|  |
| --- |
| **Lab10: File I/O and Database Connectivity** |

Designing and implementing Java programs that deal with:

1. **Reading Input From Text Files**
2. **Writing Output To Text Files**
3. **JDBC <-> ODBC**

1. Reading Input From Text Files

Designing and implementing Java programs that deal with:

1. **Reading Input From Text Files**
   1. Create a **File** object and give it the name of the file.  
      **\\ the file in the current folder**

**File myFile01 = new File(“Message.txt”);**

**\\ full path using the back slash  
File myFile02 = new File(“D:\\CSC210\\Lab10\\READ.TXT”);**

**\\ full path using the forward slash  
File myFile03 = new File(“D:/CSC210/Lab10/READ.TXT”);**

**\\ the current directory**

**File myFile04 = new File(“.”);**

* 1. Pass the **File** object to a **Scanner** object to read from it.  
     **Scanner reader01 = new Scanner(myFile01);**

🢚Steps **a** and **b** can be combined as:  
**Scanner reader01 = new Scanner(new File(“Message.txt”);**

* 1. Use the methods of the **Scanner** class normally, i.e., **hasNextInt**, **nextInt**, **next**, **nextDouble**, etc.  
     **reader01.hasNextInt();**  
     **reader01.nextInt();**

**NOTES:**

* If the file does not exist, a **FileNotFoundException** is thrown.
* ALWAYS make sure to close your files!

/\*

\* Reads a set of numbers from the file "data.txt" and finds their sum

\*/

import java.util.Scanner;

import java.io.File;

import java.io.FileNotFoundException;

public class HasNextIntDemo{

public static void main(String[] args){

Scanner input = null;

try{

System.out.println(“Trying to open \”data.txt\” file...”);

input = new Scanner(new File("data.txt"));

}

catch(FileNotFoundException e){

System.out.println("File data.txt was not found ");

System.out.println("or could not be opened.");

System.exit(0);

}

int next, sum = 0;

while (input.hasNextInt()){

next = input.nextInt();

sum = sum + next;

}

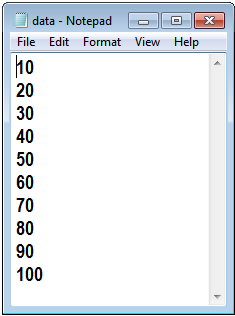
input.close();

System.out.println("Sum of the numbers is " + sum);

}

}

**Note:**

You can create your own “data.txt” (as shown in figure) file **OR** you can use the attached file below. To use the attached file, (1) double click it (2) confirm the warning message by clicking the “**Yes**” button (3) store the file in your project folder by clicking “**File**” then “**Save As…**”. 

**Note:** If your program does not find your file, use the getAbsolutePath() method of File class to find the expected path of your file in your exception handling code.

2. Writing output to Text Files

1. **Writing Output to Text Files**
   1. Create a **File** object and give it the name of the file.  
      **\\ the file in the current folder  
      File myFile01 = new File(“Message.txt”);**

**\\ full path using the back slash  
File myFile02 = new File(“D:\\CSC210\\Lab10\\WRITE.TXT”);**

**\\ full path using the forward slash**

**File myFile02 = new File(“D:/CSC210/Lab10/WRITE.TXT”);**

* 1. Pass the **File** object to a **PrintWriter** object to wirte to it  
     **PrintWriter write01 = new PrintWriter(myFile01);**

🢚Steps **a** and **b** can be combined as:  
**PrintWriter writer01 = new PrintWriter(new File(“Message.txt”);**

🢚 Steps **a** and **b** can be shorten as:  
**PrintWriter writer01 = new PrintWriter(“Message.txt”);**

* 1. Use **print** and **println** methods of the **PrintWriter** class in order to write to the file. **print** and **println** methods behave in the same way as those of **System.out** object except that they write to a file instead of the screen.  
     **writer01.print(“Hello”);**  
     **writer01.println(“ICS102 Students”);**

**NOTES:**

* If the file does not exist, it is created automatically
* If the file exists, it will be overwritten! SO, BE CAREFULL!
* If there is no writing access to the file or its folder, a **SecurityException** is thrown. It is always nice to handle it
* ALWAYS make sure to close your files! Otherwise your output can be lost!

/\*

\* Reads one line at a time from an input file and prints it to an output file with a line number

\*/

import java.util.Scanner;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

public class HasNextLineDemo{

public static void main(String[] args){

Scanner input = null;

PrintWriter output = null;

try{

System.out.println(“Trying to open \”original.txt\” file...”);

input = new Scanner(new File ("original.txt"));

output = new PrintWriter(new File("numbered.txt"));

}

catch(FileNotFoundException e){

System.out.println("Problem opening files.");

System.exit(0);

}

String line = null;

int count = 0;

while (input.hasNextLine()){

line = input.nextLine();

count++;

output.println(count + " " + line);

}

input.close();

output.close();

System.out.println("Process done successfully");

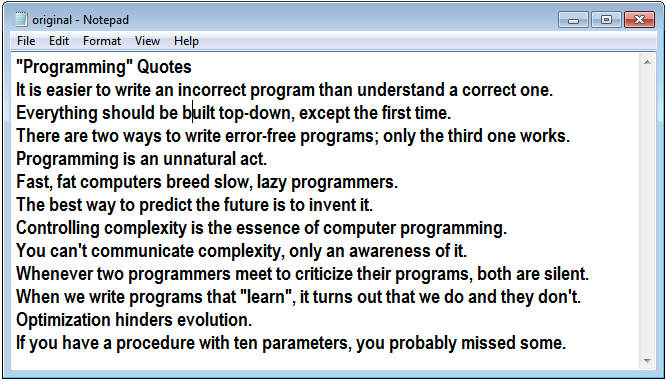
}

}

**Note:**

You can create your own “original.txt” (as shown in figure) file **OR** you can use the attached file below. To use the attached file, (1) double click it (2) confirm the warning message by clicking the “**Yes**” button (3) store the file in your project folder by clicking “**File**” then “**Save As…**”.





3. Writing and Reading Object values from Text File

**Creating a Class Account**

|  |
| --- |
| public class Account {  private String acctNum;  private String fName;  private double balance;    public Account (String num, String name, double bal) {  acctNum = num;  fName = name;  balance = bal;  }  public String getAcctNum() {  return acctNum;  }  public void setAcctNum(String acctNum) {  this.acctNum = acctNum;  }  public String getfName() {  return fName;  }  public void setfName(String fName) {  this.fName = fName;  }  public double getBalance() {  return balance;  }  public void setBalance(double balance) {  this.balance = balance;  }    public String toString() {  return "Account Number = " + acctNum + "\nFirst Name=" + fName + "\nBalance = " + balance;  }  } |

**Writing Output to Text File**

|  |
| --- |
| import java.io.IOException;  import java.util.ArrayList;  import java.util.Formatter;  public class FileOutput {  public static void main (String [] args) {    //Create an arraylist to store customers  ArrayList<Account> customers = new ArrayList<Account> ();    //create some sample customers and add to list  Account a = new Account ("123", "Sam", 123.65);  customers.add(a);  a = new Account("234", "Sue", 423.35);  customers.add(a);  a = new Account("144", "Mark", 555.25);  customers.add(a);  //open a file  Formatter output = null;    //must open the file in a try catch block  try {  output = new Formatter("data.txt");  } catch (IOException e) {  System.out.println("There is an error - exiting");  System.exit(1);  }    //write everything from list to output  for (Account x: customers)  output.format("%s %s %.2f\n",  x.getAcctNum(),  x.getfName(),  x.getBalance());    output.close();  }  } |

**Reading and Printing Output from Text File**

|  |
| --- |
| import java.io.File;  import java.io.IOException;  import java.util.ArrayList;  import java.util.Scanner;  public class FileInput {  public static void main (String [] args) {  //List to put the objects we're going to read from the file  ArrayList <Account> list = new ArrayList<Account> ();  //We'll use a scanner object to read from the file.  //Are there other ways? Of course...  Scanner s = null;  try {  s = new Scanner (new File ("data.txt"));  } catch (IOException e) {  System.out.println("Error opening file");  System.exit(1);  }  //Temp variables to store values read from file  String aNum, fName;  double bal;    while (s.hasNext()) {  //aNum = s.next();  //fName = s.next();  //bal = s.nextDouble();  //Account a = new Account(aNum, fName, bal);  Account a = new Account(s.next(),s.next(), s.nextDouble());  list.add(a);  }    //print the customers out  for(Account x: list)  System.out.println(x);    }  } |

4. JDBC

Java Database Connectivity (JDBC) provides a standard library for accessing databases. The JDBC API contains number of interfaces and classes that are extensively helpful while communicating with a database.

The java.sql package

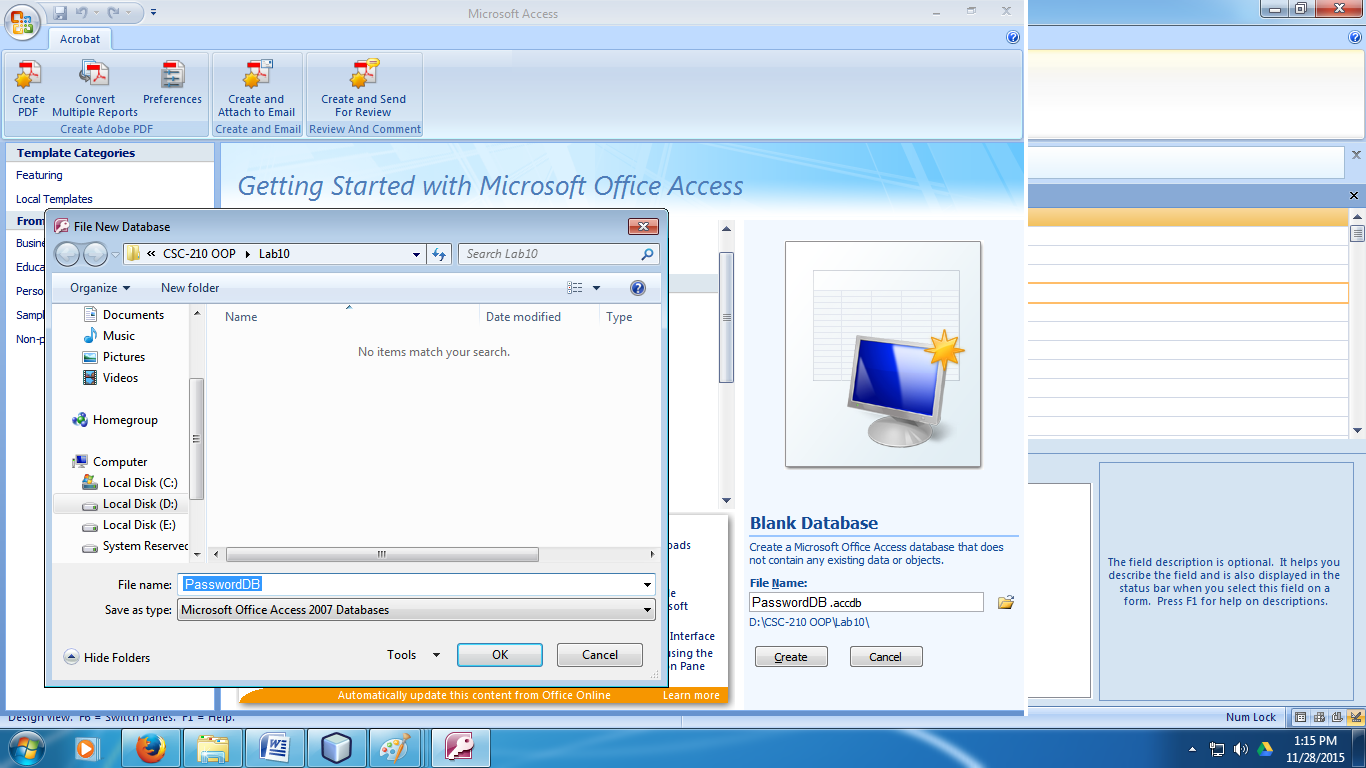
The java.sql package contains basic & most of the interfaces and classes. You automatically get this package when you download the J2SE™. You have to import this package whenever you want to interact with a relational database.

Conneting with Microsoft Access

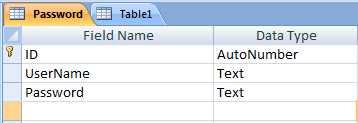
In this handout, we will learn how to connect & communicate with Microsoft Access Database.

Create Database

In start create a database “PasswordDB” using Microsoft Access. Create one table named “Password”. The schema of the table is shown in the picture.

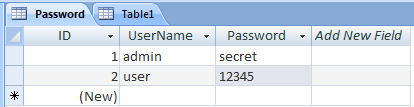


Save the database in some folder. (Your database will be saved as an .accdb file)



Save the above table with a name "Password"

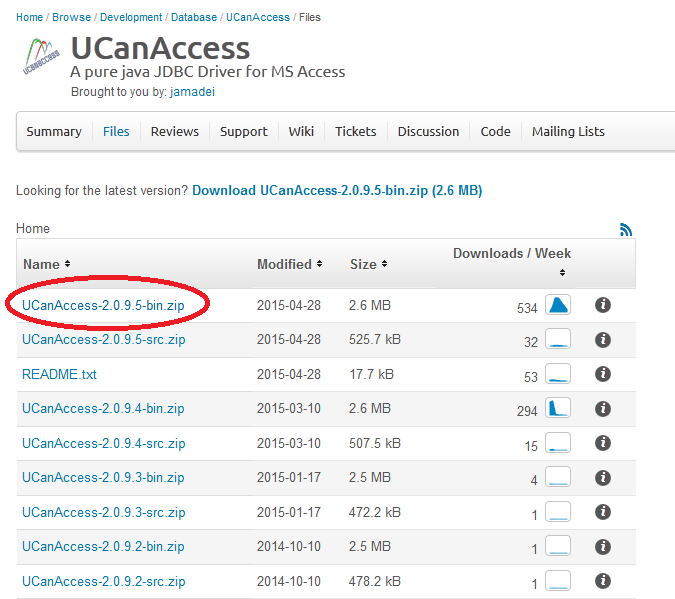
Add the following records into Person table as shown below.



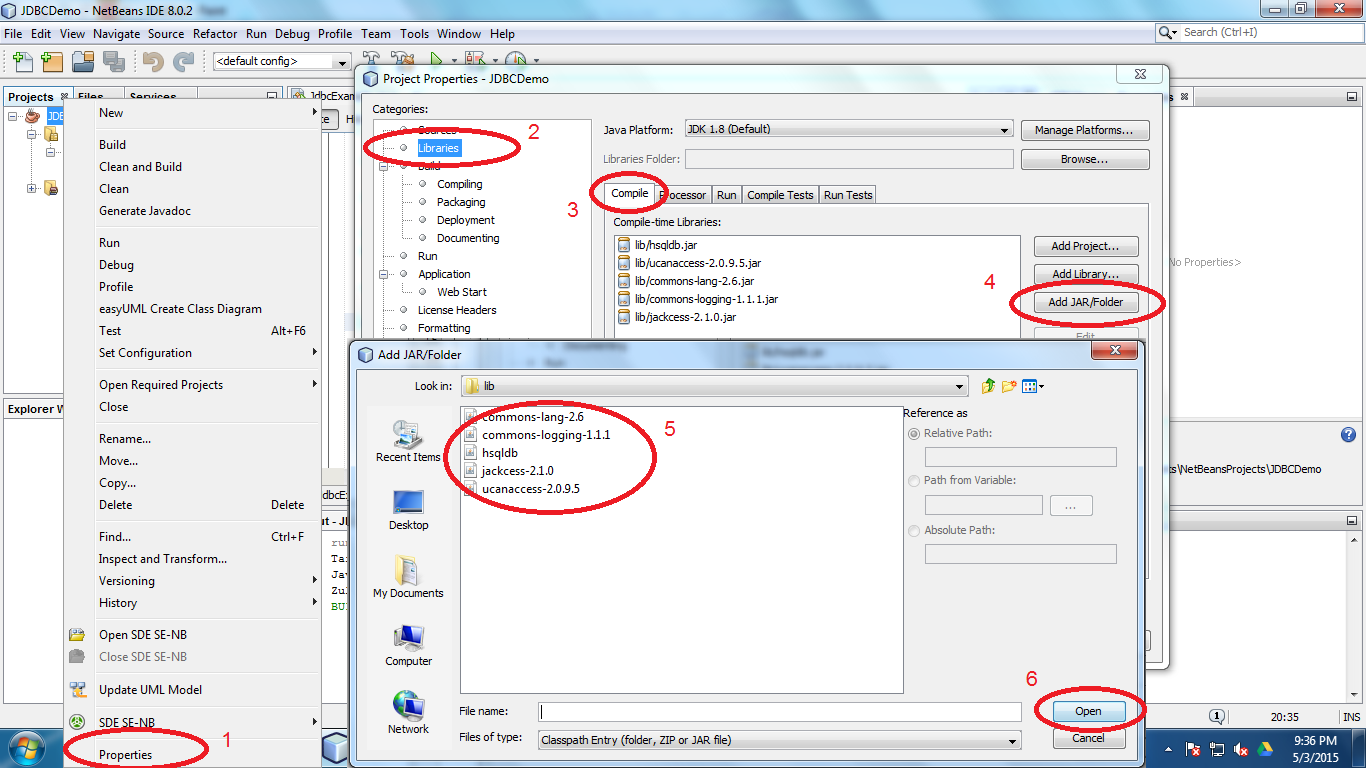
Using ucanaccess Library to Setup Database Connectivity

* Download the ucanaccess from the following link:

http://sourceforge.net/projects/ucanaccess/files/

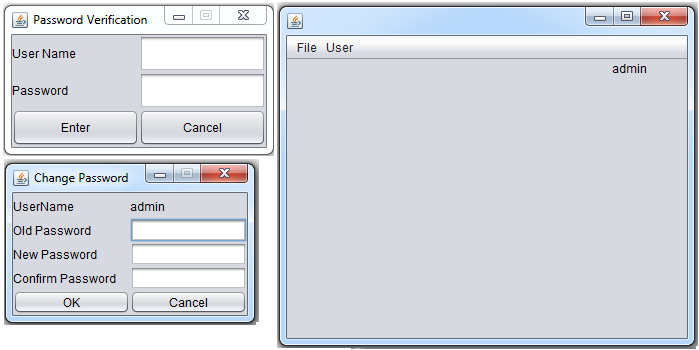


* Extract library files to your netbeans folder
* Add Library files to Project and follow these steps



Retrieving Data from ResultSet

The following example demonstrates the usage of all above explained steps. In this code example, we connect with the PasswordDB database, the one we have created earlier, and then execute the simple SQL SELECT query on Password table, and then process the query results. This example also demonstrates how we can connect multiple forms with each other.



**Password Verification Form:**

Double-click on the Enter JButton—this should take you in the Source code *btnEnterActionPerformed* method. Under the comment line, type the following code:

|  |
| --- |
| // TODO add your handling code here:  Connecting2DB c = new Connecting2DB();  String myPass=String.valueOf(txtPassword.getPassword());  if( c.matchPassword(txtUserName.getText(), myPass)){  MainFrame mf = new MainFrame();  mf.setUser(txtUserName.getText());  mf.show();  dispose();  }  else  System.out.println("Not matched..\nTry again"); |

**Main Form:**

Double-click on the Change Password Menu—this should take you in the Source code *menuChgPasswordActionPerformed* method. Under the comment line, type the following code:

|  |
| --- |
| // TODO add your handling code here:  frmChangePassword cp = new frmChangePassword();  cp.setUserName(lblUser.getText());  cp.show(); |

Double-click on the Logout Menu—this should take you in the Source code *menuLogoutActionPerformed* method. Under the comment line, type the following code:

|  |
| --- |
| // TODO add your handling code here:  frmPassword frmpass = new frmPassword();  frmpass.show();  dispose(); |

**Change Password Form:**

Double-click on the OK JButton—this should take you in the Source code *btnOKActionPerformed* method. Under the comment line, type the following code:

|  |
| --- |
| // TODO add your handling code here:  Connecting2DB c = new Connecting2DB();  String oldPass=String.valueOf(oldPassword.getPassword());  String userName = lblUserName.getText();  if( c.searchPassword(userName,oldPass)){  System.out.println("Match found");  String newpassword = String.valueOf(newPassword.getPassword());  String confirmnewpassword = String.valueOf  (confirmNewPassword.getPassword());  if(newpassword.equals(confirmnewpassword))  System.out.println("Calling updatepassword......");  c.updatePassword(userName,String.valueOf(newPassword.getPassword()));  }  else  System.out.println("Not matched..\nTry again"); |

Double-click on the Enter JButton—this should take you in the Source code btnEnterActionPerformed method. Delete the comment line and type the following code:

|  |
| --- |
| dispose(); |

The following class Connecting2DB provides the functionality to connect to database (PasswordDb.accdb) and execute the SQL commands (Select and Update) to search for the password and match it with the user name and the password. If user wants to change the password then it also updates the new password.

/\*

\* Reads the fields of Person Table from PersonalInfo database and print the screen.

\*/

public class Connecting2DB {

public Connection setConnection(){

String dataSourceName="database/PasswordDB.accdb";

String dir = System.getProperty("user.dir");

String url = "jdbc:ucanaccess://"+dir+"/" + dataSourceName;

Connection con=null;

try {

con = DriverManager.getConnection(url);

}

catch(Exception sqlEx){

System.out.println(sqlEx);

}

return con;

}

public boolean matchPassword(String user, String pass){

boolean successful =false;

try {

Connection con = setConnection();

Statement st = con.createStatement();

String sql =

"SELECT \* FROM Password where username = '"+user+"'";

ResultSet rs = st.executeQuery(sql);

while(rs.next()){

String userName = rs.getString("UserName");

String password = rs.getString("Password");

if(user.equals(userName) && pass.equals(password))

successful = true;

else

successful = false;

}

con.close();

}

catch(Exception sqlEx){

System.out.println(sqlEx);

}

return successful;

}

public boolean searchPassword(String user, String pass){

boolean found =false;

try {

Connection con = setConnection();

Statement st = con.createStatement();

String sql = "SELECT \* FROM Password where username ='"+user+"' and password = '"+pass+"'";

ResultSet rs = st.executeQuery(sql);

while(rs.next()){

String userName = rs.getString("UserName");

String password = rs.getString("Password");

if(pass.equals(password))

found = true;

else

found = false;

}

con.close();

}catch(Exception sqlEx){

System.out.println(sqlEx);

}

return found;

}

public void updatePassword(String user, String pass){

try {

Connection con = setConnection();

PreparedStatement ps = con.prepareStatement(

"UPDATE Password SET password = ? WHERE username = ? ");

ps.setString(1,pass);

ps.setString(2,user);

ps.executeUpdate();

ps.close();

}

catch(Exception sqlEx){

System.out.println(sqlEx);

}

}

}

**Note:**

We can use the following hard coded Database path but it makes it difficult to run application on

different machines unless the path to the database is modified according the location of the

database file.

String url = "jdbc:ucanaccess://C:/Users/INTEL/Documents/NetBeansProjects/prjPassword/database/PasswordDB.accdb";

Project

Today you have learnt about the File I/O and JDBC. Now add database connectivity to your group project so that you do not lose any input and output to your data.