. Implement your Java/RMI Server

```
package SimpleStocks;
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;

public class StockMarketImpl
  extends UnicastRemoteObject implements StockMarket {

  public StockMarketImpl( String name ) throws RemoteException {
    try {
      Naming.rebind( name, this );
    }
    catch( Exception e ) {
      System.out.println( e );
    }
  }

  public float get_price( String symbol ) {
    float price = 0;
    for( int i = 0; i < symbol.length(); i++ ) {
      price += (int) symbol.charAt( i );
    }
    price /= 5;
    return price;
  }

}
```

3 Implement an Application that creates your Server

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

public class StockMarketServer {

  public static void main(String[] args) throws Exception {
    if(System.getSecurityManager() == null) {
      System.setSecurityManager( new RMISecurityManager() );
    }
    StockMarketImpl myObject = new StockMarketImpl( "NASDAQ" );
```

```
    System.out.println( "RMI StockMarketServer ready..." );
  }
}
```

4. Develop your security policy file.

Grant

```
 {
 permission java.security.AllPermission "", "";
};
```

REFERENCES

1. **Core Java 2, Vol 11 Advanced features, 7th edition by Cay Horetmann, Gary Cornell Pearson Publisher**
2. **Professional Java Prigramming by Brett Spell, Wrox Publication**
3. **Advanced Java 2 Platform, How to Program, 2nd Edition, Harvey.M.Dietel, Prentice Hall**
4. **Java Complete Reference,Herbert Schildt**

WebSites

http://java.sun.com
http://www.javasoft.com
http://splash.java.com
http://www.developer.com
http://www.javology.com/javology

NEW IDEAS

Apart from this Lab I suggest some changes in syllabus provided by University that should be considered in future

1. Program List of any new Lab should be mentioned because java is very vast language it is not possible to cover whole topic in one semester

2. Some Core Java Concepts should mentioned in Practical syllabus because it is prerequisite for advanced java lab

General FAQ

JDBC

Once I have the Java 2 SDK, Standard Edition, from Sun, what else do I need to connect to a database

What's the JDBC 3.0 API?

How do I start debugging problems related to the JDBC API?

Are all the required JDBC drivers to establish connectivity to my database part of the JDK?

SERVLETS

How do I get started with Servlets using Tomcat?

How do Servlets compare to CGI?

How do I upload a file to my Servlet?

How do I debug a Servlet?

JSP

What is JavaServer Pages technology?

How does the JavaServer Pages technology work?

Which web servers support JSP technology?

How is a JSP page invoked and compiled?

JAVA BEANS

What is a Bean? Why isn't a Bean an Applet?

Is JavaBeans a complete component architecture?

What are the security implications for downloading Beans over the Internet?

What is the relationship between Sun's JFCs and JavaBeans?

RMI

Does RMI require to use an HTTP server?

Will there be debugging mechanisms built into RMI?

Where can I find a list of system properties that might be useful for implementing and debugging RMI applications?

How do RMI clients contact remote RMI servers?