File	Something	Code

Prefixes.cs	Assign getters and setters	`public string Prefix { get; set; }`
		`public string PrefixDesc { get; set; }`

frmPrefixes.cs	Design Mode – add fields and datagrid	Fields are: txtPrefixCode, btnGetDescription, btnAdd, btnModify, btnDelete, btnRefresh
frmPrefixes.cs	Create a field that will access the getters and setters in the Prefixes.cs file	`private Prefixes prefixes;`
PrefixesDB.cs	Change class to public	`public class PrefixesDB`

	Add using statement to database	`using System.Data;`
		`using System.Data.OleDb;`

PrefixesDB.cs	Create method for GetPrefixDetails	`public static Prefixes GetPrefixDetails(string preCode)`
	Create connection to database using method already set up in the BadgesDatabaseDB.cs file	`OleDbConnection connection = BadgesDatabaseDB.GetConnection();`

	Create SELECT string	`string selectStatment =`
		`    "SELECT Prefix, PrefixDesc " +`
		`    "FROM PrefixDesc " +`
		`    "WHERE Prefix = @preCode";`

	Create command by combining the SELECT string and the DB connection	`OleDbCommand selectCommand =`
		`     new OleDbCommand(selectStatement, connection);`

	Add the value of the preCode passed from the calling class	`selectCommand.Parameters.AddWithValue("@preCode", preCode);`
	Inside a try/catch/finally code block:	
	Open connection to DB	`connection.Open();`

	Execute selectCommand to grab single row from DB	`OleDbDataReader dataReader =`
		`         selectCommand.ExecuteReader(CommandBehavior.SingleRow);`

	Inside an if/else code block:	

	If a row a match is found, create a local variable to store the found information, and return that back to the calling method	`Prefixes prefixes = new Prefixes();`
		`         prefixes.Prefix = dataReader["Prefix"].ToString();`
		`         prefixes.PrefixDesc = dataReader["PrefixDesc"].ToString();`
		`         return prefixes;`

	Else no value	`return null;`
	To theh Catch block, catch and throw DB exeception	`throw ex;`
	Finally, close connection	`connection.Close();`

frmPrefixes.cs	Add Method GetPrefixDetails	`private void GetPrefixDetails(string preCode)`
	Inside a try/catch block, call the GetPrefixDetails method from the Prefixes.DB file, assigning that value to the private field first assigned in this class	`prefixes = PrefixesDB.GetPrefixDetails(preCode);`
	Else catch and convert the ex thrown in the Prefixes.DB file to a message	`MessageBox.Show(ex.Message, ex.GetType().ToString());`
frmPrefixes.cs	Add a DisplayResults method, which will use the field that accesses the getters and setters	`txtDescription.Text = prefixes.PrefixDesc;`
frmPrefixes.cs	Add function to btnGetDescription	
	Verify user has added input	`if (Validator.IsPresent(txtPrefixCode))`
	Get user input	`string prefixCode = txtPrefixCode.Text;`
	Call the GetPrefixDetails method, passing the string prefixCode	`this.GetPrefixDetails(prefixCode);`
	If no records found (if prefix == null)	`MessageBox.Show("No info found.", "Not Found");`

	Else record found, enable the Modify and Delete buttons	`btnModify.Enabled = true;`
		`btnDelete.Enabled = true;`

	And then call the DisplayResults method	`this.DisplayResults();`
frmAddModifyPrefixes.cs	Design Mode – add fields and properties	Fields are: txtPrefixCode, txtPrefixDesc, btnAccept, btnCancel Properties are: AcceptButton = btnAccept, CancelButton = btnCancel, FormBorderStyle = FixedDiaglog, MaximizeBox = False, StartPosition = CenterScreen, Text = <empty>, ControlBox = false;
frmAddModifyPrefixes.cs	Create a field that will access the getters and setters in the Prefixes.cs file	`public Prefixes prefix;`
	Add a bool field to determine if the purpose of the form is to Add or Modify, based on what the user selected from the form	`public bool addPrefix;`

	if/else statement on whether to treat form as an Add or Modify	`if (addPrefix)`
		`{ this.Text = "Add Prefix"; }`
		`else`
		`{ this.Text = "Modify Prefix";`
		`  this.DisplayPrefix(); }`

frmAddModifyPrefixes.cs	Add method DisplayPrefix	`txtPrefixCode.Text = prefix.Prefix;`

	which will use the field that accesses the getters and setters	`txtPrefixDesc.Text = prefix.PrefixDesc;`
frmAddModifyPrefixes.cs	Add bool method IsValidData to determine if required fields have content This will use the Validator class	`return Validator.IsPresent(txtPrefixCode) && Validator.IsPresent(txtPrefixDesc);`
frmAddModifyPrefixes.cs	Create PutPrefixData method That assigns values to the getters and setters	`prefix.Prefix = txtPrefixCode.Text; prefix.PrefixDesc = txtPrefixDesc.Text;`
PrefixesDB.cs	Create method for AddPrefix	`public static void AddPrefix(Prefixes prefix)`
	Create connection to database using method already set up in the BadgesDatabaseDB.cs file	`OleDbConnection connection = BadgesDatabaseDB.GetConnection();`
	Create insert string	`string insertStatement = "INSERT INTO [PrefixDesc] " + "(Prefix, PrefixDesc) " + "VALUES (@Prefix, @PrefixDesc)";`
	Create command by combining the INSERT string and the DB connection	`OleDbCommand insertCommand = new OleDbCommand(insertStatement, connection);`
	Add the values passed from the calling class	`insertCommand.Parameters.AddWithValue ("@Prefix", prefix.Prefix); insertCommand.Parameters.AddWithValue ("@PrefixDesc", prefix.PrefixDesc);`
	Open connection to DB	`connection.Open();`
	Execute insertCommand	`insertCommand.ExecuteNonQuery();`
	Close connection	`connection.Close();`
PrefixesDB.cs	Create method for UpdatePrefix Notice that this is not a void statement, but will return a bool value	`public static bool UpdatePrefix(Prefixes oldPrefix, Prefixes newPrefix)`
	Create connection to database using method already set up in the BadgesDatabaseDB.cs file	`OleDbConnection connection = BadgesDatabaseDB.GetConnection();`
	Create update string	`string updateStatement = "UPDATE Prefixes SET " + "Prefix = @NewPrefix, " + "PrefixDesc = @NewPreDesc " + "WHERE Prefix = @OldPrefix";`

	Create command by combining the UPDATE string and the DB connection	`OleDbCommand updateCommand =` `new OleDbCommand(updateStatement, connection);`
	Add the values passed from the calling class	`updateCommand.Parameters.AddWithValue` `("@NewPrefix", newPrefix.Prefix);` `updateCommand.Parameters.AddWithValue` `("@NewPreDesc", newPrefix.PrefixDesc);` `updateCommand.Parameters.AddWithValue` `("@OldPrefix", oldPrefix.Prefix);`
	Inside a try/catch/finally block:	
	Open connection	`connection.Open();`
	Create an int count that will increase by one if the update command works properly, and return a bool value as a result	`int count = updateCommand.ExecuteNonQuery();` `if (count > 0)` `return true;` `else` `return false;`
	Catch exceptions	`catch (OleDbException ex)` `{` `throw ex;` `}`
	Finally	`connection.Close();`
PrefixesDB.cs	Create method for UpdatePrefix Notice that this is not a void statement, but will return a bool value	`public static bool DeletePrefix(Prefixes prefix)`
	Create connection to database using method already set up in the BadgesDatabaseDB.cs file	`OleDbConnection connection = BadgesDatabaseDB.GetConnection();`
	Create delete string	`string deleteStatement =` `"DELETE FROM PrefixDesc " +` `"WHERE Prefix = @Prefix " +` `"AND PrefixDesc = @PrefixDesc";`
	Create command by combining the DELETE string and the DB connection	`OleDbCommand deleteCommand =` `new OleDbCommand(deleteStatement, connection);`
	Add the values passed from the calling class	`deleteCommand.Parameters.AddWithValue` `("@Prefix", prefix.Prefix);` `deleteCommand.Parameters.AddWithValue` `("@PrefixDesc", prefix.PrefixDesc);`
	Inside a try/catch/finally block:	
	Open connection	`connection.Open();`

	Create an int count that will increase by one if the delete command works properly, and return a bool value as a result	`int count = deleteCommand.ExecuteNonQuery();`
		`if (count > 0)`
		`    return true;`
		`else`
		`    return false;`
	Catch exceptions	`catch (OleDbException ex)`
		`{`
		`    throw ex;`
		`}`
	Finally	`connection.Close();`
frmPrefixes.cs	Add function to btnAdd	
	Create instance of frmAddModifyPrefixes	`frmAddModifyPrefixes addPrefixForm = new frmAddModifyPrefixes();`
	Set bool of addPrefix to true This uses the bool from the frmAddModifyPrefixes.cs file	`addPrefixForm.addPrefix = true;`
	Not sure what this line does	`DialogResult result = addPrefixForm.ShowDialog();`
		`if (result == DialogResult.OK)`
		`{`
		`    prefix = addPrefixForm.prefix;`
		`}`
frmPrefixes.cs	Add function to btnRefresh Close and re-open form	`this.Close();`
		`Form prefix = new frmPrefixes();`
		`prefix.ShowDialog();`
frmPrefixes.cs	Add function to btnModify	
	Create instance of frmAddModifyPrefxes	`frmAddModifyPrefixes modifyPrefixForm = new frmAddModifyPrefixes();`
	Set bool of addPrefix to false This uses the bool from the frmAddModifyPrefixes.cs file	`modifyPrefixForm.addPrefix = false;`
	??	`modifyPrefixForm.prefix = prefix;`
	??? This DiaglogResult.OK comes from the frmAddModifyPrefixes.cs file	`DialogResult result = modifyPrefixForm.ShowDialog();`
		`if (result == DialogResult.OK)`
		`{`
		`    prefix = modifyPrefixForm.prefix;`
		`    this.DisplayResults();`
		`}`
		`else if (result == DialogResult.Retry)`
		`{`

		`    this.GetPrefixDetails(prefix.Prefix);`
		`    if (prefix != null)`
		`        this.DisplayResults();`
		`    else`
		`        //this.ClearControls();  write this method`
		`        txtPrefixCode.Text = "";`
		`        txtDescription.Text = "";`
		`}`

frmAddModifyPrefixes.cs	Add function to btnAccept	
	Verify required fields are present in an if statement	`if (IsValidData())`
	If the user is adding a prefix code:	`if (addBlank)`
	Create new instance of a Prefix	`prefix = new Prefixes();`
	Pass this prefix instance to the PutPrefixData method	`this.PutPrefixData(prefix);`

	Inside a try/catch block, call the AddPrefix method from the PrefixesDB class	`PrefixesDB.AddPrefix(prefix);`
		`MessageBox.Show("Operation successful.", "Successful", MessageBoxButtons.OK,`
		`    MessageBoxIcon.None);`
		`this.Close();`

	If the user is modifying a prefix code:	
frmPrefixes.cs	Add function to btnDelete	

	Show messagebox with prefix details, and confirmation of deletion	`DialogResult result = MessageBox.Show("Delete " + prefix.Prefix + " ?",`
		`    "ConfirmDelte", MessageBoxButtons.YesNo, MessageBoxIcon.Question);`

		`if (result == DialogResult.Yes)`

	try/catch	`try`
		`{`
		`    if (! PrefixesDB.DeletePrefix(prefix))`
		`    {`
		`        MessageBox.Show("?", "Database Error");`
		`        this.GetPrefixDetails(prefix.Prefix);`
		`        if (prefix != null)`
		`        {`
		`            this.DisplayResults();`
		`        }`
		`        else`
		`        {`
		`            this.ClearControls();`
		`        }`
		`    }`

		```
		}
		catch (Exception ex)
		{
		    MessageBox.Show(ex.Message, ex.GetType().ToString());
		}
		```