3일차 프로젝트 (기업솔루션프로젝트)

실습 : 윈도우 기본 컨트롤1

예제1 - 리스트 추가

ListBox, TextBox, Button, Label 컨트롤을 이용하여 리스트를 추가하고, 리스트의 아이템을 선택하여 출력하는 애플리케이션 예제

프로젝트 이름 : **ListBox**

[1-1] 폼 디자인에 사용된 컨트롤의 주요 속성값

폼 컨트롤	속성	값
Form1	Name	Form1
	Text	리스트 추가
	FormBorderStyle	FixedSingle
	MaximizeBox	False
ListBox1	Name	lbView
TextBox1	Name	txtList
Button1	Name	btnAdd
	Text	추가
Label1	Name	lblResult
	Text	결과 :

[1-2] 문자 입출력 코드 구현

❶ 멤버 변수를 클래스 내부 상단에 추가

```
string OrgStr = ""; //결과 : 저장
```

❷ Form1 Load() 이벤트 핸들러 : 폼이 실행될 때 lblResult 컨트롤의 초기값을 멤버변수에 저장하는 작업 수행

```
OrgStr = this.lblResult.Text;
```

❸ btnEdit Click() 이벤트 핸들러 : txtList 컨트롤에 입력된 문자를 lbView 컨트롤에 추가하는 작업 수행

❹ lbView_**S**elected**I**ndexed**C**hanged() 이벤트 핸들러 : 선택된 리스트의 값을 lblResult컨트롤에 출력하는 작업을 수행

[1-3] 리스트 추가 예제 실행

리스트에 표시된 아이템을 선택하여 결과가 표시 되는 것을 확인해 보고, 텍스트 상자에 값을 입력한 뒤에 [추가] 버튼을 클릭하여 리스트 항목이 추가되는 것을 확인해 본다.

[Source]

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace ListBox
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        string OrgStr = ""; //결과 : 저장

        private void Form1_Load(object sender, EventArgs e)
        {
            OrgStr = this.lblResult.Text;
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            if (this.txtList.Text != "")
            {
                this.lbView.Items.Add(this.txtList.Text);
                this.txtList.Text = "";
            }
            else
            {
                MessageBox.Show("아이템을 입력하세요", "알림", MessageBoxButtons.OK,
                                                                MessageBoxIcon.Error);
                this.txtList.Focus();
            }
        }

        private void lbView_SelectedIndexChanged(object sender, EventArgs e)
        {
            this.lblResult.Text = OrgStr + this.lbView.SelectedItem.ToString();
        }
    }
}
```

예제2 - 타이머

Timer, TextBox, Button 컨트롤을 이용하여 타이머를 구현하는 애플리케이션 예제
프로젝트 이름 : **Timer**

[2-1] 폼 디자인에 사용된 컨트롤의 주요 속성값

폼 컨트롤	속성	값
Form1	Name	Form1
	Text	타이머
	FormBorderStyle	FixedSingle
	MaximizeBox	False
TextBox1	Name	txtNum
TextBox2	Name	txtCountDown
	ReadOnly	True
Button1	Name	btnCount
	Text	카운트 다운
Timer1	Name	Timer
	Interval	1000

[2-2] 타이머 코드 구현

❶ 멤버 개체 및 변수를 클래스 내부 상단에 추가
int CountOrgNum = 0; // 초기 카운터

❷ btnCount Click() 이벤트 핸들러 : txtNum 컨트롤에 입력된 값이 숫자인지를 판단하는 IntCheck() 메서드를 호출하여 txtNum 컨트롤에 입력된 값이 숫자이면 Timer 컨트롤의 Enabled 속성을 설정하여 타이머를 시작한다.

❸ IntCheck() 메서드 : txtNum 컨트롤에 입력된 값이 숫자인지를 판단하는 메서드

❹ Timer Tick() 이벤트 핸들러 : Timer 컨트롤의 Interval 속성 값에 따라 주기적으로 호출되어 프로시저 내의 코드를 실행

[2-3] 타이머 예제 실행

숫자를 입력하고 [카운트 다운] 버튼을 클릭하면 카운트 다운되는 숫자를 볼 수 있다.

[Source]

```
using System;
using System.Windows.Forms;

namespace Timer
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        int CountOrgNum = 0; // 초기 카운터

        private void btnCount_Click(object sender, EventArgs e)
        {
            if (IntCheck() == true)
            {
                CountOrgNum = Convert.ToInt32(this.txtNum.Text);
                this.txtNum.ReadOnly = true;
                this.Timer.Enabled = true;
            }
        }

        private bool IntCheck()
        {
            if (Information.IsNumeric(this.txtNum.Text))
                return true;
            else
            {
                MessageBox.Show("숫자를 입력하세요", "알림",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }
        }

        private void Timer_Tick(object sender, EventArgs e)
        {
            if (CountOrgNum < 1)
            {
                this.Timer.Enabled = false;
                this.txtNum.ReadOnly = false;
                this.txtNum.Text = "";
                MessageBox.Show("펑", "알림",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                CountOrgNum--;
                this.txtCountDown.Text = Convert.ToString(CountOrgNum);
            }
        }
    }
}
```

예제3 - 상태 진행

Progressbar, Timer, Label, Button 컨트롤을 이용하여 상태 진행을 나타내는 애플리케이션 예제

프로젝트 이름 : **Progress**

[3-1] 폼 디자인에 사용된 컨트롤의 주요 속성값

폼 컨트롤	속성	값
Form1	Name	Form1
	Text	상태 진행
	FormBorderStyle	FixedSingle
	MaximizeBox	False
ProgressBar1	Name	pbStatus
Label1	Name	lblStatus
	Text	상태 :
Button1	Name	btnRun
	Text	진행
Timer1	Name	Timer
	Interval	100

[3-2] 상태 진행 코드 구현

❶ 멤버 변수를 클래스 내부 상단에 추가

```
int Num = 0; //진행 숫자
string OrgStr = ""; //결과 : 저장
```

❷ **Form1 Load() 이벤트 핸들러** : 폼이 실행되면 lblStatus 값을 멤버 변수에 저장하는 작업 수행

❸ **btnRun Click 이벤트 핸들러** : Timer 컨트롤의 Enabled 속성을 true로 설정하면서 활성화하여 pbStatus 컨트롤에 상태 표시 진행을 수행

❹ **Timer Tick() 이벤트 핸들러** : Timer 컨트롤이 활성화 되었을 때 Timer 컨트롤의 Interval 속성값에 따라 주기적으로 호출되어 pbStatus 값을 수정하면서 상태 진행을 표시

[3-3] 상태 진행 예제 실행

[진행] 버튼을 클릭하면 상태 진행이 시작되는 것을 확인할 수 있음

[Source]

```
using System;
using System.Windows.Forms;
namespace Progress
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
```

```
        int Num = 0; //진행 숫자
        string OrgStr = ""; //결과 : 저장

        private void Form1_Load(object sender, EventArgs e)
        {
            OrgStr = this.lblStatus.Text;
        }

        private void btnRun_Click(object sender, EventArgs e)
        {
            this.Timer.Enabled = true;
        }

        private void Timer_Tick(object sender, EventArgs e)
        {
            Num++;
            if (Num > 100)
            {
                this.Timer.Enabled = false;
                return;
            }
            this.pbStatus.Value = Num;
            this.lblStatus.Text = OrgStr + Num.ToString() + "%";
        }
    }
}
```