Final Report

객체지향윈도우즈프로그래밍

담 당 : 오 세 만 교수님

학 과 : 컴퓨터 공학과

학 번 : 2017112154

성 명 : 응웬 딩 흐엉

제출일 : 2018 년 12 월 12일

dongguk UNIVERSITY

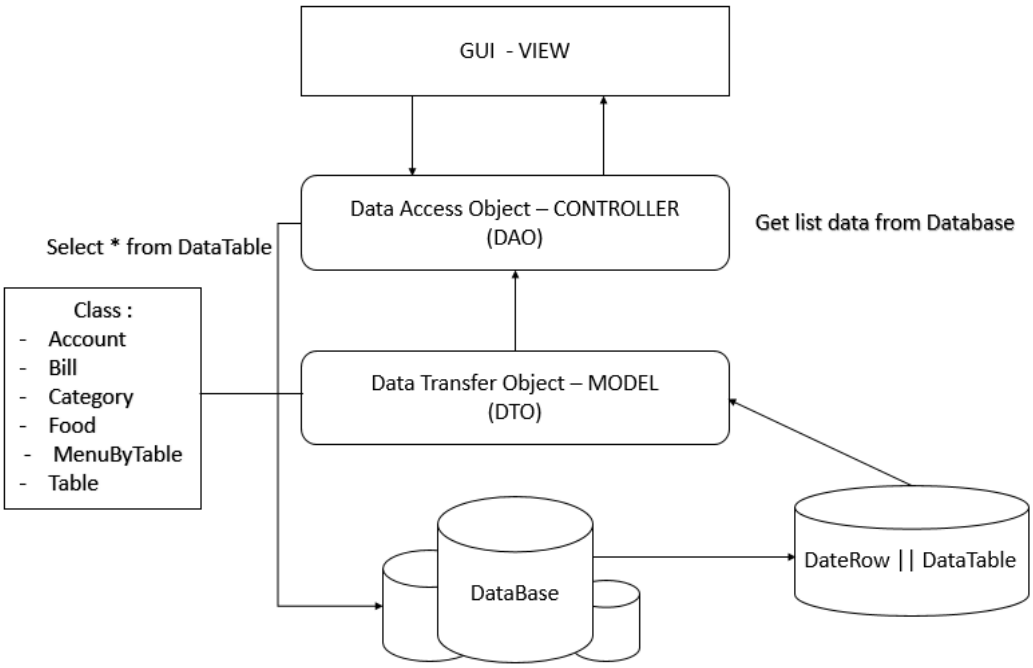
I,인사말

- 이번 학기에 과목을 신청했을때 '객체지향윈도우즈프로그래밍'이라는 과목을 신청할까 말까 많이 고미했는데 생각하다가 신청하게 되었다. 그릭고 강의를 들은 첫날에 모드 학생들이 대부분 3,4 학년 인것을 보게 되어서 많이 걱정되었다. 많은 고민이나 걱정을 불구하고 열심히 하기로 마음을 먹었다.
- 기말 전에 한 학기를 마무리로 하기 위한 프로젝트를 하나 내주셨다. 한 학기동안 배운 것을 정리하다는 기회처럼 생각해서 할 수 있는 만큼 하고자 했다. 더 많은 기능이나 이벤트를 개발하고 싶은데 보인이 프로그램 능력에 아직 부족한 점이 많이 있어서 많이 하지 못했다.
- 선택된 주제 : 외식업 포스 (POS) – 프로그램 이름: 베트남 식당 간단한 포스.

II,문제 분석

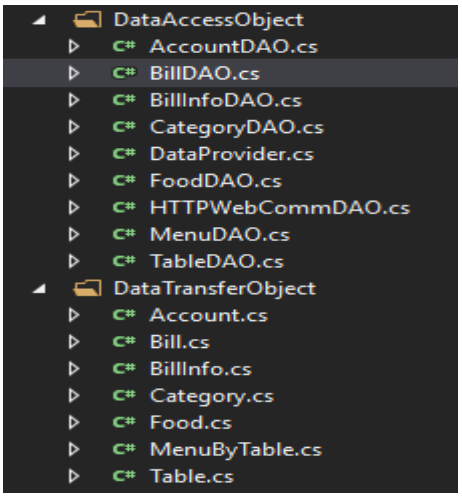
1, 프로그램의 모델

- 앱플리케이션의 모델이 많이 있는데 본인이 알고 있는 MVC 라는 모델을 선택해서 프로그램 내에 구성되어 있다.
- Model 은 데이터 구조라는 표현한다. 일반저으로 모델 클래스는 데이터를 추출,입력 등의 함수를 구성되어 있다. 모델 정보 읽어오거나 수정하는 로직포함한다. 데이터베이스와 통신하는 곳이다.
- View: 사용자에게 보여질 부분은 GUI 라는 표현한다. 즉 프로그램과 사용자과 커뮤니케이션을 하는 역할을 받는 부분이다.
- Control: 모델이 어떤 작업을 해야하는지 알아낸다.모델의 정보 수정,뷰에게 넘겨줄 새로운 모델을 작성한다.

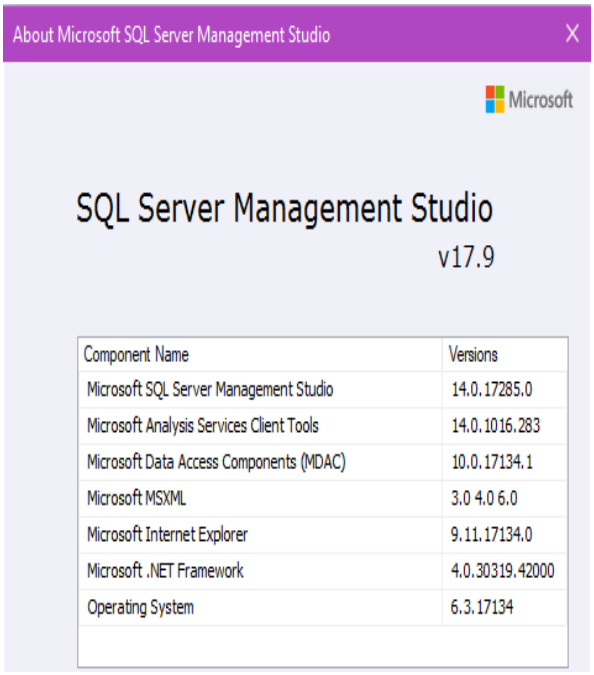


2, 프로그램 활용 기술

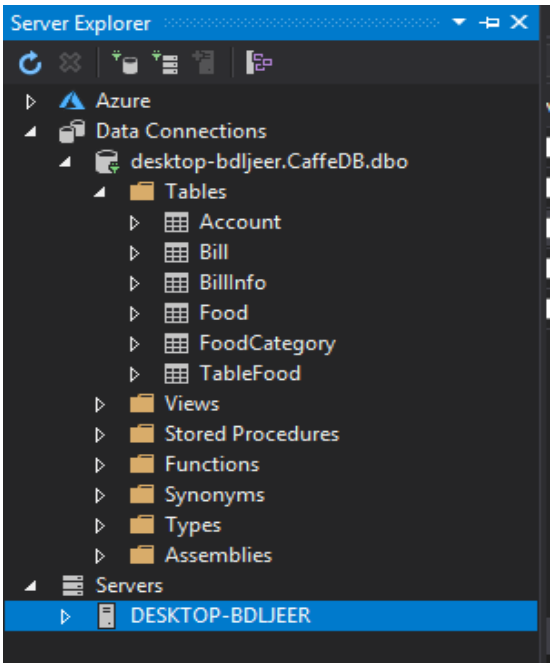
- 여러 객체를 만들어서 쓰는 것보다 인스턴스를 하나만 만들어서 하는 것은 프로그램 실행시에 효과성을 향상시키 수 있는 기술은 Design Pattert Singleton(싱글턴 패턴)를 선택했다. 즉 전체 프로그램에서 각 클래스는 인스턴스를 하마만 만들어서 사용하기 위한 패턴이다.
- 하나의 인스턴스만을 유지하기 위해서 인스턴스 생성에 틀별한 제약을 걸어둬야 한다.new 를 실행할 수 없도록 생성자에 private 접급 제어자를 지정하고, 유일한 단일 객체를 반환할 수 있도록 정적 메소드를 지원해야 한다. 또한 유일한 달일 객체를 참조할 정적 참조변수가 필요한다.
- 활용한 기술은 Design Pattert Singleton 를 전체 구성되어 있는 클래스들은 다음과 같다.



3,프로그램의 활용 데이스베이스

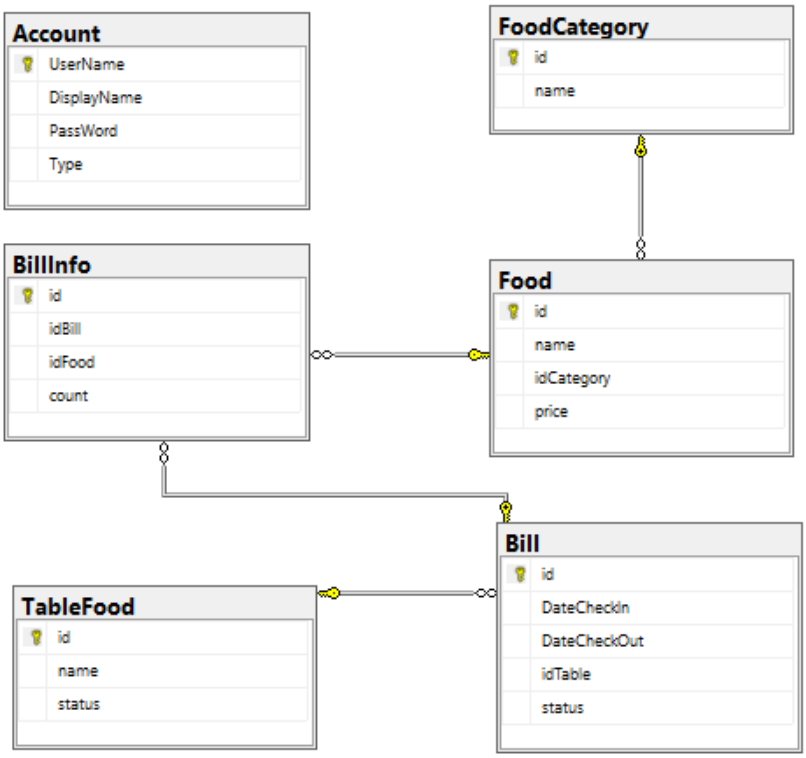


본인이 사용하고 있는 SQL 버전



데이스 베이스 과 Visual Stdio 를 연결 확인

- C# 앱플리케이션에서 많이 쓰고 있는 데이스베이스 SQL 를 활용했다.**(CaffeDB.sql 파일).** 데이터 베이스를 생선하기 위해서 먼저 **CaffeDB.sql 파일 실행하고** VisualStdio 에서 연결 경로를 확인하여 연결을 되다면 프로그램을 실행하면 된다.
- 여러 가지 업무에 공동으로 필요한 데이터를 유기적으로 결합아혀 테이블 6 개 구성되었있음



- 기준으로 데이스베시스에 저장되어 있는 값들은 다음과 같다.

+ Account 테이블

	UserName	DisplayName	PassWord	Type
1	admin	HOPE	1234	1
2	staff	STAFF	1234	0

+ TableFood 테이블

	id	name	status
1	1	TABLE 1	DONT HAVE
2	2	TABLE 2	DONT HAVE
3	3	TABLE 3	DONT HAVE
4	4	TABLE 4	DONT HAVE
5	5	TABLE 5	DONT HAVE
6	6	TABLE 6	DONT HAVE
7	7	TABLE 7	DONT HAVE
8	8	TABLE 8	DONT HAVE
9	9	TABLE 9	DONT HAVE

+ Category 테이블

	id	name
1	1	음식
2	2	음류수
3	3	맥주

+ Food 테이블

	id	name	idCategory	price
1	1	쇠국기쌀국수	1	9000
2	2	쇠고기복음밥	1	8000
3	3	해물복음밥	1	8000
4	4	분짜	1	9500
5	5	스프링롤	1	6500
6	6	사이다	2	2000
7	7	코카콜나	2	2000
8	8	베트남 커피	2	4000
9	9	아메리카노	2	3000
10	1	베트남 맥주	3	5000
11	1	한국 맥주	3	5000
12	1	유럽 맥주	3	5000

+ Bill & BillInfo 테이블 : 주문한거 없어서 아무것도 저장하지 많은 상태

	id	DateCheckIn	DateCheckOut	idTable	status

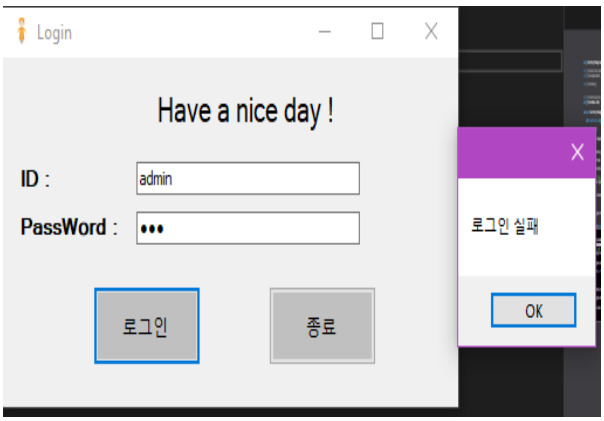
	id	idBill	idFood	count

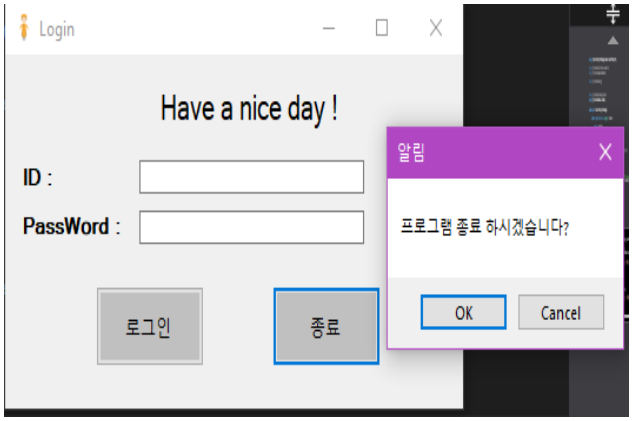
III,프로그램 기능과 실행 화면

- 각 화면의 구성되어 있는 기능과 실행 화면 결과는 다름과 같이 분석해서 설명하고자 한다.

1,로그인

- 계정 정보들을 데이터 베이스내서 저장되어 있는 것은 사용자 입력한 정보들과 비교하여 맞는지 안 맞는지 판단할 수 있다. 만약에 맞다면 본 프로그램 화면을 이동하게 될것이고 반면에 맞지 않으면 '로그인 실패' 알려줄 도록 구성되고 있다.
- '종료'라는버튼을 누를거나 로그인 장을 단을때에는 '프로그램 종료 하시겠습니까?'냐고 물어보고 선택함에 따라서 프로그램을 단음 단계를 실행하는 것이다. 실행 결과는 다음과 같다.





2,본 프로그램 화면

- 현재 데이터 베이스에서 저장되어 있는 계정 정보들이다. 보이는 것처럼 계정의 종료에 따라서 프로그램의 기능에 제한이 있는 것이다.

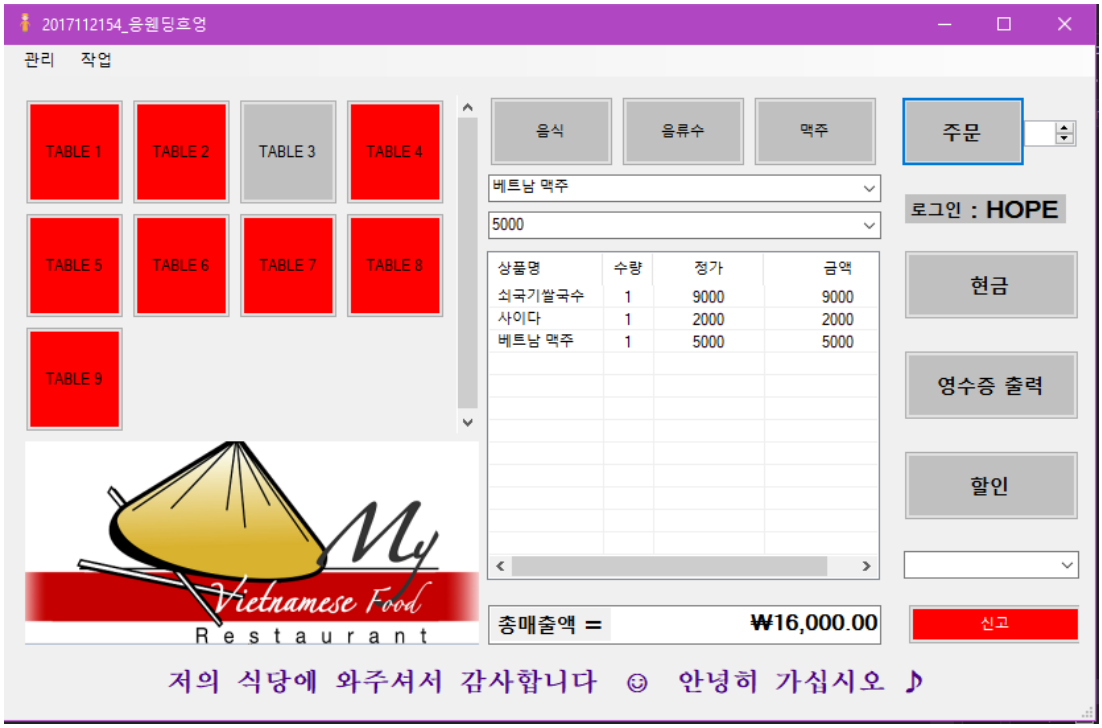
아이디	표시 이름	비밀번호	종료	권한
admin	HOPE	1234	관리자(1)	관리 기능 가능
staff	STAFF	1234	직원(0)	관리 기능 불가능

- 로그인 장에서 계정을 맞게 입력하면 본 화면으로 이동하는 것이다.(admin 나 staff 입력함에 따라서 이동한 화면이 달라지는 점이 있음).

- 일반적으로 앱 플리케이션 구성되는 것처럼 기능이나 이벤트를 처리하기 위해서 이품에서도 많은 윈도우 컨트롤을 존재하고 있다. 대표적인 말하면 테이블이나 음식 종료를 나타나기 위한 컨트롤 FlowLayoutPanel 가 2 개 구성되고 있다. 또한 계산서의 내용들을 출력하기 위한 ListView 도 있다. 그리고 Winfroms Application 에서 많이 쓰고 있는 컨트롤들은 Button, Label,

ComboxBox, Panel, NumericUpDown, MenuStrip, notifyIcon 또한 contextMenuStrip 등 구성 존재하는 것이다.

- 현재 식당에서 쓰고 있는 POS 처럼 이품에서 중요성이 높은 기능이 반드시 주문 기능,계산 기능이 처리되도록 구성되고 있는데 우선 주문 기능인 경우에는 관리는 원하는 테이블에서 원하는 음식이나 개수을 선택해서 '주문'이라는 버튼을 클릭하면 주문 하게 되었고 영수증에서 주문이 들어 있는 음식의 정보들을 옆에 FlowLayoutPanel 에서 바로 확인 할 수 있는 것이다.



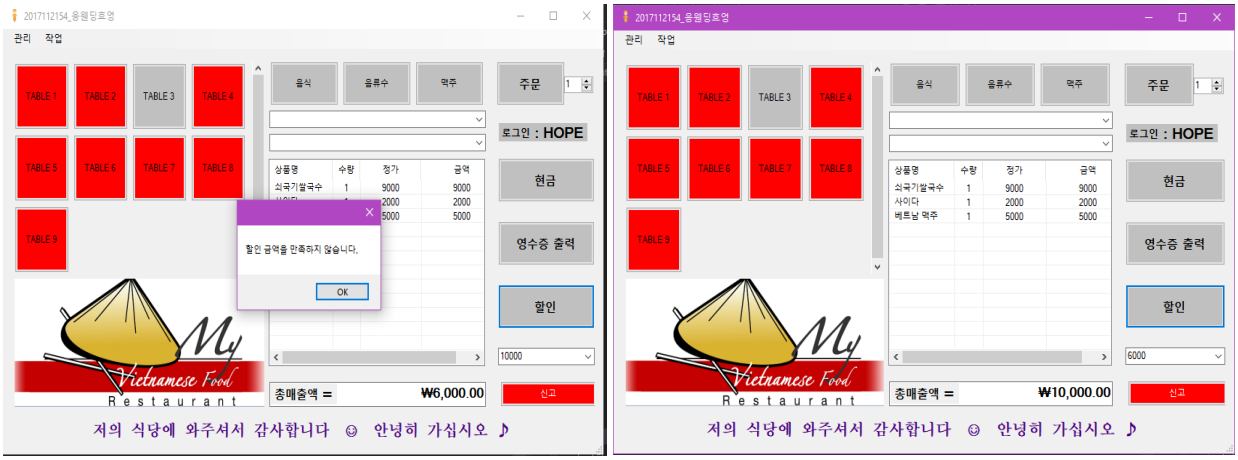
- 위에 결과 화면을 보이는 것처럼 테이블 상태(손님이 있음)과 해당하는 테이블에서 계산서를 확인 할 수 있는 것이다. 만약에 이 테이블에서 같은 음식을 추가하거나 NumericUpDown 에서 음수를 입력하면 계산서에서 음식 개수가 맞게 출력하도록 처리되고 있다.그리고 음식을 추가할때 마다 총매출액도 맞게 출력되도록 하는 것이다.

- 다른 이벤트에 대한 말하면 현금, 영수증 출력, 할인이라는 버튼이 구성되어 있는데 이 버튼들을 클릭 이벤트도 처리되고 있다. 현금이라는 버튼을 클릭하면 해당하는 테이블만 상태를 변환하고, 계산서을 없어지는 것이다. 실행하기전에 메시지도 출력해서 선택함에 따라서 프로그램을 실행하는 것이다.

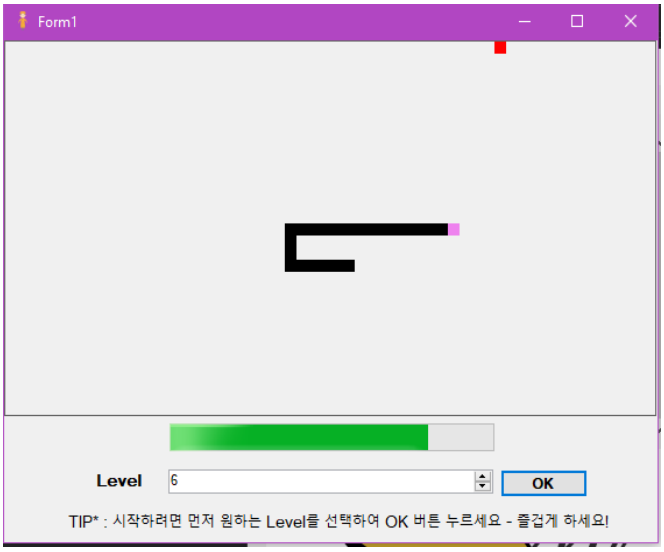
- 이미 음식을 주문한 테이블을 현금(계산)을 클릭하여 실행한 다음에 테이블 상태과 계산서 없어지는 것을 나타나는 화면 프로그램이다. 또한 주문을 안 하는 테이블에서 아무것도 안 한다.

- 그리고 영수증이라는 버튼을 클릭하는 경우에는 해당하는 계산서를 출력해서 보여즐 수 있도록 처리되고 있다. 일반적인 식당에서 출력하는 것처럼 계산서 내용이랑 많은 정보들을 표시하는 것이다. 인쇄 기능도 포함하여 처리되고 있는데 인쇄를 할때 계산서의 내용들을 인쇄될 것이다.

- 또한 할인이라는 버튼 누르면 원하는 할인을 금액을 선택하여 총매출액을 맞게 출력되도록 처리고 있은데 만약에 금액을 선택 안 하거나 할인 금액이 총매출액보다 크면 알림 메시지를 출력되도록 한다.

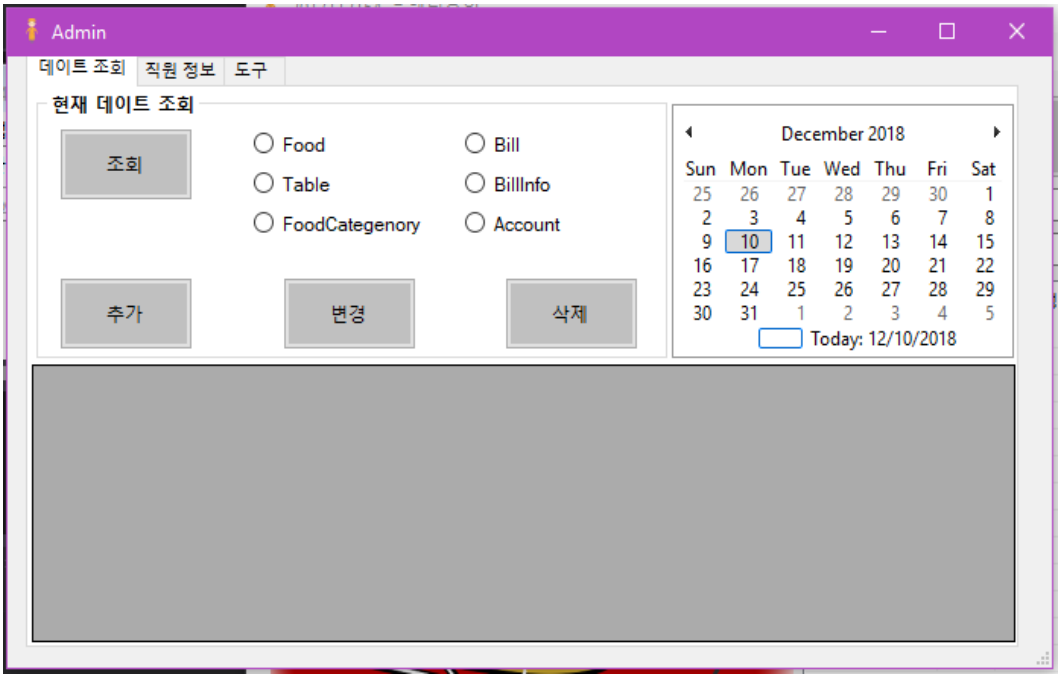


- 위에 발생하는 각 이벤트들을 대부분 계산서과 관련된 기능인데 외에 MenuStrip 에서 2 개 항목이 구성되어 있다. 먼저 '작업' 항목에서 선택하면 '로그인'과 '여가'라는 ToopStrip 가 있다. '로그인'인 경우에는 클릭하면 위에 처럼 당연히 Login 의 품을 호출되는 것이다.그것 아니라 '여가'라는 것을 클릭하면 손님이 없을때 지루지 않게 하기 위해서 실습할때 했던 Snake 게임 추가되어 있다. 실행 결과 화면은 다음과 같다.

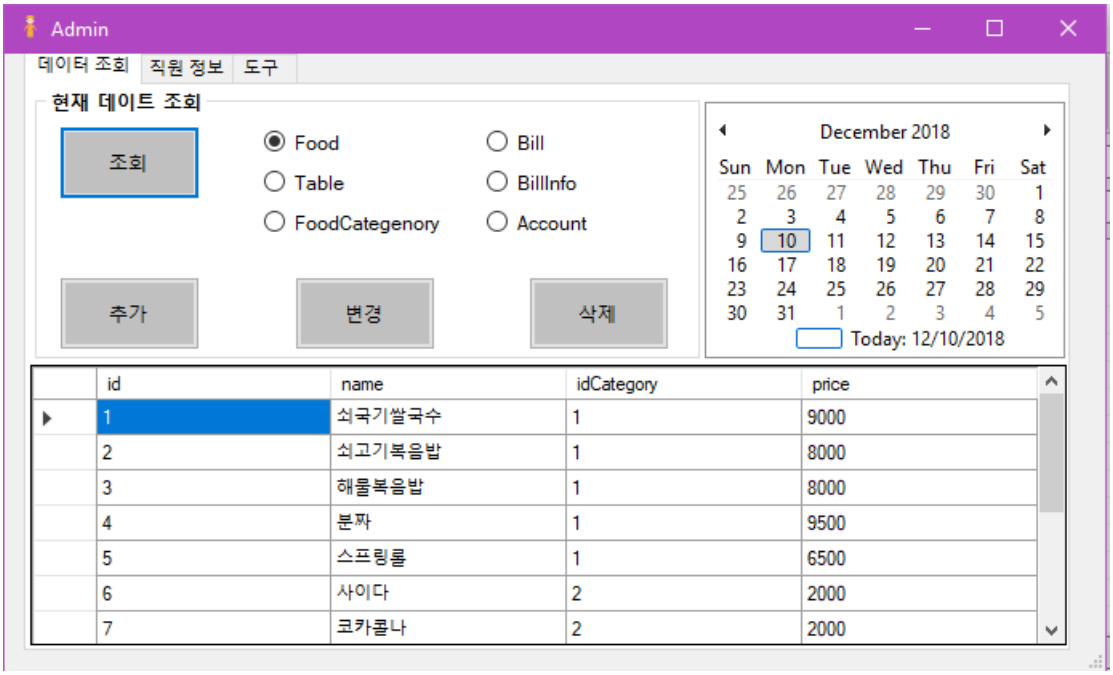


- 또한 MenuStrip 에서 첫번째 항목을 들어가면 '사업 관리'라는 ToopStrip 가 있는데 이것을 누르면 다른 폼을 이동되도록 한다. 하지만 이 기능은 위에 말하는 것처럼 관리자(Admin)인 계정만 되는 것이다. 사업 ToopStrip 를 클릭하면 출력된 화면 결과는 다음과 같다.

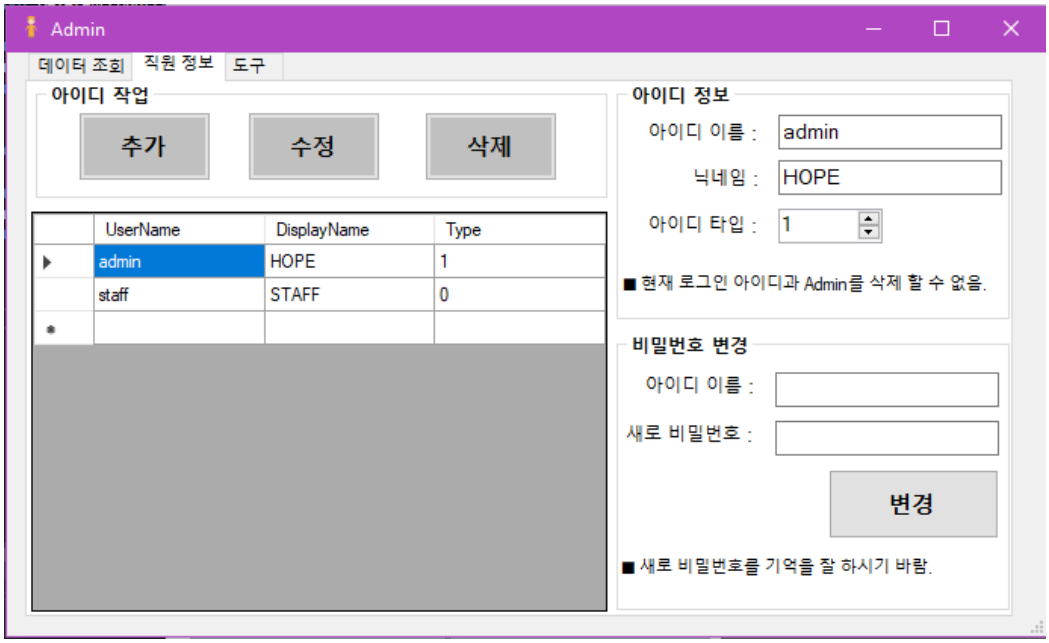
3,관리자 폼 화면



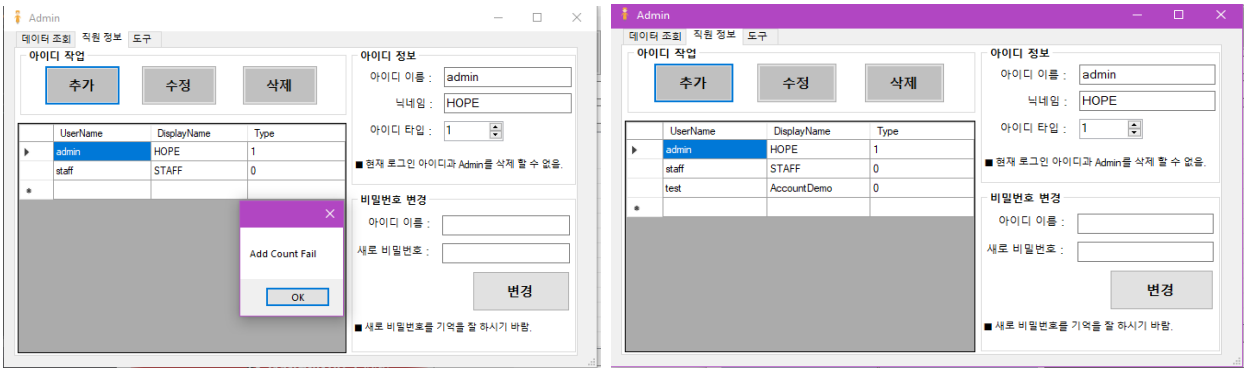
- 이폼에서 다른 폼보다 특별한 컨트롤은 TabControl 가 구성 되어 있는데 첫번째 항목을 '데이터조회'라는 태그 페이지이다 .위에 보이는 것처럼 태크 페이지의 기능은 데이터 베이스에서 저장되어 있는 데이터들을 조회할 수 있도록 한다. 따라서 원하는 항목을 하나만 선택하여 조회 버튼을 누르면 밑에 DataGridView 에서 데이터 베이스에서 저장되어 있는 데이터를 가져와서 출력되도록 처리되고 있다. 만약에 선택된 항목이 없으면 알림 메시지를 출력하게 된다. 여기서 '추가','변경','삭제'라는 기능을 아직 처리를 못하는 것이다. 밑에 출력되는 것은 'Food'의 데이터를 조회의 결과과 같다.



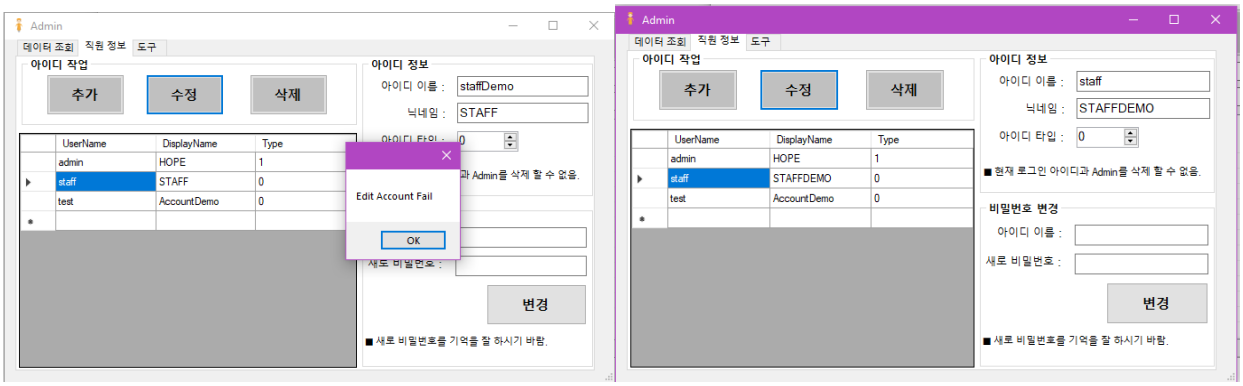
- TabControl 의 두번째 태그를 들어가면 '직원정보'를 처리되는 기능이다.



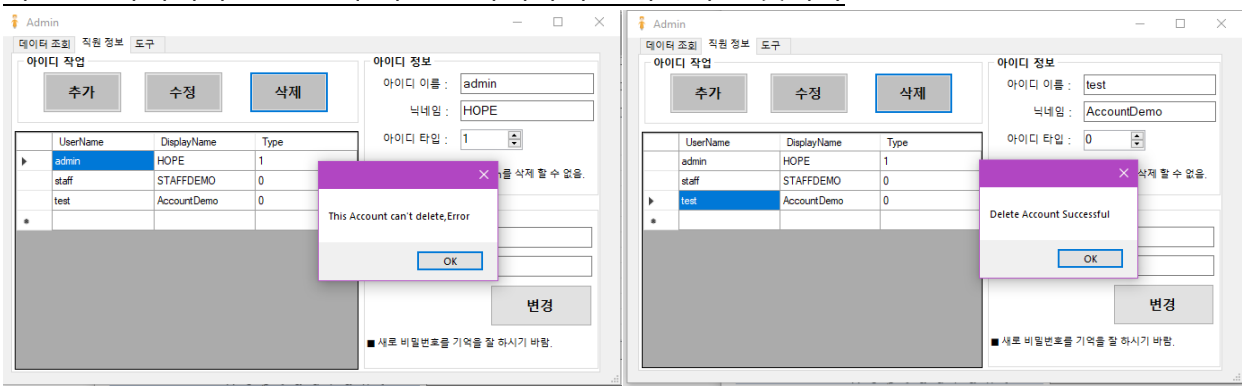
- 이폼에서 보여준 것을 바로 데이터베이스에서 저장되어 있는 계정에 대한 정보이다. 아이디 작업에대한 것을 설명하면 첫번째 추가 기능은 옆에 아이디 정보를 각 정보들을 입력하여 버튼 추가를 클릭하면 자동적으로 아이디를 로드되고 있으니 옆에 DataGridView 에서 바로 결과를 확인 할 수 있는 것이다. 그리고 비밀버번호를 기준 으로 '0'로 설정되는 것이다. 써 있는 알고리즘 때문에 이 기능의 주의해야 할점이 이름이 같은 아이디를 추가하기 불가능하는 것이다.



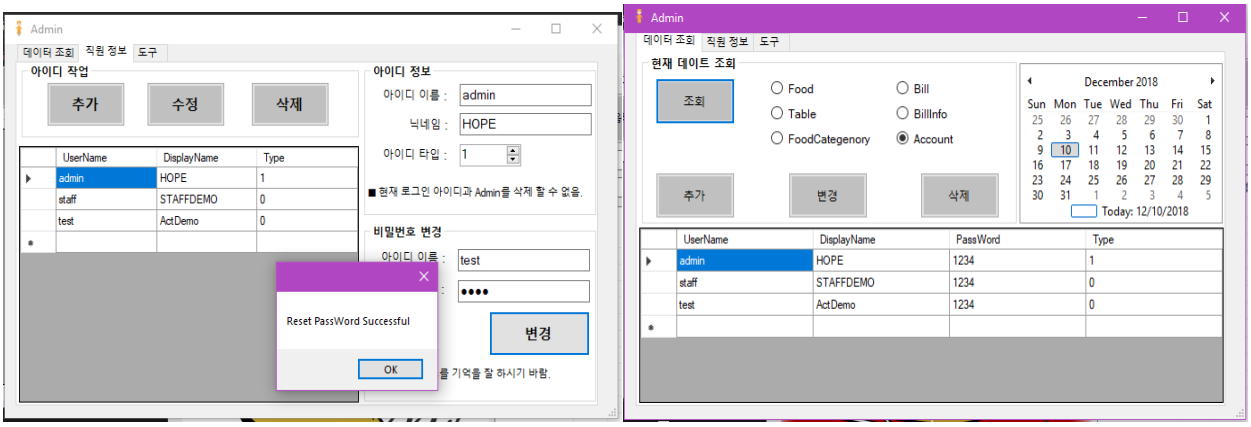
- 추가 기능과 마찬가지로 아이디의 정보를 수정하려면 아이디 정보에서 원하는 정보을 입력해서 수정 버튼을 누르면 아이디의 정보를 수정을 되도록 처리되고 있다. 하지만 아이디의 이름에 건거하여 데이트 베이스에서 저장되어 있는 아이디를 찾는 알고리즘을 써 있기 때문에 아이디의 이름을 수정하면 안 되는 것이다.



- 수정 버튼의 옆에 삭제라는 버튼이 있는데 이름 그대로 이번트을 클릭하면 해당하는 아이디를 삭제할 수 있도록 처리하고 있다. 여기서 주의 할점은 써있는 알고리즘에 따라서 현재 로그인 하는중 아이디과 Admin 의 계정을 삭제하기 불가능하는 것이다.

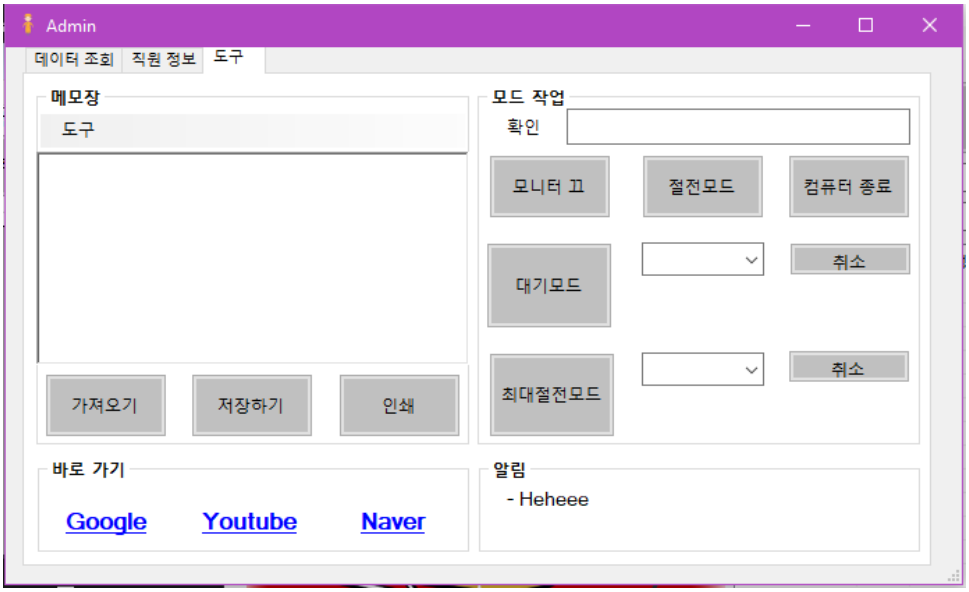


- 이폼에서 구성되어 있는 기능이 하나 더 있는데 입력된 이름에 근거하여 새로 원하는 비밀번호를 입력하면 '변경'이라는 버튼을 누르면 비밀번호를 변경해준 기능이다. 번경 버튼을 클릭한후에 데이터 조회장에서 결과를 확인 할 수 있는 것이다.



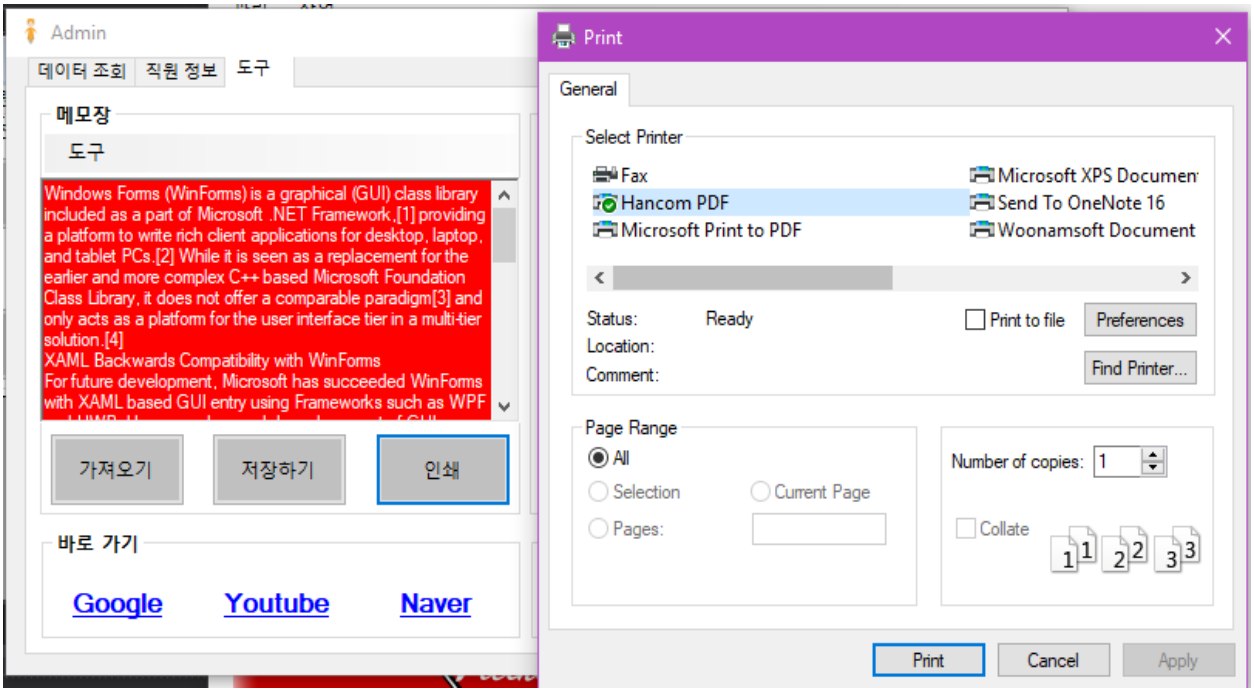
* 추가할때 비밀번호를 기준 0 으로 하는데 변경한후에 1234 가 된것이다.

- 이어서 TabControl 에서 3 번째 항목을 들어가는 모드나 파일에 대한 작업들이 구성되어 있다.

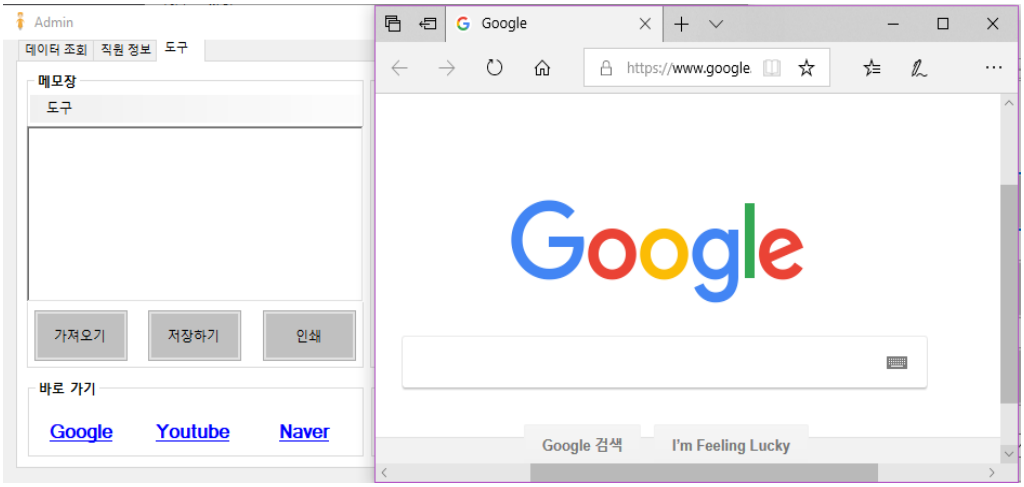


- 위에 화면을 보이는 것처럼 각 버튼 컨트롤에 따라서 이베튼를 처리되고 있는 데 버튼이나 LinkLabel 의 이름은 그대로 기능의 이름과 동일하니 실행을 해보면 결과를 자세하게 확인 할 수 있는 것이다.

- 일반 메모장의 그룹에서 파일에대한 작업은 파일 가져오기, 파일 저장 및 파일의 내용을 인쇄되도록 구성되어 있다. 그리고 파일의 내용들은 편리하게 쓰기 위하여 도구 메뉴에서 글꼴, 색깔 및 배경에 대한 변환 할 수 있도록 이벤트를 처리되어 있다. 이들은 파일을 열어서 내용들은 RichTextBox 에서 출력하고 도구 항목들을 선택함에 따라서 RichTextBox 에 있는 내용들을 변환하는 것이다. 원하는 기능을 선택해서 실행 결과은 다음음 같다.

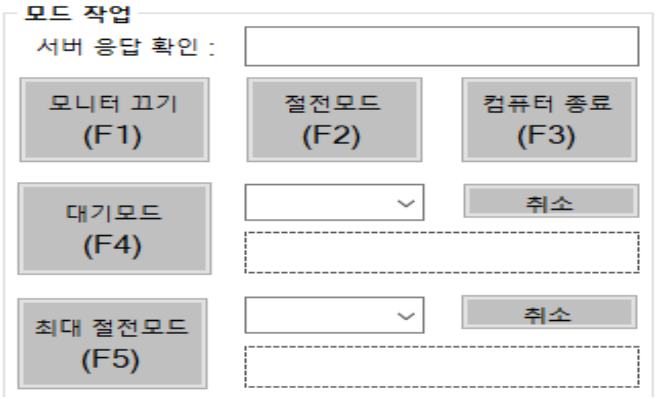


- 또한 사용자를 프로그램을 실행하다가 Google,Naver 과 같은 웹 사이트를 바로 이동하려면 '바로 가기' 그룹에서 LinkLabel 가 3 개가 구성되어 있는데 Google 링크를 선택하여 실행 결과는 다음과 같다.

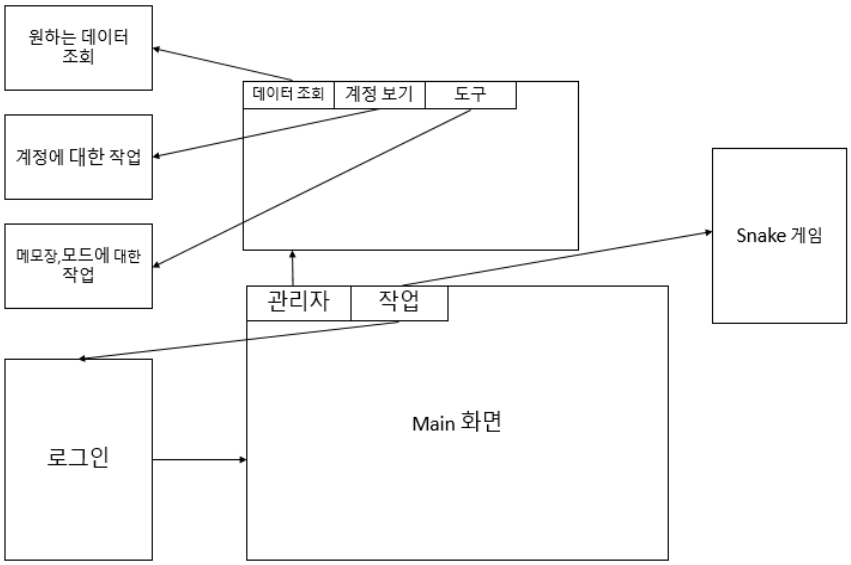


- 마직막으로 구성되어 있느 작업들은 바로 모드에 대한 작업이다. 이 그룹에서 프로젝트에서 필수적으로 5 개 모드 기능이 구성이 되어 있는데 모니터 끄기, 절전모드,컴퓨터 종료,대기 모드 및 최대절전모드이다. 각 작업들을 nircmd 를 이용해서 처리되고 있고 선택된 기능에 따라서 서버에서 기록되도록 처리되고 있다. 그리고 서버와 의 통신에 따라 로그정보는 위에 있는 테스박스에서 확인할 수 있는 것이다. 대기모드나 최대절전모드와 같은 경우에는 먼저 원하는 시간을 선택해서 해당하는 시간을 지나면 클릭한 이베트를 실행되도록 처리되고 있다. 시간을 선택하지 않으면 경고 메시지를 출력되는 것이다. 각 기능을 실행하면 제세한 결과를 바로 확인할 수 있는 것이다.

- 지금까지 설명하는 부분이 폼에서 구성되어 있는 부분인데 이 외에 프로그램 내에서 notifyICon 기능과 키도드 후킹 인벤트도 처리되고 있다. notifyIcon 인 경우에는 contextMenu 통해서 '화면보기','로그인','게임 실행','종료' 항목들을 구성되어 있다. 각 항목들을 클릭하면 위에 출력되는 결과와 동일하는 것이다. 종료 버튼인 경우에는 프로그램을 종료되도록 하는 기능이다. 키보드 후킹 이벤트 처리되는 경우에는 관리자 폼에 도구 태그 패이지만 작용 가능하는 것이다. 처리되어 있는 키보드으 이벤트 는 다음과 같다.



- 요약하면 앱 스토리보드는 다음과 같다.



IV,소감

- 프로그램을 테스트 많이 해봤는데 각 기능이나 이벤트 처리는 아직 에러가 존재할 수 있는 것이다.앞으로 더 많은 기등을 확장하여 추가할 예정이다.

- 보고서에 나와있는 내용들은 이해가 안 되는 부분이 있을 수 있으니 이해을 해주시면 감사하겠다.

- 한 학기동안 실습을 잘 가르쳐 주시느고 많으셨다.

감사합니다

V,소스코드

- 프로그램 내에 클래스가 여러 개 있는데 중요한 2 클래스 소스 코드를 다름과 같다.

1,fMain.class

```
using CShareProjectWinApp.DataAccessObject;
using CShareProjectWinApp.DataTransferObject;
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.ListViewItem;
namespace CShareProjectWinApp
{
    public partial class fMain : Form
    {
        private Account loginAccount; //현재 로그인 계정을 저장하는 변수
        public Account LoginAccount {
            get { return loginAccount; }
            set
            {
                this.loginAccount = value;
                ChangAccount(loginAccount.Type);
            }
        }
        public fMain(Account _account)
        {
            InitializeComponent();
            LoadTable();
            LoadCategory();
            this.LoginAccount = _account;
        }
        #region Load Table,Catgory,Food - Show Bill,Food,Price Access
        private void LoadCategory() //데이터베이스에서 저장되어 있는 음식 종류 함목들을 로드
        {
            List<Category> category = CategoryDAO.Instance.GetListCategory(); //로드된
데이터를 리스트에 삽입
            int btWidth = 94;
            int btHeight = 53;
            foreach (Category item in category) //각 항목들을 버튼에서 표시하여
FlowLayoutPanel에서 삽입
            {
                Button btn = new Button() { Width = btWidth, Height = btHeight };
                btn.Text = item.Name;
                btn.Tag = item; //각 버튼에 Tag값은 설정(Category 객체)
                btn.BackColor = Color.Silver;
                btn.Click += btn_ClickCategory;
                flpCategory.Controls.Add(btn);
            }
        }
private void LoadFoodListByCategory(int _idCategory)//Category의 Id에 근거하여 데이터를 꺼내서
출력하는 메소드
        {
            List<Food> listFood = FoodDAO.Instance.GetFoodByCategoryID(_idCategory);
            cbFood.DataSource = listFood; //Combobox에서 Name라는 property 출력함
            cbFood.DisplayMember = "Name";
        }
```

```
        void btn_ClickCategory(object sender, EventArgs e) //각 Category 버튼을 클릭의 이벤트
처리
        {
            int idCategory = ((sender as Button).Tag as Category).ID;
            LoadFoodListByCategory(idCategory);
        }
        private void LoadPriceListByFoodID(int id)// 전달된 idFood에 근거하여 가격을 출력할다
        {
            List<Food> listFood = FoodDAO.Instance.GetPricebyFoodID(id);
            cbPrice.DataSource = listFood;//Combobox에서 Price라는 property 출력함
            cbPrice.DisplayMember = "Price";
        }
        private void cbFood_SelectedIndexChanged(object sender, EventArgs e) //Combobox
Food에서 항목들을 선택하는 이벤트 처리
        {
            int id;
            ComboBox cb = sender as ComboBox;
            if (cb.SelectedItem == null)
                return;
            Food selected = cb.SelectedItem as Food;
            id = selected.ID;
            LoadPriceListByFoodID(id);
        }

//=============================================================================================
================
        private void LoadTable() //데이터베이스에서 저장되어 있는 테이블 데이터 로드
        {
            flpTable.Controls.Clear(); //로드할때마다 FlowLayoutPanel removes all items
            List<Table> tableList = TableDAO.Instance.LoadTableList(); //데이터 베이스테서
데이터를 얻어서 리스트를 저장함
            int btWidth = 75;
            int btHeight = 80;
            foreach(Table item in tableList)
            {
                Button btn = new Button() { Width = btWidth, Height = btHeight };
                btn.Text = item.Name + Environment.NewLine;
                btn.Click += btn_Click;
                btn.Tag = item; //태스 값을 설정
                btn.BackColor = Color.Silver;
                if (item.Status.Equals("DONT HAVE")) //테이블 상태가 주문한 항목이 없으면
배경을 변경
                {
                    btn.BackColor = Color.Red;
                }
                flpTable.Controls.Add(btn); //각 버튼을 FlowLayoutPanel에서 삽입
            }
        }
  float totalPrice;
        void Show_Bill(int _idTable)
        {
            lsvBill.Items.Clear(); //로드할때마다 ListVew removes  all items
            //List<BillInfo> listBillInfo =
BillInfoDAO.Instance.GetListBillInfo(BillDAO.Instance.GetUnCheckBillIdbyTableID(_idTable));
```

```
            List<MenuByTable> listBillInfo = MenuDAO.Instance.GetListMenuByTable(_idTable);
            totalPrice = 0; //주문한 것을 총 금액 저장하는 변수
            foreach (MenuByTable item in listBillInfo) //각 주문한 항목들을 ListView에서
출력함
            {
                ListViewItem lsvItem = new ListViewItem(item.FoodName.ToString());
                lsvItem.SubItems.Add(item.Count.ToString());
                lsvItem.SubItems.Add(item.Price.ToString());
                lsvItem.SubItems.Add(item.TotalPrince.ToString());
                totalPrice += item.TotalPrince;
                lsvBill.Items.Add(lsvItem);
            }
            CultureInfo culture = new CultureInfo("euc-KR");
            txttotalPrice.Text = totalPrice.ToString("c", culture); //금액을 한국의 돈
style로 display
        }
        void btn_Click(object sender,EventArgs e) //테이블 버튼 클릭하는 이벤트 처리
        {
            int idTable = ((sender as Button).Tag as Table).Id;
            lsvBill.Tag = (sender as Button).Tag;
            Show_Bill(idTable);
        }
        void ChangAccount(int type) //관리자이든지 직원이든지 관리라는 버튼의 상태 지정
        {
            adMStrip.Enabled = type == 1;
            lbLoginUserName.Text = "로그인 : " + this.LoginAccount.DisplayName.ToString();
//현재 로그인자 출력함
        }
        #endregion
        #region Button : Status,Click - AddFood,Print Bill,Sale
        private void 관리정보ToolStripMenuItem_Click(object sender, EventArgs e) //관리라는
버튼의 클릭 이벤트 처리
        {
            Admin admin = new Admin();
            admin.loginAccount = LoginAccount;
            admin.ShowDialog();
        }
        private void 로그인ToolStripMenuItem1_Click(object sender, EventArgs e)//로그인라는
버튼의 클릭 이벤트 처리
        {
            Login lg = new Login();
            this.Hide();
            lg.ShowDialog();
        }
        private void 여가ToolStripMenuItem_Click(object sender, EventArgs e) //여가라는
버튼으 클릭 이벤트 처리
        {
            Game game = new Game();
            game.Show();
        }
        private void btAddFood_Click(object sender, EventArgs e) //주문이라는 버튼 클릭이벤트
처리
        {
```

```
            Table table = lsvBill.Tag as Table; //현재 클릭 테이블과 ListView에서 출력하는
Bill 지정
            int idBill = BillDAO.Instance.GetUnCheckBillIdbyTableID(table.Id); //현재
테이블의 id값을 전달해서 Billd가 존재하는지 테스트
            int foodID = (cbFood.SelectedItem as Food).ID; //선택된 음식 항목 값은 저장
            int count = (int)nudFoodCount.Value; //해당하는 음식을 선택된 개수 저장
            if (idBill == -1)//현재 테이블이 Bill가 존재하지 않은 경우에는 Bill추가하면
BillInfo를 만듦
            {
                BillDAO.Instance.InsertBill(table.Id);
                BillInfoDAO.Instance.InsertBillInfo(BillDAO.Instance.GetMaxIdBill(), foodID,
count);
            }
            else//현재 테이블에 Bill가 이미 존재해 있음
            {
                BillInfoDAO.Instance.InsertBillInfo(idBill, foodID, count);
            }
            Show_Bill(table.Id); //ListView 다시 로드
            LoadTable(); //Table 다시 로드
        }
        private void checkOut_Click(object sender, EventArgs e)//결제시에 일어나는 이벤트 처리
        {
            Table table = lsvBill.Tag as Table;
            int idBill = BillDAO.Instance.GetUnCheckBillIdbyTableID(table.Id);
            if(idBill != -1) //해당하는 idTable과 idBill 맞음
            {
                if (MessageBox.Show(table.Name + "결제를 진행하시겠습니까?", "Infomation",
MessageBoxButtons.OKCancel) == DialogResult.OK)
                {
                    BillDAO.Instance.CheckOut(idBill); //Bill에서 idBill의 상태 변경
                    Show_Bill(table.Id);//ListView 다시 로드
                    LoadTable();//Table 다시 로드
                }
            }
        }
        private void btShowBill_Click(object sender, EventArgs e) //영수증이라는 버튼 클릭
이벤트 처리
        {
            Table table = lsvBill.Tag as Table;
            fBill bill = new fBill();
            bill.ShowBill(this.loginAccount.DisplayName,
txttotalPrice.Text,table.Id);//idTable를 전달해서 주문한 정보들을 출력
            bill.ShowDialog();
        }
        private void btSale_Click(object sender, EventArgs e) //할인이라는 버튼 클릭 이벤트
        {
            if (cbSale.SelectedItem == null)
            {
                MessageBox.Show("먼저 할인 금액을 선택하세요.");
                return;
```

```
            }
            float Sale = (float)Convert.ToDouble(cbSale.SelectedItem);
            totalPrice -= Sale;
            if (totalPrice <= 0f)
            {
                MessageBox.Show("할인 금액을 만족하지 않습니다.");
                return;
            }
            CultureInfo culture = new CultureInfo("euc-KR");
            txttotalPrice.Text = totalPrice.ToString("c", culture);
        }
        #endregion
        #region NotifiIcon Event Handling
        private void fMain_Resize(object sender, EventArgs e) //폼의 사이즈에 따라서
nofiIcon의 상태 지정
        {
            notifyIcon1.ContextMenuStrip = contextMenuStrip1;
            if (this.WindowState == FormWindowState.Minimized)
            {
                ShowInTaskbar = false;
                notifyIcon1.Visible = true;
            }
            if (this.WindowState == FormWindowState.Normal)
            {
                ShowInTaskbar = true;
                notifyIcon1.Visible = false;
            }
        }
        private void fMain_FormClosing(object sender, FormClosingEventArgs e) //폼의 닫을때
발생하는 이벤트 처리
        {
            e.Cancel = true;
            this.WindowState = System.Windows.Forms.FormWindowState.Minimized;
        }
        private void 화면보기ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Visible = true;
            this.ShowInTaskbar = true;
            notifyIcon1.Visible = false;
            //this.WindowState = System.Windows.Forms.FormWindowState.Normal;
        }
        private void 로그인ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Login login = new Login();
            login.ShowDialog();
        }
        private void 게임실행ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Game game = new Game();
            game.ShowDialog();
        }
        private void 종료ToolStripMenuItem_Click(object sender, EventArgs e) //종료라는 버튼
이벤트 처리
        {
            //Application.Exit();
            Environment.Exit(1);
```

```
        }
        #endregion

    }
}
```

2,Admin 클래스

```
using CShareProjectWinApp.DataAccessObject;
using CShareProjectWinApp.DataTransferObject;
using System;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Runtime.InteropServices;
using System.Text;
using System.Windows.Forms;

namespace CShareProjectWinApp
{
    public partial class Admin : Form
    {
        private static string NirCmdPath = Application.StartupPath + "\\nircmd-
x64\\nircmdc.exe"; //NirSoft 경로 setting
        BindingSource accountList = new BindingSource(); //아이디
        public Account loginAccount; //현재 로그인 아이디를 저장하는 변수
        private Timer time; //대기 기능을 위한 타이머
        public Admin()
        {
            InitializeComponent();
            LoadAccountList();
            time = new Timer();
            time.Tick += new EventHandler(Timer_Setting); //Hook up Timer's tick event
handler.
        }
        /// <summary>
        /// 데이터 베이스에서 저장되어 있는 원하는 항목을 선택해서 조회
        /// </summary>
        /// <param name="pnl"></param>
        #region Show Data Event Handling
        private void ShowSelect(Panel pnl) //체크되 항목을 검사하여 데이터를 출려
        {
            RadioButton ckb = null;
            foreach (RadioButton item in pnl.Controls)
            {
                if (item.Checked)
                {
                    ckb = item;
                    break;
                }
            }
            if (ckb != null) //선텍된 항목을 이름에 근거하여 데이터를 조회 실행
            {
                string showData = ckb.Name;
                string query = "SELECT * FROM " + showData;
```

```
                dtgvData.DataSource = DataProvider.Instance.ExcuteQuerry(query);
            }
            else { //선택된 항목이 없는 경우에는
                MessageBox.Show("먼저 조회 항목을 선택하세요",
"알림",MessageBoxButtons.OK,MessageBoxIcon.Information);
            }
        }
        private void btDateView_Click(object sender, EventArgs e) //조회 버튼을 클릭
        {
            ShowSelect(plShow);
        }    #endregion
        /// <summary>
        /// 데이터 베이스에서 Acount date를 가져와서 출력하도록
        /// Acount 추가,설정,삭제 기능의 이벤트들
        /// </summary>
        #region Account Event Handlling
        public void LoadAccountList() //데이트를 로드하는 함수
        {
            LoadAccout();
            dtgvAccount.DataSource = accountList; //Set the data source for dataGridView
dtgvAccount to BindingSource accountList
            AddAccountBinding();
        }
        void AddAccountBinding() //Add메소드를 통해서 각 테스트 박스과 DataGridView 데이터 원본
유형의 고정됨
        {
            try
            {
                txtUserName.DataBindings.Add(new Binding("Text", dtgvAccount.DataSource,
"UserName", true, DataSourceUpdateMode.Never)); //데이터를 textBox애서
                txtDisplayName.DataBindings.Add(new Binding("Text", dtgvAccount.DataSource,
"DisplayName", true, DataSourceUpdateMode.Never));
                nudType.DataBindings.Add(new Binding("Value", dtgvAccount.DataSource,
"Type", true, DataSourceUpdateMode.Never));
            }
            catch
            { }
        }
        void LoadAccout()
        {
            // Set data source for BindingSource AccountList
            accountList.DataSource = AccountDAO.Instance.GetListAccount();
        }
        //기준으로 비밀번호 = 0하고 남은 정보를 받아서 아이디를 추가
        void AddCount(string userName, string displayName, int type)
        {
            if (AccountDAO.Instance.InsertAccount(userName, displayName, type)) //아이디를
추가될 경우
                MessageBox.Show("Add Account Successful");
            else
                MessageBox.Show("Add Count Fail");
            LoadAccout();
        }
        private void btAddAccount_Click(object sender, EventArgs e)
        {
```

```
            string userName = txtUserName.Text;
            string displayName = txtDisplayName.Text;
            int type = (int)nudType.Value;
            AddCount(userName, displayName, type);
        }
        //이름에 따라서 displayName과 type를 수정,즉 userName 수정하면 안 된다.
        void EditCount(string userName, string displayName, int type)
        {
            if (AccountDAO.Instance.UpdateAccount(userName, displayName, type))//아이디를
수정될 경우
                MessageBox.Show("Edit Account Successful");
            else
                MessageBox.Show("Edit Account Fail");
            LoadAccout();
        }
        private void btEditAccount_Click(object sender, EventArgs e)
        {
            string userName = txtUserName.Text;
            string displayName = txtDisplayName.Text;
            int type = (int)nudType.Value;
            EditCount(userName, displayName, type);
            LoadAccout();
        }
        //현재 로그인된어 있는 아이디를 삭제 할 수 없음.또한 Admin 아이디도 삭제 불가능
        void DeleteCount(string userName)
        {
            if (loginAccount.UserName.Equals(userName))
            {
                MessageBox.Show("This Account can't delete,Error",
"알림",MessageBoxButtons.OK,MessageBoxIcon.Error);
                return;
            }
            if (AccountDAO.Instance.DeleteAccount(userName)) //삭제 성공된 경우
                MessageBox.Show("Delete Account Successful", "알림");
            else
                MessageBox.Show("Delete Account Fail", "알림");
            LoadAccout();
        }
        private void btDeleteAccount_Click(object sender, EventArgs e)
        {
            string userName = txtUserName.Text;
            DeleteCount(userName);
            LoadAccout();
        }
        //비밀번호를 변경하기 해댕하는 이름을 전달하고 새로 번호를 입력
        void ResetPassWord(string userName, string newPassWord)
        {
            if (AccountDAO.Instance.ResetPassWord(userName, newPassWord))
                MessageBox.Show("Reset PassWord Successful","알림");
            else
                MessageBox.Show("Reset PassWord Fail", "알림");
            LoadAccout();
        }
        private void rtResetPassWord_Click(object sender, EventArgs e)
        {
            string userName = txtNameRsPw.Text;
```

```
            string newPassWord = txtNewPassWord.Text;
            ResetPassWord(userName, newPassWord);
            LoadAccout();
        }    #endregion
        /// <summary>
        /// 전원 관리 기능에 이벤트와 메소드 구성
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        #region Monitor,Computer Event Handling
        private void btSleep_Click(object sender, EventArgs e) //모니터 끄기
        {
            string result = HTTPWebCommDAO.Instance.Connection("sleep");
            serverResult.Text = result.ToString();
            Process.Start(NirCmdPath, "monitor off");
        }
        public void btWakeUp_Click(object sender, EventArgs e) //컴퓨터 시작시에 서버에 로그
기록
        {
            string result = HTTPWebCommDAO.Instance.Connection("wakeup");
            serverResult.Text = result;
        }
        private void btShutDown_Click(object sender, EventArgs e) //컴퓨터 종료시에 서버에
로그 기록
        {
            string result = HTTPWebCommDAO.Instance.Connection("shutdown");
            serverResult.Text = result;
            Process.Start(NirCmdPath, "exitwin poweroff");
        }
        // Create two StatusBarPanel objects to display in the StatusBar.
        StatusBarPanel downTimePanel;
        StatusBarPanel barPanel;
        StatusBar bar; // Create a StatusBar control.
        decimal downTime = 0; //시간을 하나씩 감소 표시하는 변수
        void LoadStatusbar(string option)
        {
            downTimePanel = new StatusBarPanel();
            barPanel = new StatusBarPanel();
            bar = new StatusBar();

            bar.ShowPanels = true; // Display panels in the StatusBar control.
            bar.Panels.Add(barPanel); // Add both panels to the StatusBarPanelCollection of
the StatusBar.
            bar.Panels.Add(downTimePanel);

            downTimePanel.Text = "";
            barPanel.Text = "Waiting...";
            if (option.Equals("Wait"))//대기모드를 클릭할때
            {
                this.pnlWait.Controls.Add(bar); // Add the StatusBar to the Monitor wait
pabel.
                return;
            }
                this.pnlHibernation.Controls.Add(bar);
        }
        //Timer 의 Tick 이벤트를 처리
        private void Timer_Setting(object sender, EventArgs e)
```

```
        {
            if (downTime > 0)
            {
                downTime--;
               downTimePanel.Text = downTime.ToString();
            }
            else
            {
                barPanel.Text = "Finish...";
                if (!btWait.Enabled) //시간이 지나면 전원 관리 기능이 실행
                {
                    string result = HTTPWebCommDAO.Instance.Connection("suspend");
                    serverResult.Text = result; //서버에서 response의 결과 출력함
                    Process.Start(NirCmdPath, "standby force");
                    btWait.Enabled = true;
                    time.Stop();
                }
                else //최대 절전모드를 실행
                {
                    string result = HTTPWebCommDAO.Instance.Connection("hibernate");
                    serverResult.Text = result;
                    Process.Start("powercfg", "-h on");
                    Process.Start("rundll32", "powrprof.dll, SetSuspendState");
                    btHibernation.Enabled = true;
                    time.Stop();
                }

            }

        }
        //대기모드라는 버튼의 클릭의 이벤트 처리
        private void btWait_Click(object sender, EventArgs e)
        {
            if (cbWait.SelectedItem == null) //무조건 시간이 먼저 골라야 됨
            {
                MessageBox.Show("먼저 원하는 시간이 선택해주세요.", "알림",
MessageBoxButtons.OK);
                return;
            }
            time.Interval = 1000; //1초마다 Tick 이벤트를 실행
            btWait.Enabled = false;
            string timeSelect = cbWait.SelectedItem.ToString();
            timeSelect = timeSelect.Substring(0,2); //ComboBox부터 선택된 시간의 값은 얻음
            downTime = decimal.Parse(timeSelect);
            LoadStatusbar("Wait");
            time.Start();
        }
        //최대 절전모드라는 버튼의 클릭의 이벤트 처리
        private void btHibernation_Click(object sender, EventArgs e)
        {
            if (cbHibnation.SelectedItem == null)
            {
                MessageBox.Show("먼저 원하는 시간이
선택해주세요.","알림",MessageBoxButtons.OK);
                return;
            }
```

```
            time.Interval = 1000;
            btHibernation.Enabled = false;
            string timeSelect = cbHibnation.SelectedItem.ToString();
            timeSelect = timeSelect.Substring(0, 2);
            downTime = decimal.Parse(timeSelect);
            LoadStatusbar("Hibernation");
            time.Start();
        }
        //대기모드 기능의 시간을 실행하다가 취고 버튼의 이벤트
        private void btWaitNo_Click(object sender, EventArgs e)
        {
            btWait.Enabled = true;
            time.Stop();
            downTimePanel.Text = "";
        }
        //최대 절전모드 기능의 시간을 실행하다가 취고 버튼의 이벤트
        private void btHibernationNo_Click(object sender, EventArgs e)
        {
            btHibernation.Enabled = true;
            time.Stop();
            downTimePanel.Text = "";
        }  #endregion
        /// <summary>
        /// 파일을 가져오기,파일 저장하거나 파일의 내용을 인쇄 이벤트
        /// 파일의 내용을 글꼴,색깔,테스트의 박스의 배경을 변경하는 이벤트
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        #region File,File Content,Link label Event Handling
        private void btOpenFile_Click(object sender, EventArgs e) //파일 가져오기
        {
            OpenFileDialog openFile = new OpenFileDialog()
            {
                DefaultExt = "txt",
                Title = "파일을 선택해주세요",
                InitialDirectory = @"C:",
                Filter = "Text Files|*.txt|Allfiles|*.*",
                FilterIndex = 2,
            };
            if (openFile.ShowDialog() == DialogResult.OK)
            {
                richTextBox1.Text = File.ReadAllText(openFile.FileName, Encoding.Default);
            }
        }
        private void btSaveFile_Click(object sender, EventArgs e) //파일 저장
        {
            SaveFileDialog saveD = new SaveFileDialog()
            {
                DefaultExt = "txt",
                Title = "파일을 선택해주세요",
                InitialDirectory = @"C:",
                Filter = "Text Files|*.txt|Allfiles|*.*",
                FilterIndex = 2,
            };
            DialogResult result = saveD.ShowDialog();
```

```
            if (saveD.FileName != "" && result == DialogResult.OK) //저장할 파일의 이름이
있어야 함
            {
                using (StreamWriter sw = new StreamWriter(saveD.FileName))
                {
                    sw.Write(richTextBox1.Text);
                    sw.Close();
                }
            };
        }
        private void 글꼴ToolStripMenuItem_Click(object sender, EventArgs e) //내용의 글꼴을
변경
        {
            FontDialog fontD = new FontDialog();
            if (fontD.ShowDialog() == DialogResult.OK)
            {
                richTextBox1.Font = fontD.Font;
            }
        }
        private void 색깔ToolStripMenuItem_Click(object sender, EventArgs e) //내용의 색깔을
변경
        {
            ColorDialog colorD = new ColorDialog();
            if (colorD.ShowDialog() == DialogResult.OK)
            {
                richTextBox1.ForeColor = colorD.Color;
            }
        }

        private void 배경ToolStripMenuItem_Click(object sender, EventArgs e) //테스트 박스의
배경을 변경
        {
            ColorDialog colorD = new ColorDialog();
            if (colorD.ShowDialog() == DialogResult.OK)
            {
                richTextBox1.BackColor = colorD.Color;
            }
        }
        private void btPrint_Click(object sender, EventArgs e) //인쇄의 이벤트
        {
            printDialog1.Document = printDocument1;
            if (printDialog1.ShowDialog() == DialogResult.OK)
            {
                printDocument1.Print();
            }
        }
        private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
        {
            e.Graphics.DrawString(richTextBox1.Text, new Font("Arial", 20,
FontStyle.Italic), Brushes.Black, 150, 130);
        }
        private void linkGoogle_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
//Google페이지를 이동함
        {
            Process.Start("https://www.google.com/");
```

```
        }
        private void linkYoutube_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
//Youtube페이지를 이동함
        {
            Process.Start("https://www.youtube.com/?gl=KR&hl=ko");
        }
        private void linkNaver_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
//Naver페이지를 이동함
        {
            Process.Start("https://www.naver.com/");
        }  #endregion
        /// <summary>
        /// 데이트 베
        /// </summary>
        /// <param name="pnl"></param>

        /// <summary>
        /// //키보드 후킹 처리하는 변수,메소드
        /// </summary>
        /// <param name="idHook"></param>
        /// <param name="callback"></param>
        /// <param name="hInstance"></param>
        /// <param name="threadId"></param>
        /// <returns></returns>
        #region KeyboardHookHandler
        private delegate IntPtr KeyboardHookHandler(int nCode, IntPtr wParam, IntPtr
lParam);
        private KeyboardHookHandler hookHandler;
        private IntPtr hookID = IntPtr.Zero;
        public void Install()
        {
            hookHandler = HookFunc;
            hookID = SetHook(hookHandler);
        }
        public void Uninstall()
        {
            UnhookWindowsHookEx(hookID);
        }
        private IntPtr SetHook(KeyboardHookHandler proc)
        {
            using (ProcessModule module = Process.GetCurrentProcess().MainModule)
                return SetWindowsHookEx(13, proc, GetModuleHandle(module.ModuleName), 0);
        }
        private IntPtr HookFunc(int nCode, IntPtr wParam, IntPtr lParam)
        {
            if (nCode >= 0 && wParam == (IntPtr)WM_KEYDOWN)
            {
                int vkCode = Marshal.ReadInt32(lParam);
                switch (vkCode.ToString())
                {
                    case "112": //F1 - 모니터 끄기
                        this.btSleep_Click(new object(), new EventArgs());
                        break;
                    case "113": //F2 - 컴퓨터 시작시
                        this.btWakeUp_Click(new object(), new EventArgs());
                        break;
                    case "114": //F3 - 컴퓨터 끄기
                        this.btShutDown_Click(new object(), new EventArgs());
```

```
                        break;
                    case "115": //F5 - 대기모드
                        this.btWait_Click(new object(), new EventArgs());
                        break;
                    case "116": //F6 - 최대 절전모드
                        this.btHibernation_Click(new object(), new EventArgs());
                        break;
                }
                return (IntPtr)1;
            }
            else
                return CallNextHookEx(hookID, nCode,wParam, lParam);
        }

        #region WinAPI
        private const int WM_KEYDOWN = 0x100;
        private const int WM_SYSKEYDOWN = 0x104;
        private const int WM_KEYUP = 0x101;
        private const int WM_SYSKEYUP = 0x105;
         private const int WH_KEYBOARD_LL = 13;

        [DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
        private static extern IntPtr SetWindowsHookEx(int idHook, KeyboardHookHandler lpfn,
IntPtr hMod, uint dwThreadId);

        [DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
        [return: MarshalAs(UnmanagedType.Bool)]
        private static extern bool UnhookWindowsHookEx(IntPtr hhk);

        [DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
        private static extern IntPtr CallNextHookEx(IntPtr hhk, int nCode, IntPtr wParam,
IntPtr lParam);

        [DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]
        private static extern IntPtr GetModuleHandle(string lpModuleName);
        #endregion
        private void tabControl1_Selected(object sender, TabControlEventArgs e) //이장에서만
후킹 되는것임
        {
            if (tabControl1.SelectedTab == tabPage2)
            {
                Install();
            }
            else
            {
                Uninstall(); //End Hoonking
            }
        }
        #endregion
    }
}
```