C#

Object Oriented programming

## Objective

The primary objective for this lab is as follows:

* To create and use references.
* To consolidate on passing by value and passing by reference

## Part 1 – Creating and using reference types

### Step by step

1. In the **labs** project, create a class called **Account** with the following fields  
   **private int** id;   
   **private string** owner;   
   **private double** balance;   
     
   and methods:

**void Deposit(double amount) { }**

**void Withdraw(double amount) { }**

1. Create a constructor for the **Account** class to set its fields   
   (id, owner and balance).
2. Modify the **Deposit** and the **Withdraw** methods to make sure no one can Deposit a negative amount of money o r Withdraw the money they don't have (in their **balance**).
3. Create a method called **GetDetails**() to return details of the account as a **string** made up of the **id + owner + balance** (put a ‘,’ between these)
4. In the Program class, create a static method called **Lab2()**
5. Call Lab2() method from within **Main()**
6. Comment out the call to any other method.
7. In the Lab2() method create an instance of the **Account** class

And then invoke its **Deposit()** and **Withdraw()** methods.

1. Call the Account instance's **GetDetails()** method and then print the result to make sure your code works.

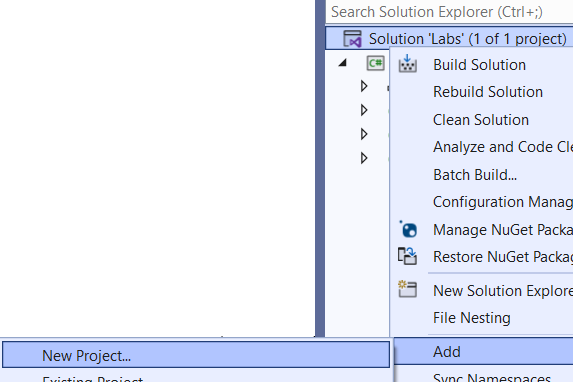
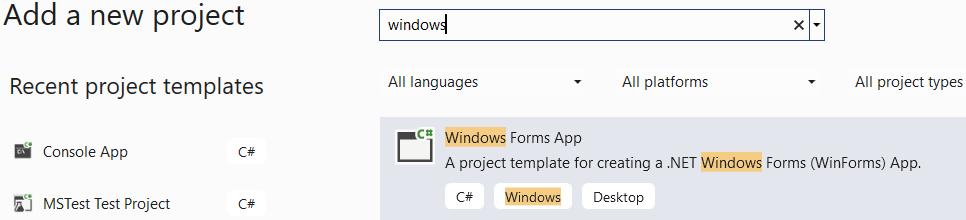
## Part 2: Create a graphical application

## Objective

You will further your understanding of object oriented programming.   
In this lab you'll create a graphical application instead of a Console app.

## Part 1

You'll soon be creating a graphical application which is new. You'll copy and paste many lines of code. You are not required to understand how the graphics code works but it is not too complex to investigate by yourself.

1. Add a new Windows Form application called **GamesApp** to your Labs solution.  
   Right mouse click on the Labs Solution and then select the   
   **Add > New project…** menus  
   
2. Create a new Windows Forms App called **GameApp** by searching for “**Windows”** and by selecting the **Windows Forms App (.Net Framework)** option  
     
   
3. Right mouse click on the newly created GameApp project and then select the “**Select as Startup Project…**” menu
4. Have a look at the **Program** class. It creates an instance of a class called **Form1** which represents a Windows Form. You'll use this class to display your graphical objects.
5. You need a canvas upon which you can draw and show your animation. In this case you'll create a number of ball objects bouncing inside a rectangle (the world).

You can get creative later and do something else but for now, please follow the steps. The aim is to investigate OO concepts and not concentrate on animation and graphics.

1. Click on the body of the Form and then press **F4** to see its properties (if not displayed already)
2. Click the lightening symbol  to see the form's events.   
   An **event** is an action which you can handle in code. For example, when a user clicks on a control, the control's click event is fired. In this application, we're after the **Paint** event which is fired when the form needs to be redrawn or repainted.
3. Double-click on the **Paint** event from the list of events. You'll then be redirected to the code behind the form.
4. Type the following code in the Paint event. This code clears the screen and draws a red reactangle. 10,10,600 and 400 is the world's dimensions.

private void Form1\_Paint(object sender, PaintEventArgs e)

{

e.Graphics.Clear(Color.Gray);

e.Graphics.FillRectangle(Brushes.Red, 10, 10, 600, 400);

}

1. Run the application to make sure everything works so far.
2. Stop running by closing the Form.
3. Double-Click on the Form1 icon in the Solution Explorer window to view it in design mode.
4. Press **F4** and then click  to see the form's events.
5. Double click on the **Load event** from the list to view an event which fires when the form is loaded for the first time.

Type the following code in the *Load event* in order to create a timer which fires every *50* millisecond.  
  
private void Form1\_Load(object sender, EventArgs e)

{

Timer t = new Timer();

t.Interval = 50;

}

You will need to press Ctrl-. (Control dot) On the word **Timer** and then select using Timer = System.Windows.Forms.Timer;

1. Add the following text to the above method **t.Tick +=**   
   and then press the Tab key to create an event handler for the Timer's tick event.

Graphical user interface, application

Description automatically generated

1. In the Tick event type: **this.Invalidate();**  
   This will force the *Paint event* to fire every 50ms.
2. Back in the *Form Load* event, activate the timer by adding this line of code: **t.Start();**From now all your code for drawing will go into the Paint event.  
   Your code will look like:

private void Form1\_Load(object sender, EventArgs e)

{

Timer t = new Timer();

t.Interval = 50;

t.Tick += T\_Tick;

t.Start();

}



### Create a class called **Ball**

1. Create a class called **Ball** with the following fields

**public int** x, y, w, h;

**private** **int** dirX, dirY;

X axis

Y axis axis

x,y axis

w

h

x and y represent the top coordinate of the shape (ball) within an upside down coordinate system. dirX and dirY are the amount by which a ball Moves in the x and y directions.  
  
**Note:** We must create these as property procedures but elected to make these public vars for simplicity.

1. Create a constructor to set every field (x,y,w,h,dirX, dirY).
2. To practice constructor chaining, create another constructor to set all the fields except dirX and dirY.

Using constructor chaining, call the first constructor and pass it dirX=1 and dirY=1.

1. Create a method called **Move()** in the Ball class.  
   In this method, increase x by dirX and y by dirY and then follow a series of tests to make sure the ball Moves within a designated world area (x=10, y=20, width=600, height=400) which you set in step 8.

if (x < 10) then set x=10 and change the sign of dirX   
Tip: dirX = **-**dirX.  
Do the same for the y coordinate(changing dirY of course).

if (x > 600 – the width of the ball) then set   
x=600 – the width of the ball and reverse the sign of dirX.

1. Do the same for the y coordinate taking the ball's height into account and the overall height of 400 for the world in which a ball can travel in.
2. Create an array field (at the beginning of the class Form1) as: **Ball[] balls;**

public partial class Form1 : Form

{

Ball[] balls = new Ball[3];

1. Back in the Form Load event handler, create three instances of Ball and place them inside the **balls** field (see below).

private void Form1\_Load(object sender, EventArgs e)

{

balls = new Ball[]

{

new Ball(30, 30, 20, 20, 2, 3),

new Ball(40, 40, 30, 30, 3, 4),

new Ball(80, 90, 20, 40, 4, 3)

};

Timer t = new Timer();

t.Interval = 50;

t.Tick += T\_Tick;

t.Start();

}

1. In the *Paint event,* create a foreach loop to go through each ball and run its **Move()** method. You will then draw the ball after moving it. Please use the code below to draw a ball:   
   e.Graphics.DrawEllipse(Pens.Yellow, b.x, b.y, b.w, b.h);  
   **// where b is the ball reference you created earlie r**
2. If everything works, you should see all 3 objects moving around within the world's rectangle.

If you have a need for speed, then increase the values of dirX and dirY of each ball.

## Part 2 - Using a struct (Optional)

**System.Drawing.Rectangle** is a struct which is defined by Microsoft and can be useful to us in this lab  
  
This struct can combine 4 attributes in one. These are X,Y,Width and Height  
It has other Properties such as Right, Bottom, Left (or X) and Top (or Y)   
Here is a sample code to guide you.

public class Ball

{

Rectangle world = new Rectangle(10, 10, 600, 400);

public Rectangle Rec { get; set; }

**// …**

public Ball(Rectangle rectangle, int dirX, int dirY)

{

Rec = rectangle;

…

…

}

**// In the Ball’s Move method:**

Rec.Offset(dirX, dirY);

if(Rec.Right > world.Right)

{

dirX = -dirX;

Rec.**Offset**(dirX, 0);

}

}

To create a new Ball do:

**Ball b = new Ball( new Rectangle(100,50,30,30), 2, 3);**

**\*\* End \*\***