Rewarding Learning

AS LEVEL

FACT FILES

Software Systems Development

AS2 Event Driven Programming

(ii) UNDERSTANDING EVENTS

For first teaching from September 2013

For first AS award in Summer 2014

For first A Level award in Summer 2015

FACT FILE

software
systems
development

FACT FILE

software systems development

AS2 Event Driven Programming
(ii) UNDERSTANDING EVENTS

Learning Outcomes

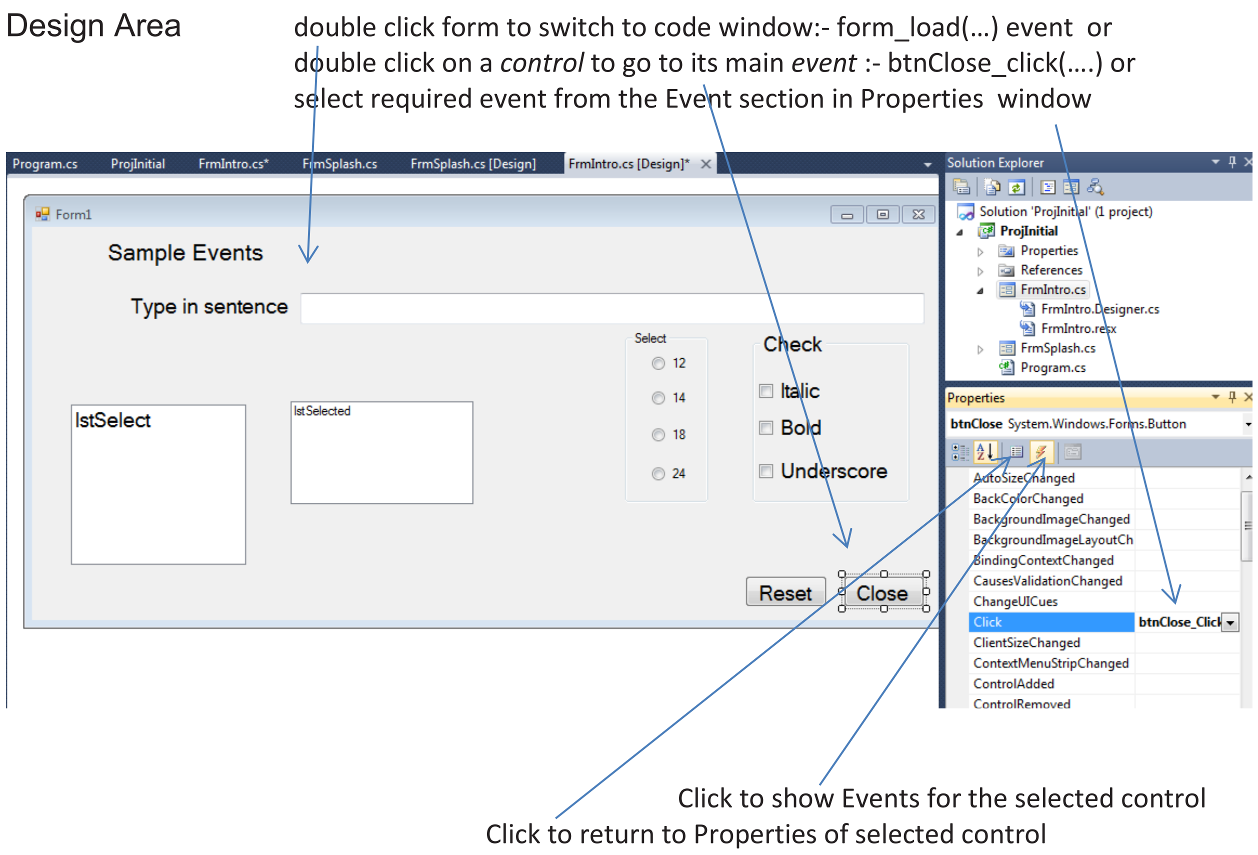
Students should be able to:

- demonstrate and apply their understanding of events in the implementation of an event driven application;
- use triggers, for example button, mouse clicks, key presses.

Course Content

An event is an action that occurs to a control which 'fires' the associated method and executes the code within it. Microsoft provides many events for the controls and sends additional information as parameters to the corresponding method. This may be triggered by a mouse or key movement.

The C# GUI Environment: - selecting an event to code



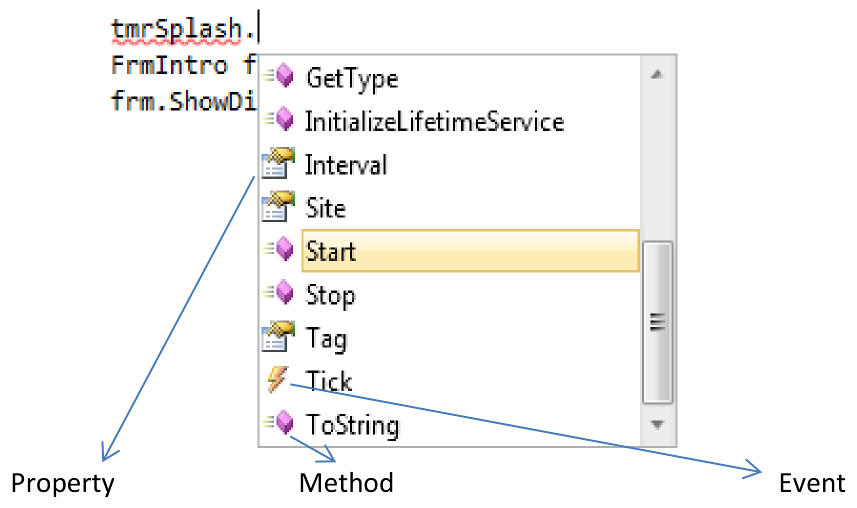
NB: The following code is added automatically when an event method is coded by the user.

```
        this.btnClose.Click += new System.EventHandler(this.btnClose_Click);
                    Event                             associated method
```

(Microsoft creates methods to handle events through the class EventHandler.)

In code the intellisense window opens when a control name is typed with a dot. The user can choose a Property, an Event or a Method. An icon precedes each and helps to categorise.

C# - Icons used to differentiate between Properties, Methods and Events.



```
tmrSplash.
FrmIntro f
frm.ShowDi
```

C# Notes on Events: - Click

http://msdn.microsoft.com/en-us/library/system.windows.forms.control.click.aspx

C# Common Windows Control Events

The most common event is the click event which applies to a large percentage of the controls provided by Microsoft. The DoubleClick event is very useful to ensure validity of user action.

The **load** event is fired automatically on instantiation of the form and can be used to initialise values and reset control Properties.

Below is a sample of the form **load event** and the **click event** for a Button control - btnClose.

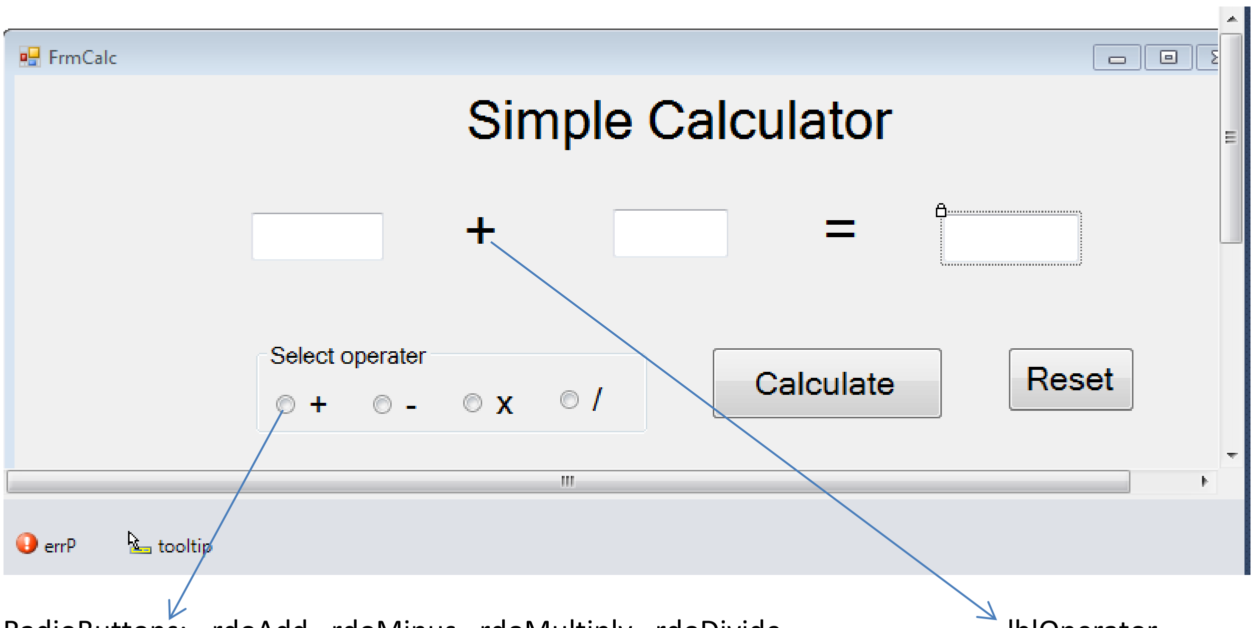
```
namespace ProjInitial
{
    public partial class FrmIntro : Form
    {
        public FrmIntro()
        {
            InitializeComponent();
        }

        private void FrmIntro_Load(object sender, EventArgs e)
        {
            lblTitle.Text = "Hangman";
            lblTitle.BackColor = Color.Azure;
            lblTitle.ForeColor = Color.DarkBlue;
            lblTitle.Font = new System.Drawing.Font("Comic Sans", 36F);

        }
        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Continued on next page.

Events Sample for RadioButtons on the Simple Calculator



RadioButtons:- rdoAdd, rdoMinus, rdoMultiply, rdoDivide   lblOperator

rdoAdd_CheckedChanged() is the common event for the RadioButton but is fired when the state changes from true to false a well as from false to true.

In the Calculator example the operator should only be set when the RadioButton is true. The Click event is more appropriate as it fires on the change from false to true.

NB: The action required is to set the lblOperator Text property to the Text Property of the RadioBuuton. As this is repeated for the four RadioButtons it is more efficient to apply the one event method to all four controls.

```
private void FrmCalc_Load(object sender, EventArgs e)
{
    // as the radio buttons do the same function i.e.
    // set the lblOperator to character in Text property
    // use the one event method to handle all four events.
       rdoDivide.Click += new System.EventHandler(rdoAdd_Click);
       rdoMultiply.Click += new System.EventHandler(rdoAdd_Click);
       rdoMinus.Click += new System.EventHandler(rdoAdd_Click);
}
```

The parameter 'sender', passed to the method, can be used to determine the operator if required.

```
private void rdoAdd_Click(object sender, EventArgs e)
{
       // sender is the control where the event was initiated
       // cast as a RadioButton to access the properties and methods of that control
       lblOperator.Text = ( (RadioButton)sender ).Text;
}
```

Setting date and time on a label of the form

From the system:-
In the *Load event* of the required form add the following code:-

    lbldate.Text = DateTime.Now;

From a *Selected date* event in the MonthCalendar control:-

    lblDate.Text = this.monthCalendar1.SelectionRange.Start.ToShortDateString();

From a *ValueChanged* event in the dateTimePicker control:-

    lblDate.Text = this.dateTimePicker1.Value. ToShortDateString();

NB: can also set dateTimePicker to a date e.g.

When using a DateTimePicker on the form it can be set to current date/time (e.g. on form load)

```
this.dateTimePicker1.Value = DateTime.Now;
```

ListView Control - events

```
private void listView1_SelectedIndexChanged(object sender, EventArgs e)
 {

 }

private void listView1_Click(object sender, EventArgs e)
 {

 }
```

Note the use of the form_load and private custom methods used to populate the listView control.

Example 1 – populating the listView from a binary file Treatments.bin :-

```
private void FrmAddAppointment_Load(object sender, EventArgs e)
        {
            ListViewItem item = new ListViewItem();
            Stream sr;
            BinaryFormatter bf = new BinaryFormatter();

            Treatment[] treats;

            // for testing purposes creation of treatments is called here
            //- comment out after executing once
            // treatment details would normally be added through form (FrmAddTreatment)
            // and be written to the binary file.

            createTreatments();
                                  //read treatment details from file to array object
            try
            {
                sr = File.OpenRead("Treatments.bin");
                treats = (Treatment[])bf.Deserialize(sr);

                //add each treatment to the ListView
                sr.Close();
                for (int x = 0; x < treats.Length; x++)
                {
                    item = lsvTreatments.Items.Add(treats[x].TreatmentNo.ToString());
                    item.SubItems.Add(treats[x].Description);
                    item.SubItems.Add(treats[x].Price.ToString());
                    item.SubItems.Add(treats[x].Duration.ToString());
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("" + ex.Message);
            }

        }

        private void  populateTreatmentArray(Treatment [] treats)
        {

            treats[0] = new Treatment(1000, "Cut", 25.0,25);
            treats[1] = new Treatment(1001, "Wash and Blow-Dry", 15, 30);
            treats[2] = new Treatment(1002, "Colour", 65.75,120);
            treats[3] = new Treatment(1003, "Extensions", 124.50,100);
            treats[4] = new Treatment(1004, "Wedding prep", 150.0,120);
        }
        private void createTreatments()
        {
            Stream sw;
            BinaryFormatter bf = new BinaryFormatter();
            Treatment[] treats = new Treatment[5];
            populateTreatmentArray(treats);
            try
            {
                // write cars array object to file
                sw = File.Open("Treatments.bin", FileMode.Create);
                bf.Serialize(sw, treats);
                sw.Close();
            }
            catch (SerializationException e)
            {
                Console.WriteLine("" + e.Message);
            }

        }
```

The following example shows how Listview controls can be used for selection/display in an Order Processing activity . Data can be selected from one Listview control ('available stock') and moved to another Listview control ('ordered stock'). It is removed from the 'available stock' and so prevents duplicate stock numbers in an order.

To select an item from a ListView use the click event (or DoubleClick event – deleting)

```
private void lsvStock_Click(object sender, EventArgs e)
{
        int no;
        ListViewItem itm = new ListViewItem();

                        //  get index of first selected row item and retrieve data to itm
        no = lsvStock.SelectedIndices[0];
        itm = lsvStock.Items[no] ;
                        // remove item from ListView as it has been selected for order
        lsvStock.Items[no].Remove();

                        // Add item to the 'buy' ListView
        lsvBuy.Items.Add(itm);
}
```

QUESTIONS

Q1 Which event would you recommend for setting today's date onto a Label, lblDate, on a form? Write the code which would show this.
Include code to reset the fontsize of the lblDate to 14.

Q2 A game project could contain an introduction form where a player could select a 'cartoon' character's name from a ListBox named lstPlayer, to represent him/her.
Which ListBox event would you activate and how would you reference the selected name in method code.
(Assume a global String, strPlayer, variable is defined to hold the selectedname)

Q3 The form in Question 2 is modified to contain an ImageList of the cartoon characters' pictures stored in the same order as the cartoon characters' names in lstPlayer. An Image control, imgSelectedPlayer, is also added to the form.

What code would you add to display the relevant player image in imgSelectedPlayer and where would you place it.

software
systems
development

Event Driven Programming