*Programming Assignment Sheet*

|  |  |
| --- | --- |
| To: | Troy Tuckett |
| From: | Elbio Iseas |
| Class: | POS/409 |
| Date: | 10/14/14 |
| Re: | Individual Assignment for Week 5 (Continuation Week Four Assignment) |

|  |
| --- |
| Design: |
| This application uses four classes to achieve the requirements of the program. A base class named: Employee, a derived class named MisDept, and two helper classes: frmMain, and frmEmpList. The Employee base class contains five attributes: ID (employee ID number 999999), last name (40 characters), first name (40 characters), DOB (Date of Birth), and the DOH (Date of Hire). The derived class MisDept (Management Information System Department) has three attributes: position in the company (alphanumeric), work area in which the employee works (alphanumeric), and the pay grade (valid range 1 to 99).  Also this derived class inherits the five attributes from its parent class Employee. The data entry screen has help to enter the position (with a combo box preloaded on the form), the work area (also preloaded on the form), and the date of birth and date of hire have date pickers.  Therefore an employee of the M.I.S. Department uses the data from the Employee class. There are three buttons on the New Employees Screen:” Submit Employee”, “Employee List”, and “Exit”.  The Submit Employee button will create a new employee with the data entered on the first screen. After the end-user entered a few new employees, he/she will be able to click on the “Employee List” button to see a list of the employees entered on dedicated screen for this purpose.  To exit the program there are two buttons: one on the bottom right corner, and the other on the top right corner like all Windows applications. They close the application.  The use of an array list is used to keep in memory the information of new employees. The end-user can enter some employees, go to see the list of employees on another screen, and come back to keep entering more new employees. Since the information is kept in memory, every time the user returns to the List of Employees the information of the first new employees will be displayed plus the information of the next group of employees entered the second time.  A data grid view control is used to display the information of the new employees. At the beginning of this document is a description of classes, attributes, constructors, and methods used to run the application, and obtain the desired output on screen.  In order to communicate between classes there are methods that send messages and return data structures in some cases.  Some useful features built-in on the form were used to help the data entry. Some of these features include: accepting uppercase regardless of what the user types; the amount of characters allowed for specific text boxes is achieved by using the properties of the controls.  Entering data out of range will trigger corresponding error messages.    A couple of methods new for this assignment:  btnLoadDatabase\_Click() and btnSaveToDatabase() will take care of reading the Access Database into an ArrayList type MisDept. This object inherits five attributes from its parent class Employee, and then it is assigned the values from the database “Employees”, table “Employees”. The connectivity to the database is through the use of the provider “Microsoft ACE OLEDB 12.0”, a dataset, a dataadapter, a datasource, a connection string, and a command string. These two methods obtain the current path into a variable path that it is used for the connection string.  The second method bntSaveToDatabase() uses the same commands to open the database, but at the time of saving the information from memory, it uses a command to zap the table of the database “dataSet.Clear();” , but first reads all the records from the table to the dataSet. Right after it creates a DataRow to load the information of each object, then it executes a command to update the dataset with the use of the dataadapter.  The method EmpList it is supposed to do exactly the same as in Week Four assignment but for some reason, I believe in the DataGridView design of the columns is not showing the information in the ArrayList EmployeeAL, but with the debugger we can see the information is there from the database, and the additional information of new employees entered with the form. |
| Source Program(s) : |
| // ====================================================================================================  // Form1.cs  // ====================================================================================================  using System;  using System.Collections;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Data.OleDb;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  using System.IO;  namespace POS409DatabaseConnection  {  public partial class frmMain : Form  {  public static ArrayList EmployeeAL = new ArrayList();  public frmMain()  {  InitializeComponent();  }  private void frmMain\_Load(object sender, EventArgs e)  {  this.txtEmployeeID.Focus();  }  private void btnExit\_Click(object sender, EventArgs e)  {  Close();  }    private void txtEmployeeID\_KeyPress(object sender, KeyPressEventArgs e)  {  // https://www.youtube.com/watch?v=EcZQghn88vE  char kp = e.KeyChar;  if (!Char.IsDigit(kp) && kp != 8)  {  e.Handled = true;  }  }  private void txtEmployeeID\_Validating(object sender, CancelEventArgs e)  {  double empID = Double.Parse(txtEmployeeID.Text);  try  {  if (empID >= 100000 && empID <= 999999)  {  // The Employee ID number is in the valid range.  e.Cancel = false;  }  else  {  // The Employee ID number is not in the valid range.  MessageBox.Show("The Employee ID needs to be in the range 100000 to 999999!");  e.Cancel = true;  this.txtEmployeeID.Text = "";  }  }  catch (Exception ex)  {  MessageBox.Show("The Employee ID needs to be in the range 100000 to 999999!");  e.Cancel = true;  this.txtEmployeeID.Text = "";  }  }  private void txtPayGrade\_KeyPress(object sender, KeyPressEventArgs e)  {  // https://www.youtube.com/watch?v=EcZQghn88vE  char kp = e.KeyChar;  if (!Char.IsDigit(kp) && kp != 8)  {  e.Handled = true;  }  }  private void txtPayGrade\_Validating(object sender, CancelEventArgs e)  {  int payGrade;  Int32.TryParse((string)txtPayGrade.Text, out payGrade);  try  {  if (payGrade >= 1 && payGrade <= 99)  {  // The Pay Grade number is in the valid range.  e.Cancel = false;  }  else  {  // The Pay Grade number is not in the valid range.  MessageBox.Show("The Pay Grade needs to be in the range 1 to 99!");  e.Cancel = true;  this.txtPayGrade.Text = "";  }  }  catch (Exception ex)  {  MessageBox.Show("The Pay Grade needs to be in the range 1 to 99!");  e.Cancel = true;  this.txtPayGrade.Text = "";  }  }  private void cBoxPosition\_Leave(object sender, EventArgs e)  {  int index = (Int32)(cBoxPosition.SelectedIndex);  if (index == -1 || cBoxPosition.Text == "")  {  cBoxPosition.Focus();  }  }  private void cBoxWorkArea\_Leave(object sender, EventArgs e)  {  int index = (Int32)(cBoxWorkArea.SelectedIndex);  if (index == -1 || cBoxWorkArea.Text == "")  {  cBoxWorkArea.Focus();  }  }  private void btnSubmit\_Click(object sender, EventArgs e)  {  EmployeeAL = createNewEmployee(EmployeeAL);  }  private ArrayList createNewEmployee(ArrayList myArray)  {  ArrayList MisAL = new ArrayList();  MisAL = myArray;  MisDept newITemployee = new MisDept();  newITemployee.Id = Convert.ToDouble(this.txtEmployeeID.Text);  newITemployee.LastName = this.txtLastName.Text;  newITemployee.FirstName = this.txtFirstName.Text;  newITemployee.DoB = Convert.ToDateTime(this.dateTimePicker1.Text);  newITemployee.DoH = Convert.ToDateTime(this.dateTimePicker2.Text);  newITemployee.Position = this.cBoxPosition.Text;  newITemployee.WorkArea = this.cBoxWorkArea.Text;  newITemployee.PayGrade = Convert.ToInt32(this.txtPayGrade.Text);  MisAL.Add(newITemployee);  BlankTextBoxes();  return MisAL;  }  private void BlankTextBoxes()  {  txtEmployeeID.Text = "";  txtLastName.Text = "";  txtFirstName.Text = "";  dateTimePicker1.Text = "";  dateTimePicker2.Text = "";  cBoxPosition.Text = "";  cBoxWorkArea.Text = "";  txtPayGrade.Text = "";  }  private void btnLoadDatabase\_Click(object sender, EventArgs e)  {  // Subroutine idea from my POS/408 Week Four Individual Assignment  OleDbConnection connexion = new OleDbConnection();  string dbProvider;  string dbSource;  string path = Directory.GetCurrentDirectory();  string dbFile = "Employees.accdb";  string file = "Employees";  DataSet dataSet = new DataSet();  OleDbDataAdapter dataAdapter = new System.Data.OleDb.OleDbDataAdapter();  OleDbCommand sql = new OleDbCommand();  dbProvider = "PROVIDER=Microsoft.ACE.OLEDB.12.0;";  dbSource = "Data Source = " + path + "\\" + dbFile;  connexion.ConnectionString = dbProvider + dbSource;  connexion.Open();  sql.CommandText = "SELECT \* FROM Employees";  dataAdapter = new System.Data.OleDb.OleDbDataAdapter(sql.CommandText, connexion);  dataAdapter.Fill(dataSet, "Employees");  connexion.Close();  MisDept newITemployee = new MisDept();  EmployeeAL.Clear();  int totalRecs = dataSet.Tables[0].Rows.Count;  for ( int i = 0; i < totalRecs; i++ )  {  newITemployee.Id = Convert.ToDouble(dataSet.Tables[file].Rows[i][0]);  newITemployee.LastName = Convert.ToString(dataSet.Tables[file].Rows[i][1]);  newITemployee.FirstName = Convert.ToString(dataSet.Tables[file].Rows[i][2]);  newITemployee.DoB = Convert.ToDateTime(dataSet.Tables[file].Rows[i][3]);  newITemployee.DoH = Convert.ToDateTime(dataSet.Tables[file].Rows[i][4]);  newITemployee.Position = Convert.ToString(dataSet.Tables[file].Rows[i][5]);  newITemployee.WorkArea = Convert.ToString(dataSet.Tables[file].Rows[i][6]);  newITemployee.PayGrade = Convert.ToByte(dataSet.Tables[file].Rows[i][7]);  EmployeeAL.Add(newITemployee);  }  }  private void btnSaveToDatabase\_Click(object sender, EventArgs e)  {  // Subroutine idea from my POS/408 Week Four Individual Assignment  OleDbConnection connexion = new OleDbConnection();  string dbProvider;  string dbSource;  string path = Directory.GetCurrentDirectory();  string dbFile = "Employees.accdb";  string file = "Employees";  DataSet dataSet = new DataSet();  OleDbDataAdapter dataAdapter = new System.Data.OleDb.OleDbDataAdapter();  OleDbCommand sql = new OleDbCommand();  dbProvider = "PROVIDER=Microsoft.ACE.OLEDB.12.0";  dbSource = "Data Source = " + path + "\\" + dbFile;  connexion.ConnectionString = dbProvider + dbSource;  connexion.Open();  OleDbCommandBuilder commMaker= new OleDbCommandBuilder(dataAdapter);  dataAdapter.Fill(dataSet, file);  dataSet.Clear();  int totalRecs = EmployeeAL.Count;  //msdn.microsoft.com/en-us/library/5ycd1034.aspx  DataRow rec;  foreach (MisDept obj in frmMain.EmployeeAL)  {  rec = dataSet.Tables[file].NewRow();  rec["ID"] = Convert.ToDouble(obj.Id);  rec["LastName"] = Convert.ToString(obj.LastName);  rec["FirstName"] = Convert.ToString(obj.FirstName);  rec["DOB"] = Convert.ToDateTime(obj.DoB);  rec["DOH"] = Convert.ToDateTime(obj.DoH);  rec["Position"] = Convert.ToString(obj.Position);  rec["WorkArea"] = Convert.ToString(obj.WorkArea);  rec["PayGrade"] = Convert.ToByte(obj.PayGrade);  dataSet.Tables[file].Rows.Add(rec);    }  commMaker.GetUpdateCommand();  dataAdapter.Update(dataSet, file);  connexion.Close();  }  private void btnEmpList\_Click(object sender, EventArgs e)  {  int elementsCount = EmployeeAL.Count;  frmEmpList frmEL = new frmEmpList();  frmEL.ShowDialog(this);  }  }  }  // ====================================================================================================  // Form2.cs  // ====================================================================================================  using System;  using System.Collections;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  namespace POS409DatabaseConnection  {  public partial class frmEmpList : Form  {  public frmEmpList()  {  InitializeComponent();  }  private void frmEmpList\_Load(object sender, EventArgs e)  {    if (frmMain.EmployeeAL.Count > 0)  {  foreach (MisDept obj in frmMain.EmployeeAL)  {    // idea from stackoverflow.com  //http://stackoverflow.com/questions/6092463/how-can-i-manually-add-data-to-a-datagridview  DataGridViewRow line = new DataGridViewRow();  dataGridViewEmpList.Rows.Add((obj.Id).ToString(),  (obj.LastName).ToString(),  (obj.FirstName).ToString(),  (obj.DoB).ToString(),  (obj.Position).ToString(),  (obj.WorkArea).ToString(),  (obj.DoH).ToString(),  (obj.PayGrade).ToString());  }  dataGridViewEmpList.Show();  }  else  {  MessageBox.Show("There are no employees in the system.!");  }  }  private void btnMainMenu\_Click(object sender, EventArgs e)  {  Close();  }  }  }  // ====================================================================================================  // Employee.cs  // ====================================================================================================  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace POS409DatabaseConnection  {  class Employee  {  private double \_ID;  private string \_lastName;  private string \_firstName;  private DateTime \_DOB;  private DateTime \_DOH;  public Employee()  {  \_ID = 100000;  \_lastName = "";  \_firstName = "";  \_DOB = DateTime.Now;  \_DOH = DateTime.Now;  }  public double Id  {  get { return \_ID; }  set { \_ID = value; }  }  public string LastName  {  get { return \_lastName; }  set { \_lastName = value; }  }  public string FirstName  {  get { return \_firstName; }  set { \_firstName = value; }  }  public DateTime DoB  {  get { return \_DOB; }  set { \_DOB = value; }  }  public DateTime DoH  {  get { return \_DOH; }  set { \_DOH = value; }  }  }  }  // ====================================================================================================  // MisDept.cs  // ====================================================================================================  using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace POS409DatabaseConnection  {  class MisDept : Employee  {  private string \_position;  private string \_workArea;  private int \_payGrade;  public MisDept()  {  \_position = "";  \_workArea = "";  \_payGrade = 0;  }  public string Position  {  get { return \_position; }  set { \_position = value; }  }  public string WorkArea  {  get { return \_workArea; }  set { \_workArea = value; }  }  public int PayGrade  {  get { return \_payGrade; }  set { \_payGrade = value; }  }  }  } |
| Output Results: |
|  |
| **Testing:** |
| // ============================================================================  Description of testing   1. The program will read the data of the Access database Employees into a data structure in memory (an array of objects called “EmployeeAL”). PASS 2. The application will add the new employees’ data in the arraylist and then save the ArrayList EmployeeAL to the Access database. PASS 3. This application will display the list of in the database and entered with the data entry screen. FAIL. 4. Entering a new employee’s data:   Employee # : 543421  Last name : SIMPSON  First name : HOMERO  DOB : using the DateTime picker select 1985 March 28  DOH : using the DateTime picker select 1986 November 5  Position : using the combobox select PROGRAMMER III  Work Area : using the combobox select PROGRAMMING  Pay Grade : 23  PASS  . |
| Tested By  Elbio Iseas |