Declaring Variables:

Syntax: ***[Public |Private| Dim] <variable name> as <Data type>***

Examples:

```
Public varItem as string
Private varQty as integer
Dim varPrice as decimal
```

Conditional Statement:

Syntax:

```
If <consdition 1> then
        <statement 1>
[elseif <condition 2> then
        <statement 2>
…
 Elseif <condtion n>  then
        <statement n>
Else
        <else statement>]
End if
```

LOOPS:

1. For … Next statement
   Syntax:
   ```
   For counter [ As datatype ] = <start value> To <end value> [ Step <increment> ]
           [ statements ]
   Next [ counter ]
   ```

2. While … End While statement
   Syntax:
   ```
   While <condition >
      [ statements]
    End While
   ```

3. Do… Loop
   Syntax:
   ```
   Do {While | Until} <condition  >
      [ statements]
   Loop
   ```

Establishing Database connection.

- Declare a connection variable
  Private <connection variable> as Common.DbConnection
  Note: must be placed outside of any event or user-defined procedures
- Create the connection string
  <connection variable> = new DB2Connection ("[server = <server name>;] database = <database name>;" + "uid = <user name>; password = <password>;")
- Open the connection
  <connection variable>. Open

Try … catch … statement

- It is an exception handler. It gives us a way to handle possible errors that may occur in a given block of code during runtime.
- In case of errors, your program can terminate "*gracefully*"

Syntax:

```
Try
        [try statements]
[Catch [ exception [ As type]]
        [ catch Statements]
End try
```

Example:

```
        Try
                Me.txtlastname = "Isaac Newton"
        Catch ex As Exception
                Msgbox(ex.tostring)
        End Try
```

DB2Command Class:

- Represents an SQL statement or stored procedure to execute against a data source.
- Methods for executing commands against a database.
  - ExecuteReader - Executes commands that return rows (SELECT)
  - ExecuteNonQuery - Executes commands such as SQL INSERT, DELETE, UPDATE, and SET statements.
- Syntax:
  - Dim <variable> As New DB2Command(<Sql as string>,<connection>) or
  - <db2Command variable> = new DB2Command(<Sql as string>,<connection>)
    NOTE: the "sql string can be defined outside of the DB2 command class"

Example:

1. insert these data, (1001, ‘Pad Paper’,50,34.50) into the ITEM table.

   ```
   Dim SqlStr as string = “insert into item values (1001,’Pad Paper’, 50, 34.50)”
   Dim Cmd as new Db2Command(SqlStr, DbConn)
   Cmd.ExcuteNonQuery
   ```

   Or

   ```
   Dim Cmd as DB2Command
   Cmd = new DB2Command(“insert into items values (1001,’Pad Paper’, 50, 34.50)”,
            DBConn)
   Cmd.ExecuteNonQuery
   ```

2. Update the quantity, reduce it by 10, of item 1001

   ```
   Dim SqlUpdate as string
   Dim CmdUpdate as DB2Command

   SqlUpdate = “update items set itemQty = ItemQty – 10 where itemNo = 1001”
   CmdUpdate = new DB2command(SqlUpdate,DBConn)
   CmdUpdate.ExecuteNoneQuery
   ```

3. Retrieving all rows from table ITEMS.

   ```
   Dim SqlRet as string
   Dim CmdSelect as DB2Command
   Dim SelectReader as DB2DataReader

   SqlRet = “select itemNo, itemDesc, itemQty, itemPrice from item”
   CmdSelect = new DB2Command(SqlRet, DBConn)
   SelectReader = CmdSelect.ExecuteReader
   ```

Data Grid:

<u>Filling Up a DataGrid View Header</u>

- Set the number of columns using ColumnCount property
  Syntax: ***<datagrid name>.ColumnCount = <no of columns>***

```
DataGridView1.ColumCount = 3
```

- Set the column names using the Name property
  Syntax: <***datagrid name>.Columns(<index>).Name = <column name>***

```
DataGridView1.Columns(0).Name = "Department"
```

- Set the column width using the Width property
  Syntax: ***<datagrid name>.Columns(<index>).Width = <column width>***

```
DataGridView1.Columns(0).Width = 150
```

Filling Up the Data Grid:

- With the data reader that holds the data populates the grid using the Data Grid Add row function.
- Syntax:
  <data grid name>.Rows.Add(<row var as array of string>)

- Example:

```
dim row as String()
while rdr.Read()
    row = new String(){rdr.GetSting(0),…, rdr.GetString(n)}
    dgView.Rows.Add(row)
End While
```

Fetching Data from the DataGrid: (with the MouseUp Event)

Syntax: ***<variable> = <datagrid name>.CurrentRow.Cells(<index>).Value[.ToString]***

```
Dim VCode as Integer
Dim VName as String

VCode = MyDataGrid.CurrentRow.Cells(0).Value
VName = MyDataGrid.CurrentRow.Cells(1).Value.ToString
```