

*Programming Assignment Sheet*

|  |  |
| --- | --- |
| To: | Troy Tuckett |
| From: | Elbio Iseas |
| Class: | POS/409 |
| Date: | 10/07/14 |
| Re: | Individual Assignment for Week 4 |

|  |
| --- |
| 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. |
| Source Program(s) : |
| // ================================================================================================  // Form1.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 POS409Inheritance  {  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 btnEmpList\_Click(object sender, EventArgs e)  {  frmEmpList frmEL = new frmEmpList();  frmEL.Show();  }  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 = "";  }  }  }  // ================================================================================================  // 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 POS409Inheritance  {  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)  {  this.Close();  }    }  }    // =============================================================================================  // Employee.cs  // =============================================================================================  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace POS409Inheritance  {  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 POS409Inheritance  {  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 new employees from a data entry screen into a data structure in memory (an array of objects). PASS 2. The application will add the new employees’ data in the array. PASS 3. This C# OO application keeps a user-friendly graphic interface to achieve its purpose. PASS. 4. The program used the information on the second screen shot to test the data entry for results on the List of Employees in the second screen. PASS. |
| Tested By  Elbio Iseas |