# **Chapter 3 Lecture Terms / Questions**

1. OLE DB
   1. Visual C# provides objects that work with Object Linking and Embedding Databases
2. ODBC
   1. Open Database Connectivity
3. MSDE
   1. SQL Server, including the Microsoft Data Engine
4. Connection object
   1. establishes a connection between your application and the database.
5. Connection object properties & their definitions

**ConnectionString** Contains the information used to establish a connection to a database.

**Database** Name of the current database.

**DataSource** Name of the current database file.

**State** Current connection state (Closed, Connecting, Open, Executing, Fetching, Broken)

1. Connection object methods & their definitions

**Open** Open the connection.

**Close** Close the connection.

**Dispose** Dispose of the connection object**.**

1. 3 C# statements needed to set up the OleDb connection to the database

**using System.Data.OleDb;**

**OleDbConnection myConnection;**

**myConnection = new OleDbConnection(myConnectionString);**

1. What should the ConnectionString look like?

**Provider=Microsoft.ACE.OLEDB.12.0; Data Source = DatabaseName**

1. What is the Load event of the form program? When does it occur?
   1. Occurs before a form is displayed for the first time
2. Command objects
   1. used to define commands to send to the database.
3. What will we use to write our commands?
   1. these commands will be SQL queries, Command objects are used by **data** **adapter** objects
4. On page 3-16, what it command to read records from the Titles table?
   1. **Select \* From Titles**
5. Data containers
   1. Datasets
6. Data adapter objects
   1. object uses a **command** **object** to transfer data between a **connection** **object** and a **dataset** containing one or more **data table(s)**.
7. Data adapter properties & their definitions

**DeleteCommand** Command object that allows the adapter to delete rows from a data table.

**InsertCommand** Command object that allows the adapter to insert rows into a data table.

**SelectCommand** Command object that the adapter uses to select rows for a data table.

**UpdateCommand** Command object that the adapter uses to update rows in a data table.

1. Data adapter methods & their definitions

**Fill** Fills the data adapter with a given data table.

**Update** Updates the data table held by the data adapter.

1. Dataset objects
   1. provides all the features you need to build, load, store, manipulate and save data in a relational database
2. 2 C# statements to declare and construct a Dataset object

**DataSet myDataSet;**

**myDataSet = new DataSet()**

1. Data table objects
   1. object represents the data contained in one table of a **dataset** object.
2. Data table properties & their definitions

**Columns** Collection of **DataColumn** objects that define information in the columns of the table.

**DataSet** The **DataSet** object (if any) that contains this **DataTable**.

**DefaultView** A **DataView** representing the **DataTable** contents.

**Rows** Collection of **DataRow** objects that define each table row.

**TableName** Name of data table.

1. Data table methods & their definitions

**Clear** Removes all rows from data table.

**Copy** Makes a copy of the data table.

**ImportRow** Copies a **DataRow** object into the data table.

**NewRow** Creates a new **DataRow** object for the data table.

1. 2 C# statements to declare and construct a Datatable object

**DataTable myTable;**

**myTable = new DataTable();**

1. C# statement to fill the table with data from the database

**myAdapter.Fill(myTable);**

1. DataRow object
   1. represents the information in one record of a data table
2. Data row properties & their definitions

**Item** Gets or sets one of the row’s fields. The parameter can be the column index or the field name.

**Table** Reference to data table containing the row.

1. Data row methods & their definitions

**BeginEdit** Puts data row in edit mode.

**CancelEdit** Cancels the current edit on the data row.

**Delete** Deletes the row from the data table.

1. Data bound control
   1. that are special controls with properties established by database fields
2. Types of controls that might be used as data bound controls

**Label** Can be used to provide display-only access to a specified text data field. The label **Text** property is usually bound.

**Text Box** Can be used to provide read/write access to a specified text data field. Probably, the most widely used data bound tool. **Text** property is usually bound.

**Check Box** Used to provide read/write access to a Boolean field. **Value** property is data bound.

**Picture Box** Used to display a graphical image from a bitmap, icon, gif, jpeg, or metafile file. Provides read/write access to a image/binary data field. **Image** property is data bound.

**List Box** Can be used to display all values of a particular field in a database.

**Combo Box** Can be used to display all values of a particular field in a database.

**Data Grid View** Can be used to display an entire database table.

1. C# statement to bind a control to a database table field (Look on p. 3-33 to see examples of this too)

**myControl.DataBindings.Add(myProperty, myTable, myField);**

1. CurrencyManager object
   1. Once associated with a **data** **table** object, the currency manager not only allows navigation through the rows of the table, but also provides several database management tasks such as editing, adding record and deleting records.
2. 4 database tasks that currency manager can do
   1. Navigation
   2. Editing
   3. Adding
   4. Deleting
3. Currency manager properties & their definitions

**Bindings** Collection of controls bound to this manager.

**Count** Number of rows in table associated with manager.

**Position** Gets or sets the current row in the data table (ranges from 0 to Count – 1).

1. Currency manager methods & their definitions

**AddNew** Add a new row to the data source.

**CancelCurrentEdit** Cancels the current editing operation.

**EndCurrentEdit** Ends the current editing operation, accepting any changes.

**Refresh** Refreshes the bound controls.

**RemoveAt** Removes the indicated data row.

1. C# line to declare a CurrencyManager object

**CurrencyManager myManager;**

1. Look on p. 3-42 and write one of the statements that changes the form’s current position in the Titles table

**private void BtnFirst\_Click(object sender, EventArgs e)**

**{**

**titlesManager.Position = 0;**

**}**

**private void BtnPrevious\_Click(object sender, EventArgs e)**

**{**

**titlesManager.Position--;**

**}**

**private void BtnNext\_Click(object sender, EventArgs e)**

**{**

**titlesManager.Position++;**

**}**

**private void BtnLast\_Click(object sender, EventArgs e)**

**{**

**titlesManager.Position = titlesManager.Count - 1;**

**}**

1. Data wizards
   1. When learning new material, many times you are taught the “hard way” to do something before learning the “easy way.” We’re going to show you an easier way to build the application we just built by using Visual C#’s
2. Benefits/drawbacks of data wizard use
   1. Even though wizards are easy to use, they have their drawbacks. That’s why you haven’t seen wizards yet – we will always use the “hard way” (data objects) to connect to databases. Wizards provide quick results, but their results are many times inflexible. For databases, the big drawback is that connection information is hard-coded into your programs, making distribution to others very difficult.