## 

## PROGRAMMING TECHNOLOGIES

Assignment 2

*Prepared for*

*Mrs Radi Doncheva*

Angela Okonedo

April 24, 2012

****

## TABLE OF CONTENTS

## Introduction ---------------------------------------------------- 3

## User Guide ----------------------------------------------------- 4

## The Code commented ---------------------------------------- 7

## Testing --------------------------------------------------------- 13

## Conclusion ---------------------------------------------------- 20

## Appendix ------------------------------------------------------21

**INTRODUCTION**

As humans and machines do not speak the same language a common ground is needed – a programming language. Programming has evolved in time, moving from what is known as a “low level” language (syntax and data interpreted as the machine does) to a” higher level” language (syntax and data concepts interpreted as we humans do). From machine code to assembly language to codes that are independent from the machine (using compilers) to procedures and functions and finally to object oriented programs.

The following task was done using the Visual C# development environment using object-oriented programming concepts.

What exactly is object-oriented programming (OOP)? OOP is a programming language model, which unlike previous programming models that focused on manipulating data “logically”, it focuses on the manipulation of the data itself also known as “objects”. This might appear intuitive because as humans we are familiar with the concept of objects in our everyday life – chairs, pens, cars, people etc. , whose characteristics(properties) can be managed . The challenge though is getting this concept to fit in with programming logic.

Usually the programmer would start off with a clear idea of the objects needed in the program. Once this has been decided, the next thing to worry about will be how these different objects will work together in the program. Think of the single object as a representative of objects of its kind. What basic characteristics would it definitely have? What “actions” or methods would need to be performed on that object? Of course the properties and methods are chosen specifically to meet the needs of the program.

What happens then is a class of objects of that kind is created. The class will have methods and properties pertaining to objects of that class. So when the programmer wants to work with an object or use its methods, they will have to create an instance of the class first, or rather declare a representative of it.

The following assignment consists of creating a program that allows the user to enter information of employees, ensure that the information has been entered correctly and retrieving information about the lowest paid employees in the list entered.

**USER GUIDE**

**Menu Form:** The first form that comes up when the program is run.



BUTTON FUNCTION

 Opens the Registration Form

 Exits the Menu Form

**Employee Registration Form:** This is where details of the employees are entered.



BUTTON FUNCTION

 Shows how many employees have been entered.

**Note:** You must enter at least 10 employees!

 Saves the data entered on database. The textboxes are

cleared ready for new input.

 Opens Display Form window. Displays data entered.

 Opens Display Form window. Displays employees with lowest

Salary

 Where the employee’s details is entered. Type in

employee’s Id, Name, Surname or Salary as indicated.

**Note:** Make sure to type in the correct data!

**Display Form:** Displays employees’ information entered and the lowest salary workers.



BUTTON FUNCTION

Closes the display window

**THE CODE COMMENTED**

**Registration form code / Form1.cs**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* Angela Okonedo Course: COMP1381, Programming Technologies \*

\* Assignment 2 to be handed in by 24/04/12 \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace assignment2\_240412

{

public partial class RegistrationForm : Form

{

private Employee[] workers = new Employee[20]; //an array workers[] where each

//element of the array is an instance of Employee class

int count = 0; //counter for number of employees entered

int id; //employee details

string name;

string surname;

double salary;

public RegistrationForm()

{

InitializeComponent();

}

private void displayButton\_Click(object sender, EventArgs e)

{

DisplayForm dForm = new DisplayForm(); //creates instance of Display form;

if(count<10) //Checks atleast 10 employees have been entered, otherwise it

// gives an error message

{

MessageBox.Show("You must enter atleast 10 employees!");

}

else //if count is more than 10 or more than 10 employees have been entered

for (int i = 0; i < count; i++) //it takes the details of each employee

{

string workerString = workers[i].employeeToString() + "\r\n"; //and puts

// them in an array - workers[]

dForm.richTextBox2.AppendText(workerString); //the string of employee

// details is sent to the display form's richtextbox

dForm.Show(); //this shows the display form with all the details

}

}

private void registerButton\_Click(object sender, EventArgs e)

{

name = nameTextBox.Text; //gets the name entered in textbox

surname = surnameTextBox.Text; //gets the surname entered in textbox

//throws a format exception where the format or type is different from expected

// in this case an integer!

Try

{

id = Convert.ToInt32(idTextBox.Text);

}

catch (FormatException exceptionObject) //catches the exception showing an

//error message!

{

MessageBox.Show("Typing Error: re-enter a valid id number!");

count--; //count should increase only when the data inputted has been

//accepted

}

//throws a format exception where the format or type is different from expected

// in this case a double!

try

{

salary = Convert.ToDouble(salaryTextBox.Text);

}

catch (FormatException exceptionObject)

{

MessageBox.Show("Typing Error: re-enter a valid salary!");

count//count should increase only when the data inputted has been

//accepted

}

workers[count] = new Employee(id, name,surname, salary);

count++; //increases for next employee

label6.Text = Convert.ToString(count);//Shows the number of employees

//typed in so far

//text boxes are cleared for next input

idTextBox.Text = "";

nameTextBox.Text = "";

surnameTextBox.Text = "";

salaryTextBox.Text = "";

}

private void lowestSalaryButton\_Click(object sender, EventArgs e)

{

PrintLowestSalary(workers); //Calls Print Lowest salary method

}

//Method that receives the array of workers and prints the ones with the lowest salary

private void PrintLowestSalary(Employee[] workers)

{

DisplayForm dForm = new DisplayForm(); //instance of display form

dForm.richTextBox2.AppendText("Lowest salary workers:" + "\r\n" + "\r\n");

double minSalary = workers[0].Salary; //sets the the first element of the

//array as the default minimum salary

for (int i = 0; i < count;

{

if (workers[i].Salary < minSalary) //compares the minimum salary with

//other salaries

{

minSalary = workers[i].Salary; //saves the new minimum salary

}

}

//prints the employees with the lowest salary

for (int i = 0; i < count; i++) //checks every employee in the array

{

if (workers[i].Salary == minSalary) // if his salary is the same as the

//minimum salary

{

string workerString = workers[i].employeeToString() + "\r\n";

dForm.richTextBox2.AppendText(workerString);//it sends the string to

//the display form

dForm.Show(); //shows the display form

}

}

}

private void RegistrationForm\_Load(object sender, EventArgs e)

{

}

private void label6\_Click(object sender, EventArgs e)

{

}

}

}

**Employee class/ Employee.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace assignment2\_240412

{

class Employee

{

private int id; // The class properties: id,name,surname and salary declared

private string firstName;

private string lastName;

private double salary;

// first constructor

public Employee()

{

id = 0; //initializing an object instance with no details

firstName = "";

lastName = "";

salary = 0.0;

}

// second constructor //initializing an object instance with no salary

public Employee(int idValue, string firstNameVal, string lastNameVal)

{

id = idValue;

firstName = firstNameVal;

lastName = lastNameVal;

salary = 0.0;

}

// third constructor //initializing an object instance with all details

public Employee(int idValue, string firstNameVal,string lastNameVal, double salaryValue)

{

id = idValue;

firstName = firstNameVal;

lastName = lastNameVal;

salary = salaryValue;

}

public int idNum

{

get

{

return id;

}

set

{

id = value;

}

}

public string FirstName

{

get

{

return firstName;

}

set

{

firstName = value;

}

}

public string LastName

{

get

{

return lastName;

}

set

{

lastName = value;

}

}

public double Salary

{

get

{

return salary;

}

set

{

salary = value;

}

}

public string employeeToString() //Method that returns the employees details

//as a string

{

return (Convert.ToString(id) + " " + " " + firstName +" " + lastName + " " + " "

+ "£ " + salary.ToString("F2")); // Adds the £ sign and gives the value of salary

// in 2 decimal places (".00")

}

}

}

**Display Form code/ DisplayForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace assignment2\_240412

{

public partial class DisplayForm : Form

{

public DisplayForm()

{

InitializeComponent();

}

private void DisplayForm\_Load(object sender, EventArgs e)

{

}

private void richTextBox2\_TextChanged(object sender, EventArgs e)

{

//In its Properties, Modifiers was set to "Public"!! So the string of details

//could be sent from RegistrationForm!

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close(); //closes "this" object

}

}

}

**TESTING**

The program is run. The menu window comes up.



We click on Register. The Registration Form window comes up.



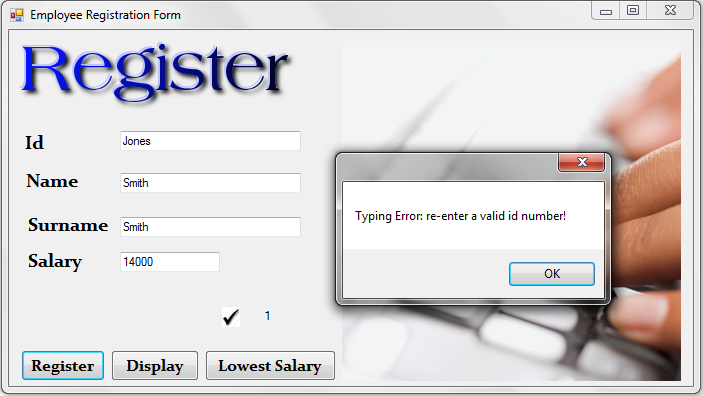
Details are entered.

**Scenario 1:** Correct data is typed in and registration goes through.



We click on Register and the tick shows “1” employee has been registered. The boxes are cleared for new input!

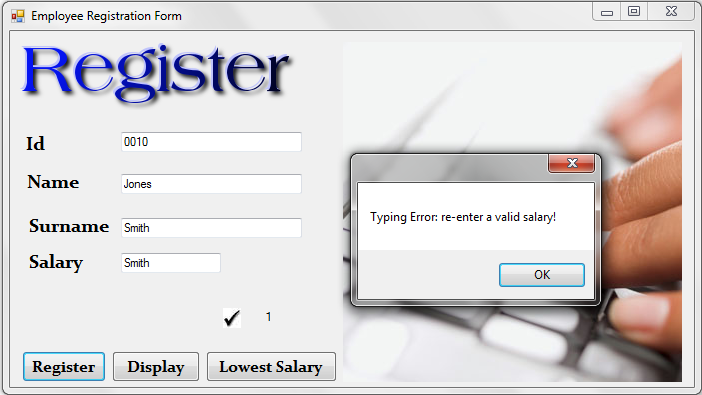


**Scenario 2:** Wrong Id number is typed in. The user mistakenly types their name instead of their Id number. A message showing typing error comes up. 

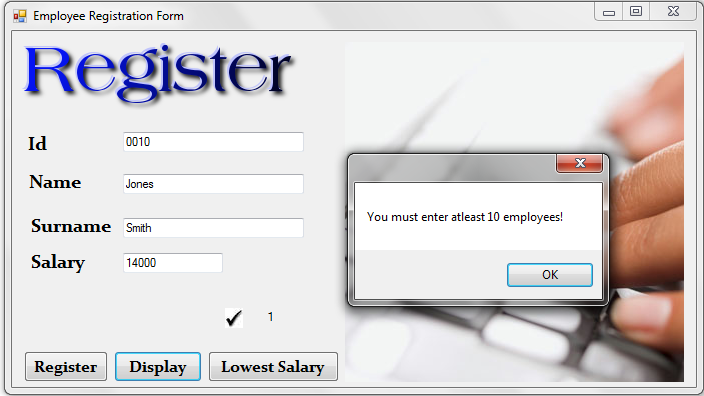
When the user presses ok to continue, the boxes are cleared ready for another input. The tick sign value remains “1”, only one correct registration has occurred so far!



**Scenario 3:** The user types in the wrong salary. They type in their surname mistakenly. An error message like the previous scenario comes up again.



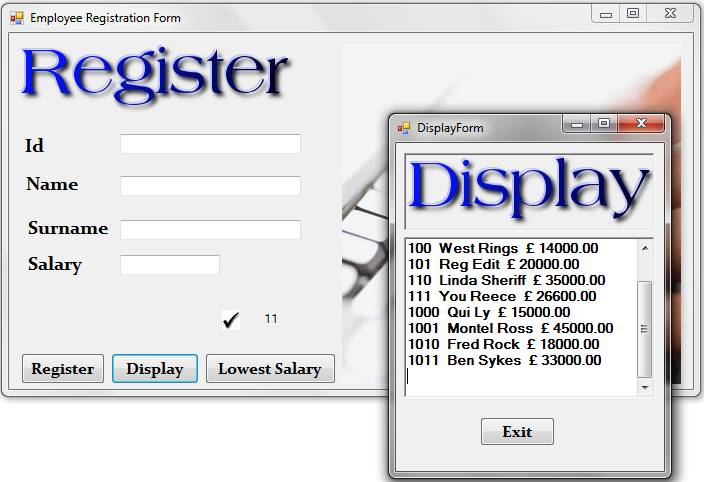
**Scenario 4:** The user enters correct data, and clicks on the Display button but doesn’t realise that they haven’t reached 10 employees yet. An elert message comes up to remind them.



They press ok. The screen remains the same and they are allowed to proceed with the registrations until they get to10.

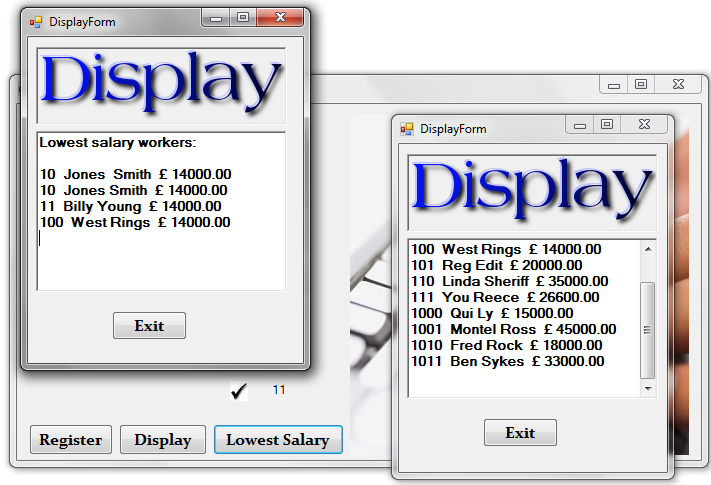


**Scenario 5:** The user finally gets to typing in all the details until he gets to 11 employees registered. And presses the display button. The display window pops up showing the list of employees.



**Scenario 6:** Has typed in the details and can see the list of employees registered. The user wants to know the list paid employees registered.

They press the Lowest Salary button. And the display window pops up with the list of the lowest salary workers.



To close display windows, user presses exit buttons. Until he is faced with the initial Menu window.



Now they can decide to make some new registrations once again, or simply exit.

The user presses the exit button and the programs stops.

**CONCLUSION**

The concept of OOP helps the programmer appreciate how instances of a class appear to be really independent. It makes working with data really easier. Once a class has been created all one needs to do if they need an object of that class is to create a new instance of it, without having to go through the process of declaring properties or methods once again. With the constructors the properties of the objects are initialized. Each constructor if more than one, having properties tailored to suit a particular instance of the class. This way of programming saves the programmer from having to re-write pieces of code, again and again.

**APPENDIX**

|  |  |
| --- | --- |
| Array  Workers[],7,8  Constructor,10  Counter,7  Count Increased,8  Count Decreased,8  Convert  To double, to int32, to string,8  Decimal places,11  Display  Form,7,8  Button,5,7  Employee,7  Class,10  Workers[],7,8  For,9  Form  Show,8  Close,12  Registration,7  Display,8  If,9  MessageBox  Show,8 | Methods  EmployeeToString,11  PrintLowestSalary,9  Call a method,8  Modifier property,12  New  Instance,7,8  Public  Method,11  RichTextBox  Appendtext,8 |