융합 UI 실습 과제
키오스크 제작

학번: 2401110252

이름: 박지수

보고서 구성

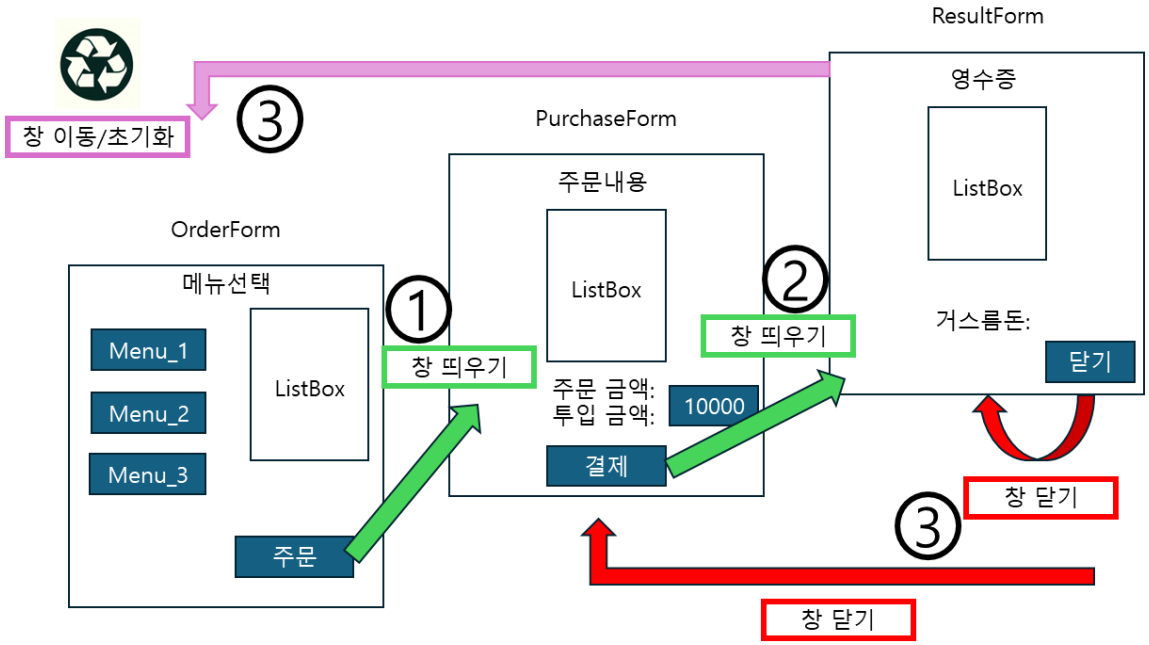
1. 개요

가) 요구 사양

첫 번째 창에서는 버튼을 이용해 해당하는 메뉴를 선택하면 ListBox로 물품이 담기고 주문할 물품들을 다 담았다면 주문하기 버튼을 통해 결제하는 새 창을 띄우게 된다.

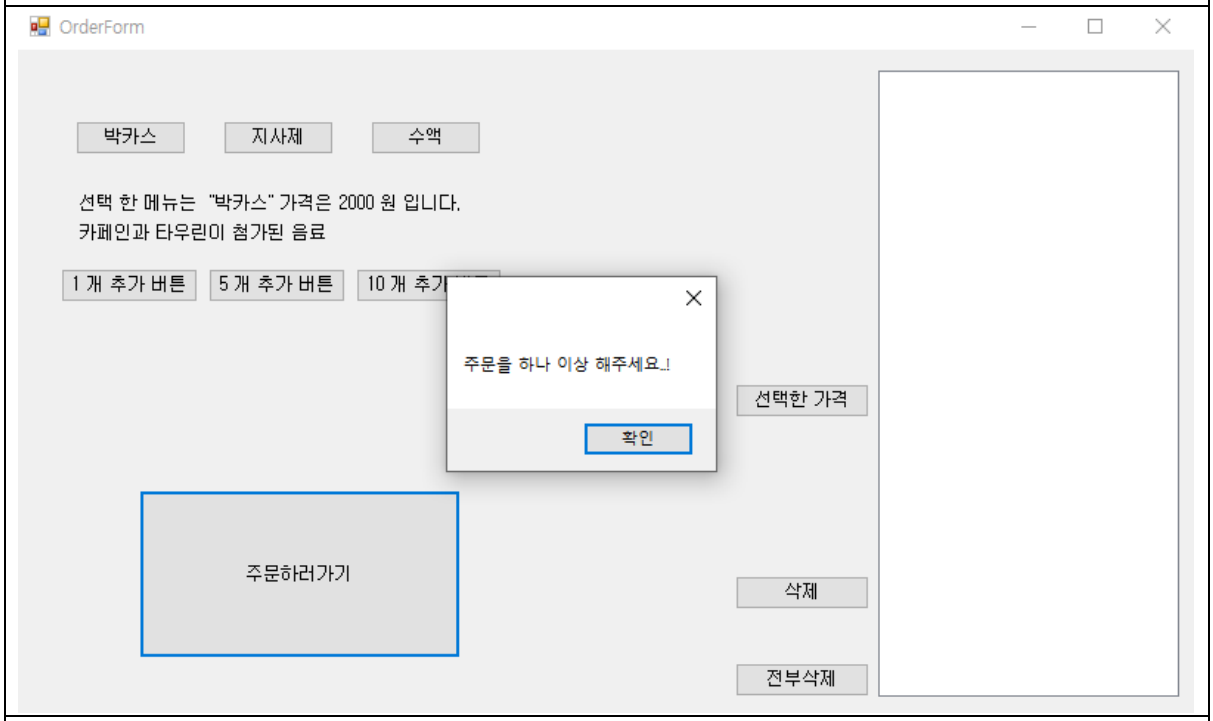
두 번째 창에서는 돈을 투입하는 버튼을 이용해 돈을 투입하여 청구된 금액 이상으로 돈을 투입하고 결제버튼을 누른다면 세 번째 창을 띄우게 되는데 구매했던 물품 내역과 결제하고 남은 거스름돈이 표시되며 세번째 창에서 닫기 버튼을 누르면 첫 번째 창만 남게 되고 구매했던 물품들은 초기화된다.

나) 키오스크 진행도

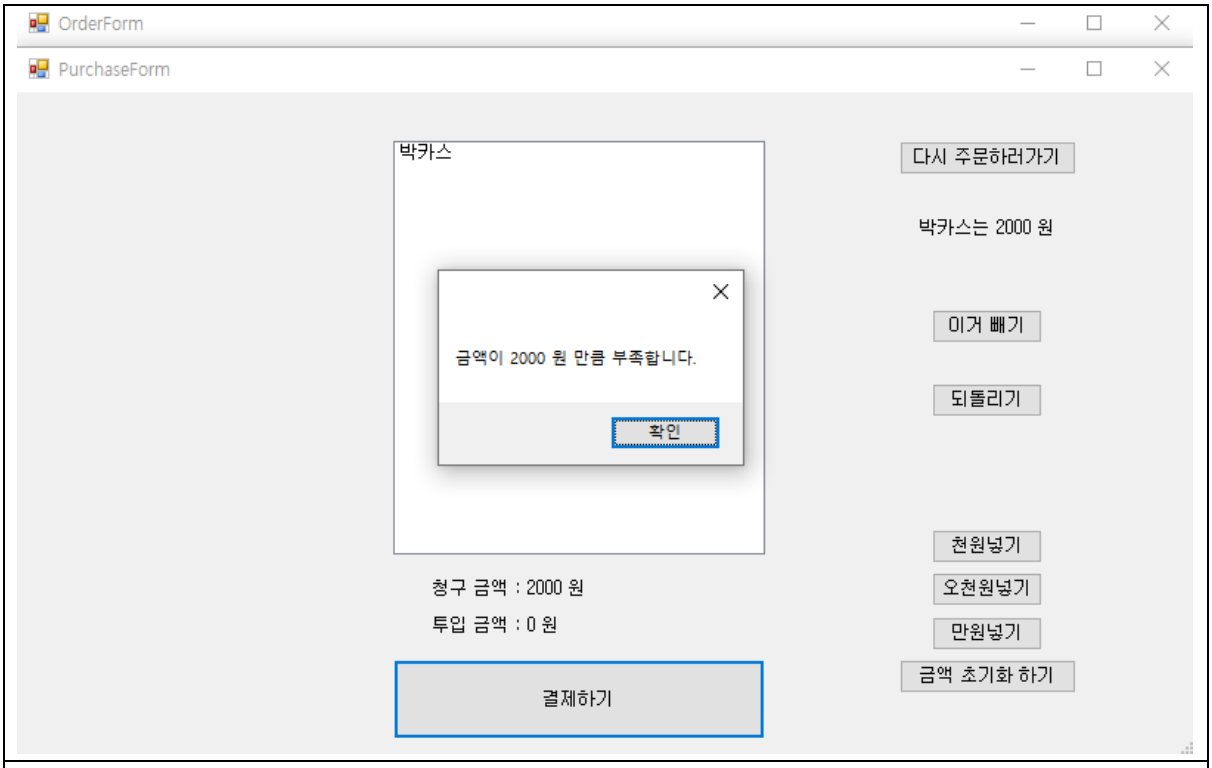


2. 실행 결과

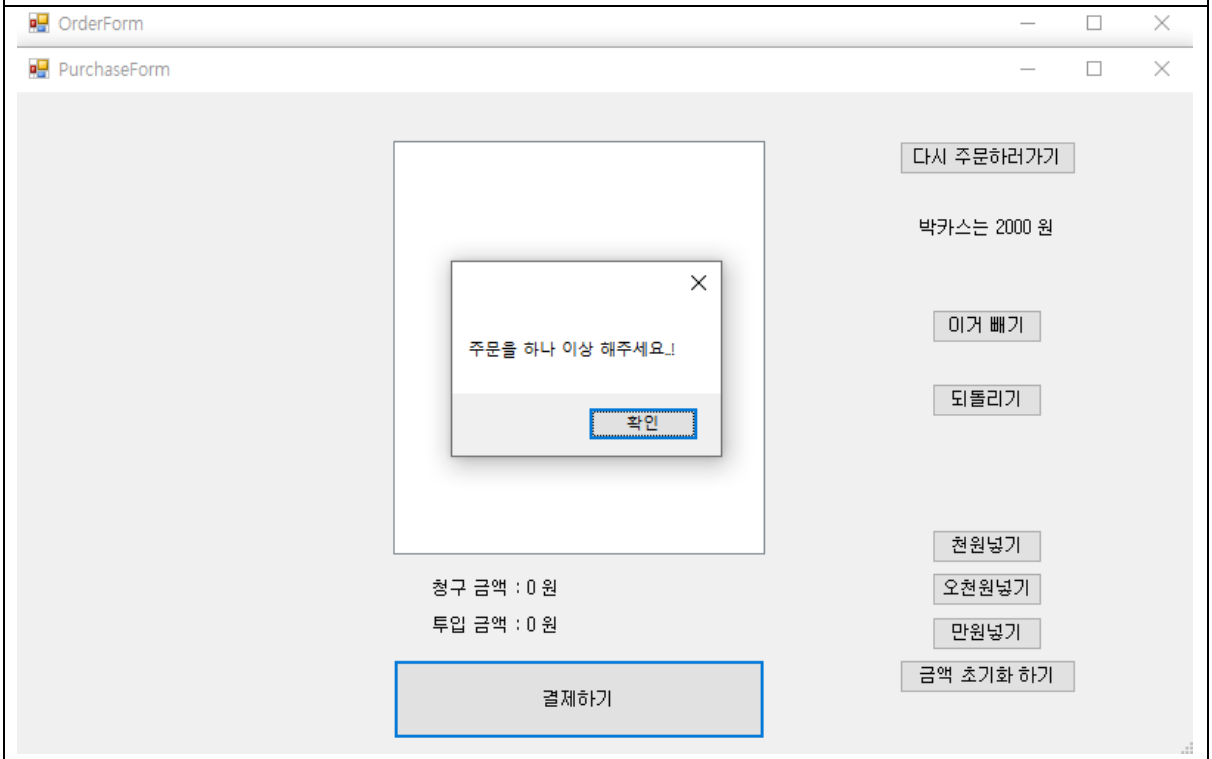
주문을 마친 경우



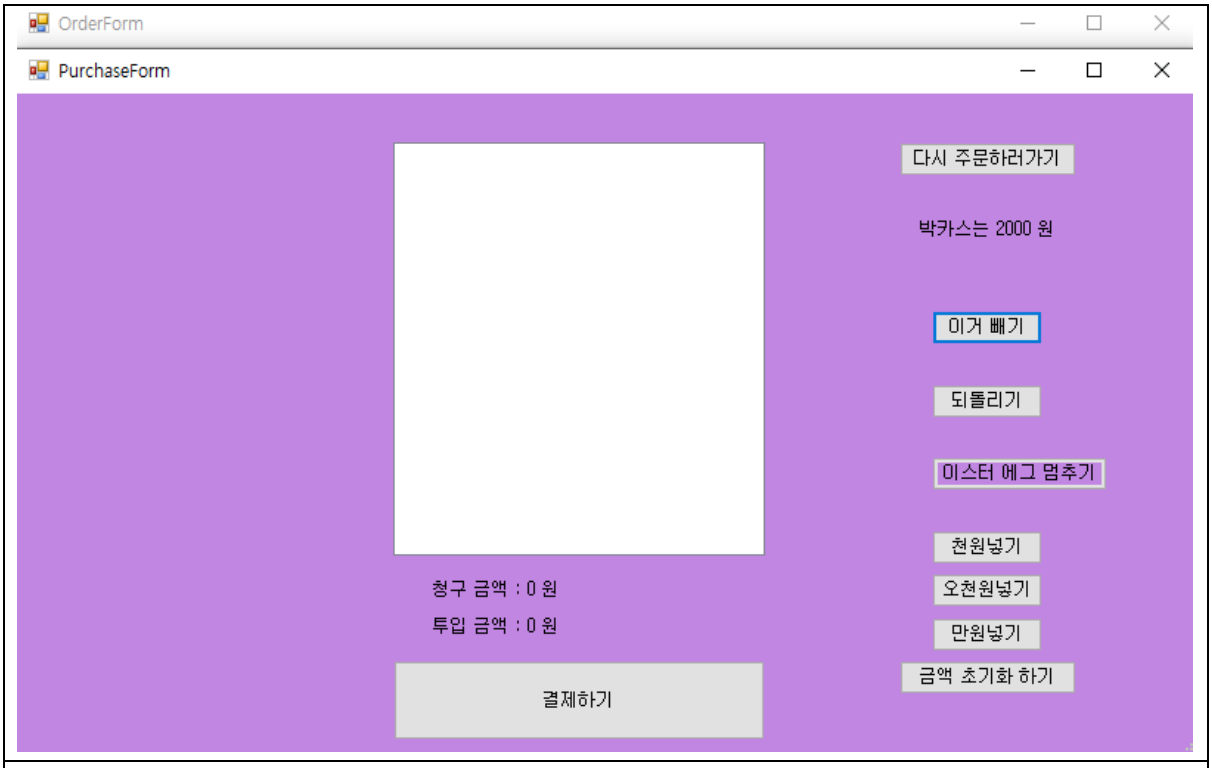
주문을 하지 않고 결제할 경우



금액을 부족하게 투입한 경우



결제 창에서 주문 내역을 다 지우고 결제하려는 경우



특정 조건을 만족했을 경우(색상이 계속 바뀜)

결제 완료된 최종 화면

3. 코드

가) 전체 코드

ㄱ) OrderForm

```
// 2024_05_16_융합 UI_실습과제
// 한국폴리텍대학_서울정수캠퍼스_인공지능소프트웨어학과
// 2401110252_박지수
// 키오스크 제작
// https://github.com/Mourn5367/Csharp_Subject

using System;
using System.Drawing;
using System.Windows.Forms;

namespace UI_Kiosk
{

    public partial class OrderForm : Form
    {
        string menuText = "";
        int price = 2000;
        int menus = 3;
        string selText = "선택 한 메뉴는 ";
        string selPrice = "선택 한 메뉴 가격은 ";
        public string[][] menuList;
        Button[] selectButtons ;
        Button[] addButtons;
        Label[] labels;
        public OrderForm() // 생성자
        {
            InitializeComponent();
            Text = "OrderForm";
            menuList = new string[][] // 첫번째는 메뉴명, 두번째는 가격,
세번째는 설명
            {
                new string[menus*3],
                new string[menus*3],
                new string[menus]

            };
            // 넣을 메뉴 입력 , 설명 넣기
            menuList[0][0] = "박카스";
```

```
            menuList[2][0] = "카페인과 타우린이 첨가된 음료";
            menuList[0][1] = "지사제";
            menuList[2][1] = "설사를 멎게하는 약으로 주의깊게 사용해야한다";
            menuList[0][2] = "수액";
            menuList[2][2] = "여러 수액이 있지만 이 메뉴는 링거액입니다.";
            for (int i = 0; i < menus; i++)
            {
                menuList[0][i + menus] = menuList[0][i] + " * 05";
                menuList[0][i + menus * 2] = menuList[0][i] + " * 10";
            }

            for (int i = 0; i < menuList[0].Length; i++) // 가격 넣기
            {
                if (i < menus)
                {
                    menuList[1][i] = ((i+1) * price).ToString();
                }
                else if (i < menus * 2)
                {
                    menuList[1][i] = ((i+1- menus) * price*5).ToString();
                }
                else
                {
                    menuList[1][i] = ((i+1 - menus * 2) *
price*10).ToString();
                }
            }


            selectButtons = new Button[menus]; // 메뉴 선택 버튼 만들기
            for (int i = 0; i < menus; i++)
            {
                int index = i;
                selectButtons[i] = new Button();
                selectButtons[i].Location = new Point(this.Size.Width / 20 +
i * 100, this.Size.Height / 10);
                selectButtons[i].Text = menuList[0][i];
                selectButtons[i].Click += (sender, e) =>
                {
                    menu_Index_Button_Click(sender, e, index);
                };
                Controls.Add(selectButtons[i]);
            }

            labels = new Label[2]; // 선택한 메뉴 라벨, 설명 라벨 만들기
```

```
            for (int i = 0; i < labels.Length; i++)
            {
                labels[i] = new Label();
                labels[i].Location = new Point(this.Size.Width / 20,
selectButtons[0].Location.Y + i*20  + 50);
                labels[i].AutoSize = true;
                Controls.Add(labels[i]);
            }
            labels[0].Text = selText;
            labels[1].Text = "설명";


            addButtons = new Button[3]; // 1 개 추가하기 버튼 5 개 추가하기 버튼
10 개 추가하기 버튼
            for ( int i = 0; i < 3; i++)
            {
                int index = i;
                addButtons[i] = new Button();
                addButtons[i].Text = i == 0 ? $"{i+1} 개 추가 버튼" : $"{i*5}
개 추가 버튼";
                addButtons[i].Location = new Point(this.Size.Width / 27 + i
* 100, labels[0].Location.Y + 50);
                addButtons[i].AutoSize = true;
                addButtons[i].Click += (sender, e) =>
                {
                    menu_Add_Button_Click(sender, e, index);
                };
                Controls.Add(addButtons[i]);
            }
            listBox1.Sorted = true;
        }


        // 메뉴가 5 개,10 개모였는지 감지하여 묶어버리는 함수
        public void detectMenuCount()
        {
            int detected = 0;
            int detectedSet = 0;
            int findMenuIndex = 0;
            int findMenuIndexSet = 0;
            for (int i = 0; i <  menus; i++)
            {
                int equalCount = 0;
                if (detected != 0) break;
                foreach (String items in listBox1.Items)
                {
```

```
                if (menuList[0][i] == items)
                {
                    equalCount++;
                    if (equalCount >= 10)
                    {
                        detected++;
                        findMenuIndex = i;
                    }
                    else if (equalCount >= 5)
                    {
                        detected++;
                        findMenuIndex = i;
                    }
                }
            }
        }
        if ( detected >= 5)
        {
            listBox1.Items.Add(menuList[0][findMenuIndex + menus*2]);
            for (int i = 0; i < 10; i++)
            {
                listBox1.Items.Remove(menuList[0][findMenuIndex]);
            }
        }
        else if ( detected >= 1)
        {
            listBox1.Items.Add(menuList[0][findMenuIndex + menus]);
            for (int i = 0; i < 5; i++)
            {
                listBox1.Items.Remove(menuList[0][findMenuIndex]);
            }
        }

        for (int i = menus; i < menus * 2; i++)
        {
            int equalCount = 0;
            if (detectedSet != 0)
            {
                break;
            }
            foreach (String items in listBox1.Items)
            {

                if (menuList[0][i] == items)
                {
```

```
                        equalCount++;
                        if (equalCount >= 2)
                        {
                            detectedSet++;
                            findMenuIndexSet = i;
                        }
                    }
                }
            }

            if (detectedSet >= 1)
            {
                listBox1.Items.Add(menuList[0][findMenuIndexSet + menus]);
                for (int i = 0; i < 2; i++)
                {
                    listBox1.Items.Remove(menuList[0][findMenuIndexSet]);
                }
            }
        }

        // 메뉴의 이름을 비교하여 메뉴 값을 계산하는 함수
        public void calPrice()
        {
            int cnt = listBox1.Items.Count;
            int m = 0;
            for (int i = 0; i < menuList[0].Length; i++)
            {
                for (int j = 0; j < cnt; j++)
                {
                    if (listBox1.Items[j].ToString() == menuList[0][i])
                    {
                        m += int.Parse(menuList[1][i]);
                    }
                }
            }
            label3.Text = "총 가격 : " + m.ToString() + " 원";
        }

        // 결제 도중 다시 주문하기 위해 돌아올때 다시 ListBox items를 받아오는
함수
        public void reOrder(ListBox listBox, int price)
        {
            listBox1.Items.Clear();
            listBox1.Items.AddRange(listBox.Items);
            int cnt = listBox.Items.Count;
```

```
            label3.Text = "총 가격 : " + price.ToString() + " 원";
        }

        //결제까지 다 끝난후 주문상태를 초기화 하는 함수
        public void clearListBox()
        {
            labels[0].Text = selText;
            labels[1].Text = "설명";
            label2.Text = selPrice;
            label3.Text = "총 가격";
            listBox1.Items.Clear();
        }
        // 메뉴 버튼 클릭 이벤트시 작동하는 함수
        private void menu_Index_Button_Click(object sender, EventArgs e, int
index)
        {
            menuText = selectButtons[index].Text;
            labels[0].Text = selText + " \"" + menuText + "\"" + " 가격은 " +
menuList[1][index] + " 원 입니다.";
            labels[1].Text = menuList[2][index];

        }
        // 메뉴 추가 버튼 클릭 이벤트시 작동하는 함수
        private void menu_Add_Button_Click(object sender, EventArgs e, int
index)
        {
            if (menuText != "")
            {
                switch (index)
                {
                    case 0:
                        listBox1.Items.Add(menuText);
                        break;
                    case 1:
                        for (int i = 0; i < 5; i++)
                        {
                            listBox1.Items.Add(menuText);
                        }
                        break;
                    case 2:
                        for (int i = 0; i < 10; i++)
                        {
                            listBox1.Items.Add(menuText);
                        }
                        break;
```

```
                }
                detectMenuCount();
                calPrice();
            }
        }

        // ListBox items 클릭 후 삭제하기 버튼 클릭 이벤트시 작동하는 함수
        private void deleteButton_Click(object sender, EventArgs e)
        {
            if (listBox1.SelectedIndex >= 0)
            {
                listBox1.Items.RemoveAt(listBox1.SelectedIndex);
            }
            calPrice();
        }

        // ListBox items 전부 비우기
        private void deleteAllButton_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
            calPrice();
        }

        // ListBox items 클릭 후 가격 확인 버튼 클릭 이벤트시 작동하는 함수
        private void button2_Click(object sender, EventArgs e)
        {
            if (listBox1.SelectedIndex >= 0)
            {
                for (int i = 0; i < menuList[0].Length; i++)
                {
                    if (listBox1.SelectedItem.ToString() == menuList[0][i])
                    {
                        label2.Text = menuList[0][i] + " 가격은" +
menuList[1][i] + " 원";
                    }
                }
            }
        }

        // 주문하기 버튼 클릭 이벤트시 작동하는 함수 새 창을 연다.
        private void purchaseButton_Click(object sender, EventArgs e)
        {
            if (listBox1.Items.Count > 0)
            {
                PurchaseForm purchaseForm = new PurchaseForm();
```

```
                purchaseForm.SetForm(this);
                purchaseForm.PurchaseListBox(listBox1);
                purchaseForm.ShowDialog();
            }
            else
            {
                MessageBox.Show("주문을 하나 이상 해주세요..!");
            }
        }
    }
}
```

ㄴ) PurchaseForm

```
using System;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;
namespace UI_Kiosk
{

    public partial class PurchaseForm : Form
    {
        OrderForm orderForm;
        Random ran;
        Thread easterEggThread;
        Button eggStop;
        int bills = 0;
        int input = 0;
        int change = 0;
        string undoMenuName = "";
        bool threadFunc = false;
        public PurchaseForm()
        {
            InitializeComponent();
            ran = new Random();
            listBox1.Click += listBox_Index_Click;
        }

        public void SetForm(OrderForm form)
        {
            orderForm = form;
        }
        private void listBox_Index_Click( object sender, EventArgs e)
```

```
        {
            if (listBox1.SelectedIndex >= 0)
            {
                int selectIndex = listBox1.SelectedIndex;
                for (int i = 0; i < orderForm.menuList[1].Length; i++)
                {
                    if (listBox1.Items[selectIndex].ToString() ==
orderForm.menuList[0][i])
                    {
                        label1.Text = orderForm.menuList[0][i] + "는 " +
orderForm.menuList[1][i] + " 원";
                    }
                }
            }
        }

        public void PurchaseListBox(ListBox listBox_) // orderForm 의 ListBox
item 가져오기
        {
            foreach (String items in listBox_.Items)
            {
                listBox1.Items.Add(items);
            }
            CalPrice();
        }
        private void CalPrice() // 주문한 메뉴들 값 계산하기
        {
            int cnt = listBox1.Items.Count;
            bills = 0;
            for (int i = 0; i < orderForm.menuList[0].Length; i++)
            {
                for (int j = 0; j < cnt; j++)
                {
                    if (listBox1.Items[j].ToString() ==
orderForm.menuList[0][i])
                    {
                        bills += int.Parse(orderForm.menuList[1][i]);
                    }

                }
            }
            bill.Text = "청구 금액 : " + bills.ToString() + " 원";
        }

        private void input_1000_Click(object sender, EventArgs e)
```

```
        {
            input += 1000;

            inputMoney.Text = "투입 금액 : " + input.ToString() + " 원";
        }

        private void input_5000_Click(object sender, EventArgs e)
        {
            input += 5000;

            inputMoney.Text = "투입 금액 : " + input.ToString() + " 원";
        }

        private void input_10000_Click(object sender, EventArgs e)
        {
            input += 10000;

            inputMoney.Text = "투입 금액 : " + input.ToString() + " 원";
        }

        // 투입금액 초기화
        private void resetMoney_Click(object sender, EventArgs e)
        {
            input = 0;
            inputMoney.Text = "투입 금액 : " + input.ToString() + " 원";
        }

        // 결제하기 버튼 클릭 이벤트시 작동하는 함수 투입금액이 주문금액 보다
크다면 새 창을 연다.
        private void purchaseButton_Click(object sender, EventArgs e)
        {
            if (listBox1.Items.Count == 0)
            {
                MessageBox.Show("주문을 하나 이상 해주세요..!");
                return;
            }
            if (input >= bills)
            {
                threadFunc = false;
                change = input - bills;
                ResultForm purchaseForm = new ResultForm();
                if (easterEggThread != null)
                {
                    if (easterEggThread.IsAlive == true)
                        easterEggThread.Join();
```

```
                }
                purchaseForm.SetForm(this);
                purchaseForm.ResultListBox(listBox1, change, input, bills);
                purchaseForm.ShowDialog();
            }
            else
            {
                MessageBox.Show("금액이 " + (bills - input) + " 원" + " 만큼
부족합니다.");
            }
        }

        // 결제 다 끝났을때 OrderForm ListBox 초기화, Purchase 폼 닫기위한 함수
        public void closePurchaseForm()
        {
            orderForm.clearListBox();
            if (easterEggThread != null)
            {
                if (easterEggThread.IsAlive == true)
                    easterEggThread.Join();
            }
            Close();
        }

        // 메뉴 빼는 버튼 클릭 이벤트시 작동하는 함수
        private void select_Delete_Click(object sender, EventArgs e)
        {
            if (listBox1.SelectedIndex >= 0)
            {
                undoMenuName =
listBox1.Items[listBox1.SelectedIndex].ToString();
                listBox1.Items.RemoveAt(listBox1.SelectedIndex);
                easterEggCondition();
                CalPrice();
            }
        }

        // 메뉴 되돌리는 버튼 클릭 이벤트시 작동하는 함수
        // 메뉴 빼는 버튼을 잘못 눌렀을 시 되돌려 주는 함수
        private void undo_Button_Click(object sender, EventArgs e)
        {
            if (undoMenuName != "")
            {
                foreach (String item in orderForm.menuList[0])
                {
```

```
                    if (undoMenuName.Equals(item))
                    {
                        listBox1.Items.Add(undoMenuName);
                        undoMenuName = "";
                        easterEggCondition();
                        break;
                    }
                }
                CalPrice();
            }
        }

        // 빼먹은 주문이 있어 창을 닫고 OrderForm 창으로 돌아가는 함수
        private void close_Button_Click(object sender, EventArgs e)
        {
            threadFunc = false;
            if (easterEggThread != null)
            {
                if (easterEggThread.IsAlive == true)
                    easterEggThread.Join();
            }
            orderForm.reOrder(listBox1, bills);
            Close();
        }

        // 특정 조건시 발동하기 위한 함수 및 조건들
        private void randBackColor()
        {
            while (threadFunc == true)
            {
                this.BackColor = Color.FromArgb(ran.Next(0, 255),
ran.Next(0, 255), ran.Next(0, 255));
                Thread.Sleep(200);
            }
        }
        private void easterEgg_Stop_Button(object sender, EventArgs e)
        {
            this.BackColor = default(Color);
            threadFunc = false;
            if (easterEggThread != null)
            {
                if (easterEggThread.IsAlive == true)
                    easterEggThread.Join();
            }
```

```
            Controls.Remove(eggStop);

        }
        private void easterEggCondition()
        {
            if (listBox1.Items.Count == 3)
            {
                if
(listBox1.Items[0].ToString().Equals(orderForm.menuList[0][0]) &&
listBox1.Items[1].ToString().Equals(orderForm.menuList[0][1]) &&
listBox1.Items[2].ToString().Equals(orderForm.menuList[0][2]))
                {
                    if (!Controls.Contains(eggStop))
                    {
                        eggStop = new Button();
                        eggStop.Location = new Point(input_1000.Location.X,
input_1000.Location.Y - 50);
                        eggStop.Text = "이스터 에그 멈추기";
                        eggStop.AutoSize = true;

                        eggStop.Click += (send, even) =>
                        {
                            easterEgg_Stop_Button(send, even);
                        };
                        Controls.Add(eggStop);
                    }
                    if (easterEggThread != null)
                    {
                        threadFunc = false;
                        easterEggThread.Join();
                    }
                    threadFunc = true;
                    easterEggThread = new Thread(randBackColor);
                    easterEggThread.Start();

                }
            }
        }
    }
}
```

ㄷ) ResultForm

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

namespace UI_Kiosk
{
    public partial class ResultForm : Form
    {
        PurchaseForm purchase;

        public ResultForm()
        {
            InitializeComponent();
            this.ControlBox = false;
        }
        public void SetForm(PurchaseForm form)
        {
            purchase = form;
        }
        public void ResultListBox(ListBox listBox_, int change, int input,
int bills)
        {
            foreach (String items in listBox_.Items)
            {
                listBox1.Items.Add(items);
            }
            label_change.Text = "요금 "+bills + " 원\n\n투입 금액 " + input
+" 원 \n\n거스름돈 "+change.ToString()+ " 원";
        }

        // 버튼을 눌렸을때 실행되는 함수 PurchaseForm 까지 닫고 OrderForm 창
초기화 한다.
        private void closeResultForm_Click(object sender, EventArgs e)
        {
            purchase.closePurchaseForm();
            Close();
        }


        // ResultForm 이 닫길때 실행되는 함수 PurchaseForm 까지 닫고 OrderForm
창 초기화 한다.
```

```
        private void ResultForm_FormClosed(object sender,
FormClosedEventArgs e)
        {
            purchase.closePurchaseForm();
        }
    }
}
```

나) 주요 코드

ㄱ) OrderForm

A) detectMenuCount

메뉴가 추가될 때마다 낱개가 5개 모였는지 5개 묶음이 두 개인지 검사하는 함수이다. 5개 또는 10개가 모일 때 지워버리고 새로 추가한다.

B) calPrice

ListBox에 있는 item이 추가 또는 삭제되는 변경이 있을 때마다 작동하며 총가격을 계산해 주는 함수이다.
ListBox에 담긴 items들의 문자열과 메뉴 문자열을 비교하여 조건이 맞을 때마다 메뉴의 가격을 하나씩 더하고 총합을 라벨의 텍스트로 변환한다.

C) reOrder

현재 PurchaseForm 창일 때 주문한 메뉴를 삭제한 상태에서 다시 주문하기 버튼을 눌러 OrderForm으로 넘어올 때 PurchaseForm의 ListBox items를 OrderForm ListBox와 똑같이 만들어 주기 위한 함수이다.

D) purchaseButton_Click

PurchaseForm으로 넘어가기 위한 버튼을 눌렀을 때 발생하는 이벤트이다. 아무것도 주문하지 않을 때에는 넘어가지 않고 최소한 한 개 이상 담겨있을 때
PurchaseForm창으로 넘어가면서 PurchaseForm에서 OrderForm을 지정하고 items들을 넘긴 다음 창을 띄운다.
만약 한 개도 안 담은 상태라면 MessageBox를 띄워 안내한다.

ㄱ) PurchaseForm

A) listBox_Index_Click

기존 OrderForm에서는 ListBox item을 누른 후 선택한 가격이 출력되게 하였다면 이번에는 item을 누르자마자 라벨로 입력받게 만든 함수이다.

ListBox를 클릭했을 때 발생하는 이벤트 핸들러를 통해 ListBox의 인덱스가 클릭 되었을 때를 조건으로 두어서 클릭한 인덱스의 값을 가져와서 Label.Text로 처리한다.

B) undo_Button_Click

메뉴를 실수로 삭제하는 경우가 있을 수 있어 삭제 버튼을 통해 삭제할 때 삭제한 ListBox item 이름을 저장하고 undo_Button_Click 이벤트시 삭제한 이름을 토대로 다시 추가한다.

C) easterEggCondition

ListBox item을 삭제하거나 삭제 취소를 했을 때 발생한다.
조건이 맞다면 Thread를 작동시키고 Thread에 적용한
함수의 bool 자료형 또한 true로 만들어 작동하게 만든다.

Thread 작동 중일 때 또 Thread를 생성하고 작동하면 안 되니
존재 여부를 확인하고 중단한 다음 다시 실행한다.

ㄴ) ResultForm

A) closeResultForm_Click

종료 버튼을 클릭하면 발생하는 이벤트이다.

PurchaseForm의 closePurchaseForm 함수를 실행시켜 OrderForm창을 초기화시키고 PurchaseForm창을 닫고 ResultForm 창까지 닫고 함수가 종료된다.

4. 고찰

이번 과제를 진행하면서 크게 3가지 막힌 부분이 있었다.

첫번째는 ListBox를 가져오는 것이었다.
ListBox items를 가져오기 위해 객체를 함수 파라미터에 넣고(ref 사용 가능)
"=" 할당 연산자를 이용해 변수나 배열처럼 가져올 수는 있었지만 얕은 복사라서
실행해 보니 OrderForm의 창에서 뚝 때서 다음 창으로 가져오는 식이었다.

정말 무식하게 가져오는 식이라 Purchase창에서 역으로 OrderForm창으로 이동할 때
OrderForm에서 ListBox가 보이지 않는다.
위치 좌표를 미리 저장한 다음 새로 지정하고 Controls.Add() 함수를 해야
다시 나타난다.

메모리 관리 면에서는 이런 방식이 좋을 것 같지만 요즘 하드웨어를 생각한다면
굳이 이렇게까지 할 필요 없이 깊은 복사를 통해 새로운 객체를 만드는 것도
나름 방법이 될 수도 있고 그렇게 만들었다.
다음에 이런 박스를 정말 여러 창을 통해 거쳐 가야 한다면 참조하는 방식을
사용해야겠다.

현재 코드는 OrderForm의 ListBox item들을 다 가져와서 Purchase, ResultForm에
하나씩 다 넣는 방향으로 진행했다. 이때에는 기존 List 배열 함수인 AddRange
함수를 사용해도 되고 강의 시간에 배운 foreach를 사용해도 된다.

두 번째는 Forms의 이해도가 낮아서 나온 LabelSize로 인한 문제였다.

두 문자열을 합치기 위해 + 연산자를 사용하여 합쳤는데 뒷글자가 안 나오는
상황이었다. 여느 때처럼 합칠 때를 디버깅하여 보았는데도 문자열이 둘 다 제대로
들어가 있고 라벨의 텍스트도 제대로 들어가 있었다.

순간 이게 진짜 버그인지라고 생각할 만큼 당혹스러웠지만 문자열 두개의 순서를 바꿔보니 앞의 문자열만 출력되고 문자열 한 개만 출력하니 제대로 출력되었다. 아무래도 라벨 속성의 문제인 것 같아 찾아보니 LabelSize를 Auto로 설정하니 잘 출력되었다. 해결하니 정말 어이없는 문제였지만 당시에는 정말 힘든 순간이었다.

세 번째는 Thread 사용할 때 에러가 떠서 작동이 멈추는 상황이 자주 나와 작동 방식을 조금 더 공부해야겠다고 느꼈다.

검색해 보니 ManualResetEvent, AutoResetEvent 클래스를 이용하면 조금 더 핸들링에 도움이 되는 도구로 보이지만 개념을 잘 잡고 가야 할 것 같아 쓰는 것을 그만두고 기본 Thread 클래스 내에서 제어하기 위해 bool 자료형 타입의 변수와 Thread.Start, Join만 이용해 Form의 창을 반짝반짝 빛나게 구현해 보았다.

툭하면 에러 메시지를 뱉는 위험한 부분이라 최대한 많은
경우의 수를 가지고 테스트해 보면서 민감한 Thread 핸들링 방식을 조금은 알 것만 같았다.

다음에는 쿠폰을 넣어서 특정 상품을 넣었을 때 할인되고 사용하면 없어지는 기능까지 구현해 보고 싶다.