Laboratory Manual 5.4
5.4 Design and Build a Report

Objectives
After completing this lab session, you should be able to do the following:-

- ➢ Install NetBeans Reporting tool plugins
- ➢ Become familiar with Java open source reporting tools
- ➢ Design and build report using iReport tool
- ➢ Call and integrate JasperReports XML/jasper file in Java application

Required Software
- ➢ **iReport 5.5.0 plugin**
- ➢ **NetBeans IDE 7.4 or later**

5.4.1 Introduction
Report is a Meaningful, well summarized nicely formatted way of presenting data from the database. Usually the routine activities are automated and data summarized into a decision-supporting format "Reports". Reports act as wonders when it converts the usual messy data into attractive charts, and graphical representations.
Open source JAVA reporting tools,
- ➢ JasperReports
    - – is an engine that takes an XML file and forms a report out of that file using the data source specified in the XML file.
- ➢ iReport and
    - – is a visual tool to obtain XML files for JasperReports.
- ➢ JFreeChart
    - – is a free Java class library for generating charts.
    - – Installing iReport Plugins

5.4.2 Installing iReport plugin

     i.   Get the from your lab Assistant/Instructor or Download the latest version iReport plugin (as of this writing is 5.5.0) from www.netbens.org that is compatible to your NetBeans IDE version (Figure 1)

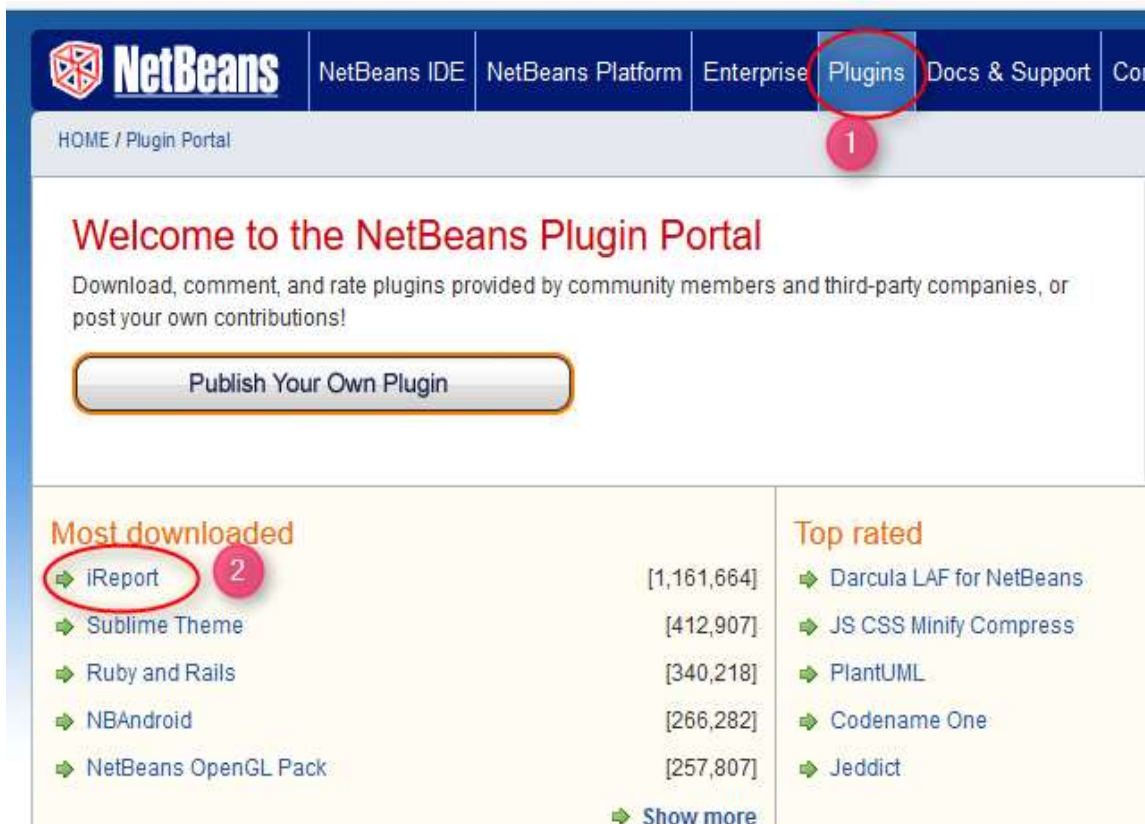   ii.   Go to www.netbens.org, click "Plugins" (Figure 1[1]) link, click "iReport" (Figure 1[2])link

Figure 1

     iii.   Select Your Netbeans IDE version (Figure 2[1]) and download iReport plugins for your IDEDownload (Figure 2[2])

Figure 2

iv.   Unpack the file "1383067774_iReport-5.5.0-plugin" (Figure 3[1]), you will get four files (Figure 3[2])
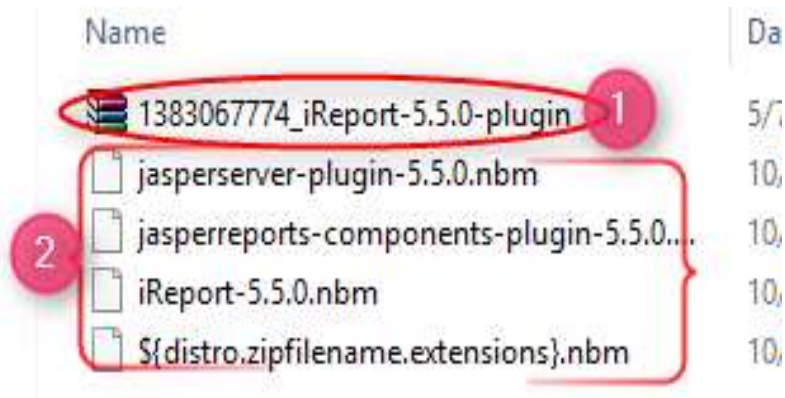
Figure 3

v.   Open your NetBeans IDE,
 i.   Go to "**Tools**" menu, select "**plugin**" (Figure 4)

Figure 4

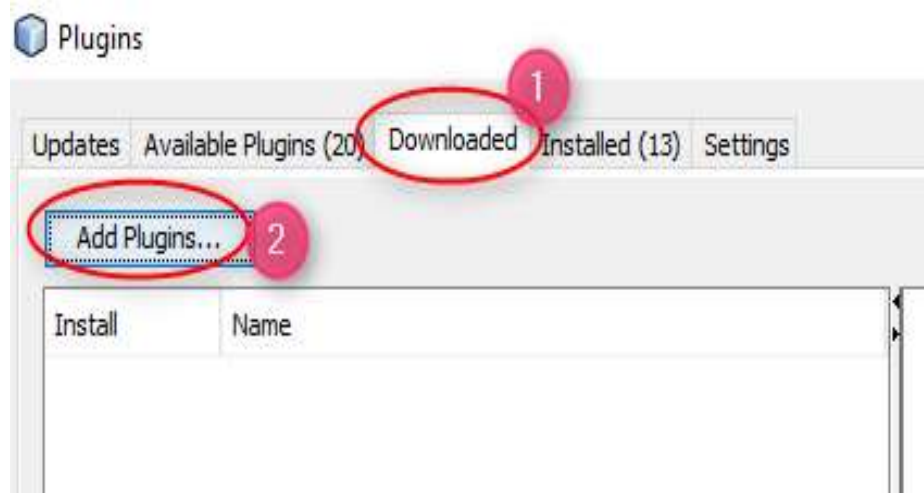 ii.   Click "**Download**" tab, click "**Add plugin…**" (Figure 5)

Figure 5

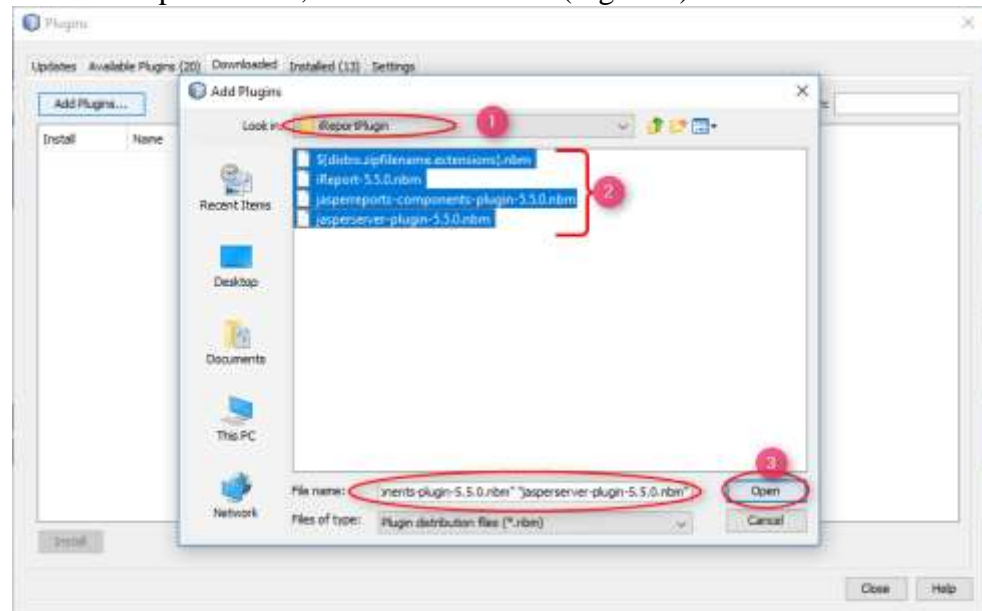iii. Browse to the unpacked file, select all four files (Figure 6)

Figure 6
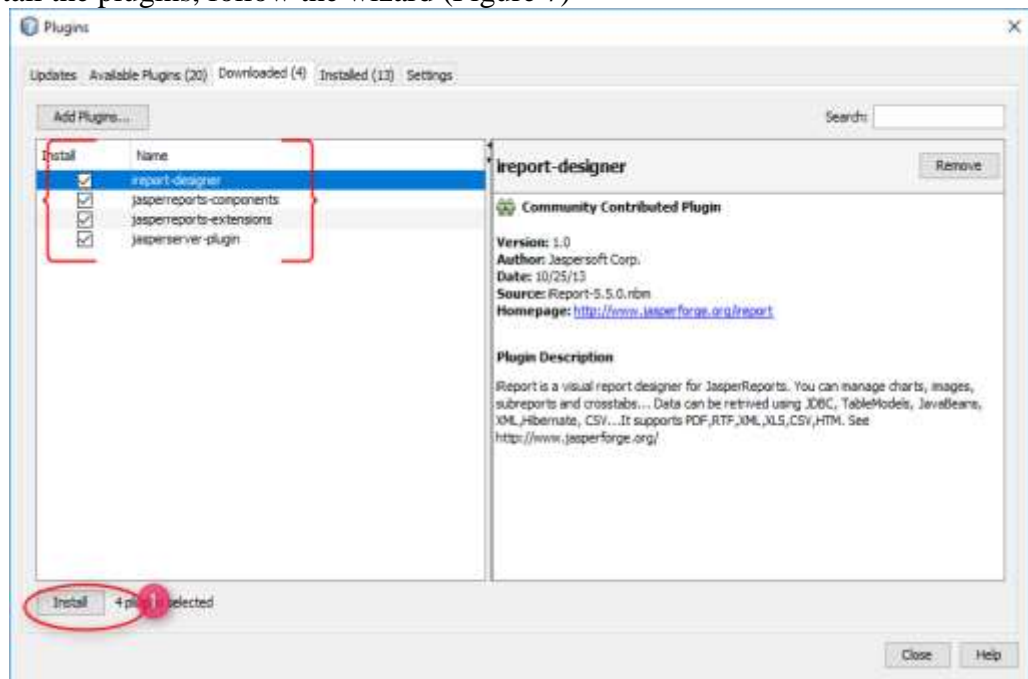
iv. Install the plugins, follow the wizard (Figure 7)

Figure 7

v.   Follow the installation wizard. (Figure 8)
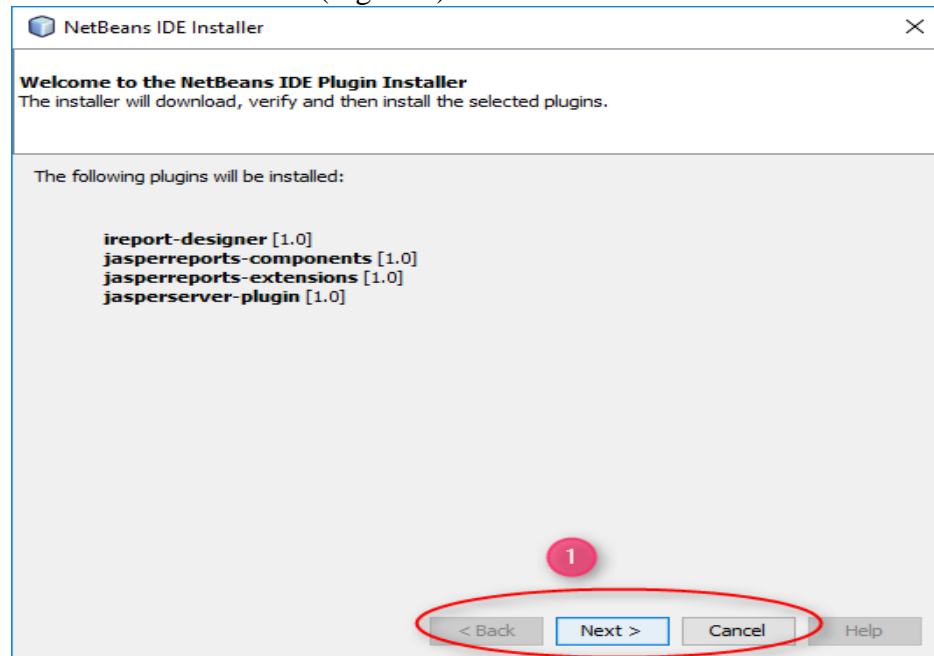
Figure 8

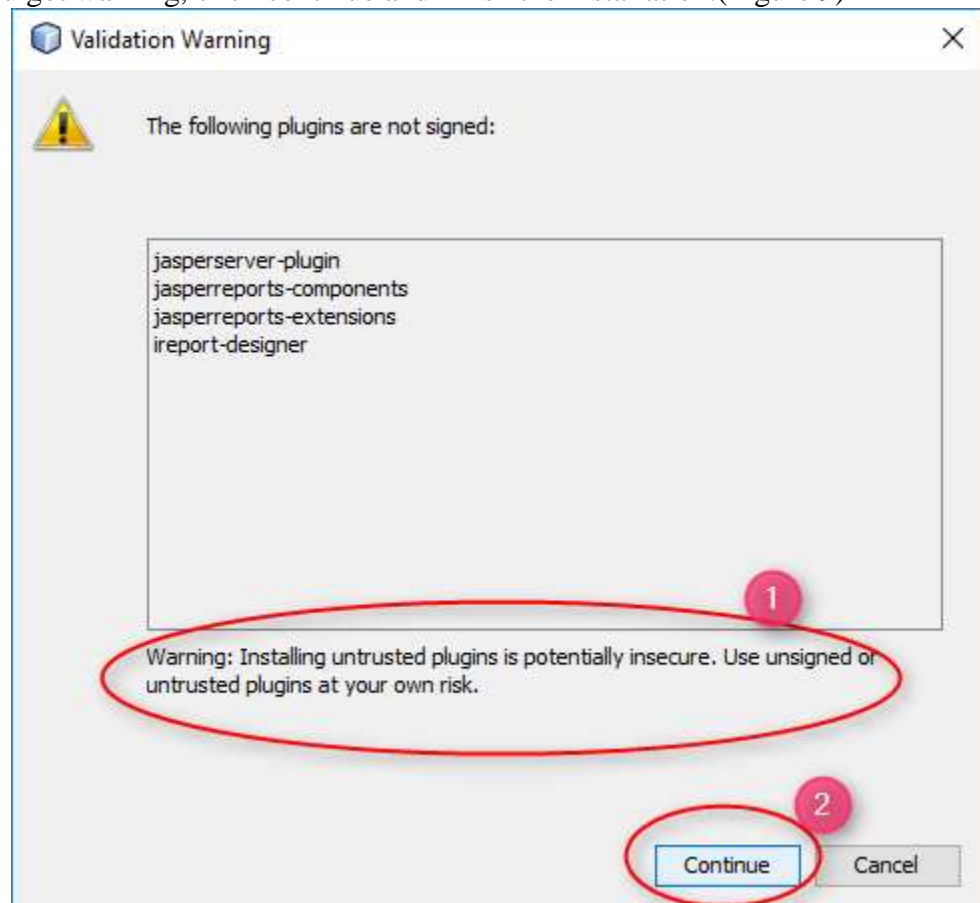vi. If you got warning, click continue and finish the installation.(Figure 9)

Figure 9

vii.   Restart NetBeans IDE (close and Reopen)
viii. The NetBeans IDE brings "Welcome window"(Figure 10 [1]) and steps how to Get started in designing report using iReport (Figure 10[2])
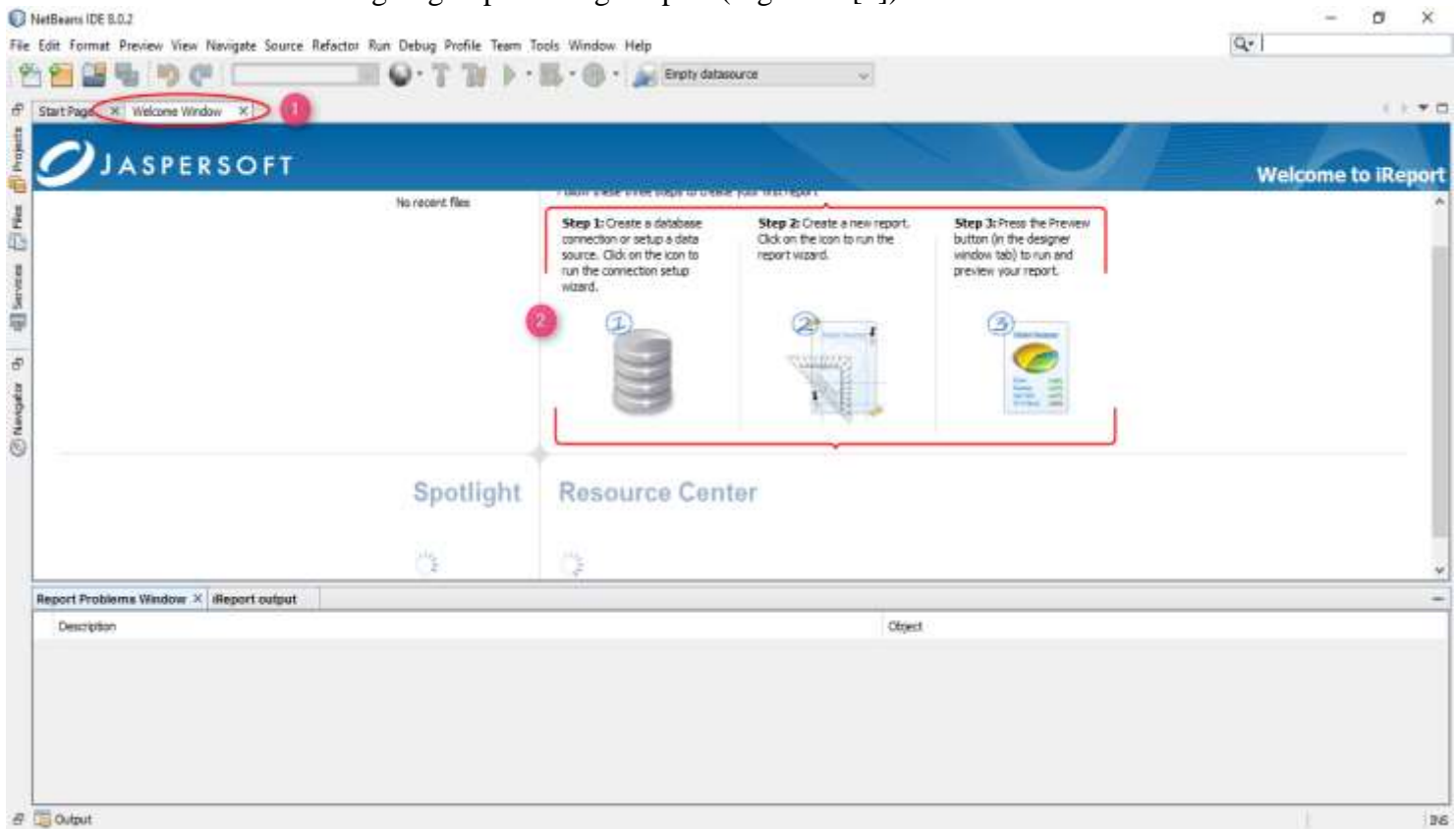

Figure 10 : iReport welcome page in NetBeansIDE

5.4.3 Using iReport to Design a Report
Start designing a report using iReport plugin features
  i.   Open the "RegistrarSystem" project
  ii.   Create a package called "**report**" (it is optional, recommended)
  iii. Right click on your project(package"report), **New**, select **Others**
  iv.   Select **Report** category, Select **Report wizard** (Figure 11)**, Next**
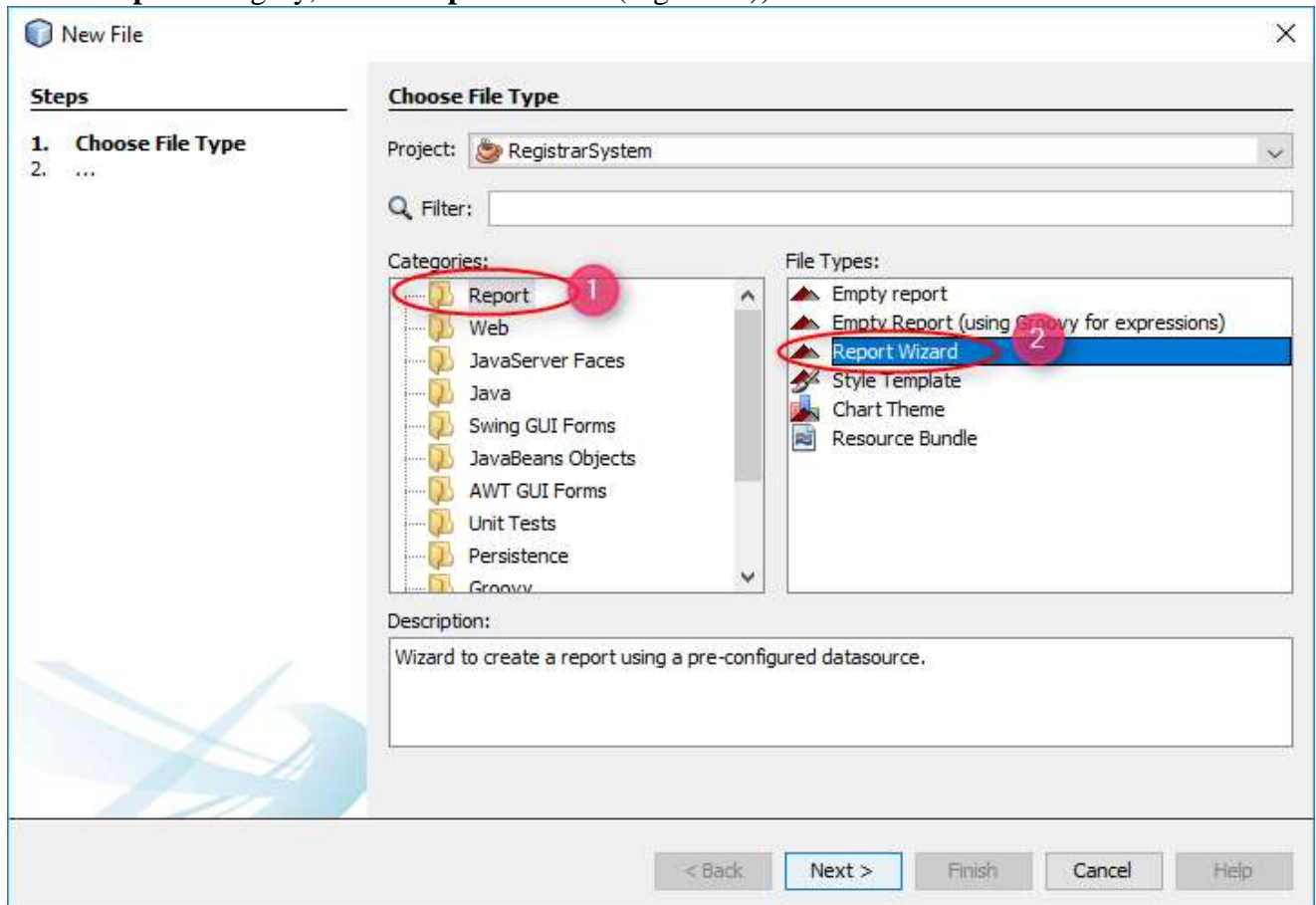

Figure 11

  v.   Select report template/layout
        Select "Blank A4 Landscape" to start from scratch, or you can select an initial template (Figure 12),
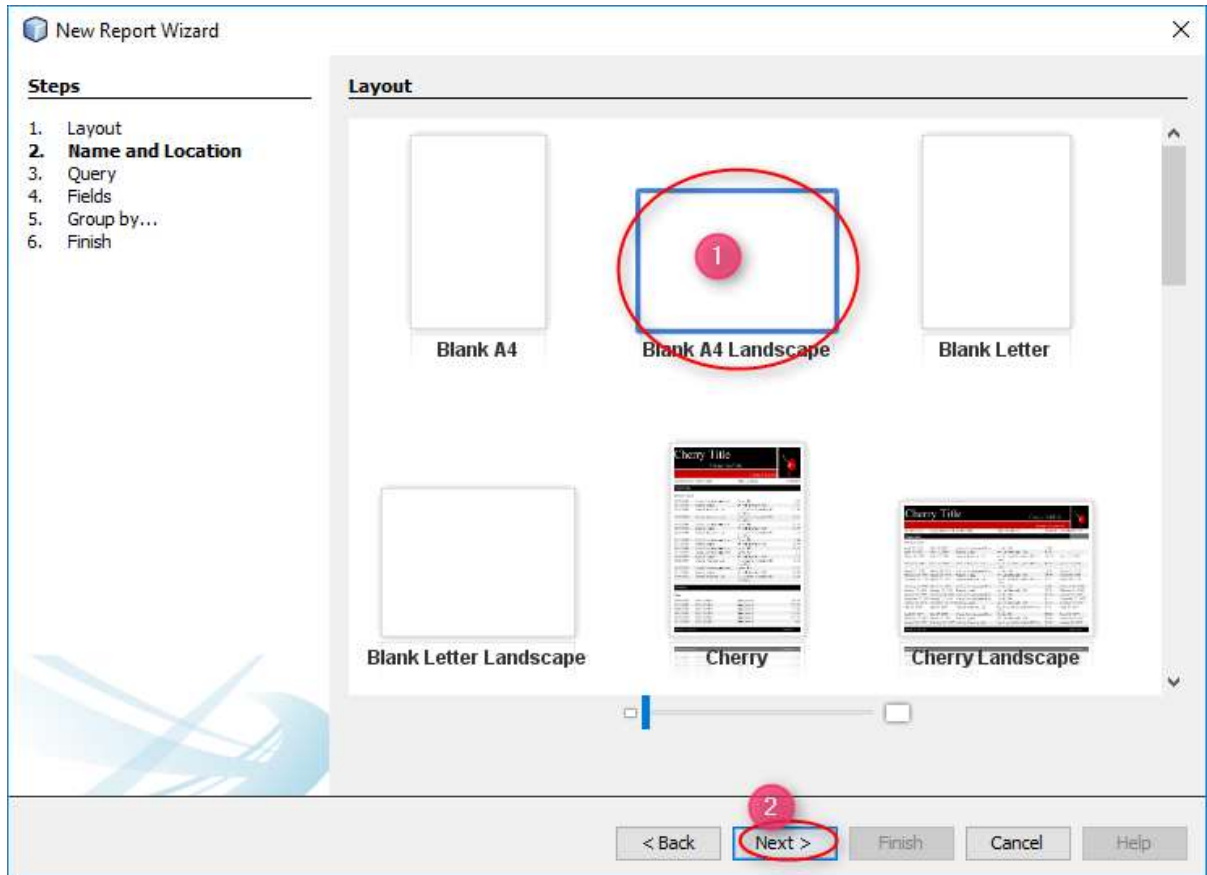   Click Next

Figure 12

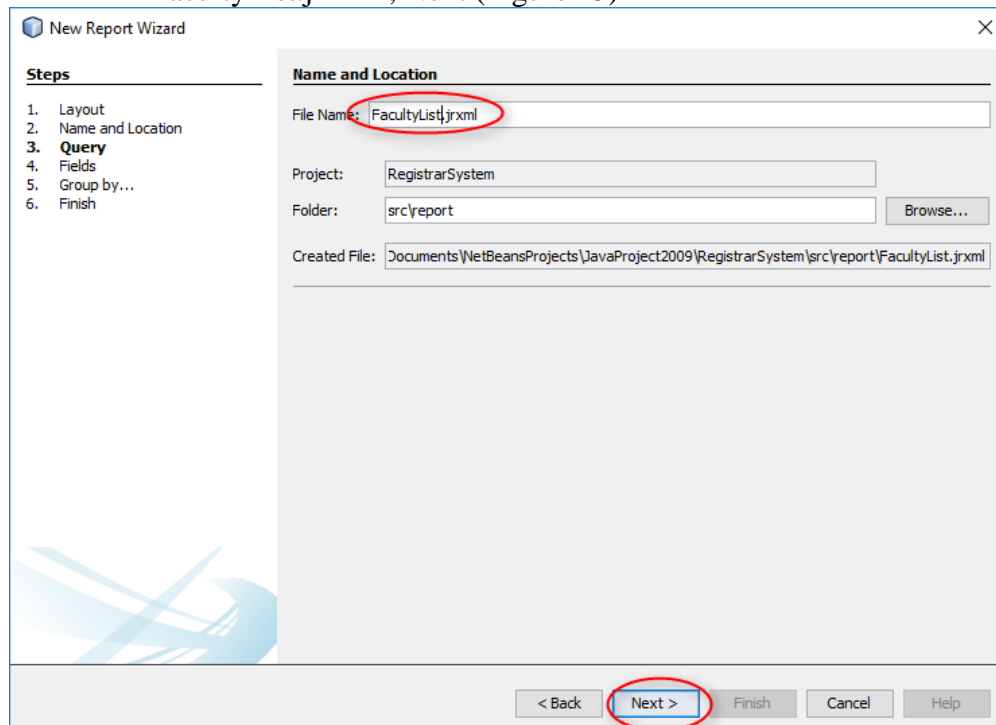vi.   Give report file name "FacultyList.jrxml", Next (Figure 13)

Figure 13

vii.   Configure database connection string
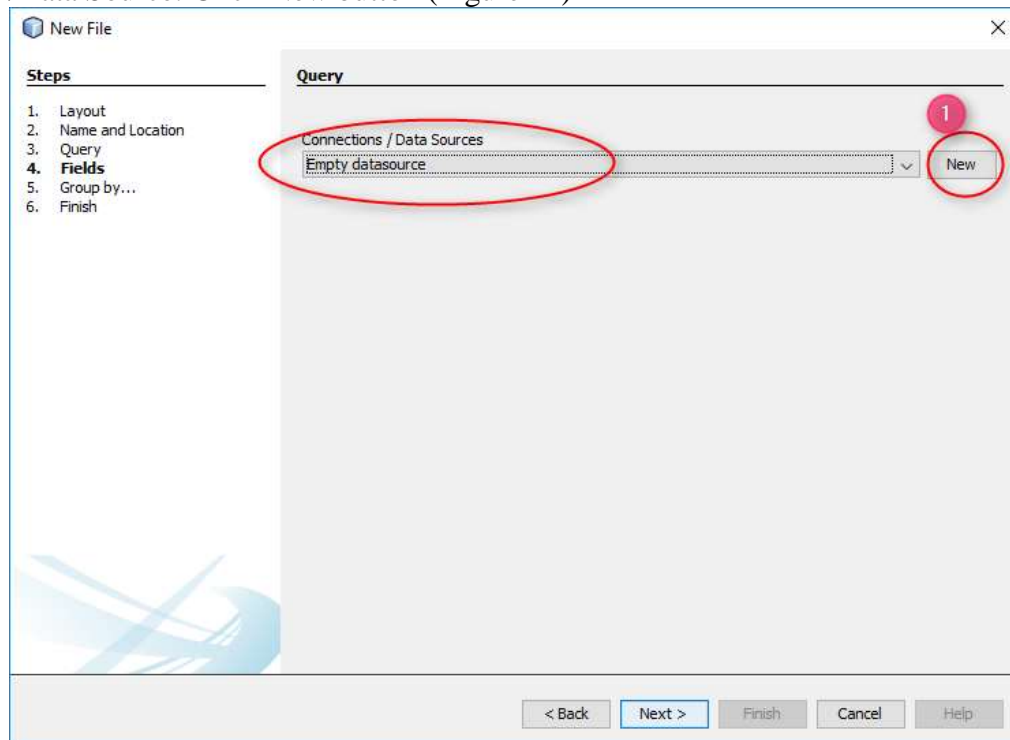   Connection/Data Source: Click New button (Figure 14)

Figure 14

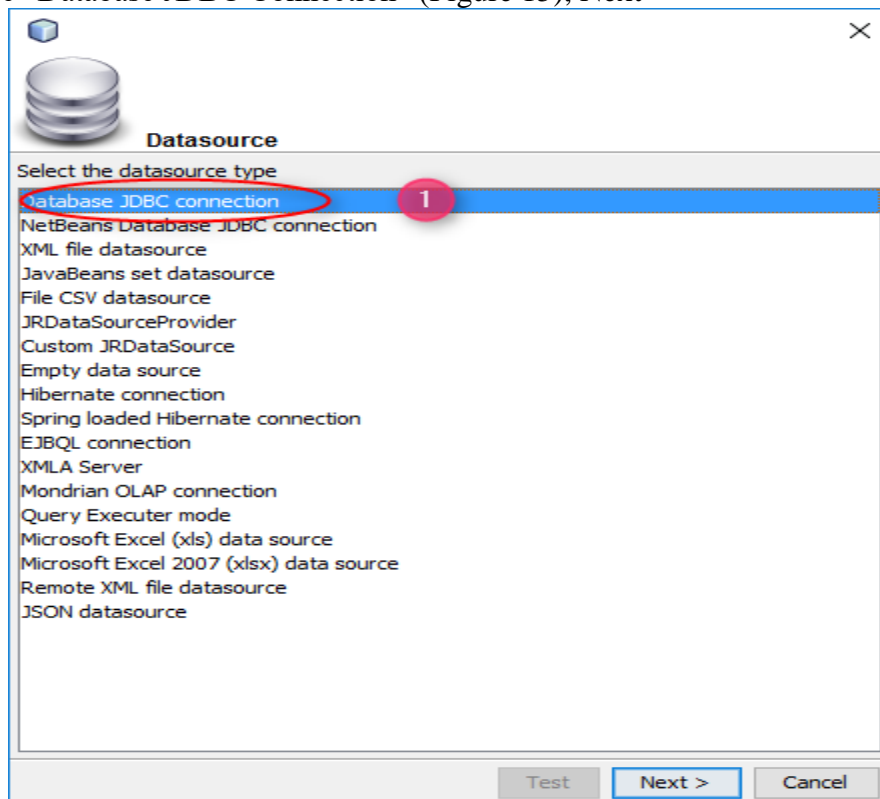viii.   Select Data Source "Database JDBC Connection" (Figure 15), Next

Figure 15

ix. Fill Database JDBC connection credentials, click "Test" and then click "Save" (Figure 16)
Name: RegistrarSystem
JDBC Driver: Oracle (oracle.jdbc.driver.OracleDriver)
JDBC URL : jdbc:oracle:thin:@localhost:1521:XE
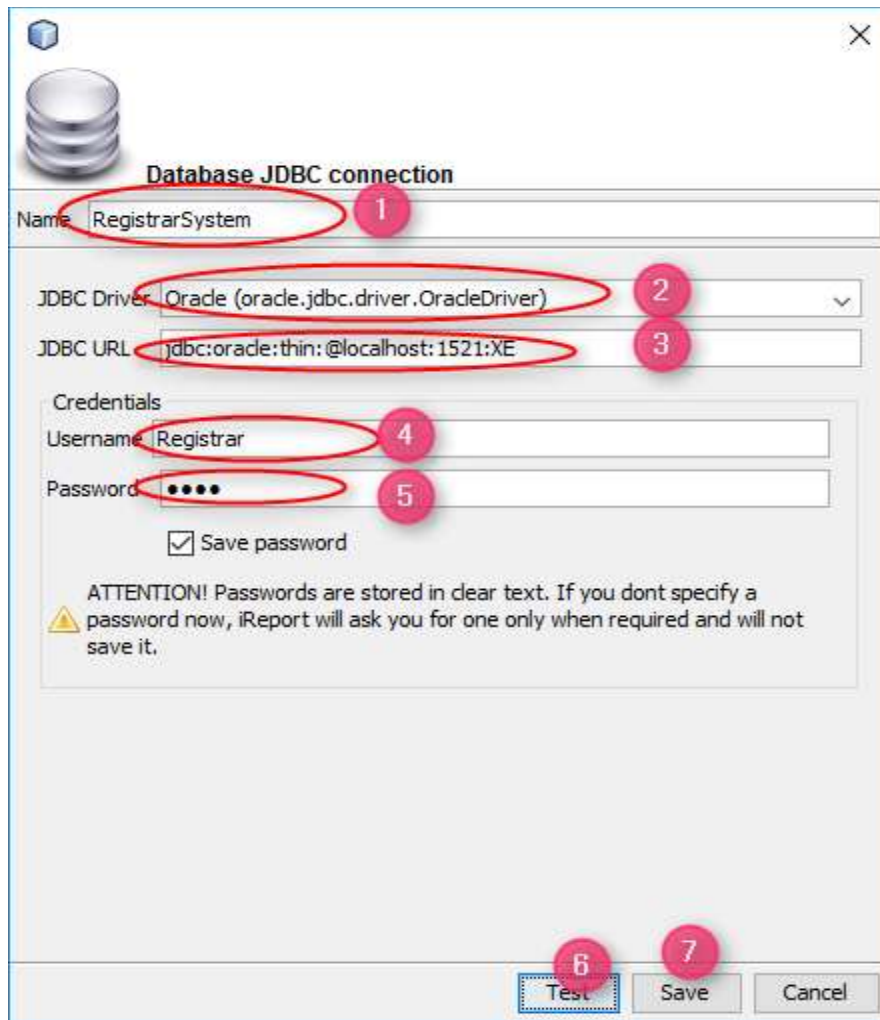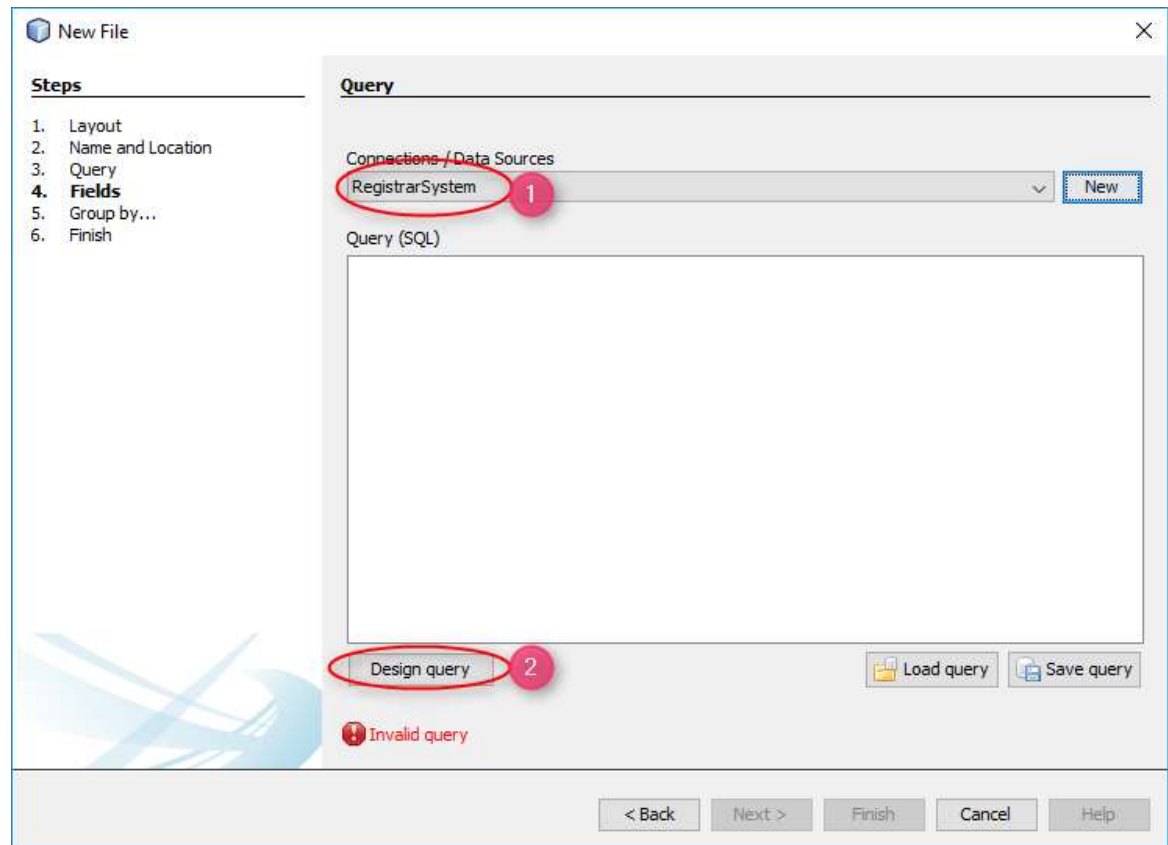Username : Registrar
Password : 1234

Figure 16

x. Design query for the report
    i. Select Your connection "RegistrarSystem" (Figure 17[1])
    ii. Click "Design Query" button (Figure 17[2])

Figure 17

   iii.   Select "Select" under "ROOTQUERY" (Figure 18[1])
   iv.   Select "Registrar" database (Figure 18[2])
   v.   Select "Table" object (Figure 18[3])
   vi.   Drag "FACULTY" table to the Design area (Figure 18[4])
   vii.   Select all fields except Faculty_Code (Figure 18[5])
  viii.   You click "Syntax" tab and view the SQL statement for your query
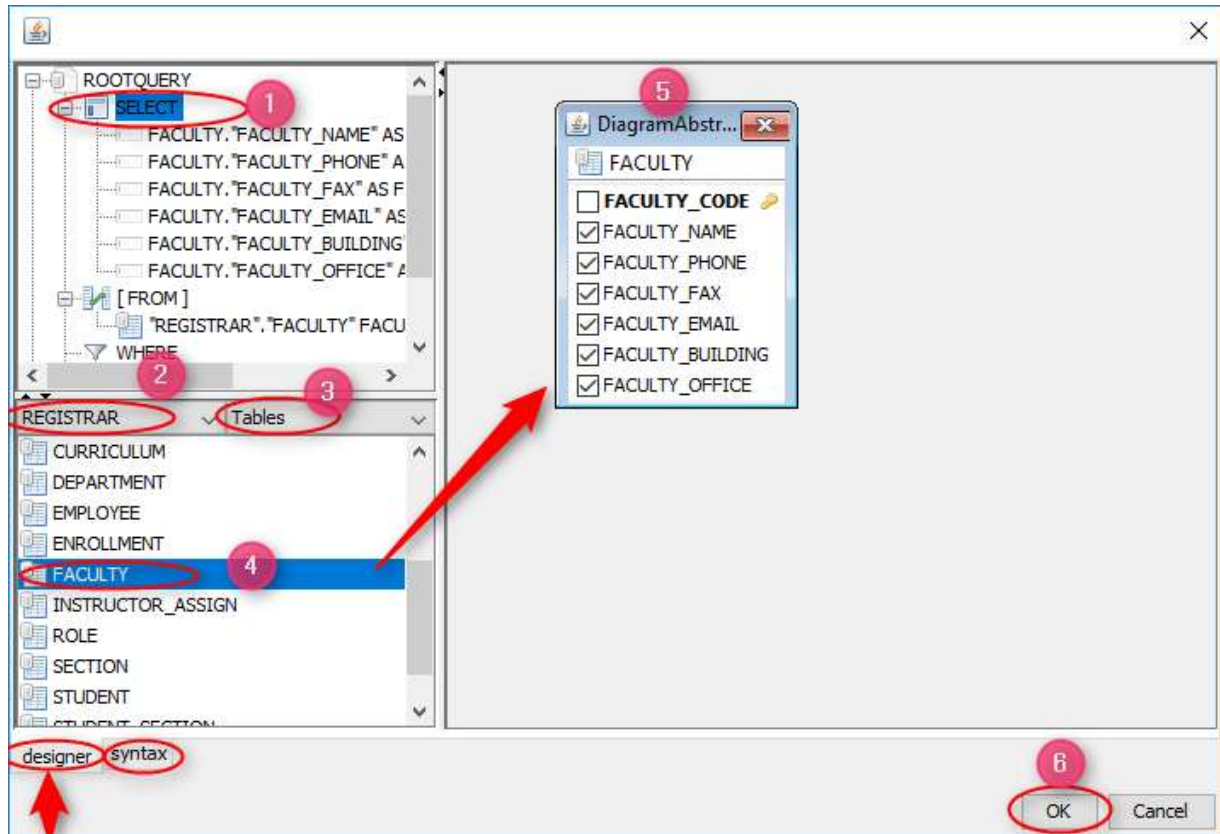   ix.   Click "Ok"

Figure 18

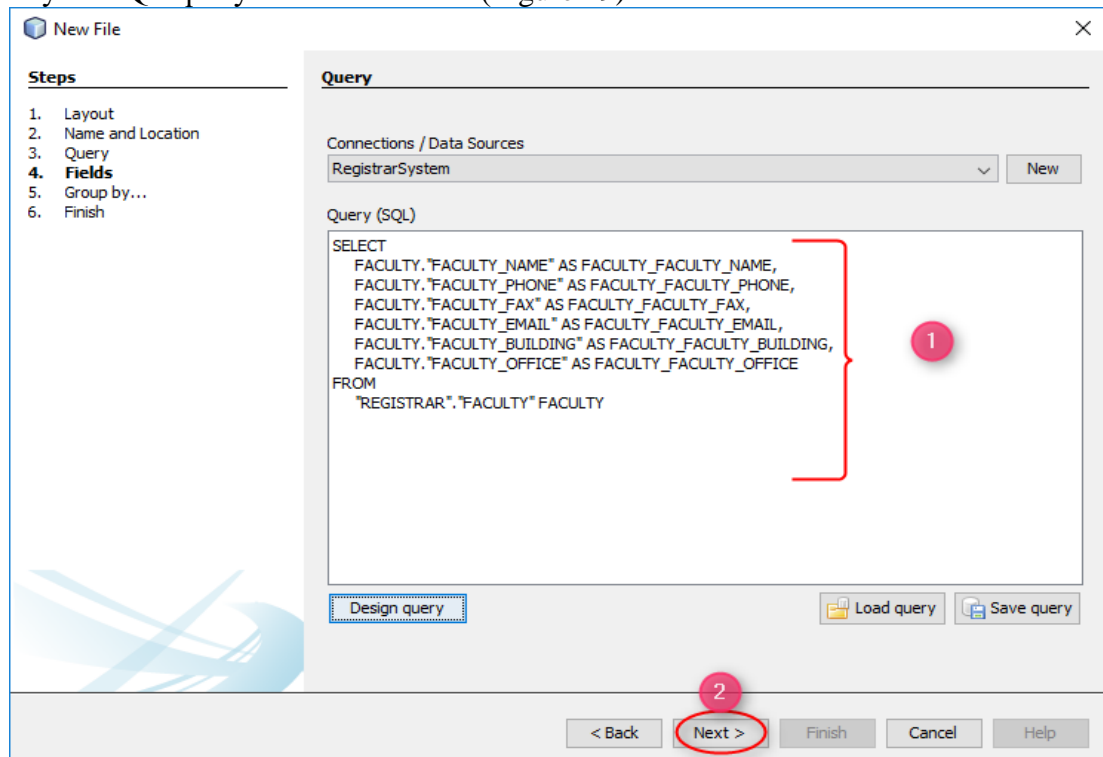x.   Check your SQL query and click "Next" (Figure 19)

Figure 19

xi.   Select fields that you want to appear on the report, select all fields now (Figure 20)
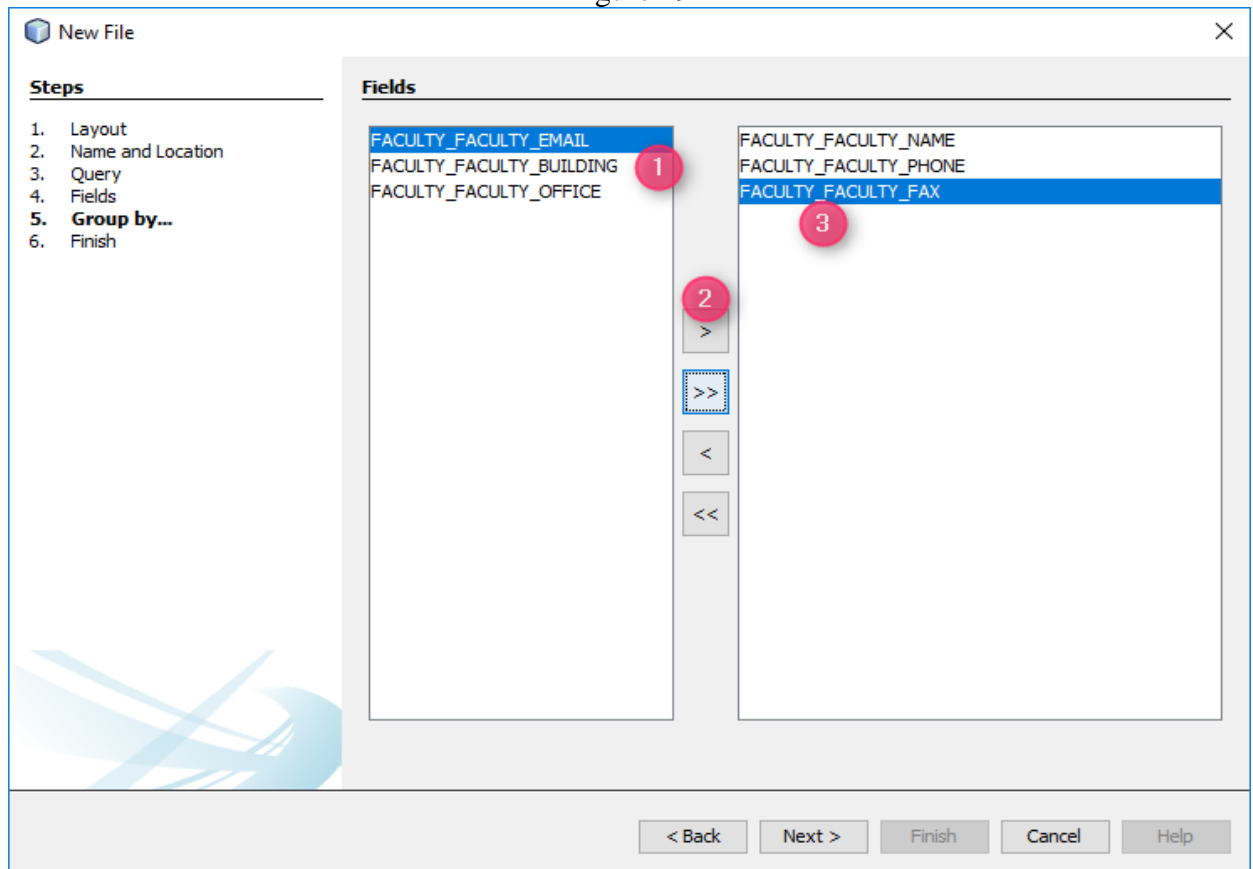
Figure 19

Figure 20

xii.   Click next and Select grouping if any, for now do nothing, click "Next" (Figure 21)
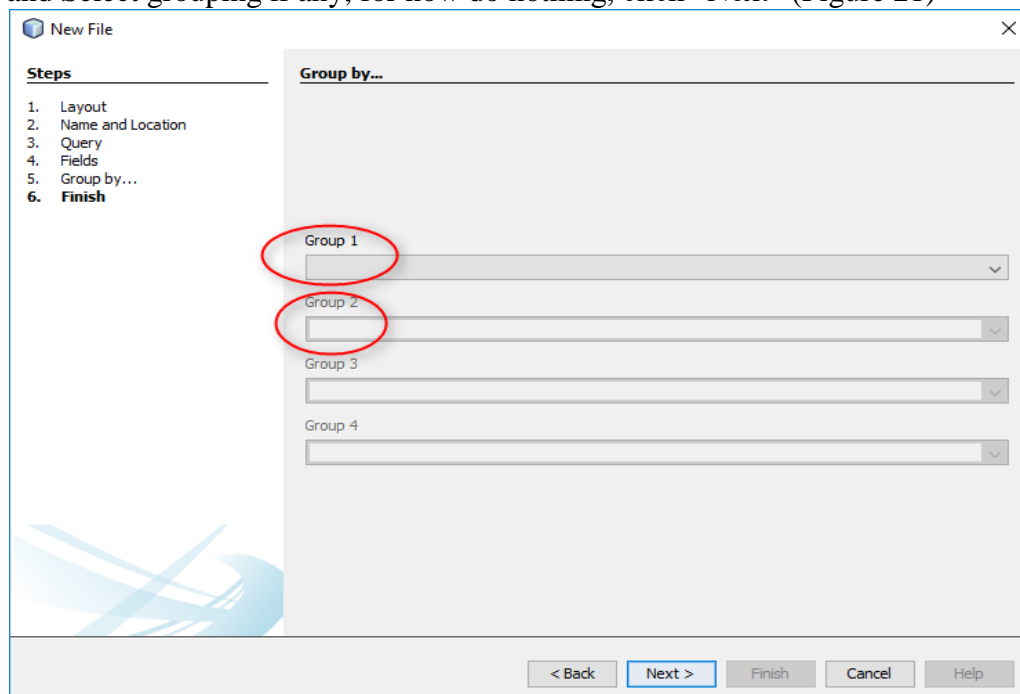

Figure 21

xiii.   Finish the wizard, click "Finish" (Figure 22)
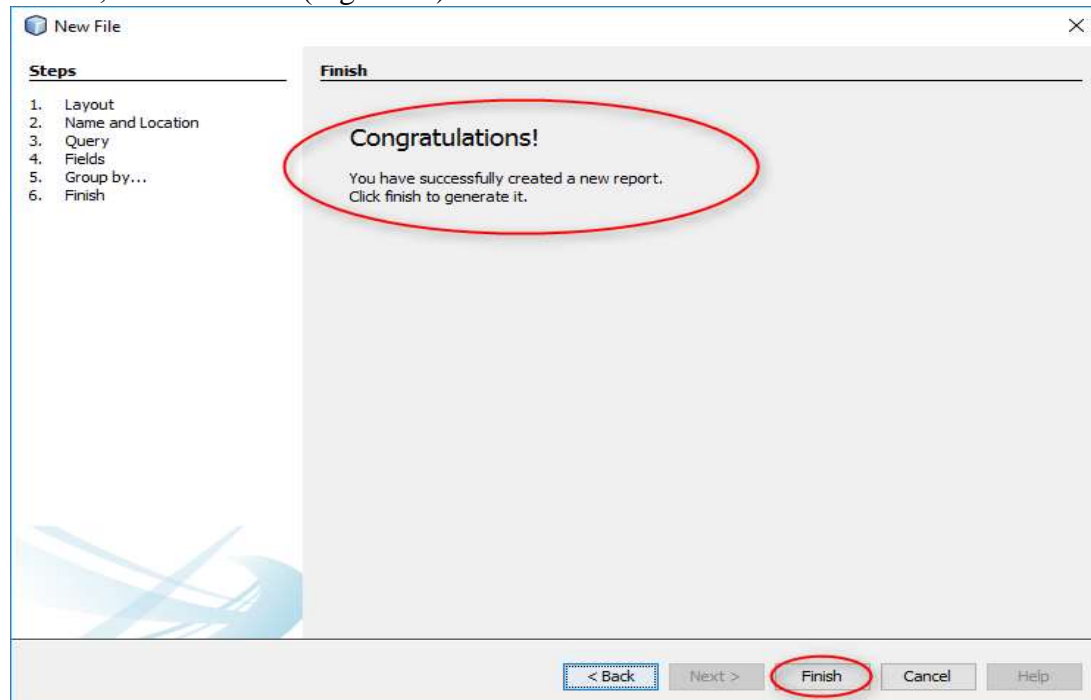

Figure 22

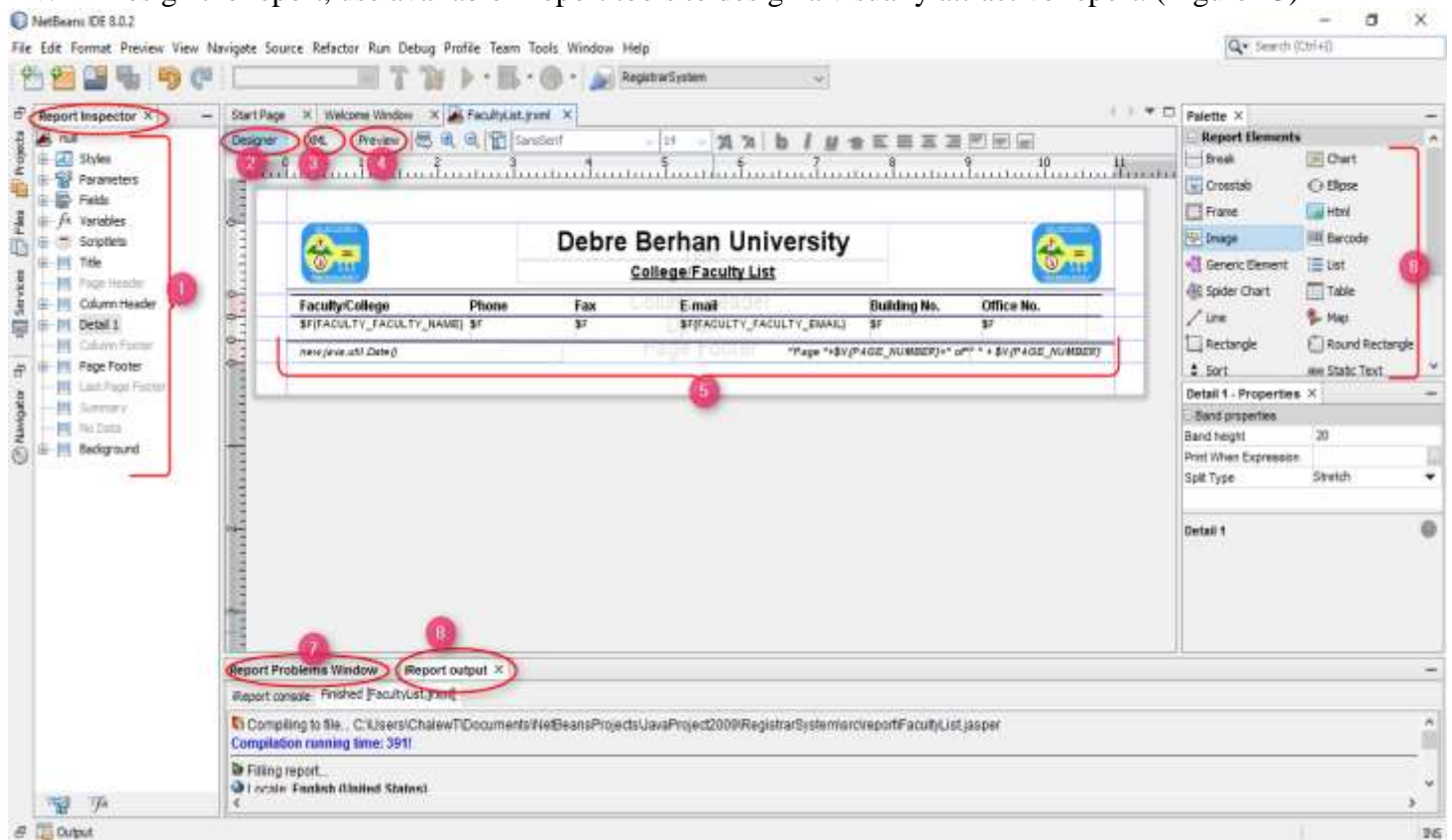xiv.   Design the report, use available iReport tools to design a visually attractive report. (Figure 23)
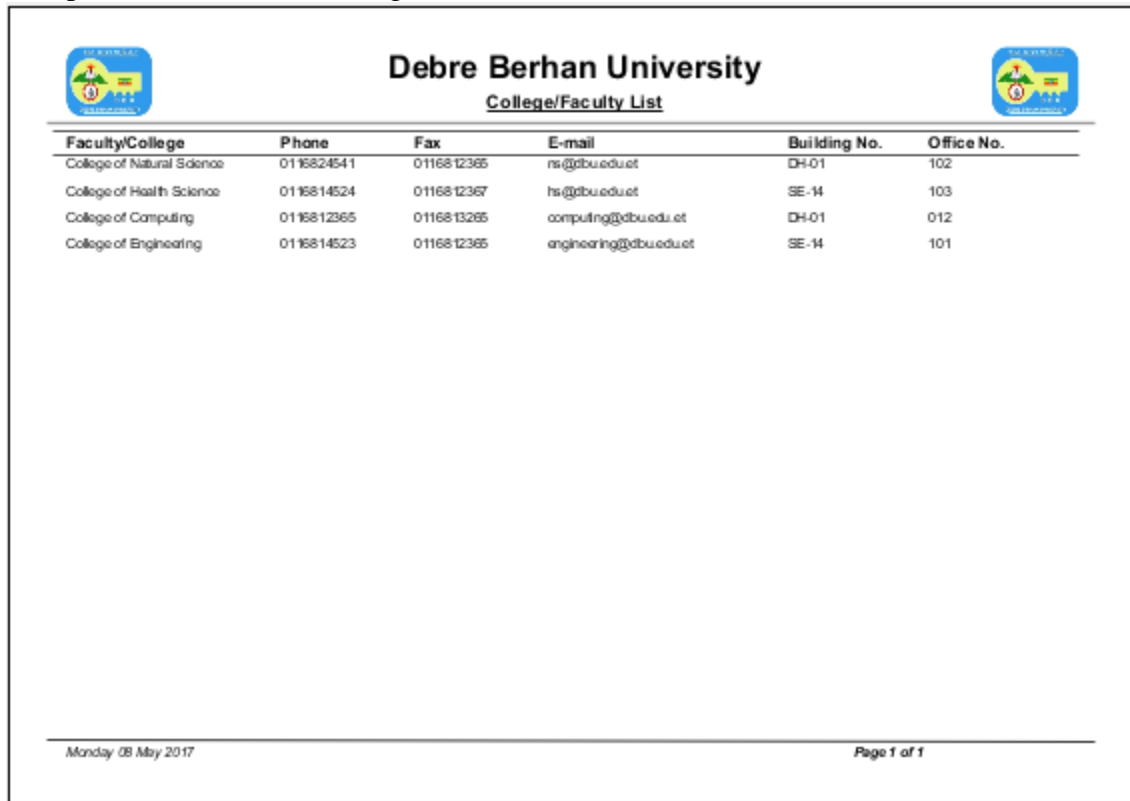

Figure 23

xv.   View the report with Preview tab (Figure 23[4])

Figure 24

5.4.4 Call iReport from a Java App

i.   Import required libraries
   – Select libraries -> click Add jar/Folder
   – JasperSoft libraries
      • Browse the following ".jar" files in
      • C:\Users\username\AppData\Roaming\NetBeans\8.0.2\modules and \ext or C:\Users\username\.netbeans\8.0.2\modules and \ext folder in Window OS
      • It is recommend to copy all the libraries to your project folder first
   – Select the following libraries one by one
      • jasperreports-5.5.0.jar
      • commons-beanutils-1.8.2.jar
      • commons-collections-3.2.1.jar
      • commons-digester-2.1.jar
      • commons-javaflow-20060411.jar
      • commons-logging-1.1.jar
      • groovy-all-2.0.1.jar
      • servlet-api-2.4.jar
      • jdt-compiler-3.1.1.jar
      • iText-2.1.7.js2.jar
      • poi-3.7-20101029.jar

ii. Call the report from java code
   – Add the following import packages
      • import net.sf.jasperreports.engine.JRException;
      • import net.sf.jasperreports.engine.JasperCompileManager;
      • import net.sf.jasperreports.engine.JasperFillManager;
      • import net.sf.jasperreports.engine.JasperPrint;
      • import net.sf.jasperreports.engine.JasperReport;
      • import net.sf.jasperreports.view.JasperViewer;
   – create a method that call a faculty list jasper report under "FacultyList" form/class

```
private void loadFacultyReport() {
    try {
        Connection conn = DbConnection.getConn();
        String facultyReport = "src\\report\\FacultyList.jrxml";
        JasperReport jspReport = JasperCompileManager.compileReport(facultyReport);
        JasperPrint jspPrint = JasperFillManager.fillReport(jspReport, null, conn);
        JasperViewer vr = new JasperViewer(jspPrint, false);
        vr.setTitle(" Faculty List");
        vr.setVisible(true);       } catch (SQLException | JRException e) {
        JOptionPane.showMessageDialog(this, e.getMessage());
    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, e.getMessage());
    }
}
```

iii.   add a button control (Print, btnPrint) to FacultyList form

iv.   call the method under print button action performed event

private void btnPrintActionPerformed(java.awt.event.ActionEvent evt) {

        loadFacultyReport();
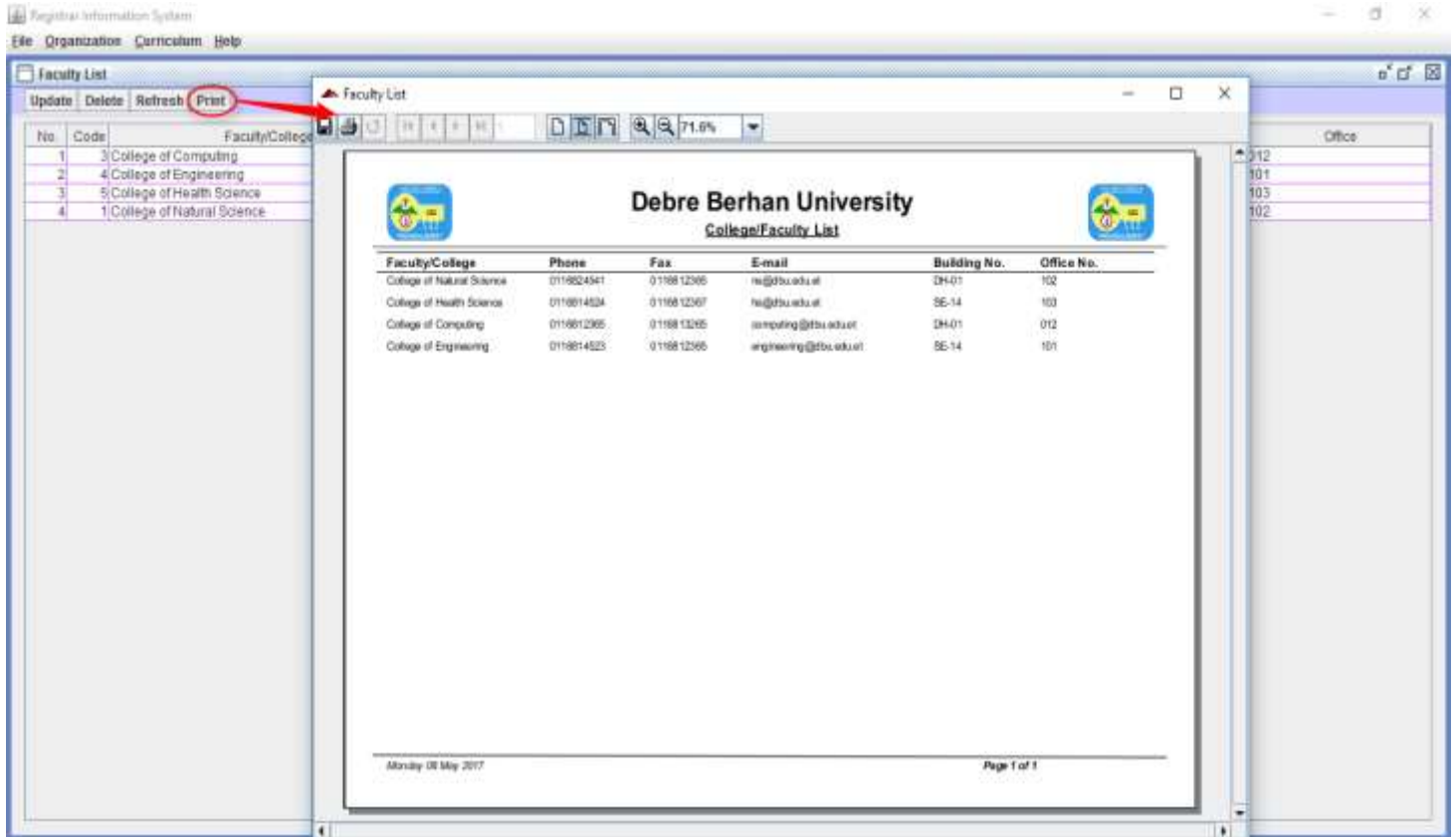
}

v.   run and test

Figure 25

5.4.5 Passing parameters to Jasper Report
- ➢ Parameters used to pass
  - – dynamic values from the java program to the report
  - – Report selection criteria
- ➢ To pass parameters
  1. Define array of parameters
  2. Assign the value of java control values to the name of parameters defined in the report
  3. Call the report by passing
     - ✓ Parameter values along with
     - ✓ the report name and
     - ✓ connection string

Design a Curriculum Details report, which list detail information about the curriculum and available course with breakdown under the curriculum.
- i.   Add a report called "**CurriculumDetails**" (Follow the steps in (0 Using iReport to Design a Report))
- ii.   Design the report to the following ()
     - a. Define parameter (Figure 26[1])
     - b. Add the parameter to the report (Figure 26[2])
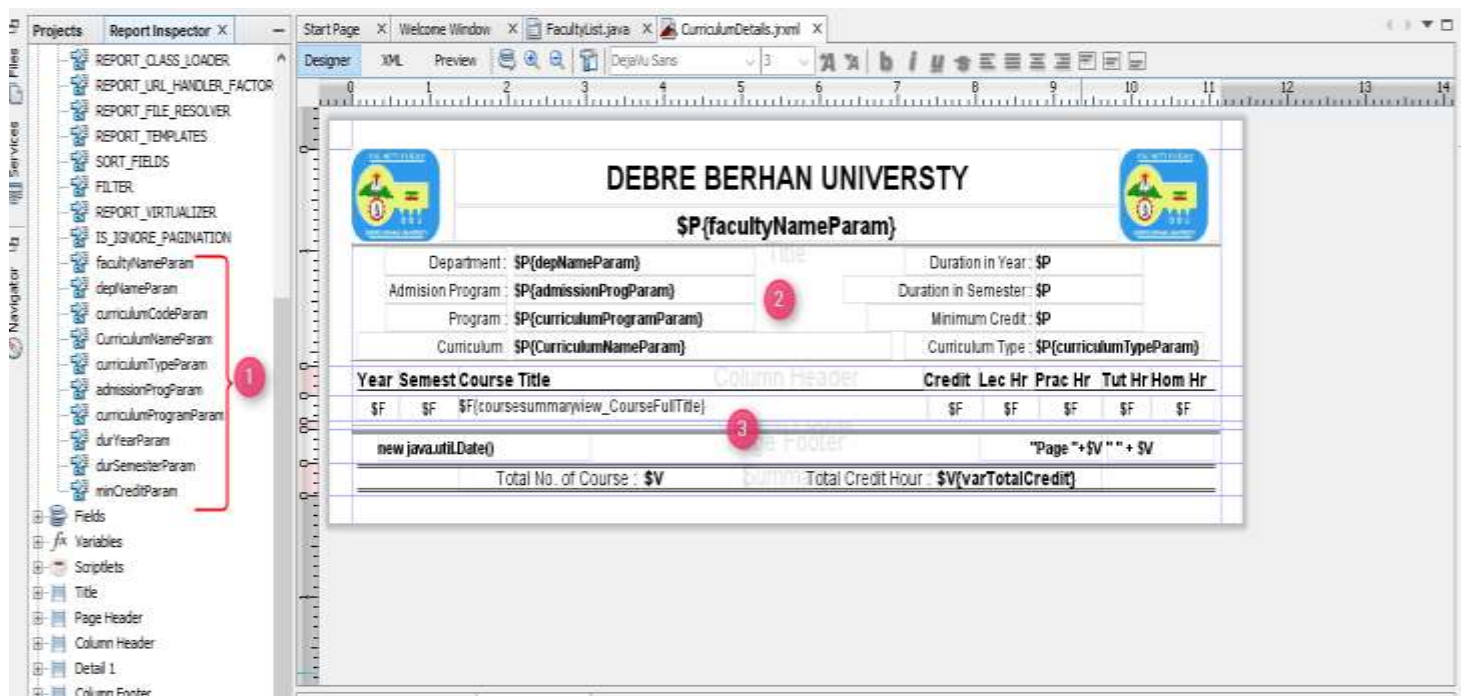
Figure 26

- iii.   Create a method called "loadCurriculumReport" which:
     - – Define the parameters
     - – Pass parameter and
     - – Display "CurriculumDetails" jasper report a

```java
private void loadCurriculumReport() {
    if (cmbCurriculum.getSelectedIndex() != -1 && cmbProgram.getSelectedIndex() != -1) {
        this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
        try {
            SisComboBoxItem currItem = (SisComboBoxItem) cmbCurriculum.getSelectedItem();
            SisComboBoxItem curriculumBdItem = (SisComboBoxItem) cmbProgram.getSelectedItem();

            Connection conn = DbConnection.getConn();
            // Put values from controls into a parameter map.
            Map currParams = new HashMap<>();
            currParams.put("facultyNameParam", cmbFaculty.getSelectedItem().toString());
            currParams.put("depNameParam", cmbDepartment.getSelectedItem().toString());
            currParams.put("curriculumCodeParam", currItem.getCodeValue());
            currParams.put("CurriculumNameParam", cmbCurriculum.getSelectedItem().toString());
            currParams.put("curriculumTypeParam", cmbCurrType.getSelectedItem().toString());
            currParams.put("admissionProgParam", cmbProgram.getSelectedItem().toString());
            currParams.put("curriculumProgramParam", currItem.getCurrProgram());
            currParams.put("durYearParam", curriculumBdItem.getNoYear());
            currParams.put("durSemesterParam", curriculumBdItem.getNoSemster());
            currParams.put("minCreditParam", currItem.getCurrminCredit());
            // Give the parameters to the report filler.
            String courseReport = "src\\reports\\CurriculumDetails.jrxml";
            JasperReport jspReport = JasperCompileManager.compileReport(courseReport);
            JasperPrint jspPrint = JasperFillManager.fillReport(jspReport, currParams, conn);
            //Display the report
            JasperViewer vr = new JasperViewer(jspPrint, false);
            vr.setTitle(cmbCurriculum.getSelectedItem().toString() + " Curriculum Details");
            vr.setVisible(true);
        } catch (SQLException | JRException e) {
            JOptionPane.showMessageDialog(this, e.getMessage());
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, e.getMessage());
        }
        this.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
    }
}
```

iv.   Add jInternalForm called "CurriculumList" to your project, design to look the following picture (Figure 27)

Figure 27

v.   Call the method "**loadCurriculumReport**" under action performed event of print (btnPrintCurriculum) button control.

private void btnPrintCurriculumPerformed(java.awt.event.ActionEvent evt) {
        loadCurriculumReport();
    }

vi.   Run and test
        You should have to get an output similar to the following picture (Figure 28)

Figure 28 Curriculum Detail Report output

================================== ********** ==================================