동국대학교
dongguk university

메소드, 생성자

HyoJoon Han
동국대학교
han6343@dongguk.edu

목차

접근 제한자

- 접근 제한자의 형태

```
[접근 제한자] [자료형] [변수 이름]
[접근 제한자] [반환형] [메서드 이름]([매개변수])
{
    [메서드 코드]
}
```

- 접근 제한자의 범위

접근 제한자	클래스 내부	동일 패키지	상속받은 클래스	이외의 영역
private	●	X	X	X
protected	●	●	●	X
public	●	●	●	●

접근 제한자

- private 접근 제한자
  - 접근 제한자 입력하지 않으면 자동으로 private 접근 제한자 설정
  - 예

```
private void Form1_Load(object sender, EventArgs e)
{

}

private void button1_Click(object sender, EventArgs e)
{

}
```

  - Form_Load() , Button_Click() 와 같은 이벤트들은 주로 private 메서드로 생성
  - private 접근 제한자 적용 되면 자신의 클래스 내부에서만 해당 메서드 사용 가능

접근 제한자

- 다른 클래스를 만들고 다른 클래스에서 private 메서드 호출

```
class abc
{
    private void test1()
    {

    }
}

class def
{
    private void test2()
    {
        abc.test1();
    }
}
```

void abc.test1()

보호 수준 때문에 'Form1.abc.test1()'에 액세스할 수 없습니다.

접근 제한자

- public 접근 제한자
  - 모두가 접근 할 수 있는 영역
  - 예

```
private void button1_Click(object sender, EventArgs e)
{
    int a = plus(10, 20);
    int b = minus(10, 20);
}
public int plus(int a, int b)
{
    return a + b;
}
public int minus(int a, int b)
{
    return a - b;
}
```

접근 제한자

- 다른 클래스를 만들고 다른 클래스에서 public 메서드 호출

```
class abc
{
    public static void test1()
    {

    }
}

class def
{
    public static void test2()
    {
        abc.test1();
    }
}
```

생성자

- 생성자 (Constructor)
  - 클래스를 생성할 때 자동으로 호출되는 메서드
  - 인스턴스 생성자의 생성 조건
    - 이름이 클래스 이름과 같아야 함
    - 접근 제한자는 public
    - 반환형태 없음
    - 매개 변수의 개수는 상관 없음
  - 생성자의 형태

```
public [클래스 이름]([매개변수])
{

}
```

그림 6-12 생성자 형태

생성자

- 생성자의 인스턴스 변수 초기화

```
class FruitSeller
{
    string fruit;
    int myMoney;
    int numOfFruit;
    int fruitPrice;

    public FruitSeller()
    {
        fruit = "apple";
        myMoney = 0;
        numOfFruit = 10;
        fruitPrice = 1000;
    }

}
```

생성자

- 클래스의 암호화
  - 클래스 내부 변수에 타인이 접근하지 못하도록 함
  - private로 변수 선언

```
class FruitSeller
{
    public string fruit;
    public int myMoney;
    public int numOfFruit;
    public int fruitPrice;

    public FruitSeller()
    {
        fruit = "apple";
        myMoney = 0;
        numOfFruit = 10;
        fruitPrice = 1000;
    }

}

class FruitBuyer
{
    public void BuyFruit()
    {
        FruitSeller fruitSeller = new FruitSeller();
        fruitSeller.fruitPrice = 500;
    }
}
```

```
class FruitSeller
{
    private string fruit;
    private int myMoney;
    private int numOfFruit;
    private int fruitPrice;

    public FruitSeller()
    {
        fruit = "apple";
        myMoney = 0;
        numOfFruit = 10;
        fruitPrice = 1000;
    }

}

class FruitBuyer
{
    public void BuyFruit()
    {
        FruitSeller fruitSeller = new FruitSeller();
        fruitSeller.fruitPrice = 500;
    }
}
```

struct System.Int32
부호 있는 32비트 정수를 나타냅니다. 이 유형에 대 한.NET Framework 소스 코드를 찾아보려면 참조는 Reference Source합니다.

보호 수준 때문에 'FruitSeller.fruitPrice'에 액세스할 수 없습니다.

생성자

- Getter & Setter
  - 변수를 바로 수정할 수는 없지만 변수 변경 메서드를 만들고, 메서드를 호출해 변경

```
private string fruit;
private int myMoney;
private int numOfFruit;
private int fruitPrice;

// Getter
public string getFruitName() { return this.fruit; }
public int getMoney() { return this.myMoney; }
public int getNumOfFruit() { return this.numOfFruit; }
public int getFruitPrice() { return this.fruitPrice; }

// Setter
public void setFruitName(string name){ this.fruit = name; }
public void setMoney(int money) { this.myMoney = money; }
public void setNumOfFruit(int num) { this.numOfFruit = num; }
```

생성자

- Getter & Setter 쉽게 만드는 방법

```
private int [변수 이름];
public int [속성 이름]
{
    get { return [변수 이름]; }
    set { [변수 이름] = value; }
}
```

```
[인스턴스 이름].[속성 이름]          // 겟터 호출
[인스턴스 이름].[속성 이름] = [값]    // 셋터 호출
```

```
private string fruit;
private int myMoney;
private int numOfFruit;
private int fruitPrice;

public string FruitName
{
    get { return fruit; }
    set { fruit = value; }
}
```

```
public void test()
{
    FruitSeller appleSeller = new FruitSeller();

    string name = appleSeller.FruitName;
    appleSeller.FruitName = "orange";
}
```

생성자

- 생성자 오버로딩
  - 인스턴스마다 다른 값을 가질 경우를 고려
  - 오버로딩 기능을 사용하여 여러 개의 생성자를 생성

```
public void test()
{
    FruitSeller appleSeller = new FruitSeller();
    FruitSeller orangeSeller = new FruitSeller("orange", 0, 20, 2000);
}
```

```
class FruitSeller
{
    private string fruit;
    private int myMoney;
    private int numOfFruit;
    private int fruitPrice;

    public FruitSeller()
    {
        fruit = "apple";
        myMoney = 0;
        numOfFruit = 10;
        fruitPrice = 1000;
    }
    public FruitSeller(string name, int myMoney,int numOfFruit, int price)
    {
        this.fruit = name;
        this.myMoney = myMoney;
        this.numOfFruit = numOfFruit;
        this.fruitPrice = price;
    }

    public FruitSeller(int numOfFruit, int price)
    {
        this.fruit = "apple";
        this.myMoney = 0;
        this.numOfFruit = numOfFruit;
        this.fruitPrice = price;

    }
```

소멸자

- 소멸자 (Destructor)
  - 인스턴스가 소멸될 때에 호출
  - 변수의 불사용이 확실할 때 객체 소멸시키며 소멸자 호출(객체 소멸시기 불명확)
  - 소멸자 생성 규칙
    - 이름은 클래스 이름 앞에 ~ 기호 붙음
    - 접근 제한자 사용 하지 않음
    - 반환과 관련된 선언 하지 않음
    - 매개변수와 관련된 선언 하지 않음
    - 하나의 클래스에는 하나의 소멸자만
  - 소멸자의 형태

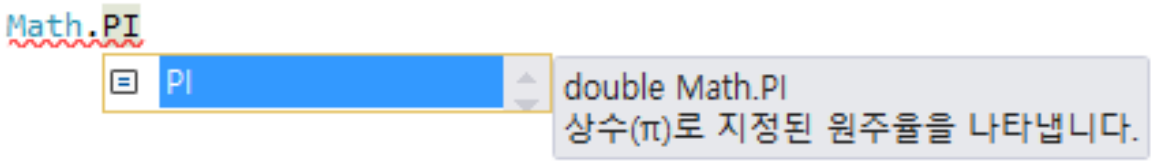
```
~[클래스 이름]()
{

}
```

그림 6-13 소멸자 형태

클래스 상수

- 일반적인 변수는 값 계속 변경 가능, 상수로 선언된 변수는 값 변경 불가능



코드 6-22 상수 변경 /6장/Constants

```
static void Main(string[] args)
{
    Math.PI = 20;

    int r = 10;
    Console.WriteLine("원의 둘레: " + (2 * Math.PI * r));
    Console.WriteLine("원의 넓이: " + (Math.PI * r * r));
}
```

```
Math.PI = 20;
```

double Math.PI
상수(π)로 지정된 원주율을 나타냅니다.

오류:
할당식의 왼쪽은 변수, 속성 또는 인덱서여야 합니다.

클래스 상수

- 상수 만들기

코드 6-23 상수 생성 /6장/Constants

```
class MyMath
{
    public const double PI = 3.141592;
}
```

코드 6-24 메서드 내부에서 상수 사용 /6장/Constants

```
static void Main(string[] args)
{
    const int value = 10;

    Console.WriteLine(value);
}
```

클래스 상수

- readonly 키워드
  - 읽기 전용 변수
  - 변수 선언 시점과 생성자 메서드에서만 값 변경 가능(이 외에는 오류 발생)

코드 6-25 readonly 키워드 /6장/Constants

```
class Product
{
    private static int count;
    public readonly int id;
    public string name;
    public int price;

    public Product(string name, int price)
    {
        id = count++;
        this.name = name;
        this.price = price;
    }
}
```

readonly 키워드를 지정했습니다.

생성자에서는 readonly 키워드를 적용한 변수를 변경할 수 있습니다.

```
product.id = 10;
```

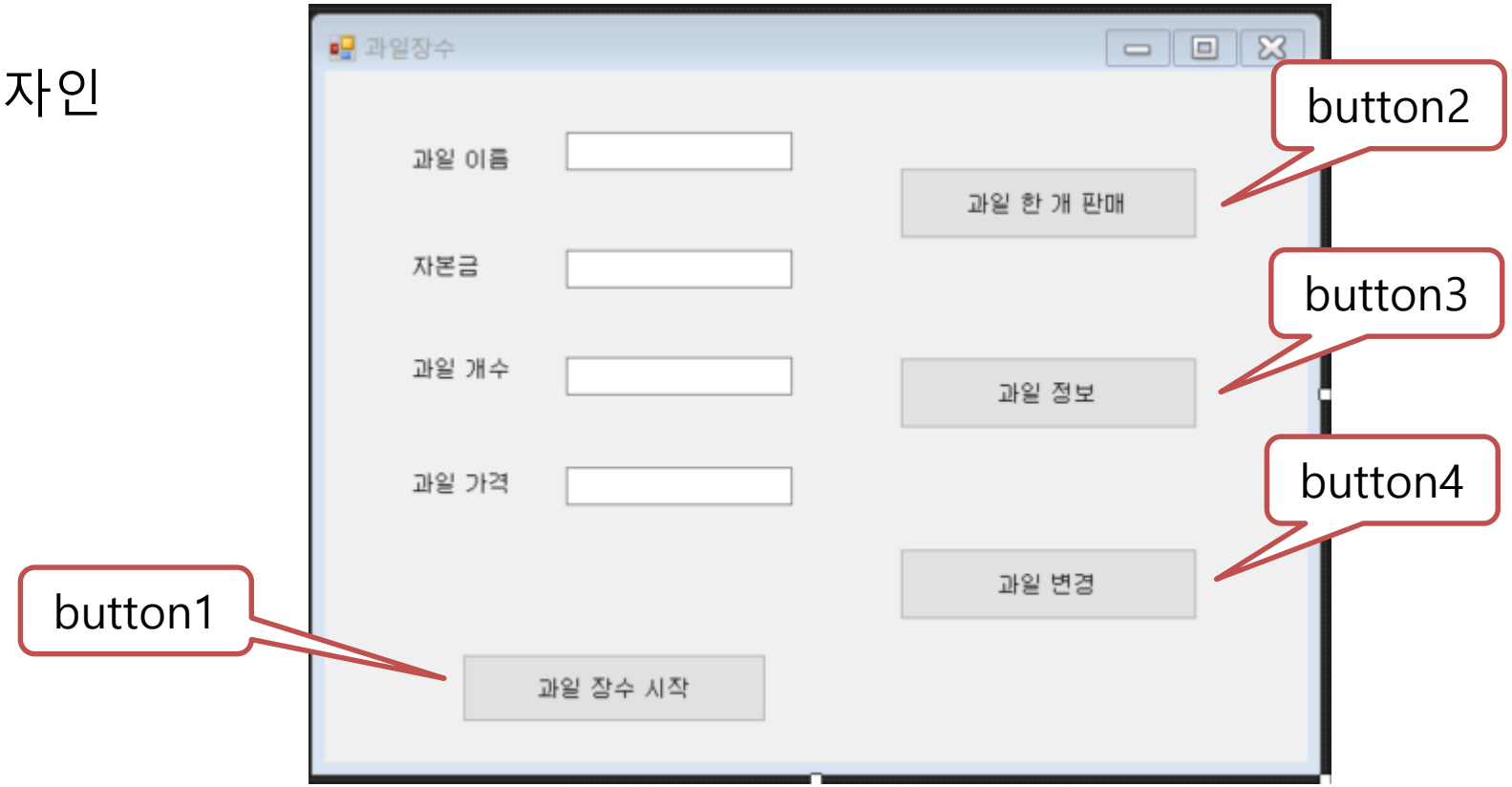
int Product.id

오류:
읽기 전용 필드에는 할당할 수 없습니다. 단 생성자 또는 변수 이니셜라이저에서는 예외입니다.

실습

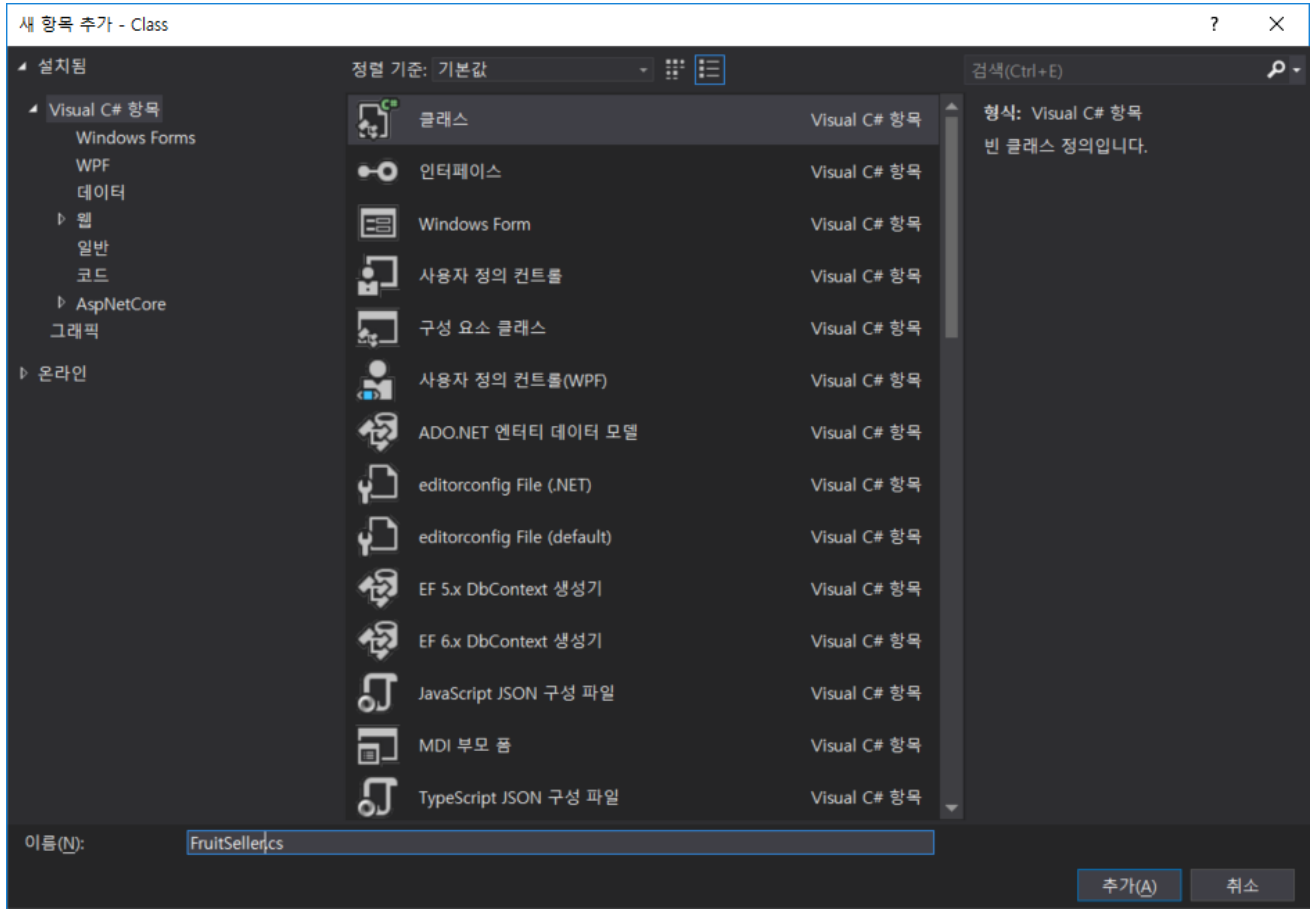
- 과일 장수 실습

1) 윈도우 폼 디자인



실습

2) 클래스 추가


새 항목 추가 - Class
설치됨
Visual C# 항목
Windows Forms
WPF
데이터
웹
일반
코드
AspNetCore
그래픽
온라인
정렬 기준: 기본값
검색(Ctrl+E)
클래스 Visual C# 항목
인터페이스 Visual C# 항목
Windows Form Visual C# 항목
사용자 정의 컨트롤 Visual C# 항목
구성 요소 클래스 Visual C# 항목
사용자 정의 컨트롤(WPF) Visual C# 항목
ADO.NET 엔터티 데이터 모델 Visual C# 항목
editorconfig File (.NET) Visual C# 항목
editorconfig File (default) Visual C# 항목
EF 5.x DbContext 생성기 Visual C# 항목
EF 6.x DbContext 생성기 Visual C# 항목
JavaScript JSON 구성 파일 Visual C# 항목
MDI 부모 폼 Visual C# 항목
TypeScript JSON 구성 파일 Visual C# 항목
형식: Visual C# 항목
빈 클래스 정의입니다.
이름(N): FruitSeller.cs
추가(A)
취소

실습

3) 클래스 코드 입력

```
class FruitSeller
{
    private string fruit;
    private int myMoney;
    private int numOfFruit;
    private readonly int fruitPrice;

    public string FruitName
    {
        get { return fruit; }
        set { fruit = value; }
    }

    public int MyMoney
    {
        get { return myMoney; }
        set { myMoney = value; }
    }

    public int NumOfFruit
    {
        get { return numOfFruit; }
        set { numOfFruit = value; }
    }
```

```
    public int FruitPrice
    {
        get { return fruitPrice; }
    }


    public FruitSeller()
    {
        fruit = "apple";
        myMoney = 0;
        numOfFruit = 10;
        fruitPrice = 1000;
    }
    public FruitSeller(string name, int myMoney, int numOfFruit, int price)
    {
        this.fruit = name;
        this.myMoney = myMoney;
        this.numOfFruit = numOfFruit;
        this.fruitPrice = price;
    }

    public FruitSeller(int numOfFruit, int price)
    {
        this.fruit = "apple";
        this.myMoney = 0;
        this.numOfFruit = numOfFruit;
        this.fruitPrice = price;
    }
```

실습

4) 폼 코드 입력

```
public partial class Form1 : Form
{
    FruitSeller fruitSeller = null;

    public Form1()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        string fruitName = textBox1.Text;
        int myMoney = int.Parse(textBox2.Text);
        int numOfFruit = int.Parse(textBox3.Text);
        int price = int.Parse(textBox4.Text);

        fruitSeller = new FruitSeller(fruitName, myMoney, numOfFruit, price);
        MessageBox.Show("과일장수 시작");

    }
```

실습

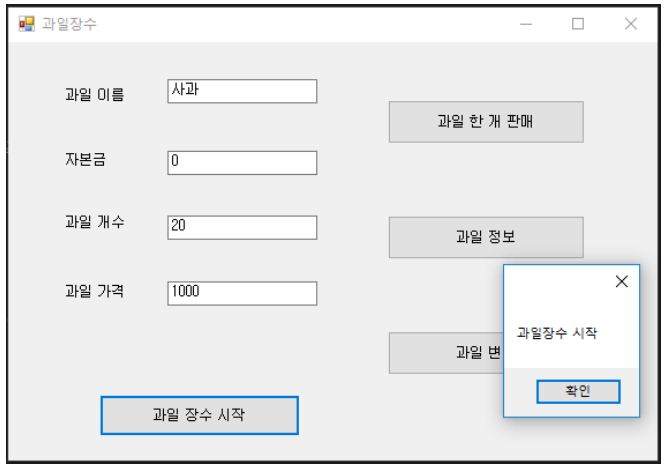
```
private void button2_Click(object sender, EventArgs e)
{
    if (fruitSeller == null)
    {
        MessageBox.Show("과일장수 시작 버튼을 눌러 주세요.");
    }
    else
    {
        fruitSeller.NumOfFruit--;
        fruitSeller.MyMoney += fruitSeller.FruitPrice;
        MessageBox.Show("과일을 한 개 판매했습니다.");
    }
}
```

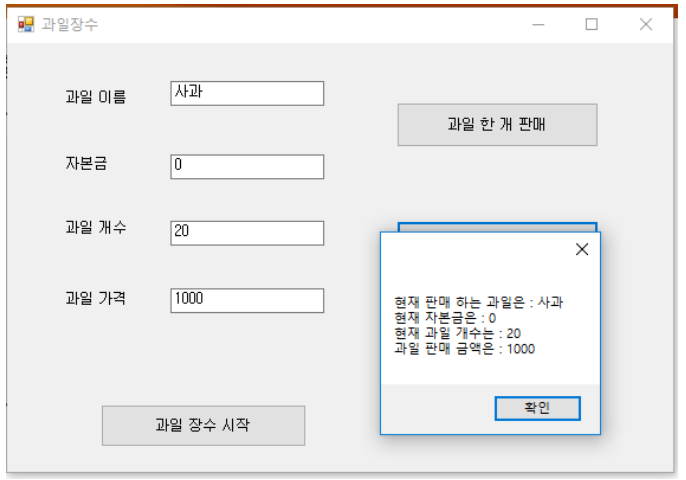
```
private void button4_Click(object sender, EventArgs e)
{
    if (fruitSeller == null)
    {
        MessageBox.Show("과일장수 시작 버튼을 눌러 주세요.");
    }
    else
    {
        string newName = textBox1.Text;
        fruitSeller.FruitName = newName;
        MessageBox.Show("과일 이름이 변경되었습니다.");
    }
}
```

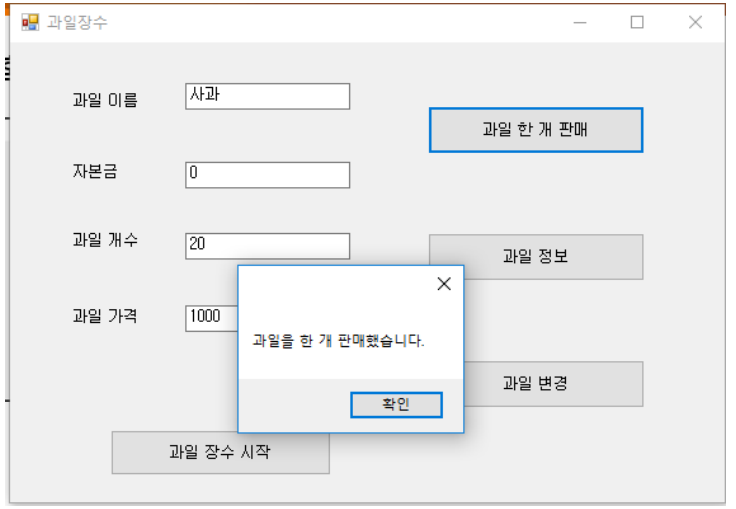
```
private void button3_Click(object sender, EventArgs e)
{
    if (fruitSeller == null)
    {
        MessageBox.Show("과일장수 시작 버튼을 눌러 주세요.");
    }
    else
    {
        string fruitName = fruitSeller.FruitName;
        int myMoney = fruitSeller.MyMoney;
        int numOfFruit = fruitSeller.NumOfFruit;
        int price = fruitSeller.FruitPrice;
        MessageBox.Show("현재 판매 하는 과일은 : " + fruitName
            + "₩n현재 자본금은 : " + myMoney
            + "₩n현재 과일 개수는 : " + numOfFruit
            + "₩n과일 판매 금액은 : " + price);
    }
}
```

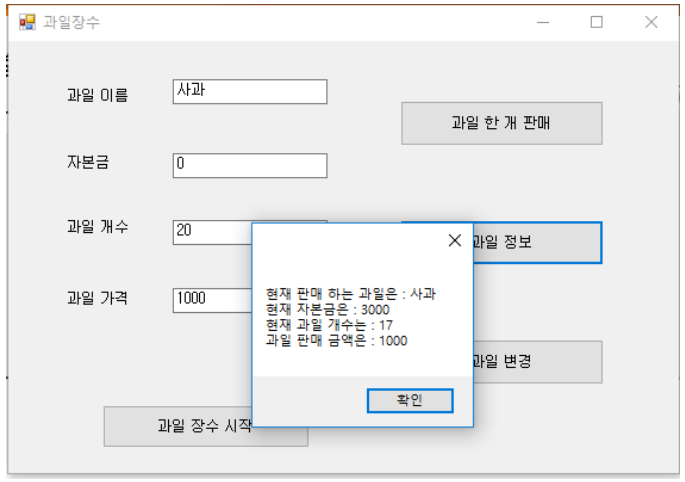
실습

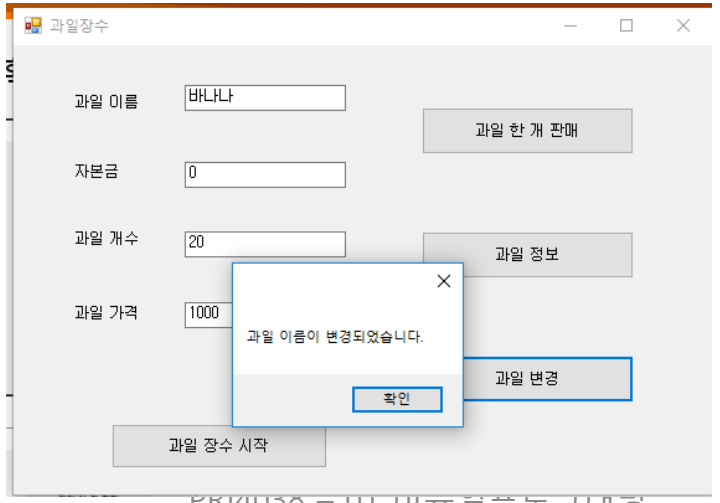
5) 실행 결과 확인

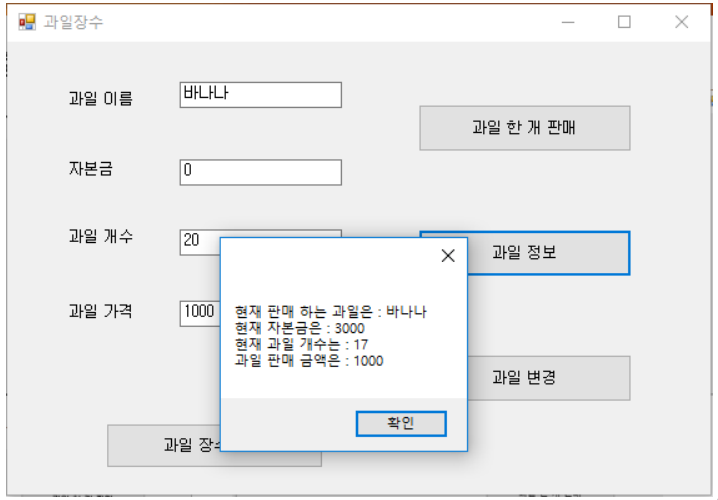

과일장수
과일 이름
사과
과일 한 개 판매
자본금
0
과일 개수
20
과일 정보
과일 가격
1000
과일 변
과일장수 시작
확인
과일 장수 시작


과일장수
과일 이름
사과
과일 한 개 판매
자본금
0
과일 개수
20
과일 가격
1000
현재 판매 하는 과일은 : 사과
현재 자본금은 : 0
현재 과일 개수는 : 20
과일 판매 금액은 : 1000
확인
과일 장수 시작


과일장수
과일 이름
사과
과일 한 개 판매
자본금
0
과일 개수
20
과일 정보
과일 가격
1000
과일을 한 개 판매했습니다.
확인
과일 변경
과일 장수 시작


과일장수
과일 이름
사과
과일 한 개 판매
자본금
0
과일 개수
20
과일 정보
과일 가격
1000
현재 판매 하는 과일은 : 사과
현재 자본금은 : 3000
현재 과일 개수는 : 17
과일 판매 금액은 : 1000
확인
과일 변경
과일 장수 시작


과일장수
과일 이름
바나나
과일 한 개 판매
자본금
0
과일 개수
20
과일 정보
과일 가격
1000
과일 이름이 변경되었습니다.
확인
과일 변경
과일 장수 시작


과일장수
과일 이름
바나나
과일 한 개 판매
자본금
0
과일 개수
20
과일 정보
과일 가격
1000
현재 판매 하는 과일은 : 바나나
현재 자본금은 : 3000
현재 과일 개수는 : 17
과일 판매 금액은 : 1000
확인
과일 변경