융합 UI 실습 과제  
키오스크 제작

학번: 2401110252

이름: 박지수

보고서 구성

1. **개요**1

#### 요구사양2

#### 키오스크 진행도[2](#_코드_설명_)

1. **실행 결과**5
2. **코드**

#### 코드 전체

#### 주요 코드

1. **고찰**[8](#_고찰)

# [**개요**](#_top)

#### [요구 사양](#_top) 첫 번째 창에서는 버튼을 이용해 해당하는 메뉴를 선택하면 ListBox로 물품이 담기고 주문하기 버튼을 누르면 ListBox와 물품들의 금액이 출력된다. 버튼을 이용해 돈을 투입하여 결제까지 마친다면 또 새로운 창이 뜨게 되는데 산 물품과 결제하고 남은 돈이 표시되며 닫기 버튼을 누르면 첫 번째 창만 남게 되고 샀던 물품들은 초기화된다.

#### [키오스크 진행도](#_top)

#### 

# **코드**

# 전체 코드

# 

# 주요 코드

# OrderForm

##### detectMenuCount 메뉴가 추가될 때마다 5개 모였는지 5개 묶음이 두 개인지 검사하는 함수이다. 5개 또는 10개가 모일 때 지워버리고 기존 메뉴 배열에 있는 문자열을 추가하는 단순한 방식이다.

##### calPrice ListBox에 있는 items들의 변경이 있을 때마다 작동하며 총가격을 계산해 주는 함수이다. ListBox에 담긴 items들의 문자열과 메뉴의 문자열을 비교하여 똑같다면 해당 배열에 문자열로 등록된 값을 가져와 int.Parse 함수를 통해 변환시켜 하나씩 더하고 라벨의 텍스트로 다시 문자열로 변환한다.

##### reOrder PurchaseForm에서 제어하기 위해 만들었다. 주문하다가 현재 상황 그대로 돌아가서 다시 주문하기 위해 ListBox items를 추가하는 함수이다.

##### purchaseButton\_Click PurchaseForm으로 넘어가기 위한 버튼을 눌렀을 때 발생하는 이벤트이다. 아무것도 주문하지 않고 최소한 한 개 이상 담겨있을 때 PurchaseForm창으로 넘어가면서 PurchaseForm에서 OrderForm을 기억할 수 있도록 지정하고 items들을 넘긴 다음 창을 띄운다. 만약 한 개도 안 담은 상태라면 MessageBox를 띄워 안내한다.

##### PurchaseForm

##### 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.

##### 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.

##### 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.

##### ResultForm

##### closeResultForm\_Click 종료 버튼을 클릭하면 발생하는 이벤트이다. PurchaseForm의 closePurchaseForm 함수를 실행시켜 OrderForm창을 초기화시키고 PurchaseForm창을 닫고 ResultForm 창까지 닫고 함수가 종료된다.

# **실행 결과**

|  |
| --- |
|  |
| 주문을 마친 경우 |
|  |
| 주문을 안하고 결제하러 갈 경우 |
|  |
| 금액을 부족하게 투입한 경우 |
|  |
| 결제 창에서 다지우고 결제하려는 경우 |
|  |
| 특정 조건을 만족했을 경우(색상이 계속 바뀜) |
|  |
| 결제 완료된 최종 화면 |

# **[고찰](#_top)**

이번 과제를 진행하면서 3가지 정도 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 1차원적인 생각으로 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 두 개를 합치는 것인 만큼 순서를 바꿔보니 앞의 문자열만 출력되고  
문자열 한 개만 출력하니 제대로 출력되는 것이었다.  
  
아무래도 라벨 속성의 문제인 것 같아 찾아보니 LabelSize를 Auto로 설정하니  
  
잘 출력되었다. 해결하니 정말 어이없는 문제였지만 당시에는 정말 힘든 순간이었다.  
  
세 번째는 Thread 사용이 어려워 따로 공부해야겠다고 느꼈다.  
검색해 보니 ManualResetEvent, AutoResetEvent 클래스를 이용하면 조금 더 핸들링에 도움이 되는 도구로 보이지만 개념을 잘 잡고 가야 할 것 같아 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.