Lesson 13
Files And Databases:

The Ever Present Source of Pure Knowledge

Wholeness Statement

File structures make storing and retrieving persistent data efficiently. A file structure is an organization of data within disk blocks. The Absolute field of pure consciousness contains the infinite organizing power of natural law.

Files

- Java provides convenient tools for reading and writing files.
- Java supports reading and writing text streams and binary streams.
- In this lesson we focus on the API for text streams.
- Java represents each character with the UTF-16 encoding.
- Need to close files when finished.
- Prior to j2se5.0: Pass the File to a FileReader then pass from FileReader to a BufferedReader
- J2se5.0: Pass the File to a Scanner

The File Class

- The `File` class is an abstraction that represents either a file or a directory on the native system's directory system.
- Methods available in `File` include:
  - `boolean isFile`
  - `boolean isDirectory`
  - `boolean exists`
  - `String getAbsolutePath`
  - `String getParent`
  - `File getParentFile`
  - `boolean mkdir`
  - `boolean mkdirs`
  - `boolean delete`

File Writer / PrintWriter

- **FileWriter** creates a **Writer** that you can use to write to a file. It is most commonly used.
- constructors are shown here:
  - FileWriter(String *fileName*) throws IOException
  - FileWriter(String *fileName,* boolean *append*) throws IOException
- If *append* is **true**, then output is appended to the end of the file. Otherwise, the file is overwritten.
- **FileWriter** is derived from **OutputStreamWriter** and **Writer**. Thus, it has access to the methods defined by these classes.
- **PrintWriter**
  - Prints formatted representations of objects to a text-output stream.

FileReader

- The **FileReader** class creates a **Reader** that you can use to read the contents of a file.
- Its most commonly used constructor is shown here:
  - FileReader(String *fileName*) throws FileNotFoundException
- Here, *fileName* is the name of a file with full path. It throws a **FileNotFoundException** if the file does not exist. **FileReader** is derived from **InputStreamReader** and **Reader**.

Example(Read file using scanner)

```
File f = new File("word_test.txt");
Scanner s = new Scanner(f);
// Scanner s = new Scanner(new File("word_test.txt"));
String word;
try {
        while( (word = s.next()) != null){
                System.out.println(word);
        }
}
catch(Exception e){   }
s.close();
```

Example(Read File using BufferedReader)

```
File f = new File("word_test.txt");
FileReader fr = new FileReader(f);
BufferedReader reader =
        new   BufferedReader(fr);

 // To read many lines, loop like this:
 String line;
 while((line = reader.readLine()) != null) {
        //do something with the line
 }
reader.close();
```

Example

```
File file = new File("scores.txt");
PrintWriter output = new PrintWriter(file);
    // Write formatted output to the file
    output.print("Ram Sudhan ");
    output.println(90);
    output.print("Jane Smith ");
    output.println(85);
            System.out.println("Write Successfully");
    // Close the file
    output.close();
```

Demo code - Files Examples

- ReadData.java
- WriteData.java
- FileReadWriteDemo

Main Point 1

Reading a File in Java is accomplished by using a FileReader (or Scanner). Writing to a File is accomplished by using a FileWriter/PrintWriter.

More generally, "input" in human life is handled by the senses; "output" is handled by the organs of action; both have their source in the field of pure creative intelligence.

Interacting With A Database

- JDBC provides an API for interacting with a database using SQL – part of the jdk distribution.

- To interact efficiently with a database, you use the database vendor's *driver* that allows communication between the JVM and the database. Java comes with JavaDB, adapted from Apache's Derby dbms, which is a pure Java dbms implementation.
- Follow the guidelines from your Lesson13\JavaDBFiles\javadb_files\Setup.txt

Set-up steps for using JDBC

- Install the database software. In this course we will use JavaDb.

- Install the JDBC driver. This is the platform specific software that the JDBC API uses to communicate with your database. (JavaDb driver is automatically installed when Java is installed.)

  For example, MySql has a custom JDBC *driver* (most database systems provide such a driver for use in Java programs). The configuration step is to add the driver package as a *jar file* to your project.

Coding steps for JDBC

1. Assemble your database url that will tell JDBC the name of your data source– typically, store this database url as a constant.
2. In your code, first load the driver (if the dbms does not provide automatic registration – JavaDB does provide automatic registration).
3. Then create a connection object – you pass in the url at this stage. Typically try to reuse the connection object because it is costly to open up this connection, especially when you do it frequently.
4. Create a Statement object.

5. Prepare a String consisting of an SQL query, and pass it to the Statement instance, using either executeQuery(String sql) or updateQuery(String sql). executeQuery is for reads, updateQuery for updating or inserting.

6. Return value of these query methods in a ResultSet instance. A ResultSet encapsulates the rows of data that have been read. A pointer is (semantically) present that points to position –1 when you first get the ResultSet. Loop through using the next() method, which returns a boolean, to read the rows in the table. Read a row by specifying the column names you wish to look at. The getString(columnName) method returns the value in the current row of the result set having the specified column name, if that value is of String type. Some other options are getInt, getDouble, getDate.

7. When you have finished with the database, close the Statement instance with the close() method. When you are done with the Connection instance, close it with its close() method.
8. Installation Steps in the Folder : javadb_files\setup.txt

SQL Queries

- SELECT
  - used to read values from a table
- INSERT
  - used to insert a new row
- DELETE
  - used to remove one or more rows
- UPDATE
  - used to change values in already-existing rows

SELECT

```
SELECT custid, password
FROM Customer
WHERE custid = '1'
```

- This statement locates all records in the Customer table in which the custid column has value '1', and returns rows restricted to the two columns custid and password

INSERT

```
INSERT INTO ShippingAddresses
(custid,street,city,state,zip)
VALUES('1','B St','Fairfield','IA','52556')
```

- This statement inserts a new record into the ShippingAddresses table, placing values 1, B St, Fairfield, IA, and 52556 in the custid, street, city , state, and zip fields, respectively

DELETE

```
DELETE FROM Company
WHERE comp_name = 'Pepsi‘
```

- This statement deletes from the Company table all rows in which the comp_name field is equal to 'Pepsi'

UPDATE

```
UPDATE Employees
SET salary = salary * 1.10
WHERE dep_code = 32
```

- This statement changes the value in the salary field to 1.10 times the original value, for all records in which the dep_code equals 32

To enhance flexibility, we want to separate the User Interface (UI) from the Business Logic and the Database (DB).

Presentation Layer UI	Business Logic Layer	Persistence Layer DB

Business logic is the programming that manages communication between an end user interface and a database.

The purpose of your application's persistence layer is to use a session at run time to associate mapping data and a data source in order to create, read, update, and delete persistent objects using queries and expressions, as well as transactions

Main Point 2

JDBC provides an API for interacting with a database using SQL. To interact efficiently with a database, you typically use the database vendor's driver that allows communication between the JVM and the database. *This is reminiscent of the Principle of Diving – once the initial conditions have been met, a good dive is automatic. (Here, the initial conditions are correct configuration of the data source and code to load the database driver; once the set-up is right, interacting with the database is "effortless".)*

Demo program

Database Examples Folder

On Sakai – Refer-Fiolders

JavaDBPrograms

JavaDBFiles

UNITY CHART

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

Expansion of consciousness leads to expanded territory of influence

1. Since Java is an OO language, it supports storage and manipulation of data within appropriate objects.

2. To work with real data effectively, Java supports interaction with external data stores (databases) through the use of various JDBC drivers, and the JDBC API.

Transcendental Consciousness: TC is the field of truth, the field of *Sat*. "Know that by which all else is known." – *Upanishads*

Impulses within the Transcendental field: *These infinitely diverse impulses which create the whole universe, flow within the one holistic field of pure consciousness, transcendental consciousness.*

Wholeness moving within Itself*: In Unity Consciousness, the final truth about life is realized in a single stroke of knowledge.*