**OOP Basics**

**User Defined Type in C#**

In C#, you get two types of data type:

1. Primitive data type (int, double, string etc.)
2. User defined type (Employee, Customer, Member, Loan, etc.)

Also .Net framework has lots pre-made types (Console, Form, Convert, Button etc.) which can be used in C#. You have been already familiar with primitive type and some other data types provided by .Net framework. In this lesson, you will make your own data type and use these for solving your problem.

Consider the following example:

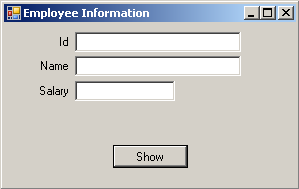


Figure 1: Employee information user interface

Here, user will enter employee’s information (See Figure 2) and press ‘Show’ button, all information will be cleared from textboxes and will be shown in a Messagebox as shown in Figure 3.

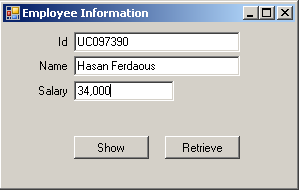


Figure 2: User enters employee’s information

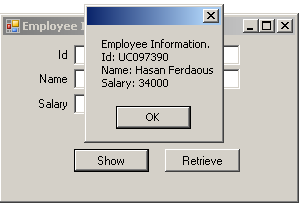


Figure 3: After pressing ‘Show’ button user will see the employee’s information in a MessageBox.

Already all data has been cleared from textboxes. But when user will press ‘Retrieve’ button, he will get the employee information into related textboxes (See Figure 4)

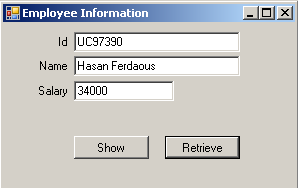


Figure 4: Pressing ‘Retrieve’ button user will see employee’s information in textboxes which has been entered just before.

If you use primitive data type your code might be as follows (Open the solution inside *User Define Type\EmployeeExample\PrimitiveType*)

public partial class EmployeeInformationUI : Form

{

private string id;

private string name;

private double salary;

public EmployeeInformationUI()

{

InitializeComponent();

}

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

{

idTextBox.Text = id;

nameTextBox.Text = name;

salaryTextBox.Text = salary.ToString();

}

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

{

id = idTextBox.Text;

name = nameTextBox.Text;

salary = Convert.ToDouble(salaryTextBox.Text);

ClearEmployeeInformationFromTextBoxes();

MessageBox.Show("Employee Information." + "\nId: " + id +

"\nName: " + name + "\nSalary: " + salary);

}

private void ClearEmployeeInformationFromTextBoxes()

{

idTextBox.Text = "";

nameTextBox.Text = "";

salaryTextBox.Text = "";

}

}

Here, three variables: *id*, *name* as string and *salary* as double have been declared. these variables have been outside of all methods to get the access in all methods.

The above code is self-descriptive and so description is not needed.

Interesting thing is that the above example can be solved by creating your own data type.

It is possible to keep several data (related) in a single user defined type data. Here, employee id, name and salary can be placed in a single user defined data type. Suppose the name of this type is *Employee*.

Follow the steps to solve the problem by using *user defined data type*:

1. Click right button on the project
2. Click Add>Class
3. In Name TextBox, type Employee.cs and press ‘Add’ button

You will see Employee.cs in solution explorer and see the following code inside Employee.cs file:

class Employee

{

}

Update the Employee class as follows:

class Employee

{

public string id;

public string name;

public double salary;

}

Then update your previous code as following code:

public partial class EmployeeInformationUI : Form

{

Employee employeeObj = new Employee();

public EmployeeInformationUI()

{

InitializeComponent();

}

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

{

idTextBox.Text = employeeObj.id;

nameTextBox.Text = employeeObj.name;

salaryTextBox.Text = employeeObj.salary.ToString();

}

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

{

employeeObj.id = idTextBox.Text;

employeeObj.name = nameTextBox.Text;

employeeObj.salary = Convert.ToDouble(salaryTextBox.Text);

ClearEmployeeInformationFromTextBoxes();

MessageBox.Show("Employee Information." + "\nId: "

+ employeeObj.id + "\nName: " + employeeObj.name +

"\nSalary: " + employeeObj.salary);

}

private void ClearEmployeeInformationFromTextBoxes()

{

idTextBox.Text = "";

nameTextBox.Text = "";

salaryTextBox.Text = "";

}

}

In primitive data type example, we use three variables to store employee information in our application.

private string id;

private string name;

private double salary;

But here, we replace these three variables by a single variable, employeeObj.

employeeObj variable is an Employee type, which is a user defined type. Actually, three data of an employee has been embedded inside employeeObj variable.

Employee employeeObj = new Employee();

The above line is very simple as any other primitive data type such as

int noOfStudents = 30;

Compare these two lines. In second line, your variable name is noOfStudents, in first line variable name is employeeObj. In second line, variable type it int, whereas in first line type is Employee. In second line you initialize noOfStudents by 30, in first line you initiialize employeeObj by the instance (or object) of Employee type.

**Practice-1:**

Open the solution file *StudentInformationPrimitiveDataType* located in *Basic Practices* folder. Here, the code has been written using primitive type. Find a User Defined Type here (guess what) and follow the above example (employee information) to replace variables with this type.

See the solution *StudentInformationUsingUserDefinedType* located in *Basic Practices*

**Method inside User Defined Type:**

If you see the provided solution of Practice-1, you will get Student as a user defined data type and this type is used in StudentInformationUI class.

In the third example, StudentInformationUsingUserDefinedTypeWithMethod

inside StudentExample you will see a method, GetFullName() in Student data type and this method is used in MessageBox. So, you see, it is possible to write method inside a user defined type.

**Object Oriented Concept**

**Introduction**

An object is a unit of code that is responsible to accomplish a particular task located inside code. In other way, we can tell that an object is a data structure which solves a particular problem. To do its work, an object may need some data (i.e. fields) and methods.

In this chapter, we will learn ‘How to think in Object Oriented way’ and how to write Object Oriented code to solve our problem. Here, we will discuss Object Oriented Though process, attributes and responsibilities of an object, what object and class means. At the end, we will see the comparison between several designs of objects.

**How to Start Object Oriented Programming?**

Most of us started Object Oriented Programming (OOP) by jumping into a language book like Complete Reference of C++/Java/C# etc. So, from these books we might have learned how to use their classes (class library) or even learned how to write your own classes and making object from them. But OOP is more than these. Designing beautiful classes, keep them simple, make them for reusability and easy to change is the ultimate goal of OO code and design.

So, what should we know first? First of all we should know: what’s an object and how to think in Object Oriented way.

**OOP Thought Process:**

Starting of Object Oriented Programming is of course the concept of object i.e. what object means? This concept is really simple. Let’s talk about it. Before starting, I am requesting you to forget everything about programming, database design, software design i.e. all kinds of software engineering and programming language details for some couple of minutes (not forever!!).

**Everything can be an Object :**

Look around us, what we see? ALL ARE OBJECTS. We see computer, chair, table, pen, air cooler, carpet etc, all these are objects. Why these are objects? Pretty simple, they have some related attributes and some related activities.

Think about computer, it has processor, RAM, hard drive, keyboard, mouse, etc. All these are the attributes of a computer. Now, what’re the activities of a computer? It has a lot: it runs antivirus program, displays information, accepts requests and executes these properly. All these are the activities of a computer. So, computer is an object.

In the same way, we will get the attributes and activities of chair, table, etc. So, to be an object, something should have attributes and/or should have activities. Finally, we could say: Everything can be an object.

Is it possible that any event, like war or raining, singing, thinking etc. (which are not visible or tangible) can be an object? The answer is YES. These can be objects also. Think, raining has attributes: whether is it drizzling or “cats and dogs”? Singing can be an object if we consider, in which frequency-range it is sung?

‘Name’? – is it an attribute or an object? Actually, it depends on problem domain. if we think about a banking solution then customer ‘Name’ will be an attribute of customer. But if it is a business from where you can purchase the ‘Name’ of your new-born baby then ‘Name’ will be an object to this business. Because ‘Name’ has attributes: name of baby boy or girl?, meaning, type, etc.

Note: In some OOP books, authors may use data instead attribute and responsibility instead of activity.

**Object’s data and responsibility depend on problem domain**

If I tell you and your friend to find out the data and activity of customer object, suppose, you get customer with name, email, phone attributes and purchase, bargain activities. At the same time your friend may get customer with first name, middle name, last name, gender, credit card attributes and open account activity.

Why it happens? It happens because you and your friend were on different platform with different point of view. You have thought about the customer of a grocery store whereas your friend has worked with the customer of a bank. But both of you are right. So, object and its data, responsibility everything totally depends on problem domain.

**What is class?**

Class is the blue print of an object. Suppose, Nirob Hasan, Linda Suzana and James Scott are students. Here, Nirob Hasan, Linda Suzana and James Scott are objects and Student is the class. More specifically, Student is the class which has Regi no, Email and Name attributes and there are three student objects whose names are Nirob Hasan, Linda Suzana and James Scott. See the following figure and get an idea of class and object.

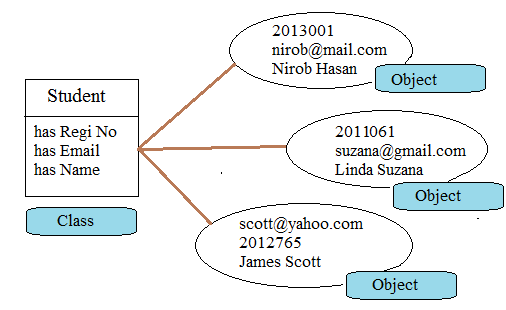


Fig: Object and Class

**Walkthrough 1: Mapping Object Oriented Thoughts to Object Oriented Programming**

So, we know what is class and what is object. So, let’s try a very simple example.

Our considering problem domain as follows:

“User wants a program where (s)he will enter first name, middle name and last name of a person, press a button, full name and reverse full name will be shown just below.”

We will identify the objects first. Here, we will get ‘Person’ object as follows:

**Person**

**Has first name**

**Has middle name**

**Has last name**

**Can tell its full name**

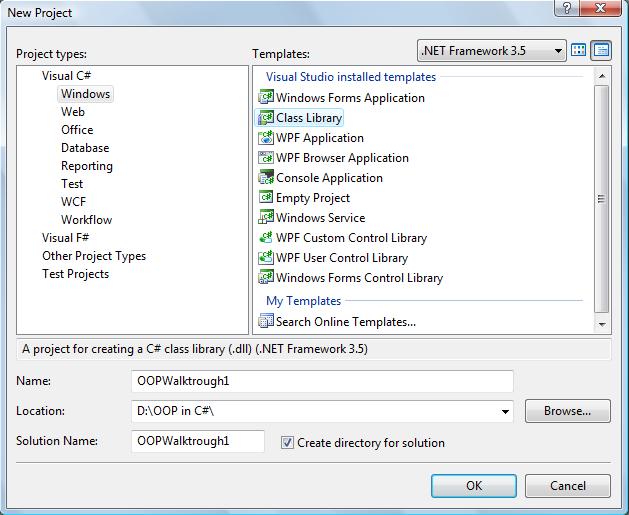
**Can tell its reverse name**

Now we will implement our above thought in C# by following steps:

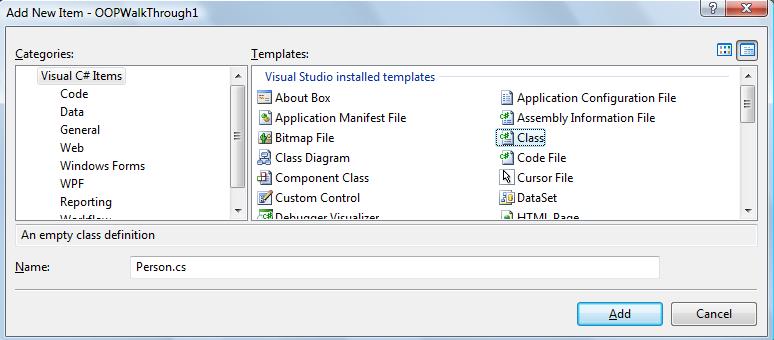
Steps:

1. Start Microsoft Visual Studio 2010
2. Click File>New>Project

Select ‘Class Library’ template from the list of templates, give a meaningful name and select your preferable location where VS 2010 will create all the related folders and files.



1. Click ‘Ok’ Button
2. In Solution Explorer, we will see Class1.cs file. Delete this file.
3. Select project (not solution) OOPWalkThrough1 and click right button. Then select Class from the templates.



1. We have found an object, ‘Person’ in our problem domain and now we are creating class for it. So, here we will give ‘Person.cs’ in Name as a file name.
2. Click ‘Add’ button and we will see the following code

using System;

namespace OOPWalkThrough1

{

class Person

{

}

}

1. Now update the Person class as follows:

namespace OOPWalkThrough1

{

class Person

{

public string firstName;

public string middleName;

public string lastName;

public string GetMyFullName()

{

return firstName + " " + middleName + " " + lastName;

}

public string GetMyReverseName()

{

string reverseName = "";

string fullName = GetMyFullName();

for (int index = fullName.Length - 1; index >= 0; index--)

{

reverseName += fullName[index];

}

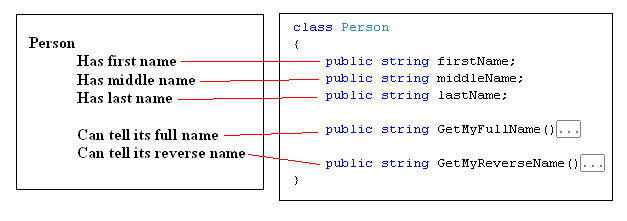
return reverseName;

}

}

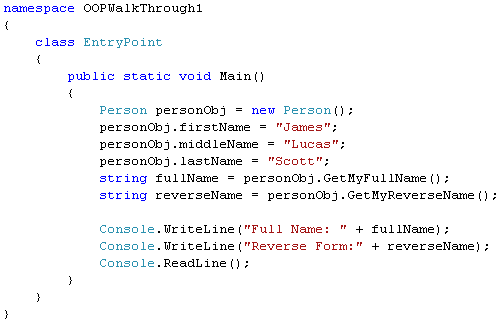
}

Have a look inside the Person class. Compare the identified object, Person with the Person class written above.



We see *data of Person are represented by attributes and responsibilities are represented by methods*. So any object creates from Person class will represent our domain object, Person.

1. Now, add another class in this project. Here file name will be EntryPoint.cs (or any meaningful name you like) and write Main() method inside this class as follows:



Here, we create a Person object, personObj from Person class using the following line:

Person personObj = new Person();

Then assign some data to the attributes of personObj as follows:

personObj.firstName = "James";

personObj.middleName = "Lucas";

personObj.lastName = "Scott";

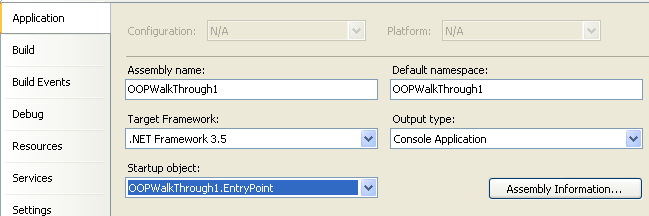
Finally, call its two methods to get full name and reverse name from it:

string fullName = personObj.GetMyFullName();

string reverseName = personObj.GetMyReverseName();

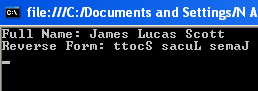
We have started our project as a class library so if you run it you will get an error. Since the output of a class library is DLL it can’t be run.

1. To make the project executable, we need to change the project properties. Select the project, click right button, select Properties from pop-up menu; we will get the following tab to change project properties.



Select Output Type as Console Application and Startup object as OOPWalkThrough1.EntryPoint.

1. Run the application and we will the output as follows:



1. Now we will update the code so that we can take input from user instead of hard-coded value. Replace these three lines

personObj.firstName = "James";

personObj.middleName = "Lucas";

personObj.lastName = "Scott";

with the following three lines

personObj.firstName = Console.ReadLine();

personObj.middleName = Console.ReadLine();

personObj.lastName = Console.ReadLine();

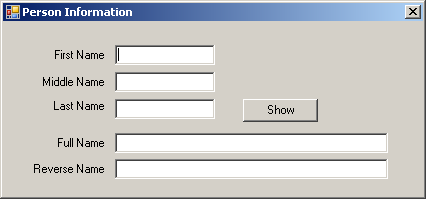
Now run the application and test it for several inputs. Find the code of this walkthrough in *OOPWalkThrough1* inside *WalkThroughs* folder.

Let’s have look into the files and folder created in the selected folder for the above project.

Walkthrough 2: Convert the application of Walkthrough 1 to Windows application.

These walkthrough is based on Walkthrough 1. So, open the project of Walkthrough1 and do the following steps:

1. Add a new Form class in your project (Right-click on project, click Add>Windows Form). In name textbox, type PersonInformationUI.cs and click Add button. Then you will see a windows form.
2. Design the windows form as following UI and give the appropriate name of each control.



1. Write the following code inside Show button\_click event:

Person personObject = new Person();

personObject.firstName = firstNameTextBox.Text;

personObject.middleName = middleNameTextBox.Text;

personObject.lastName = lastNameTextBox.Text;

fullNameTextBox.Text = personObject.GetMyFullName();

reverseNameTextBox.Text = personObject.GetMyReverseName();

1. Change your Main() method of EntryPoint class with following code:

PersonInformationUI personInformationUIObj =

new PersonInformationUI();

Application.Run(personInformationUIObj);

1. Right-Click on project of solution explorer. You will see a popup and click properties. Select Application Tab, Change output type to Windows Application.
2. Run the application and see it works.

By following the steps of two above walkthroughs, we have created a small desktop application by which user can get full name and reverse name giving first name, middle name and last name. First we have identified domain object, Person and then we have created class, Person using C# code without starting with Windows Form.

We have started with Class Library template instead of Console or Windows Form Application, because we have planned to write all the classes from scratch needed for this project. But it’s doesn’t mean that we should following this way for any desktop application.

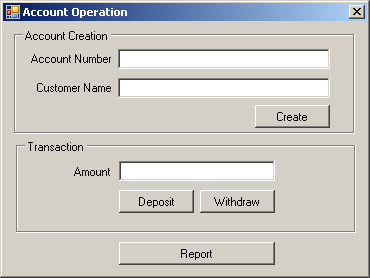
**Practice 1: Application for bank account transaction**

User story:

“As a customer

I want to create an account and want to deposit and withdraw in my account.”

See the UI prototyping:



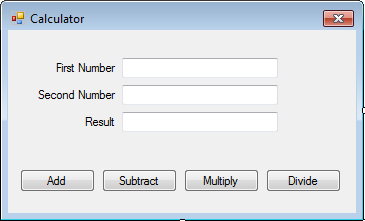
When report button is clicked, information will be displayed as follows:



Hint: Do the brain-storming work first. Don’t start coding right now. Find out an object and write its name, attributes and responsibilities in pen and paper. Do these by discussing with your peer and after that start coding.

Now, see the solution in *Solutions* folder. Project name: *BankAccountOperation* and find the difference between your code and given code. Let’s discuss it.

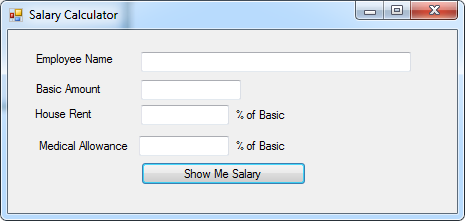
Practice 2: Make a calculator as follow:



Hint: Remember, you should do the brain-storming work first. Don’t start coding right now. Find out an object and write its name, attributes and responsibilities in pen and paper. Do these by discussing with your peer and after that start coding.

Have a look into the given solution in *Solutions* folder. Project Name: *CalculatorApp.* Have you seen any difference? Let’s discuss these.

**Practice 3: Make a salary calculator**



User will input data and click *Show Me Salary* button and then total aalary will be shown in a MessageBox, like: Sumon your salary is: 78500

Hint: Please do brain-storming work with your peer and do it before coding. There’s a solution in *Solutions* folder. Project name *EmployeeSalaryApp.* Compare your code with it.

**Practice 4: Make a circle class by which user can get diameter, perimeter, and area of it providing the radius of it. Write Main() method inside Program class to demonstrate your work.**

Comparison between alternative designs of objects:

Let’s start a discussion on two alternative designs of each object so far we have found.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Person | | firstName: string  middleName: string  lastName: string | | GetFullName():string  GetReverseName(): string | | |  | | --- | | Person | |  | | GetFullName(fN: str, mN: str, lN:str):str  GetReverseName(fN: str, mN: str, lN:str): string | |
| |  | | --- | | Calculator | | firstNumber: double  secondNumber: double  result: double | | Add():double  Subtract(): double  Multiply(): double  Divide(): double | | |  | | --- | | Calculator | |  | | Add(frN:double, sndN:double):double  Subtract(frN:double, sndN:double):double  Multiply(frN:double, sndN:double):double  Divide(frN:double, sndN:double):double | |
| |  | | --- | | BankAccount | | number: string  name: string  balance: double | | Withdraw(amount: double):string  Deposit(amount:double): string | | |  | | --- | | BankAccount | |  | | Withdraw(no:str, name:str, blnc: dbl):string  Deposit(no:str, name:str, blnc:dbl): string | |
| |  | | --- | | Circle | | radius: double | | GetArea():double  GetPerimeter():double  GetDiameter():double | | |  | | --- | | Circle | |  | | GetArea(r:double):double  GetPerimeter(r:double):double  GetDiameter(r:double):double | |

Everything can’t be covered in a sample example or even in a big one:

Here, to keep the example simple, I’ve violated encapsulation (keep the data public) principle.

To solve a problem, mapping a domain object directly to software object sometimes may not be a good solution.