**Module Objectives:**

* Become proficient in the RDz Java Perspective
* Be able to extract data from a DB2 table using Java

**Creating a Java program to access DB2 9.1 on System z**

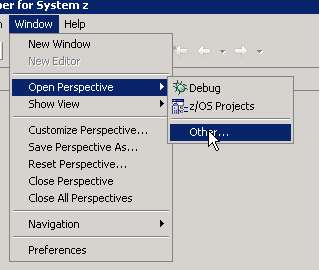
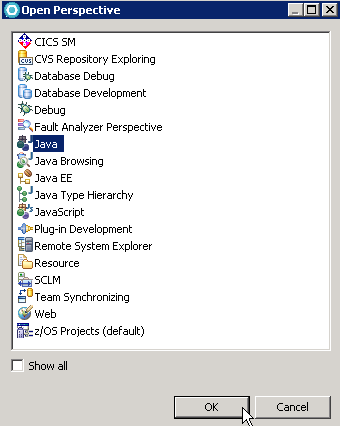
## Create a Java Application

This example creates a Java application that runs against DB2 9.1 on the host (z10). One must create the DB2 9.1 table before this project will run.

Open RDz and ensure you have a z/OS connection (Remote System). To create a Java Project, you will need to open the Java Perspective.

**To switch to the Java Perspective:**

1. Perspectives that have been opened in RDz are kept usually as expandable tabs in the upper right corner of your Remote System perspective. If you have not previously opened the Java perspective, click **Window => Perspective => other**. A dialog box displays.

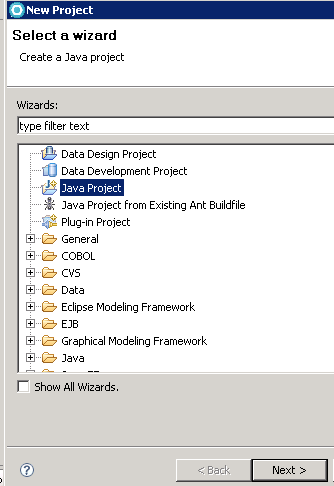


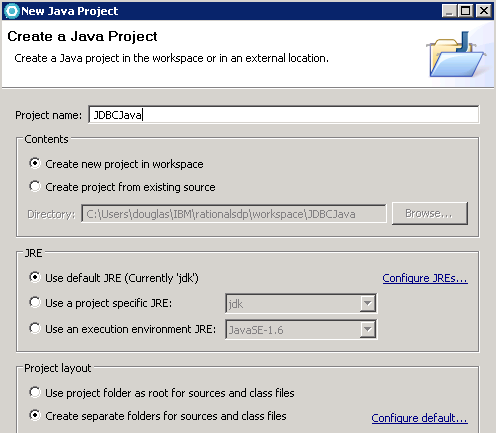
2. Select **Java** and click **OK**. RDz automatically arranges the windows, views, and editors for Java development tasks. Package Explorer (only Packa displayed) is shown below.

****

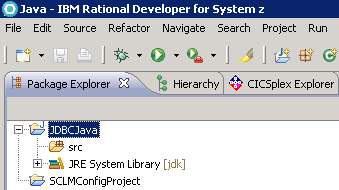
**To create a Java project:**

1. Click **File** => **New** => **Project** => **Java Project**



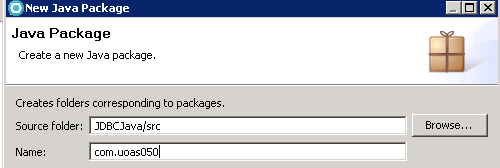
2. Click the Next button and name the new project –JDBCJava for this example.

Accept the default values and click the Finish button (not shown). The result is shown below.

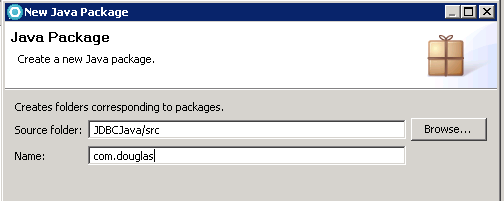


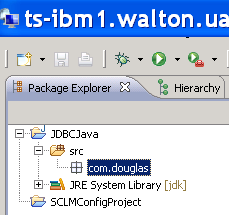
**To create a Java package within the project:**

1. Right-click the project (JDBC Java in this case) in the Package Explorer window and select **New** => **Package**



3. Enter a name for the new package such as com.acctid (com.douglas in this case) and click the Finish button (not shown)



The package is created as shown

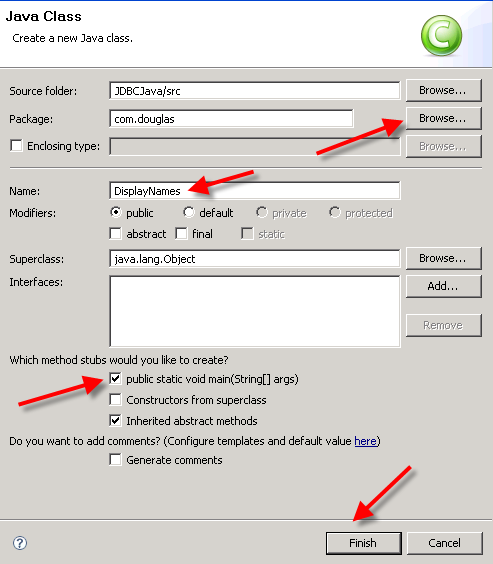
**To create a Java class:**

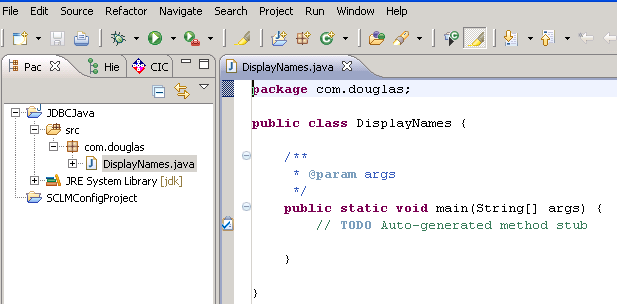
1. Right-click the project (JDBCJava) in the Package Explorer window and select **New** => **Class**.

3. Use the Browse button to select your package, for example com.douglas

4. Enter your class name; for example, DisplayNames

5. Click the box for **public static void main** and then click the Finish button.





RDz creates a Java programming template as shown below.

**The Program--copy the following Java Program into the DisplayNames Java program**

The package statement should be in your Java editor—my specific package is

package com.douglas;

Enter the following import statements--between the package name and the class--to import the Java classes needed for this application

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.sql.Statement;

The class and main subroutine headers are already created in the Java Editor. Thus, do not copy and paste them.

----------------------------------------------------

public class DisplayNames {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

Copy the following code into the main program stub—note some comments that show the DB2 9.1 table structure

System.out.println("<<< Entering DisplayNames >>>");

if (args.length != 3) {

System.out.println("required: location\_name<userid<password>");

System.exit(0);

}

try {

Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();

String url = "jdbc:db2://stplex4a.svl.ibm.com:8028/" + args[0];

Connection connection = DriverManager.getConnection(url,

args[1], args[2]);

showPeople(connection);

connection.close();

} catch (SQLException ex) {

System.out.println("\n\*\*\*SQLExecption 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 langEx) {

System.out.println("\n\*\*\*java.lang.Execption caught\*\*\*\n");

langEx.printStackTrace();

}

}

/\*\*

\* showPeople method

\*

\* CREATE TABLE FRIENDZ (

\* FNAME CHAR(10) NOT NULL,

\* LNAME CHAR(10) NOT NULL,

\* PHONE CHAR(10) NOT NULL,

\* EMAIL CHAR(30) NOT NULL

\* )

\* AUDIT NONE

\* DATA CAPTURE NONE

\* CCSID EBCDIC;

\*

\*\*/

protected static void showPeople(Connection connection) throws SQLException {

System.out.println("<<< showPeople method >>> ");

Statement stmt = connection.createStatement();

String sql = "SELECT LNAME, FNAME FROM DOUGLAS.FRIENDZ " +

"ORDER BY LNAME ";

boolean hasResultSet = stmt.execute(sql);

if (hasResultSet) {

showResultSet(stmt);

} else {

System.out.println("There was no result set to display.");

}

}

/\*\*

\* Prints a result set

\*\*/

protected static void showResultSet(Statement stat) throws SQLException {

System.out.println("<<< showResultSet method >>> ");

ResultSet result = stat.getResultSet();

ResultSetMetaData metaData = result.getMetaData();

int columnCount = metaData.getColumnCount();

for (int i = 1; i <= columnCount; i++) {

if (i > 1) {

System.out.print(", ");

}

System.out.print(metaData.getColumnLabel(i));

}

System.out.println();

System.out.println("============================");

while (result.next()) {

for (int col = 1; col <= columnCount; col++) {

if (col == 1) {

System.out.print(result.getString(col) + ", ");

} else /\* col > 1 \*/{

System.out.print(result.getString(col) + " ");

}

}

System.out.println();

}

result.close();

Notes:

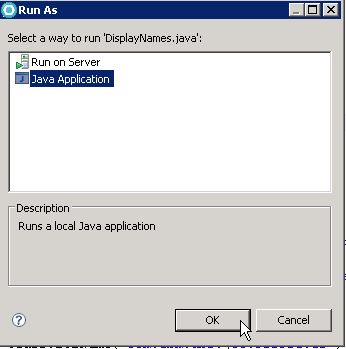
* Two closing} characters are created with the Java template and thus are not shown at the end of this program. Otherwise, you will have two extra ones in your code and thus an error.
* There are two highlighted places in your code; indications of required changes  
  \*\*The JDBC string should be as shown below for our z10 system\*\*

## "jdbc:db2://130.184.26.148:446/"

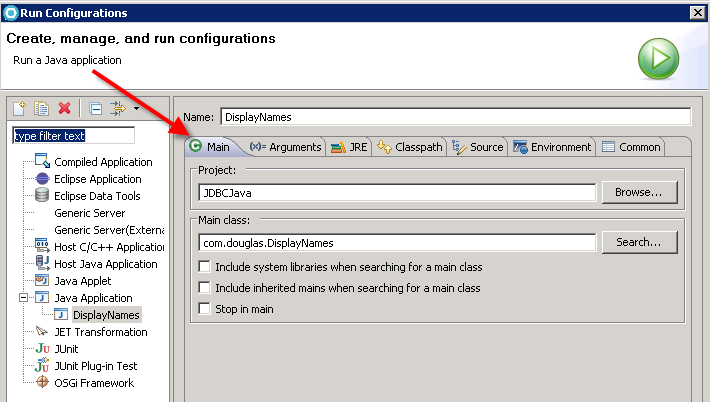
* Also, change the DB2 9.1 table qualifier to be your account
* Save your Java program by clicking the disk icon
* In Java perspective, an Outline window is created to assist in program development. As shown below, clicking on an entry in the Outline window (DisplayNames in this example) will move to that location in the Java code.

## 

## Setup a Run Configuration

To create a run configuration, click Run from the main menu and then click the Run option. Select Java application and then click the OK button.

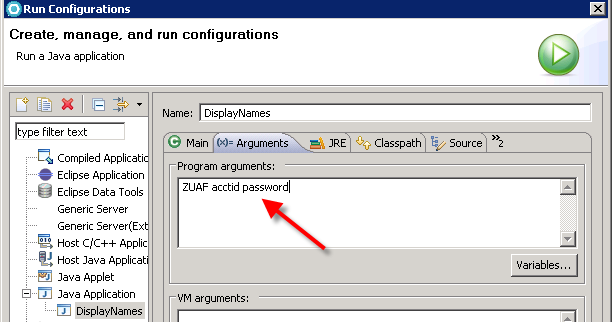
Click Run from the main menu and select Run Configurations to open the following Run Configuration dialog with the Main tab as the default tab. Browse for the Project: and Main class: names. This example uses JDBCJava and com.douglas.DisplayNames.



Click the Arguments tab—this is where you will provide database information.

Note the parameters passed to the main method which are for this example

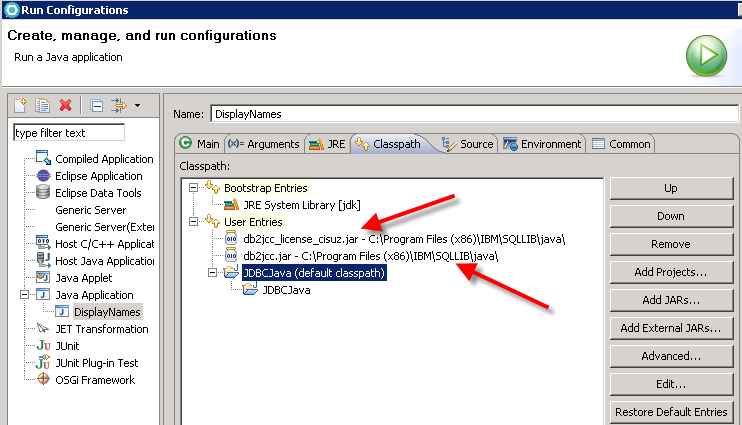
* Database name --ZUAF
* AcctID-- douglas
* Password—yourpassword



Yours should be:

ZUAF uoasxxx yourpass

Click the Classpath tab. The Classpath is very important and must contain the location of the DB2 driver along with the License Jar for DB2/z. Under User Entries, click JDBCJava and then click the Add External Jars… button as shown below. Browse and select each of the entries shown with the red arrows.

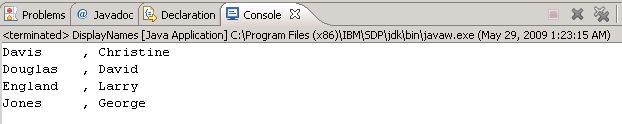


***The classpaths for the remote terminal Server are the same as those indicated above—these files must be on your local machine to make this app run. . . they are probably not on your laptop or in most labs. . .***

## Click the Apply button at the bottom of the Dialog window.

## Run the application within RDz

Click the Run button; this will establish the environment defined by the Run Configuration. The output in the Console window will be similar to below—depending on the data in your database.



If you have an error in your program, there are several ways to rerun the program. One way is to right-click on the program in the Outline tab; Run As 🡺 Run Configurations 🡺 Run (this run will from the Classpaths tab shown above). The most intuitive way is to click the dropdown on the green go button; select Rus As and then Java Application as shown below.

