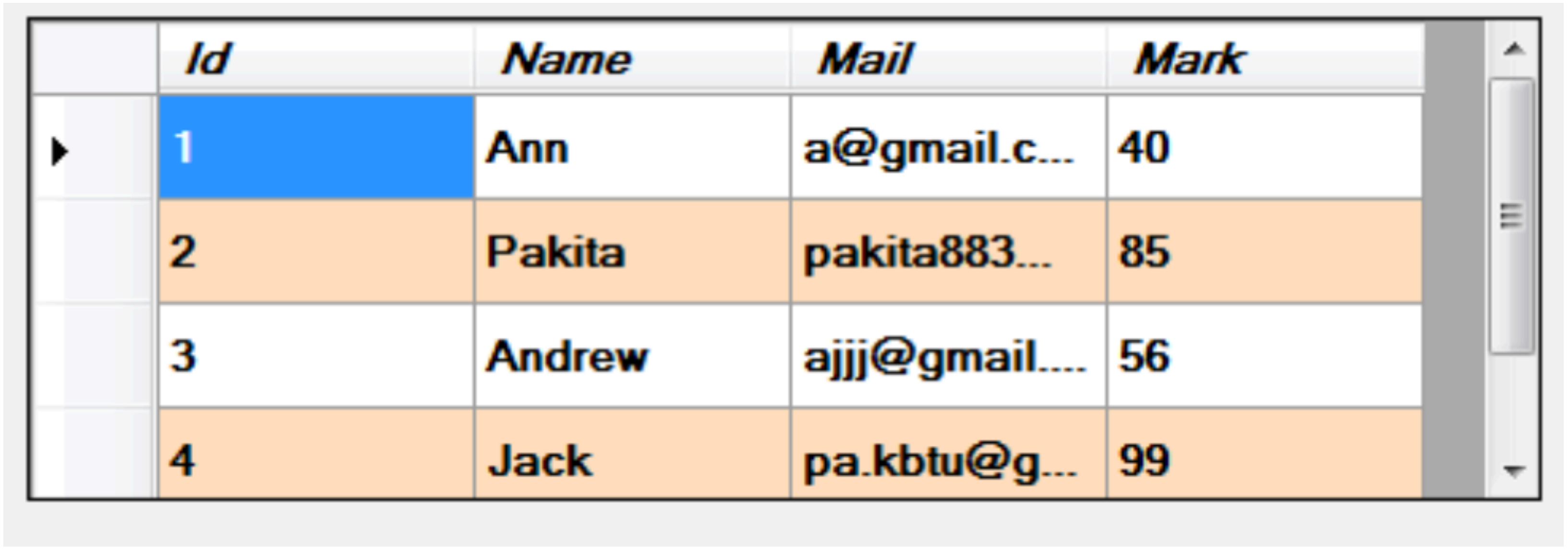
Advanced WF Controls, User-defined Controls

By Pakita Shamoi

Spring 2014

DataGridView



Id	Name	Mail	Mark
1	Ann	a@gmail.c...	40
2	Pakita	pakita883...	85
3	Andrew	ajjj@gmail....	56
4	Jack	pa.kbtu@g...	99

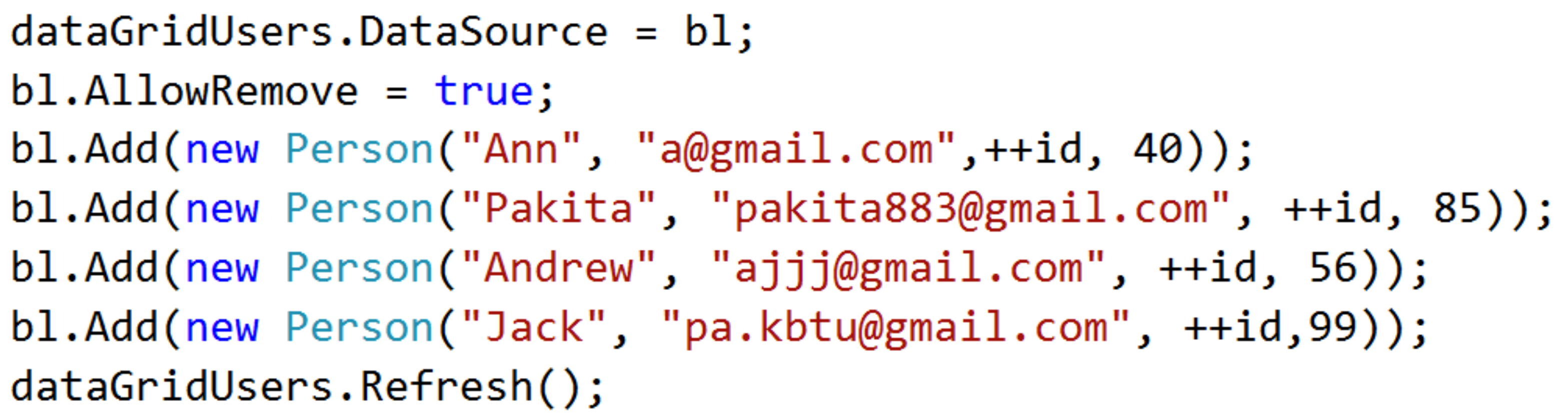
```
class Person
{
    string name;
    string mail;
    int id;
    int mark;
    public Person(string name, string mail, int id, int mark) {
        this.id = id;
        this.name = name;
        this.mail = mail;
        this.mark = mark;
    }
    public Person() { }
    public int Id
    {
        get { return id; }
        set { id = value; }
    }
    public string Name
    {
        get { return name; }
        set { name = value; }
    }
```

Steps to fill DataGridView

- Create class or structure
- Store it in a Binding List or any other collection

  ```
  BindingList<Person> bl = new BindingList<Person>();
  ```
- Set the data source of your grid to a collection you created
- Add objects to a collection
- Refresh grid

Steps 3-5



```
dataGridUsers.DataSource = bl;
bl.AllowRemove = true;
bl.Add(new Person("Ann", "a@gmail.com",++id, 40));
bl.Add(new Person("Pakita", "pakita883@gmail.com", ++id, 85));
bl.Add(new Person("Andrew", "ajjj@gmail.com", ++id, 56));
bl.Add(new Person("Jack", "pa.kbtu@gmail.com", ++id,99));
dataGridUsers.Refresh();
```

Notes

- Keep in mind that when grid allocates columns corresponding to the fields of your class, it looks at **properties.** So, e.g. if we delete the property for *Name*, you won't see such column in your grid
- The columns are added in the same order in which properties appear in your class.
- Remember that you can always fill grid manually

Useful events and properties

- **Events**

CellBeginEdit, CellClick, CellEndEdit, CellLeave, CellDoubleClick, CellValueChanged,…

- **Properties**

AllowUserToAddRows, AllowUserToDeleteRows, AllowUserToOrdeColumns, AlternatingRowsDefaultStyle, DataSource, DefaultCellStyle, …

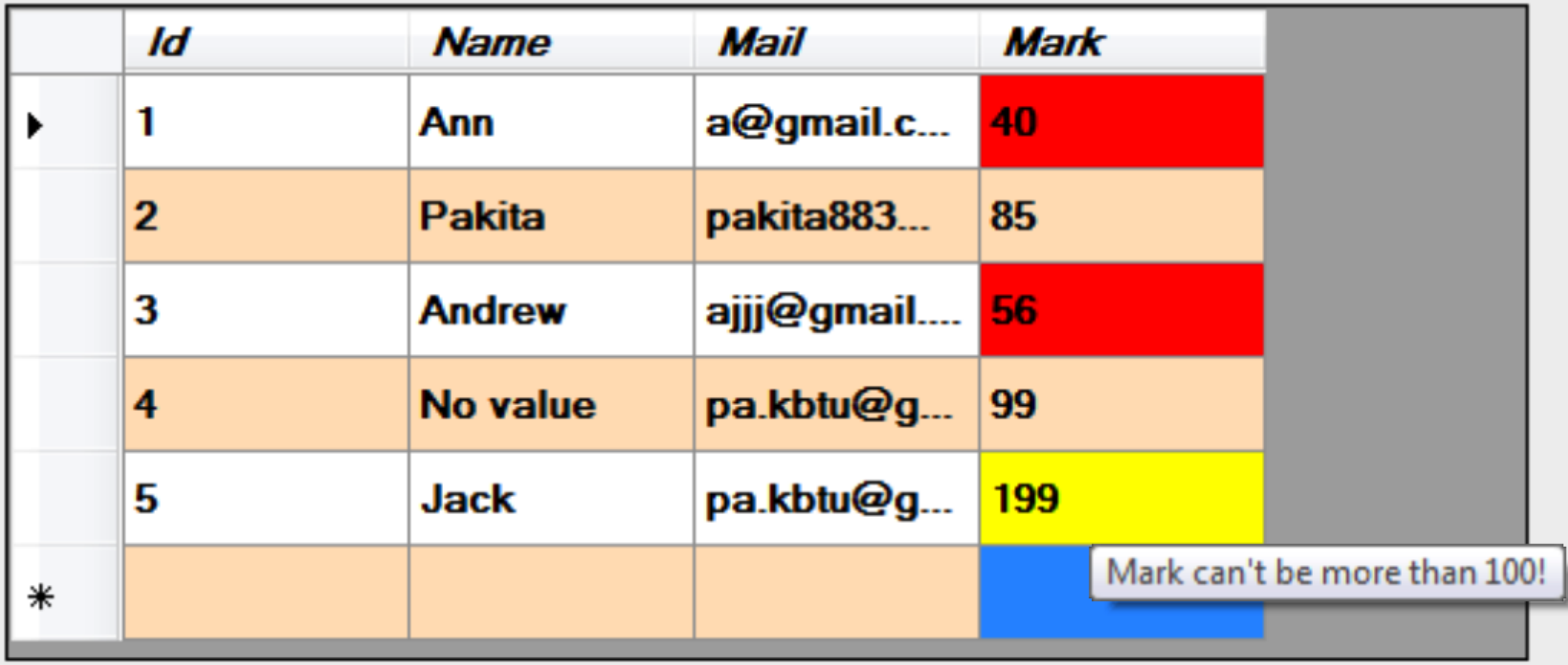
Formatting Data in DataGridView

- You can validate data in your grid for various reasons. For our grid, e,g., we can do this to:
  - Highlight with red all students with low marks
  - Highlight all marks that in the incorrect format and set the corresponding hint (tooltip)
  - Replace all empty values with some value.

Formatting Data in DataGridView

```
for (int i = 0; i < dataGridUsers.RowCount; i++)
    if (dataGridUsers[3, i].Value != null && (int)dataGridUsers[3, i].Value < 70)
        dataGridUsers.Rows[i].Cells[3].Style.BackColor = Color.Red;

foreach (DataGridViewRow row in dataGridUsers.Rows)
{
    foreach (DataGridViewColumn col in dataGridUsers.Columns)
    {
        if (dataGridUsers[col.Index, row.Index].Value == null) return;
        if (dataGridUsers.Rows[row.Index].Cells[col.Index].Value.ToString()=="")
        {
            dataGridUsers[col.Index, row.Index].Value = "No value";
        }
        if(col.Name.ToString()=="Mark")
            if ((int)dataGridUsers[col.Index, row.Index].Value > 100)
            {
                dataGridUsers.Rows[row.Index].Cells[col.Index].Style.BackColor = Color.Yellow;
                dataGridUsers.Rows[row.Index].Cells[col.Index].ToolTipText = "Mark can't be more than 100!";
            }
    }
```

Formatting Data in DataGridView



	Id	Name	Mail	Mark
▸	1	Ann	a@gmail.c...	40
	2	Pakita	pakita883...	85
	3	Andrew	ajjj@gmail....	56
	4	No value	pa.kbtu@g...	99
	5	Jack	pa.kbtu@g...	199
*				

Mark can't be more than 100!

Defining User Controls

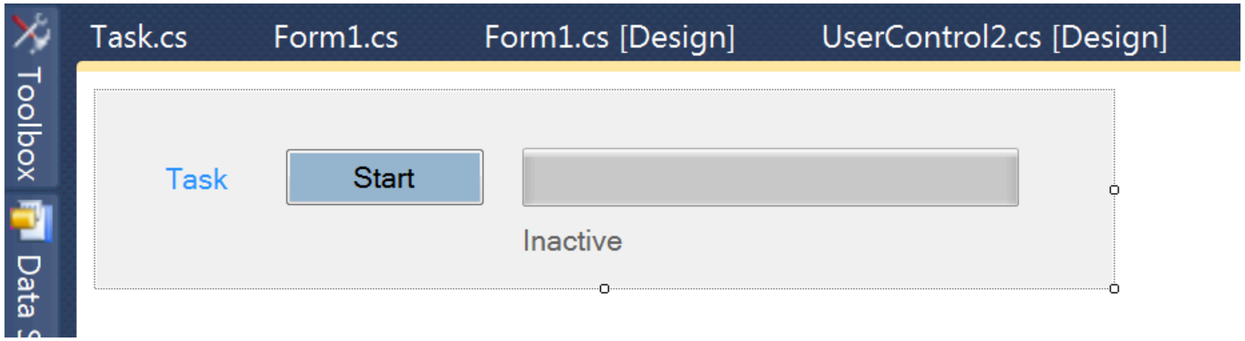
- NET platform allows you to create your own user controls. This can be very useful when you have a group of controls that is repeated several time in the app.

When displaying data on Windows Forms, you can choose existing controls from the Toolbox, or you can author custom controls if your application requires functionality not available in the standard controls.

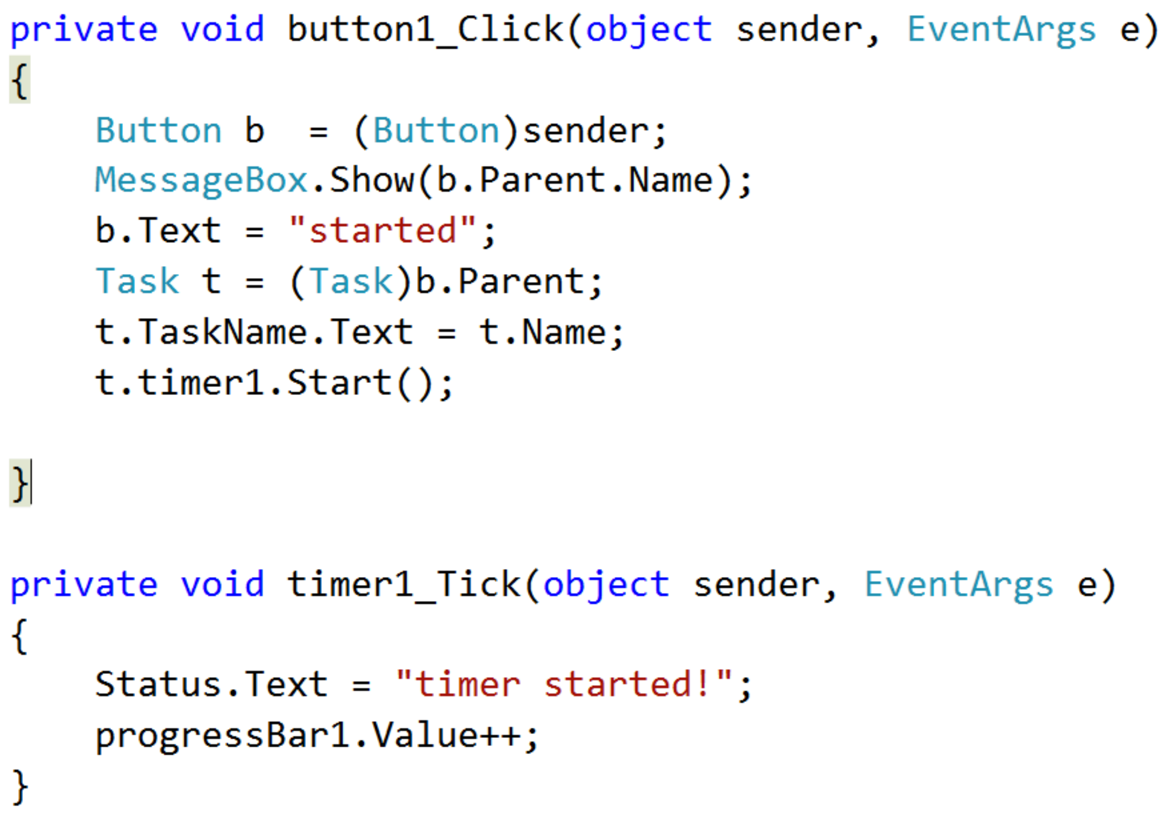
Step 1 - Adding User Control

- From the Project menu, select Add User Control.
- Type name for your control in the Name area, and then click Add.
- Then you will see that control with the corresponding name is added to Solution Explorer and opens in the designer.

Step 2 – Design User Control

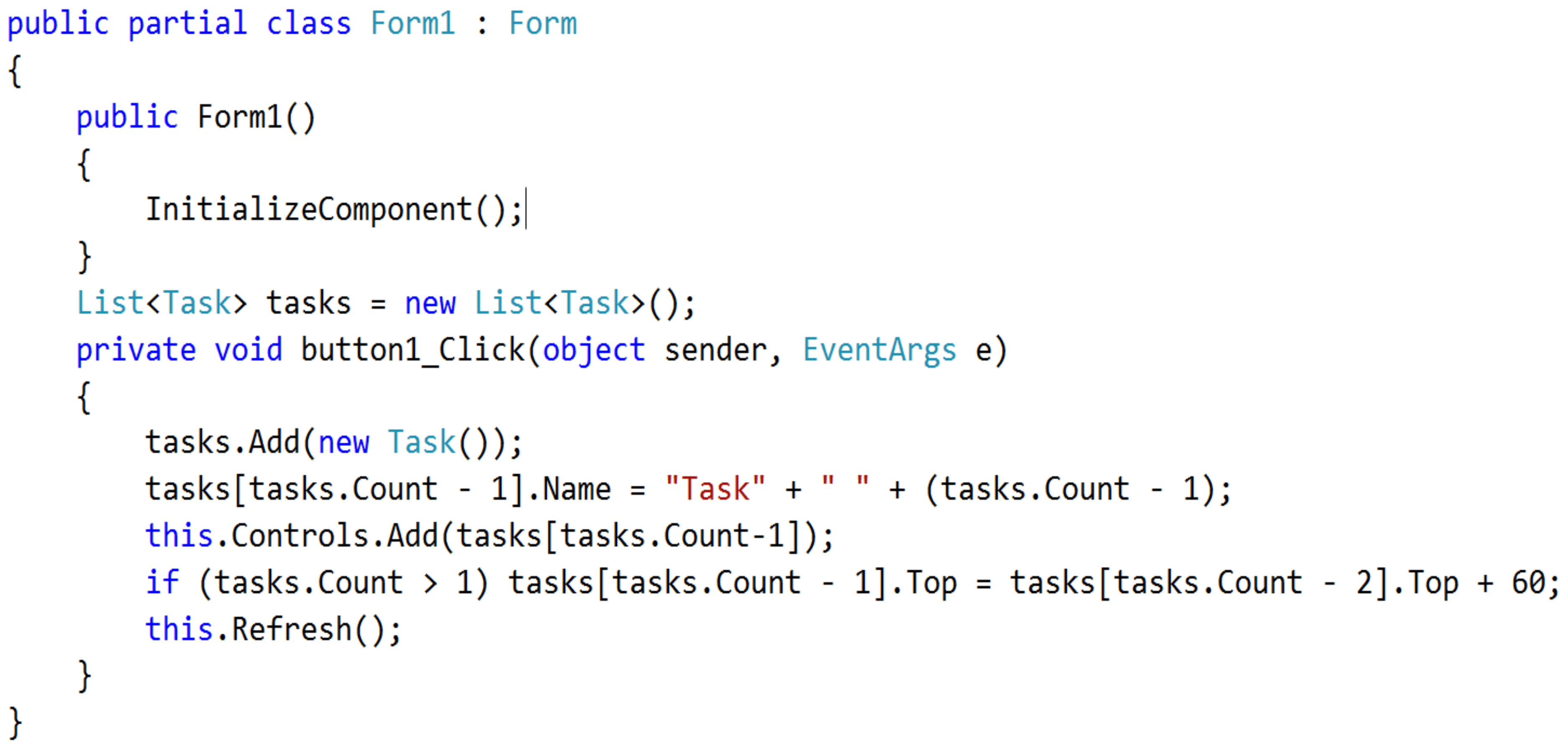


Step 3 – Implement User Control



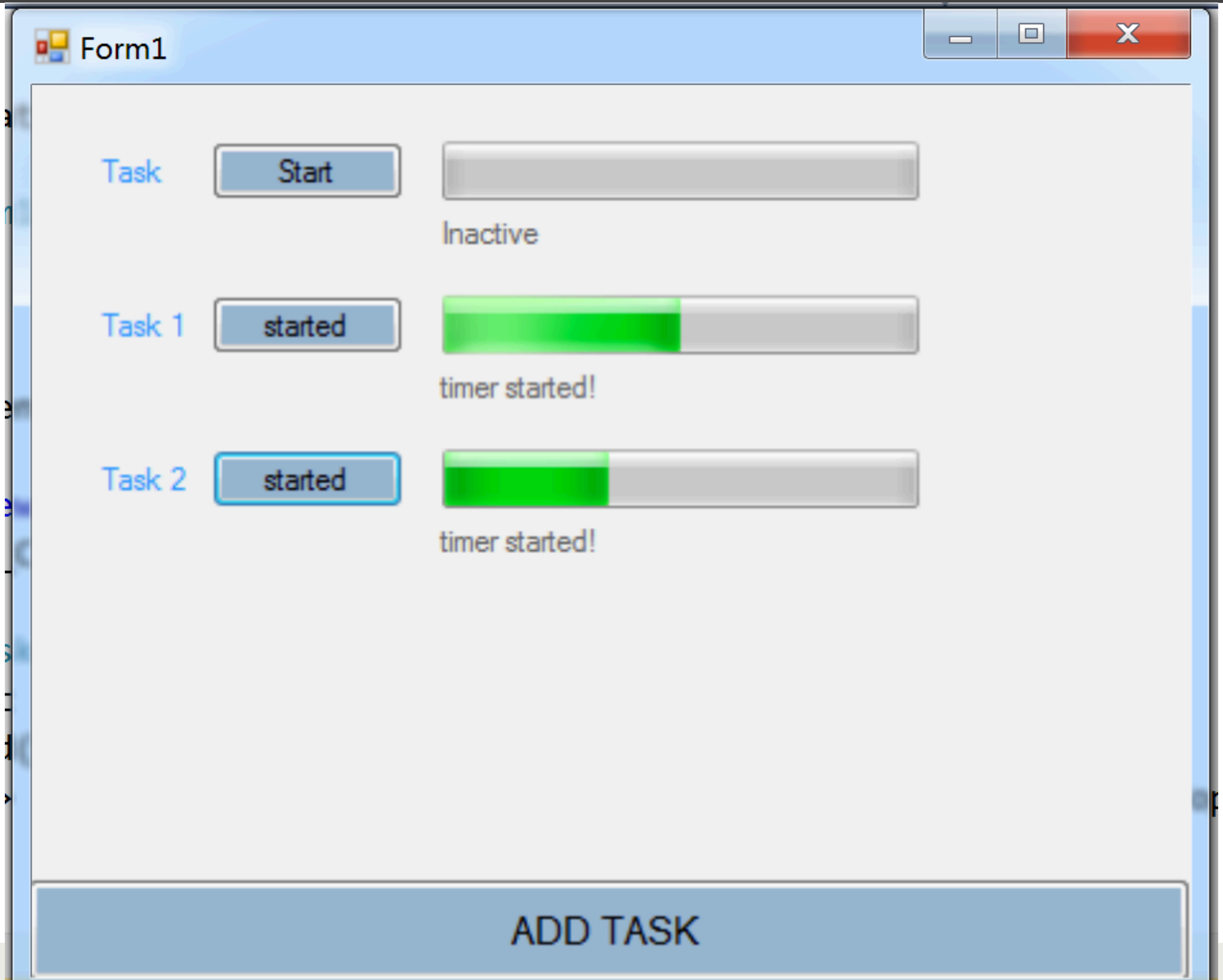
```
private void button1_Click(object sender, EventArgs e)
{
    Button b  = (Button)sender;
    MessageBox.Show(b.Parent.Name);
    b.Text = "started";
    Task t = (Task)b.Parent;
    t.TaskName.Text = t.Name;
    t.timer1.Start();

}

private void timer1_Tick(object sender, EventArgs e)
{
    Status.Text = "timer started!";
    progressBar1.Value++;
}
```

Step 4 – Use objects of the control in a Form



```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    List<Task> tasks = new List<Task>();
    private void button1_Click(object sender, EventArgs e)
    {
        tasks.Add(new Task());
        tasks[tasks.Count - 1].Name = "Task" + " " + (tasks.Count - 1);
        this.Controls.Add(tasks[tasks.Count-1]);
        if (tasks.Count > 1) tasks[tasks.Count - 1].Top = tasks[tasks.Count - 2].Top + 60;
        this.Refresh();
    }
}
```

How it looks like



Inactive

timer started!

timer started!

Conclusion

- There are many-many extra features not mentioned here.
- Please, investigate them on your own – everything you need is Visual Studio and time