Program Cover Sheet

|  |
| --- |
| Name: Tajbid Hasib |
| Assignment: 4 |
| List any parts of the assignment that do not work/were not completed:  (none) |

|  |
| --- |
| Instructor’s Comments: |
| Grade: |

Program Submission Requirements: (1) all files, zipped and uploaded to Canvas and (2) a completed cover sheet, program execution screenshots and source code printed, **stapled** and turned in during class. Failure to follow the submission requirements will result in points lost on that particular assignment.

**dbModule.vb**

Imports System.Data.SqlClient

'------------------------------------------------------------

'- File Name : dbModule.vb -

'- Part of Project: Assign4 -

'------------------------------------------------------------

'- Written By: Tajbid Hasib -

'- Written On: 10/15/2019 -

'------------------------------------------------------------

'- File Purpose: -

'- This module creates and sets up databases

'------------------------------------------------------------

'- Program Purpose: -

'- -

'- This program can be used to manage an auto service store

'- by communicating with databases

'------------------------------------------------------------

'- Global Variable Dictionary: -

'- (none)

'------------------------------------------------------------

Module dbModule

Const fileName = "dbModule.vb"

'------------------------------------------------------------

'- Subprogram Name: Main -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine initiates the module.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (none)

'------------------------------------------------------------

'- Local Variable Dictionary -

'- strDBPATH - string path to DB

'- strCONNECTION - DB Connection string

'------------------------------------------------------------

Sub Main()

Const strDBNAME As String = "Assignment4" 'Name of database

'Name of the database server

Const strSERVERNAME As String = "(localdb)\MSSQLLocalDB"

'Path to database in executable

Dim strDBPATH As String = My.Application.Info.DirectoryPath &

"\" & strDBNAME & ".mdf"

'This is the full connection string

Dim strCONNECTION As String = "SERVER=" & strSERVERNAME & ";DATABASE=" &

strDBNAME & ";Integrated Security=SSPI;AttachDbFileName=" &

strDBPATH

DeleteDatabase(strSERVERNAME, strDBNAME)

'If the database doesn't exist, create it

If Not (System.IO.File.Exists(strDBPATH)) Then

CreateDatabase(strSERVERNAME, strDBNAME, strDBPATH, strCONNECTION)

End If

'Make sure all tables are cleaned out each time we run this

CleanOutMechanicsTable(strCONNECTION)

CleanOutCustomersTable(strCONNECTION)

CleanOutScheduleTable(strCONNECTION)

CleanOutVehiclesTable(strCONNECTION)

CleanOutServicesTable(strCONNECTION)

'Put some data into the tables

PopulateMechanicsTable(strCONNECTION) '1 student record

PopulateServicesTable(strCONNECTION) '6 courses

'DeleteDatabase(strSERVERNAME, strDBNAME)

frmMain.ShowDialog()

End Sub

'------------------------------------------------------------

'- Subprogram Name: CreateDatabase -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine initiates the program.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- strSERVERNAME – string server name-

'- strDBNAME - string DB name -

'- strDBPATH - string DB Path

'- strCONNECTION - string connection to DB

'------------------------------------------------------------

'- Local Variable Dictionary -

'- strSQLCmd - String containing SQL command

'- mechanics - array containing Mechanic objects

'------------------------------------------------------------

Sub CreateDatabase(ByVal strSERVERNAME As String, ByVal strDBNAME As String,

ByVal strDBPATH As String, ByVal strCONNECTION As String)

'Let's build a SQL Server database from scratch

Dim DBConn As SqlConnection

Dim strSQLCmd As String

Dim DBCmd As SqlCommand

'All we need to do initially is just point at the server

DBConn = New SqlConnection("Server=" & strSERVERNAME)

'Let's write a SQL DDL Command to build the database

'There are a lot of other parameters but we can let them default

'All we need are these three

strSQLCmd = "CREATE DATABASE " & strDBNAME & " On " &

"(NAME = '" & strDBNAME & "', " &

"FILENAME = '" & strDBPATH & "')"

DBCmd = New SqlCommand(strSQLCmd, DBConn)

Try

'Open the connection and try running the command

DBConn.Open()

DBCmd.ExecuteNonQuery()

'MessageBox.Show("Database was successfully created", "",

'MessageBoxButtons.OK, MessageBoxIcon.Information)

Catch ex As Exception

'If we can't build the database, we are dead in the water so bail...

'MessageBox.Show(ex.ToString())

'MessageBox.Show("Cannot build database! Closing program down...")

End

End Try

'We are currently pointing at the [MASTER] database, so we

'need to close the connection and reopen it pointing at the

'Registration database...

If (DBConn.State = ConnectionState.Open) Then

DBConn.Close()

End If

'Now we need to use the full connection string with the Integrated

'Security line, et cetera

DBConn = New SqlConnection(strCONNECTION)

DBConn.Open()

'Build the Customers Table

DBCmd.CommandText = "CREATE TABLE Mechanics(" &

"TUID varchar(6), " &

"Name varchar(50)," &

"Rate varchar(2))"

DBCmd.Connection = DBConn

Try

DBCmd.ExecuteNonQuery()

'MessageBox.Show("Created Mechanics Table")

Catch Ex As Exception

'MessageBox.Show("Mechanics Table Already Exists")

End Try

'Build the Customers Table

DBCmd.CommandText = "CREATE TABLE Customers(" &

"TUID varchar(6), " &

"Name varchar(50))"

DBCmd.Connection = DBConn

Try

DBCmd.ExecuteNonQuery()

'MessageBox.Show("Created Customers Table")

Catch Ex As Exception

'MessageBox.Show("Customers Table Already Exists")

End Try

'Build the Registration Table

DBCmd.CommandText = "CREATE TABLE Vehicles(" &

"TUID varchar(6), " &

"Customer\_TUID varchar(6)," &

"Vehicle\_Description varchar(100))"

DBCmd.Connection = DBConn

Try

DBCmd.ExecuteNonQuery()

'MessageBox.Show("Created Vehicles Table")

Catch Ex As Exception

'MessageBox.Show("Vehicles Table Already Exists")

End Try

'Build the Registration Table

DBCmd.CommandText = "CREATE TABLE Services(" &

"TUID varchar(6), " &

"ServiceName varchar(50)," &

"TimeRequired varchar(6))"

DBCmd.Connection = DBConn

Try

DBCmd.ExecuteNonQuery()

'MessageBox.Show("Created Services Table")

Catch Ex As Exception

'MessageBox.Show("Services Table Already Exists")

End Try

'Build the Registration Table

DBCmd.CommandText = "CREATE TABLE Schedule(" &

"TUID varchar(6), " &

"Week varchar(6)," &

"Mechanic varchar(100)," &

"Appointment\_Day varchar(10)," &

"Customer varchar(100)," &

"Vehicle varchar(100)," &

"Service varchar(100)," &

"Start\_Time varchar(10)," &

"End\_Time varchar(10))"

DBCmd.Connection = DBConn

Try

DBCmd.ExecuteNonQuery()

'MessageBox.Show("Created Schedule Table")

Catch Ex As Exception

'MessageBox.Show("Schedule Table Already Exists")

End Try

'We can check to see if we're open before trying to

'issue a connection close

If (DBConn.State = ConnectionState.Open) Then

DBConn.Close()

End If

End Sub

'------------------------------------------------------------

'- Subprogram Name: CleanOutMechanicsTable -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine clears mechanics table from DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- strConn - string containing DB Connection

'------------------------------------------------------------

'- Local Variable Dictionary -

'- DBConn – SQL Connection Object-

'- DBCmd - SQL Command object -

'------------------------------------------------------------

Sub CleanOutMechanicsTable(ByVal strConn As String)

Dim DBConn As SqlConnection

Dim DBCmd As SqlCommand = New SqlCommand()

'Now try to open up a connection to the database

DBConn = New SqlConnection(strConn)

DBConn.Open()

'Use SQL DML to zap the contents of the table

DBCmd.CommandText = "DELETE FROM Mechanics"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

'MessageBox.Show("Deleted Everything In Mechanics")

DBConn.Close()

End Sub

'------------------------------------------------------------

'- Subprogram Name: CleanOutCustomersTable -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine clears customers table from DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- strConn - string containing DB Connection

'------------------------------------------------------------

'- Local Variable Dictionary -

'- DBConn – SQL Connection Object-

'- DBCmd - SQL Command object -

'------------------------------------------------------------

Sub CleanOutCustomersTable(ByVal strConn As String)

Dim DBConn As SqlConnection

Dim DBCmd As SqlCommand = New SqlCommand()

'Now try to open up a connection to the database

DBConn = New SqlConnection(strConn)

DBConn.Open()

'Use SQL DML to zap the contents of the table

DBCmd.CommandText = "DELETE FROM Customers"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

'MessageBox.Show("Deleted Everything In Customers")

DBConn.Close()

End Sub

'------------------------------------------------------------

'- Subprogram Name: CleanOutScheduleTable -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine clears schedule table from DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- strConn - string containing DB Connection

'------------------------------------------------------------

'- Local Variable Dictionary -

'- DBConn – SQL Connection Object-

'- DBCmd - SQL Command object -

'------------------------------------------------------------

Sub CleanOutScheduleTable(ByVal strConn As String)

Dim DBConn As SqlConnection

Dim DBCmd As SqlCommand = New SqlCommand()

'Now try to open up a connection to the database

DBConn = New SqlConnection(strConn)

DBConn.Open()

'Use SQL DML to zap the contents of the table

DBCmd.CommandText = "DELETE FROM Schedule"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

'MessageBox.Show("Deleted Everything In Schedule")

DBConn.Close()

End Sub

'------------------------------------------------------------

'- Subprogram Name: CleanOutVehiclesTable -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine clears vehicles table from DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- strConn - string containing DB Connection

'------------------------------------------------------------

'- Local Variable Dictionary -

'- DBConn – SQL Connection Object-

'- DBCmd - SQL Command object -

'------------------------------------------------------------

Sub CleanOutVehiclesTable(ByVal strConn As String)

Dim DBConn As SqlConnection

Dim DBCmd As SqlCommand = New SqlCommand()

'Now try to open up a connection to the database

DBConn = New SqlConnection(strConn)

DBConn.Open()

'Use SQL DML to zap the contents of the table

DBCmd.CommandText = "DELETE FROM Vehicles"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

'MessageBox.Show("Deleted Everything In Vehicles")

DBConn.Close()

End Sub

'------------------------------------------------------------

'- Subprogram Name: CleanOutServicesTable -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine clears services table from DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- strConn - string containing DB Connection

'------------------------------------------------------------

'- Local Variable Dictionary -

'- DBConn – SQL Connection Object-

'- DBCmd - SQL Command object -

'------------------------------------------------------------

Sub CleanOutServicesTable(ByVal strConn As String)

Dim DBConn As SqlConnection

Dim DBCmd As SqlCommand = New SqlCommand()

'Now try to open up a connection to the database

DBConn = New SqlConnection(strConn)

DBConn.Open()

'Use SQL DML to zap the contents of the table

DBCmd.CommandText = "DELETE FROM Services"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

'MessageBox.Show("Deleted Everything In Services")

DBConn.Close()

End Sub

'------------------------------------------------------------

'- Subprogram Name: PopulateServicesTable -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine populates services table from DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- strConn - string containing DB Connection

'------------------------------------------------------------

'- Local Variable Dictionary -

'- DBConn – SQL Connection Object-

'- DBCmd - SQL Command object -

'------------------------------------------------------------

Sub PopulateServicesTable(ByVal strConn As String)

Dim DBConn As SqlConnection

Dim DBCmd As SqlCommand = New SqlCommand()

'Now try to open up a connection to the database

DBConn = New SqlConnection(strConn)

DBConn.Open()

'Add a student using SQL DML

DBCmd.CommandText = "INSERT INTO Services(TUID, ServiceName, TimeRequired) " &

"VALUES ('1','Oil Change','0.5')"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

DBCmd.CommandText = "INSERT INTO Services(TUID, ServiceName, TimeRequired) " &

"VALUES ('2','Tire Replacement','1')"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

DBCmd.CommandText = "INSERT INTO Services(TUID, ServiceName, TimeRequired) " &

"VALUES ('3','Brakes','3')"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

DBCmd.CommandText = "INSERT INTO Services(TUID, ServiceName, TimeRequired) " &

"VALUES ('4','Transmission Filter Replacement','2')"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

DBCmd.CommandText = "INSERT INTO Services (TUID, ServiceName, TimeRequired) " &

"VALUES ('5','Cooling System Cleaning','4')"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

DBConn.Close()

' MessageBox.Show("Services Table Populated")

End Sub

'------------------------------------------------------------

'- Subprogram Name: PopulateMechanicsTable -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine populates mechanics table from DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- strConn - string containing DB Connection

'------------------------------------------------------------

'- Local Variable Dictionary -

'- DBConn – SQL Connection Object-

'- DBCmd - SQL Command object -

'------------------------------------------------------------

Sub PopulateMechanicsTable(ByVal strConn As String)

Dim DBConn As SqlConnection

Dim DBCmd As SqlCommand = New SqlCommand()

'Now try to open up a connection to the database

DBConn = New SqlConnection(strConn)

DBConn.Open()

'Add student registration using SQL DML

DBCmd.CommandText = "INSERT INTO Mechanics(TUID, Name, Rate) " &

"VALUES ('1','Sue', '10')"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

DBCmd.CommandText = "INSERT INTO Mechanics(TUID, Name, Rate) " &

"VALUES ('2','Steve', '9')"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery()

DBConn.Close()

'MessageBox.Show("Mechanics Table Populated")

End Sub

'------------------------------------------------------------

'- Subprogram Name: DeleteDatabase -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine deletes entire DB.

'------------------------------------------------------------

Sub DeleteDatabase(ByVal strSERVERNAME As String, ByVal strDBNAME As String)

'This routine shows how to delete a database completely

'from code. It does not consider deleting the data from

'the tables nor dropping the tables -- it just zaps the

'database completely

Dim DBConn As SqlConnection

Dim strSQLCmd As String

Dim DBCommand As SqlCommand

'We need to point back at the [Master] database itself

DBConn = New SqlConnection("Server=" & strSERVERNAME)

'Try to force single ownership of the database so that we have the

'permissions to delete it

strSQLCmd = "ALTER DATABASE [" & strDBNAME & "] SET " &

"SINGLE\_USER WITH ROLLBACK IMMEDIATE"

DBCommand = New SqlCommand(strSQLCmd, DBConn)

Try

DBConn.Open()

DBCommand.ExecuteNonQuery()

'MessageBox.Show("Database set for exclusive use", "",

'MessageBoxButtons.OK, MessageBoxIcon.Information)

Catch ex As Exception

MessageBox.Show(ex.ToString())

End Try

If (DBConn.State = ConnectionState.Open) Then

DBConn.Close()

End If

'Now drop the database

strSQLCmd = "DROP DATABASE " & strDBNAME

DBCommand = New SqlCommand(strSQLCmd, DBConn)

Try

DBConn.Open()

DBCommand.ExecuteNonQuery()

'MessageBox.Show("Database has been deleted", "", MessageBoxButtons.OK,

'MessageBoxIcon.Information)

Catch ex As Exception

MessageBox.Show(ex.ToString())

End Try

If (DBConn.State = ConnectionState.Open) Then

DBConn.Close()

End If

End Sub

End Module

**frmMain.vb**

Imports System.Data.SqlClient

Imports System.ComponentModel

'------------------------------------------------------------

'- File Name : frmMain.vb -

'- Part of Project: Assign4 -

'------------------------------------------------------------

'- Written By: Tajbid Hasib -

'- Written On: 10/15/2019 -

'------------------------------------------------------------

'- File Purpose: -

'- This is main form for Turbo Auto Service

'------------------------------------------------------------

'- Program Purpose: -

'- -

'- This program can be used to manage an auto service store

'- by communicating with databases

'------------------------------------------------------------

'- Global Variable Dictionary: -

'- dsMechanics - dataset for mechanics

'- dsVehicles - dataset for vehicles

'- dsCustomers - dataset for customers

'- dsServices - dataset for services

'- dsShedule - dataset for schedule

'- customerID - incrementing ID for new customer

'- vehicleID - incrementing ID for new vehicle

'- scheduleID - incrementing ID for new appointment

'- DBConn - SQL Connection Object

'- DBCmd - SQL Command Object

'- strSQLCmd - String containing SQL command

'- strDBPath - string containing path to DB

'- strConnection - String containing DB connection

'- myConn - SQL Connection Object

'- DBAdaptMechanics - SQL Adapter for mechanics

'- DBAdaptCustomers - SQL Adapter for customers

'- DBAdaptServices - SQL Adapter for services

'- DBAdaptSchedule - SQL Adapter for schedule

'- DBAdaptVehicles - SQL Adapter for vehicles

'- arrMechanics - Array containing mechanics

'------------------------------------------------------------

Public Class frmMain

Const fileName As String = "frmMain.vb"

'Create a dataset to point to each table -- do it here so that we don't

'have to keep passing things around

Dim dsMechanics As New DataSet

Dim dsVehicles As New DataSet

Dim dsCustomers As New DataSet

Dim dsServices As New DataSet

Dim dsSchedule As New DataSet

'Beginning IDs

Dim customerID As Integer = 1

Dim vehicleID As Integer = 1

Dim scheduleID As Integer = 1

'Frequently used variables

Dim DBConn As SqlConnection

Dim DBCmd As SqlCommand = New SqlCommand()

Dim strSQLCmd As String

'Here's the connection string related pieces - the same as in the module

'A smarter way would be to declare them once, but I wanted the code to

'be as simple as possible

Const strDBNAME As String = "Assignment4" 'Name of database

'Name of the database server

Const strSERVERNAME As String = "(localdb)\MSSQLLocalDB"

'Path to database in executable

Dim strDBPATH As String = My.Application.Info.DirectoryPath &

"\" & strDBNAME & ".mdf"

'This is the full connection string

Dim strCONNECTION As String = "SERVER=" & strSERVERNAME & ";DATABASE=" &

strDBNAME & ";Integrated Security=SSPI;AttachDbFileName=" &

strDBPATH

'We'll also create a SqlConnection object since we will execute some

'straight SQL rather than relying on the DBAdapters...

Dim myConn As New SqlConnection(strCONNECTION)

'Likewise create three data adapters so we don't mess stuff up

'trying to be cute with one adapter

Dim DBAdaptMechanics As SqlDataAdapter

Dim DBAdaptSchedule As SqlDataAdapter

Dim DBAdaptCustomers As SqlDataAdapter

Dim DBAdaptVehicles As SqlDataAdapter

Dim DBAdaptServices As SqlDataAdapter

Dim arrMechanics() As Mechanic

'------------------------------------------------------------

'- Subprogram Name: frmMain\_Load -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine initiates the program.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- sender – Identifies which particular control raised the –

'- click event -

'- e – Holds the EventArgs object sent to the routine -

'------------------------------------------------------------

'- Local Variable Dictionary -

'- strSQLCmd - String containing SQL command

'- mechanics - array containing Mechanic objects

'------------------------------------------------------------

Private Sub frmMain\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Dim strSQLCmd As String

'Load up all Mechanics and Services since they will never change while the program runs

strSQLCmd = "Select \* From Mechanics"

DBAdaptMechanics = New SqlDataAdapter(strSQLCmd, strCONNECTION)

DBAdaptMechanics.Fill(dsMechanics, "Mechanics")

'Create mechanic objects

Dim mechanics(dsMechanics.Tables(0).Rows.Count) As Mechanic

For index As Integer = 0 To (dsMechanics.Tables(0).Rows.Count - 1)

Dim mID As String = dsMechanics.Tables(0).Rows(index).Item("TUID").ToString

Dim mName As String = dsMechanics.Tables(0).Rows(index).Item("Name").ToString

Dim mRate As Integer = CInt(dsMechanics.Tables(0).Rows(index).Item("Rate").ToString)

mechanics(index) = New Mechanic(mID, mName, mRate)

cmbSdlMechanic.Items.Add(dsMechanics.Tables(0).Rows(index).Item("Name").ToString)

Next

arrMechanics = mechanics

'Load up all Mechanics and Services since they will never change while the program runs

strSQLCmd = "Select \* From Services"

DBAdaptServices = New SqlDataAdapter(strSQLCmd, strCONNECTION)

DBAdaptServices.Fill(dsServices, "Services")

For index As Integer = 0 To (dsServices.Tables(0).Rows.Count - 1)

cmbAptService.Items.Add(dsServices.Tables(0).Rows(index).Item("ServiceName").ToString)

Next

End Sub

'------------------------------------------------------------

'- Subprogram Name: btnImportFile\_Click -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine is called when btnImportFile is clicked

'- in order to import a file

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- sender – Identifies which particular control raised the –

'- click event -

'- e – Holds the EventArgs object sent to the routine -

'------------------------------------------------------------

'- Local Variable Dictionary -

'- fileLocation - string containing file location

'- splitLine - array containing words from each line of file

'- customerName - string containing customer name

'- serviceName - string containing service name

'- serviceTimeRequired - double containing service time

'- vehicleName - string containing vehicle name

'------------------------------------------------------------

Private Sub btnImportFile\_Click(sender As Object, e As EventArgs) Handles btnImportFile.Click

Dim fileLocation As String

fileLocation = InputBox("Enter the location of .txt file to import:", "File Import")

If System.IO.File.Exists(fileLocation) Then

'Now try to open up a connection to the database

DBConn = New SqlConnection(strCONNECTION)

DBConn.Open()

For Each line As String In System.IO.File.ReadLines(fileLocation)

If line.Chars(0) = "C" Then

Dim splitLine() As String = Split(line, vbTab)

Dim customerName As String = Trim(splitLine(1))

addCustomer(customerName)

ElseIf line.Chars(0) = "V" Then

Dim splitLine() As String = Split(line, vbTab)

Dim currCustomerName As String = splitLine(1)

Dim vehicleName As String = splitLine(2)

addVehicle(currCustomerName, vehicleName)

ElseIf line.Chars(0) = "S" Then

Dim splitLine() As String = Split(line, vbTab)

Dim currCustomerName As String

Dim currVehicleName As String

Dim serviceName As String = ""

Dim serviceTimeRequired As Double

'Find ID of Customer

strSQLCmd = "Select \* From Customers WHERE Name = '" & splitLine(1) & "'"

DBAdaptCustomers = New SqlDataAdapter(strSQLCmd, strCONNECTION)

dsCustomers.Clear()

DBAdaptCustomers.Fill(dsCustomers, "Customers")

currCustomerName = dsCustomers.Tables(0).Rows(0).Item("Name").ToString

'Find ID of Vehicle

strSQLCmd = "Select \* From Vehicles WHERE Vehicle\_Description = '" & splitLine(2) & "'"

DBAdaptVehicles = New SqlDataAdapter(strSQLCmd, strCONNECTION)

dsVehicles.Clear()

DBAdaptVehicles.Fill(dsVehicles, "Vehicles")

currVehicleName = dsVehicles.Tables(0).Rows(0).Item("Vehicle\_Description").ToString

'Find Service ID and time required for service

For index As Integer = 0 To (dsServices.Tables(0).Rows.Count - 1)

If splitLine(3) = dsServices.Tables(0).Rows(index).Item("ServiceName").ToString Then

serviceName = dsServices.Tables(0).Rows(index).Item("ServiceName").ToString

serviceTimeRequired = CDbl(dsServices.Tables(0).Rows(index).Item("TimeRequired"))

Exit For

End If

Next

addAppointment(currCustomerName, currVehicleName, serviceName, serviceTimeRequired)

End If

Next

DBConn.Close()

refreshDgvs()

refreshWeeks()

Else

MessageBox.Show("File Does Not Exist!", "Error")

End If

End Sub

'------------------------------------------------------------

'- Subprogram Name: btnAddCustomer\_Click -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine attempts to add new customer to DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- sender – Identifies which particular control raised the –

'- click event -

'- e – Holds the EventArgs object sent to the routine -

'------------------------------------------------------------

'- Local Variable Dictionary -

'- customerName - String containing new customer name

'------------------------------------------------------------

Private Sub btnAddCustomer\_Click(sender As Object, e As EventArgs) Handles btnAddCustomer.Click

If txtCustomerName.Text.Length < 3 Then

MessageBox.Show("Customer name must be at least 3 characters.")

Else

DBConn = New SqlConnection(strCONNECTION)

DBConn.Open()

Dim customerName As String = txtCustomerName.Text

addCustomer(customerName)

DBConn.Close()

refreshDgvs()

MessageBox.Show("Added Customer Successfully")

End If

End Sub

'------------------------------------------------------------

'- Subprogram Name: btnAddVehicle\_Click -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine attempts to add new vehicle to DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- sender – Identifies which particular control raised the –

'- click event -

'- e – Holds the EventArgs object sent to the routine -

'------------------------------------------------------------

'- Local Variable Dictionary -

'- currCustomerName - String containing new customer name

'- vehicleName - String containing new vehicle name

'------------------------------------------------------------

Private Sub btnAddVehicle\_Click(sender As Object, e As EventArgs) Handles btnAddVehicle.Click

If txtVehicleName.Text.Length < 3 Then

MessageBox.Show("Vehicle Make/Model must be at least 3 characters.")

ElseIf IsNothing(cmbVehicleCustomer.SelectedItem) Then

MessageBox.Show("Please select a customer.")

Else

DBConn = New SqlConnection(strCONNECTION)

DBConn.Open()

Dim currCustomerName As String = cmbVehicleCustomer.SelectedItem

Dim vehicleName As String = txtVehicleName.Text

addVehicle(currCustomerName, vehicleName)

DBConn.Close()

refreshDgvs()

refreshVehicles()

MessageBox.Show("Added Vehicle Successfully")

End If

End Sub

'------------------------------------------------------------

'- Subprogram Name: btnAddAppointment\_Click -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine attempts to add new new appointment to DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- sender – Identifies which particular control raised the –

'- click event -

'- e – Holds the EventArgs object sent to the routine -

'------------------------------------------------------------

'- Local Variable Dictionary -

'- currCustomerName - String containing customer name

'- currVehicleName - String containing vehicle name

'- serviceName - String containing service name

'- serviceTimeRequired - double containing time required

'------------------------------------------------------------

Private Sub btnAddAppointment\_Click(sender As Object, e As EventArgs) Handles btnAddAppointment.Click

If IsNothing(cmbAptCustomer.SelectedItem) Then

MessageBox.Show("Please select a customer.")

ElseIf IsNothing(cmbAptService.SelectedItem) Then

MessageBox.Show("Please select a service.")

ElseIf IsNothing(cmbAptVehicle.SelectedItem) Then

MessageBox.Show("Please select a vehicle.")

Else

DBConn = New SqlConnection(strCONNECTION)

DBConn.Open()

Dim currCustomerName As String

Dim currVehicleName As String

Dim serviceName As String = ""

Dim serviceTimeRequired As Double

'Find ID of Customer

strSQLCmd = "Select \* From Customers WHERE Name = '" & cmbAptCustomer.SelectedItem & "'"

DBAdaptCustomers = New SqlDataAdapter(strSQLCmd, strCONNECTION)

dsCustomers.Clear()

DBAdaptCustomers.Fill(dsCustomers, "Customers")

currCustomerName = dsCustomers.Tables(0).Rows(0).Item("Name").ToString

'Find ID of Vehicle

strSQLCmd = "Select \* From Vehicles WHERE Vehicle\_Description = '" & cmbAptVehicle.SelectedItem & "'"

DBAdaptVehicles = New SqlDataAdapter(strSQLCmd, strCONNECTION)

dsVehicles.Clear()

DBAdaptVehicles.Fill(dsVehicles, "Vehicles")

currVehicleName = dsVehicles.Tables(0).Rows(0).Item("Vehicle\_Description").ToString

'Find Service ID and time required for service

For index As Integer = 0 To (dsServices.Tables(0).Rows.Count - 1)

If cmbAptService.SelectedItem = dsServices.Tables(0).Rows(index).Item("ServiceName").ToString Then

serviceName = dsServices.Tables(0).Rows(index).Item("ServiceName").ToString

serviceTimeRequired = CDbl(dsServices.Tables(0).Rows(index).Item("TimeRequired"))

Exit For

End If

Next

addAppointment(currCustomerName, currVehicleName, serviceName, serviceTimeRequired)

DBConn.Close()

refreshDgvs()

refreshWeeks()

MessageBox.Show("Added New Appointment")

End If

End Sub

'------------------------------------------------------------

'- Subprogram Name: btnViewSchedule\_Click -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine shows weekly schedule for employee.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- sender – Identifies which particular control raised the –

'- click event -

'- e – Holds the EventArgs object sent to the routine -

'------------------------------------------------------------

'- Local Variable Dictionary -

'- name - String containing employee name

'- rate - double containing employee pay rate

'- hoursWorked - double containing weekly hours worked

'- estPay - double containing estimated pay for employee

'- week - integer containing week number

'------------------------------------------------------------

Private Sub btnViewSchedule\_Click(sender As Object, e As EventArgs) Handles btnViewSchedule.Click

If IsNothing(cmbSdlMechanic.SelectedItem) Then

MessageBox.Show("Please select a mechanic.")

ElseIf IsNothing(cmbSdlWeek.SelectedItem) Then

MessageBox.Show("Please select a week.")

Else

DBConn = New SqlConnection(strCONNECTION)

DBConn.Open()

strSQLCmd = "SELECT \*

FROM Schedule

WHERE Mechanic = '" & cmbSdlMechanic.SelectedItem & "' " &

"AND Week = '" & cmbSdlWeek.SelectedItem & "'" &

"ORDER BY CASE Appointment\_Day

WHEN 'Monday' THEN '1'

WHEN 'Tuesday' THEN '2'

WHEN 'Wednesday' THEN '3'

WHEN 'Thursday' THEN '4'

WHEN 'Friday' THEN '5'

END"

DBAdaptSchedule = New SqlDataAdapter(strSQLCmd, strCONNECTION)

dsSchedule.Clear()

DBAdaptSchedule.Fill(dsSchedule, "Schedule")

'Refresh the DataGridView showing the schedule so that it's accurate

frmEmpSchedule.dgvSchedule.DataSource = dsSchedule.Tables("Schedule")

frmEmpSchedule.dgvSchedule.Columns(0).Visible = False

frmEmpSchedule.dgvSchedule.Columns(1).Visible = False

frmEmpSchedule.dgvSchedule.Columns(2).Visible = False

frmEmpSchedule.dgvSchedule.Refresh()

Dim name As String = ""

Dim rate As Double = 0

Dim hoursWorked As Double = 0

Dim estPay As Double = 0

Dim week As Integer = CInt(cmbSdlWeek.SelectedItem)

If arrMechanics(0).getName = cmbSdlMechanic.SelectedItem Then

name = arrMechanics(0).getName

rate = arrMechanics(0).getRate

hoursWorked = arrMechanics(0).getHoursWorked(week - 1)

estPay = CDbl(rate) \* hoursWorked

ElseIf arrMechanics(1).getName = cmbSdlMechanic.SelectedItem Then

name = arrMechanics(1).getName

rate = arrMechanics(1).getRate

hoursWorked = arrMechanics(1).getHoursWorked(week - 1)

estPay = CDbl(rate) \* hoursWorked

End If

frmEmpSchedule.lblMechanic.Text = "Week " & cmbSdlWeek.SelectedItem & " Schedule For " & name

frmEmpSchedule.lblRate.Text = rate.ToString("$0.00")

frmEmpSchedule.lblHours.Text = hoursWorked.ToString

frmEmpSchedule.lblPay.Text = estPay.ToString("$0.00")

frmEmpSchedule.ShowDialog()

DBConn.Close()

End If

End Sub

'------------------------------------------------------------

'- Subprogram Name: addCustomer -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine adds new customer to DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- customerName - name of new customer

'------------------------------------------------------------

'- Local Variable Dictionary -

'- cmd - SQL Command for executing query

'- result - result of SQL Query

'------------------------------------------------------------

Public Sub addCustomer(ByVal customerName As String)

Dim cmd As New SqlCommand("Select Name From Customers WHERE Name = '" & customerName & "'", DBConn)

Dim result = cmd.ExecuteScalar()

If result = "" Then

'Use SQL to insert a new row into Customers

DBCmd.CommandText = "INSERT INTO Customers (TUID, Name) VALUES ('" &

customerID.ToString & "','" & customerName & "')"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery() 'Since it's a non-SELECT statement

'Add Customer to combo box

cmbVehicleCustomer.Items.Add(customerName)

cmbAptCustomer.Items.Add(customerName)

customerID += 1

Else 'Customer exists

MessageBox.Show("Customer " & customerName & " already exists in system.")

End If

End Sub

'------------------------------------------------------------

'- Subprogram Name: addVehicle -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine adds new vehicle to DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- customerName - name of customer

'- vehicleName - name of new vehicle

'------------------------------------------------------------

'- Local Variable Dictionary -

'- customerID - string containing customer id

'- cmd - SQL Command for executing query

'- result - result of SQL Query

'------------------------------------------------------------

Public Sub addVehicle(ByVal customerName As String, ByVal vehicleName As String)

'Find ID of Customer

strSQLCmd = "Select \* From Customers WHERE Name = '" & customerName & "'"

DBAdaptCustomers = New SqlDataAdapter(strSQLCmd, strCONNECTION)

dsCustomers.Clear()

DBAdaptCustomers.Fill(dsCustomers, "Customers")

Dim customerID As String = dsCustomers.Tables(0).Rows(0).Item("TUID").ToString

'Check if this vehicle already exists

Dim cmd As New SqlCommand("Select TUID From Vehicles WHERE Customer\_TUID = '" & customerID &

"' AND Vehicle\_Description = '" & vehicleName & "'", DBConn)

Dim result = cmd.ExecuteScalar()

If result = "" Then

'Use SQL to insert a new row into Registration

DBCmd.CommandText = "INSERT INTO Vehicles (TUID, Customer\_TUID, Vehicle\_Description) VALUES ('" &

vehicleID.ToString & "','" & customerID & "', '" & vehicleName & "')"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery() 'Since it's a non-SELECT statement

'Add Customer to combo box

vehicleID += 1

Else

MessageBox.Show("Vehicle " & vehicleName & " belonging to " & customerName &

" already exists in system.")

End If

End Sub

'------------------------------------------------------------

'- Subprogram Name: addAppointment -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine adds new appointment to DB.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- currCustomerName - name of customer

'- currVehicleName - name of new vehicle

'- serviceName - name of service

'- serviceTimeRequired - time for service

'------------------------------------------------------------

'- Local Variable Dictionary -

'- cmd - SQL Command for executing query

'- result - result of SQL Query

'- mechanicName - name of mechanic

'- week - week number

'- day - week day

'- startTime - appointment start time

'- endTime - appointment end time

'- arrDayTimes - array containing week, day and times

'------------------------------------------------------------

Public Sub addAppointment(ByVal currCustomerName As String,

ByVal currVehicleName As String,

ByVal serviceName As String,

ByVal serviceTimeRequired As Double)

'Check if this appointment already exists

Dim cmd As New SqlCommand("Select TUID From Schedule WHERE Customer = '" & currCustomerName &

"' AND Vehicle = '" & currVehicleName &

"' And Service = '" & serviceName & "'", DBConn)

Dim result = cmd.ExecuteScalar()

If result = "" Then

Dim mechanicName As String = ""

Dim week As String

Dim day As String

Dim startTime As String

Dim endTime As String

Dim arrDayTimes(3) As String

Dim arrMech1DayTime() As Double = arrMechanics(0).getAvailableDayTimeWeek(serviceTimeRequired)

Dim arrMech2DayTime() As Double = arrMechanics(1).getAvailableDayTimeWeek(serviceTimeRequired)

'Find Mechanic ID, Available Day and Time for both mechanics

'Compare weeks

If arrMech1DayTime(2) = arrMech2DayTime(2) Then

'Compare Days

If arrMech1DayTime(0) < arrMech2DayTime(0) Then

mechanicName = dsMechanics.Tables(0).Rows(0).Item("Name")

arrDayTimes = arrMechanics(0).addToSchedule(serviceTimeRequired)

ElseIf arrMech1DayTime(0) > arrMech2DayTime(0) Then

mechanicName = dsMechanics.Tables(0).Rows(1).Item("Name")

arrDayTimes = arrMechanics(1).addToSchedule(serviceTimeRequired)

ElseIf arrMech1DayTime(0) = arrMech2DayTime(0) Then

'Compare CurrentTimes

If arrMech1DayTime(1) = arrMech2DayTime(1) Then

mechanicName = dsMechanics.Tables(0).Rows(0).Item("Name")

arrDayTimes = arrMechanics(0).addToSchedule(serviceTimeRequired)

ElseIf arrMech1DayTime(1) < arrMech2DayTime(1) Then

mechanicName = dsMechanics.Tables(0).Rows(0).Item("Name")

arrDayTimes = arrMechanics(0).addToSchedule(serviceTimeRequired)

Else

mechanicName = dsMechanics.Tables(0).Rows(1).Item("Name")

arrDayTimes = arrMechanics(1).addToSchedule(serviceTimeRequired)

End If

End If

'Check which mechanic has Earlier week available:

ElseIf arrMech1DayTime(2) < arrMech2DayTime(2) Then

mechanicName = dsMechanics.Tables(0).Rows(0).Item("Name")

arrDayTimes = arrMechanics(0).addToSchedule(serviceTimeRequired)

Else

mechanicName = dsMechanics.Tables(0).Rows(1).Item("Name")

arrDayTimes = arrMechanics(1).addToSchedule(serviceTimeRequired)

End If

day = arrDayTimes(0)

startTime = arrDayTimes(1)

endTime = arrDayTimes(2)

week = arrDayTimes(3)

'Use SQL to insert a new row into Registration

DBCmd.CommandText = "INSERT INTO Schedule (TUID, Week, Mechanic, Customer,

Vehicle, Service, Appointment\_Day,

Start\_Time, End\_Time) VALUES ('" & scheduleID.ToString &

"','" & week & "', '" & mechanicName & "', '" & currCustomerName & "', '" &

currVehicleName & "', '" & serviceName & "', '" &

day & "', '" & startTime & "', '" & endTime & "')"

DBCmd.Connection = DBConn

DBCmd.ExecuteNonQuery() 'Since it's a non-SELECT statement

scheduleID += 1

Else

MessageBox.Show(serviceName & " Service is already scheduled for Customer " &

currCustomerName & " on Vehicle " & currVehicleName)

End If

End Sub

'------------------------------------------------------------

'- Subprogram Name: cmbMechanic\_SelectedIndexChanged -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine calls refreshWeeks subroutine.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- sender – Identifies which particular control raised the –

'- click event -

'- e – Holds the EventArgs object sent to the routine -

'------------------------------------------------------------

'- Local Variable Dictionary -

'- (none)

'------------------------------------------------------------

Private Sub cmbMechanic\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles cmbSdlMechanic.SelectedIndexChanged

refreshWeeks()

End Sub

'------------------------------------------------------------

'- Subprogram Name: cmbAptCustomer\_SelectedIndexChanged -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine calls refreshVehicles subroutine.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- sender – Identifies which particular control raised the –

'- click event -

'- e – Holds the EventArgs object sent to the routine -

'------------------------------------------------------------

'- Local Variable Dictionary -

'- (none)

'------------------------------------------------------------

Private Sub cmbAptCustomer\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles cmbAptCustomer.SelectedIndexChanged

refreshVehicles()

End Sub

'------------------------------------------------------------

'- Subprogram Name: refreshWeeks -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine fills weeks combo box

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (none)

'------------------------------------------------------------

'- Local Variable Dictionary -

'- cmd - SQL Command string

'- result - SQL Command result

'------------------------------------------------------------

Public Sub refreshWeeks()

DBConn = New SqlConnection(strCONNECTION)

DBConn.Open()

Dim cmd As New SqlCommand("Select MAX(Week) From Schedule WHERE Mechanic = '" & cmbSdlMechanic.SelectedItem & "'", DBConn)

cmbSdlWeek.Items.Clear()

Dim result = cmd.ExecuteScalar()

If Not TypeName(result) = "DBNull" Then

'Add week numbers to combo box

For i As Integer = 1 To CInt(result)

cmbSdlWeek.Items.Add(i)

Next

End If

DBConn.Close()

End Sub

'------------------------------------------------------------

'- Subprogram Name: refreshDgvs -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine refreshes datagrid views

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (none)

'------------------------------------------------------------

'- Local Variable Dictionary -

'- (none)

'------------------------------------------------------------

Public Sub refreshDgvs()

'Refresh dataset

strSQLCmd = "Select \* From Customers"

DBAdaptCustomers = New SqlDataAdapter(strSQLCmd, strCONNECTION)

dsCustomers.Clear()

DBAdaptCustomers.Fill(dsCustomers, "Customers")

strSQLCmd = "Select \* From Vehicles"

DBAdaptCustomers = New SqlDataAdapter(strSQLCmd, strCONNECTION)

dsVehicles.Clear()

DBAdaptCustomers.Fill(dsVehicles, "Vehicles")

strSQLCmd = "Select \* From Schedule"

DBAdaptSchedule = New SqlDataAdapter(strSQLCmd, strCONNECTION)

dsSchedule.Clear()

DBAdaptSchedule.Fill(dsSchedule, "Schedule")

dgvSchedule.DataSource = dsSchedule.Tables("Schedule")

dgvSchedule.Refresh()

End Sub

'------------------------------------------------------------

'- Subprogram Name: refreshVehicles -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine fills vehicles combo box

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (none)

'------------------------------------------------------------

'- Local Variable Dictionary -

'- cmd1 - SQL Command string

'- cmd2 - SQL Command string

'- cID - SQL Command execution

'------------------------------------------------------------

Private Sub refreshVehicles()

'Clear

cmbAptVehicle.Items.Clear()

'Retrieve customer ID

DBConn = New SqlConnection(strCONNECTION)

DBConn.Open()

Dim cmd1 As New SqlCommand("Select TUID From Customers WHERE Name = '" & cmbAptCustomer.SelectedItem & "'", DBConn)

Dim cID = cmd1.ExecuteScalar()

Dim cmd2 As New SqlCommand("Select Vehicle\_Description From Vehicles WHERE Customer\_TUID = '" & cID & "'", DBConn)

Dim reader As SqlDataReader = cmd2.ExecuteReader

While reader.Read()

cmbAptVehicle.Items.Add(reader.Item(0))

End While

reader.Close()

DBConn.Close()

End Sub

'------------------------------------------------------------

'- Subprogram Name: txtCustomerName\_KeyPress -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine restricts keypresses in customer TB

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- sender - object sending

'- e - KeyPressEventArgs argument

'------------------------------------------------------------

'- Local Variable Dictionary -

'- allowedChars - string containing allowed characters

'------------------------------------------------------------

Private Sub txtCustomerName\_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtCustomerName.KeyPress

If Not (Asc(e.KeyChar) = 8) Then

Dim allowedChars As String = "abcdefghijklmnopqrstuvwxyz1234567890-' "

If Not allowedChars.Contains(e.KeyChar.ToString.ToLower) Then

e.KeyChar = ChrW(0)

e.Handled = True

End If

End If

End Sub

'------------------------------------------------------------

'- Subprogram Name: txtCustomerName\_KeyPress -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine restricts keypresses in vehicle TB

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- sender - object sending

'- e - KeyPressEventArgs argument

'------------------------------------------------------------

'- Local Variable Dictionary -

'- allowedChars - string containing allowed characters

'------------------------------------------------------------

Private Sub txtVehicleName\_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtVehicleName.KeyPress

If Not (Asc(e.KeyChar) = 8) Then

Dim allowedChars As String = "abcdefghijklmnopqrstuvwxyz1234567890-' "

If Not allowedChars.Contains(e.KeyChar.ToString.ToLower) Then

e.KeyChar = ChrW(0)

e.Handled = True

End If

End If

End Sub

End Class

**frmEmpSchedule.vb**

'------------------------------------------------------------

'- File Name : frmEmpSchedule.vb -

'- Part of Project: Assign4 -

'------------------------------------------------------------

'- Written By: Tajbid Hasib -

'- Written On: 10/15/2019 -

'------------------------------------------------------------

'- File Purpose: -

'- This is schedule form for Turbo Auto Service

'------------------------------------------------------------

'- Global Variable Dictionary: -

'- (none)

'------------------------------------------------------------

'------------------------------------------------------------

Public Class frmEmpSchedule

Const fileName As String = "frmEmpSchedule.vb"

Private Sub btnPrint\_Click(sender As Object, e As EventArgs) Handles btnPrint.Click

PrintForm1.PrintAction = Printing.PrintAction.PrintToPreview

PrintForm1.Print()

End Sub

End Class

**Mechanic.vb**

'------------------------------------------------------------

'- File Name : Mechanic.vb -

'- Part of Project: Assign4 -

'------------------------------------------------------------

'- Written By: Tajbid Hasib -

'- Written On: 10/15/2019 -

'------------------------------------------------------------

'- File Purpose: -

'- This is the mechanic class that holds information about

'- each mechanic

'------------------------------------------------------------

'- Global Variable Dictionary: -

'- (none)

'------------------------------------------------------------

Public Class Mechanic

Const fileName As String = "Mechanic.vb"

Private strID As String

Private strName As String

Private intRate As Integer

Private intWeekNumber As Integer

Private dblHoursWorked(51) As Double

Private dayMon(51) As Day

Private dayTue(51) As Day

Private dayWed(51) As Day

Private dayThu(51) As Day

Private dayFri(51) As Day

'------------------------------------------------------------

'- Subprogram Name: New -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine creates new mechanic object.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (none)

'------------------------------------------------------------

Public Sub New()

setID("")

setName("")

setRate(0)

intWeekNumber = 0

setHoursWorked(intWeekNumber, 0)

addWeek(intWeekNumber)

End Sub

'------------------------------------------------------------

'- Subprogram Name: New -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine creates new mechanic object.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- newID - ID of mechanic

'- newName - name of mechanic

'- newRate - hourly rate of mechanic

'------------------------------------------------------------

Public Sub New(ByVal newID As String, ByVal newName As String, ByVal newRate As Integer)

setID(newID)

setName(newName)

setRate(newRate)

intWeekNumber = 0

setHoursWorked(intWeekNumber, 0)

addWeek(intWeekNumber)

End Sub

'------------------------------------------------------------

'- Subprogram Name: addWeek -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine adds days to a week.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- inputWeekNumber - week number

'------------------------------------------------------------

Private Sub addWeek(inputWeekNumber)

dayMon(inputWeekNumber) = New Day("Monday", inputWeekNumber)

dayTue(inputWeekNumber) = New Day("Tuesday", inputWeekNumber)

dayWed(inputWeekNumber) = New Day("Wednesday", inputWeekNumber)

dayThu(inputWeekNumber) = New Day("Thursday", inputWeekNumber)

dayFri(inputWeekNumber) = New Day("Friday", inputWeekNumber)

End Sub

'------------------------------------------------------------

'- Function Name: getAvailableDayTimeWeek -

'------------------------------------------------------------

'- Function Returns: -

'- -

'- This function returns available day and time.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- inputHours - hours for job

'------------------------------------------------------------

Public Function getAvailableDayTimeWeek(ByVal inputHours As Double)

Dim arrDayTime(2) As Double

For i As Integer = 0 To intWeekNumber

If dayMon(i).getHoursRemaining >= inputHours Then

arrDayTime(0) = 1

arrDayTime(1) = dayMon(i).getCurrentTime()

arrDayTime(2) = i

Exit For

ElseIf dayTue(i).getHoursRemaining >= inputHours Then

arrDayTime(0) = 2

arrDayTime(1) = dayTue(i).getCurrentTime()

arrDayTime(2) = i

Exit For

ElseIf dayWed(i).getHoursRemaining >= inputHours Then

arrDayTime(0) = 3

arrDayTime(1) = dayWed(i).getCurrentTime()

arrDayTime(2) = i

Exit For

ElseIf dayThu(i).getHoursRemaining >= inputHours Then

arrDayTime(0) = 4

arrDayTime(1) = dayThu(i).getCurrentTime()

arrDayTime(2) = i

Exit For

ElseIf dayFri(i).getHoursRemaining >= inputHours Then

arrDayTime(0) = 5

arrDayTime(1) = dayFri(i).getCurrentTime()

arrDayTime(2) = i

Exit For

Else

'NO Available time this week, check if the next week exist, otherwise create one

If dayMon(i + 1) Is Nothing Then

intWeekNumber += 1

addWeek(intWeekNumber)

'Since new week is created, first available day is 1

arrDayTime(0) = 1

arrDayTime(1) = dayMon(intWeekNumber).getCurrentTime

arrDayTime(2) = intWeekNumber

End If

'next week exists, so do nothing

End If

Next

Return arrDayTime

End Function

'------------------------------------------------------------

'- Function Name: addToSchedule -

'------------------------------------------------------------

'- Function Returns: -

'- -

'- This function returns array of day, week, times.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- inputHours - hours for job

'------------------------------------------------------------

Public Function addToSchedule(ByVal inputHours As Double)

Dim arrDayTimes(3) As String

For i As Integer = 0 To intWeekNumber

If dayMon(i).getHoursRemaining >= inputHours Then

arrDayTimes = dayMon(i).addJob(inputHours)

dblHoursWorked(i) += inputHours

Exit For

ElseIf dayTue(i).getHoursRemaining >= inputHours Then

arrDayTimes = dayTue(i).addJob(inputHours)

dblHoursWorked(i) += inputHours

Exit For

ElseIf dayWed(i).getHoursRemaining >= inputHours Then

arrDayTimes = dayWed(i).addJob(inputHours)

dblHoursWorked(i) += inputHours

Exit For

ElseIf dayThu(i).getHoursRemaining >= inputHours Then

arrDayTimes = dayThu(i).addJob(inputHours)

dblHoursWorked(i) += inputHours

Exit For

ElseIf dayFri(i).getHoursRemaining >= inputHours Then

arrDayTimes = dayFri(i).addJob(inputHours)

dblHoursWorked(i) += inputHours

Exit For

End If

Next

Return arrDayTimes

End Function

'------------------------------------------------------------

'- SubProgram Group: Setters and getters-

'------------------------------------------------------------

'- Purpose: -

'- -

'- Set attributes

'- return attributes

'------------------------------------------------------------

Public Sub setID(ByVal inputID As String)

strID = inputID

End Sub

Public Function getID()

Return strID

End Function

Public Sub setName(ByVal inputName As String)

strName = inputName

End Sub

Public Function getName()

Return strName

End Function

Public Sub setRate(ByVal inputRate As Integer)

intRate = inputRate

End Sub

Public Function getRate()

Return intRate

End Function

Public Sub setHoursWorked(ByVal weekNumber As Integer, ByVal inputHoursWorked As Double)

dblHoursWorked(weekNumber) = inputHoursWorked

End Sub

Public Function getHoursWorked(ByVal weekNumber As Integer)

Return dblHoursWorked(weekNumber)

End Function

End Class

**Day.vb**

'------------------------------------------------------------

'- File Name : Day.vb -

'- Part of Project: Assign4 -

'------------------------------------------------------------

'- Written By: Tajbid Hasib -

'- Written On: 10/15/2019 -

'------------------------------------------------------------

'- File Purpose: -

'- This is the Day class that holds information about

'- each mechanic

'------------------------------------------------------------

'- Global Variable Dictionary: -

'- (none)

'------------------------------------------------------------

Public Class Day

Const fileName As String = "Day.vb"

Private strDayName As String

Private dblHoursRemaining As Double

Private dblCurrentTime As Double

Private dictTime As Dictionary(Of Double, String)

Private weekNumber As Integer

Public Sub New()

setDayName("")

setHoursReamining(0)

dblCurrentTime = 0 'Clock Time

initializeDictionary()

weekNumber = 0

End Sub

Public Sub New(ByVal newDayName As String, ByVal newWeekNumber As Integer)

strDayName = newDayName

setHoursReamining(8)

dblCurrentTime = 8 'Clock Time

initializeDictionary()

weekNumber = newWeekNumber

End Sub

'------------------------------------------------------------

'- Function Name: addJob -

'------------------------------------------------------------

'- Function Returns: -

'- -

'- This function adds job to day.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- inputHours - hours for job

'------------------------------------------------------------

Public Function addJob(ByVal inputHours As Double)

dblHoursRemaining = dblHoursRemaining - inputHours

Dim strTimes(3) As String

'Day number

strTimes(0) = strDayName

'start time

strTimes(1) = dictTime.Item(dblCurrentTime)

dblCurrentTime += inputHours

'end time

strTimes(2) = dictTime.Item(dblCurrentTime)

'IF end time is 1 PM, make it 12PM

If strTimes(2) = "01.00PM" Then

strTimes(2) = "12.00PM"

End If

strTimes(3) = weekNumber + 1

Return strTimes

End Function

'------------------------------------------------------------

'- Subprogram Name: initializeDictionary -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- This subroutine initalizes dictionary.

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (none)

'------------------------------------------------------------

Public Sub initializeDictionary()

dictTime = New Dictionary(Of Double, String)

' Add entries to dictionary.

dictTime.Add(8.0, "08.00AM")

dictTime.Add(8.5, "08.30AM")

dictTime.Add(9.0, "09.00AM")

dictTime.Add(9.5, "09.30AM")

dictTime.Add(10.0, "10.00AM")

dictTime.Add(10.5, "10.30AM")

dictTime.Add(11.0, "11.00AM")

dictTime.Add(11.5, "11.30AM")

dictTime.Add(12.0, "01.00PM")

dictTime.Add(12.5, "01.30PM")

dictTime.Add(13.0, "02.00PM")

dictTime.Add(13.5, "02.30PM")

dictTime.Add(14.0, "03.00PM")

dictTime.Add(14.5, "03.30PM")

dictTime.Add(15.0, "04.00PM")

dictTime.Add(15.5, "04.30PM")

dictTime.Add(16.0, "05.00PM")

End Sub

'------------------------------------------------------------

'- SubProgram Group: Setters and getters-

'------------------------------------------------------------

'- Purpose: -

'- -

'- Set attributes

'- return attributes

'------------------------------------------------------------

Public Sub setDayName(ByVal inputDayName As String)

strDayName = inputDayName

End Sub

Public Function getDayName()

Return strDayName

End Function

Public Sub setHoursReamining(ByVal inputHoursRemaining As Double)

dblHoursRemaining = inputHoursRemaining

End Sub

Public Function getHoursRemaining()

Return dblHoursRemaining

End Function

Public Sub setCurrentTime(ByVal inputCurrentTime As Double)

dblCurrentTime = inputCurrentTime

End Sub

Public Function getCurrentTime()

Return dblCurrentTime

End Function

End Class

**Screenshots:**





