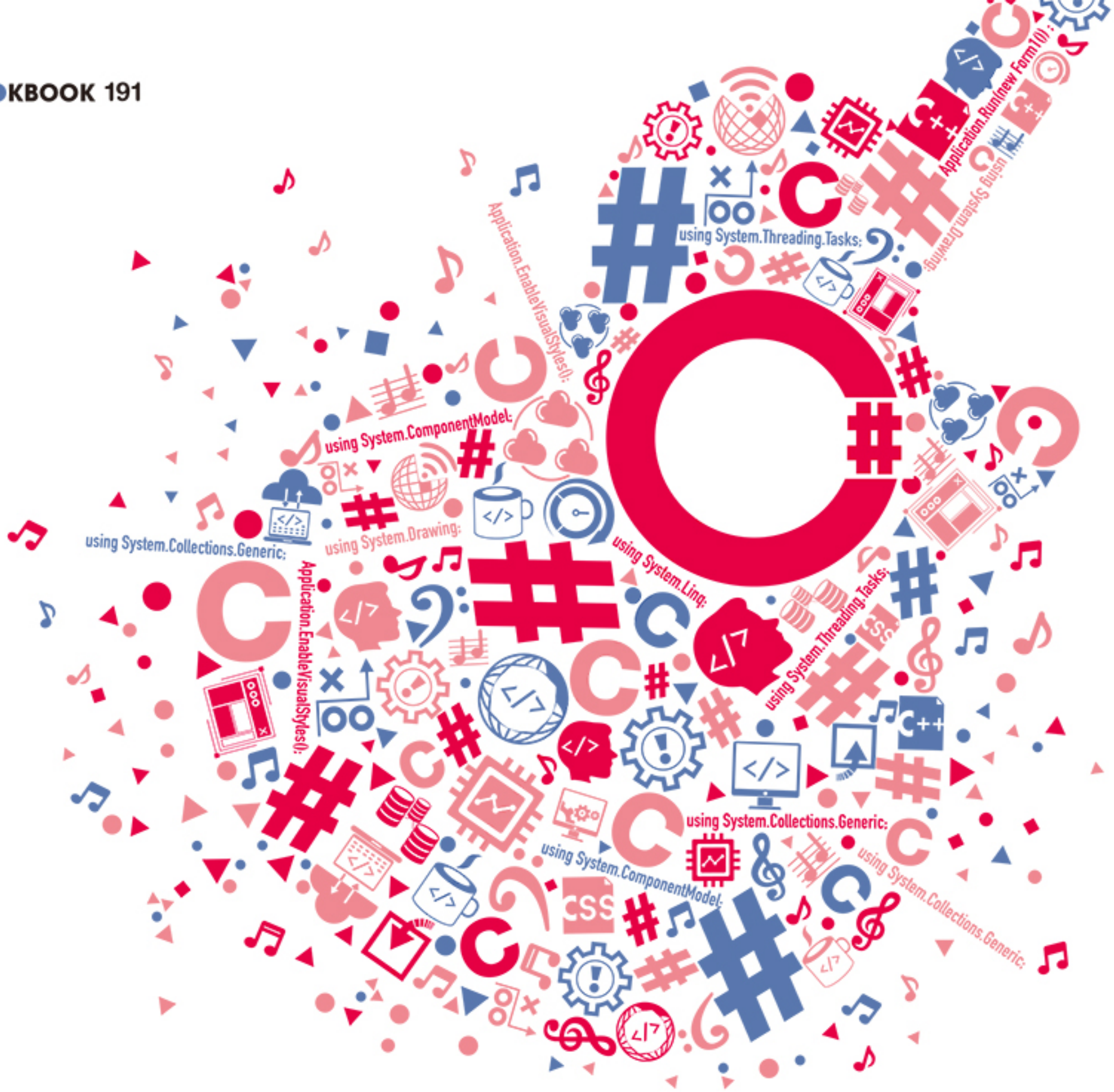
IT COOKBOOK 191



원리를 알면 IT가 맛있다

C# 프로그래밍

프로그래밍 기초부터 객체 지향 핵심까지

윤인성 지음

한빛아카데미 Hanbit Academy, Inc.

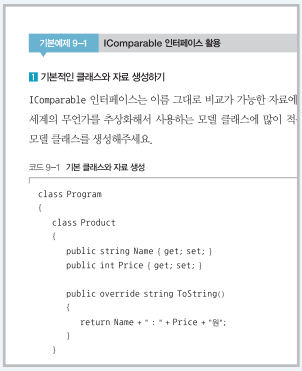
누구를 위한 책인가

C# 프로그래밍 언어를 처음 배우는 학부생을 대상으로 하며, 처음 프로그래밍을 공부한다고 해도 따라 할 수 있게 구성했습니다. 또한 클래스 기반의 객체 지향 언어를 최대한 일반화해서 다루었기 때문에 이후에 다른 객체 지향 언어를 공부할 때에도 많은 도움이 될 것입니다. C# 프로그래밍 언어의 기초부터 가장 기본적인 프레임워크(윈도 폼)를 다루는 방법까지 다룹니다. 따라서 사전에 알아야 하는 내용은 없습니다. 다만 다른 프로그래밍 언어를 공부한 경험이 있다면 내용을 더 쉽게 이해할 수 있을 것입니다.

강의 보조 자료

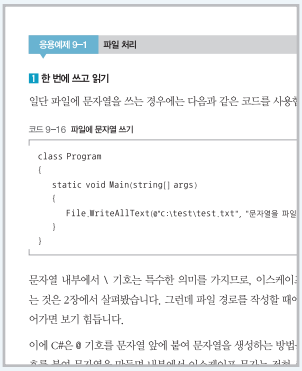
한빛 홈페이지(http://www.hanbit.co.kr)에서 '교수회원'으로 가입하신 분은 인증 후 교수용 강의 보조 자료를 제공받으실 수 있습니다. 한빛 홈페이지 우측 상단의 〈교수회원전용〉 아이콘을 클릭해 주세요.

이 책의 구성요소



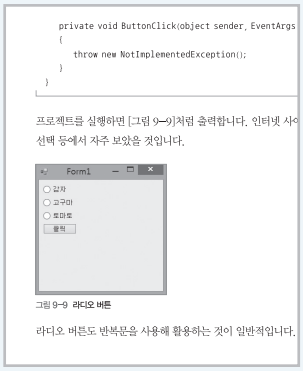
기본예제

프로그래밍 기본 개념과 문법을 익힐 수 있습니다.



응용예제

기본예제에서 배운 내용을 응용하고, 알고 있으면 유용한 기능을 추가로 익힐 수 있습니다.



윈도 폼

5장부터 각 장의 이론을 윈도 폼에 적용하여 비주얼한 실행 결과를 확인할 수 있습니다.

무엇을 다루는가

1장 : C# 프로그래밍 첫걸음

C#의 탄생과 발전, C#으로 할 수 있는 일을 알아본 후 실습을 위한 환경을 구축해봅니다.

1부(2장~4장) : 프로그래밍 기초

대부분의 프로그래밍 언어에서 공통적으로 사용되는 용어, 기본 자료형, 구문 등의 기초 내용을 알아봅니다. 조금 딱딱하고 지루할 수 있어 간단한 구문은 그림으로 표현했습니다. 다른 프로그래밍 언어를 공부한 적이 있다면 가볍게 복습해도 좋습니다.

2부(5장~8장) : 클래스와 객체 지향

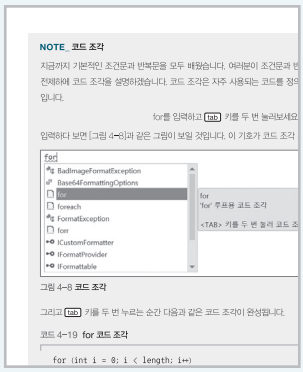
C#은 클래스 기반의 객체 지향 언어입니다. 클래스를 사용자 정의 자료형으로 정의하는 방법을 살펴보고, 객체 지향 언어의 가장 큰 특징인 상속과 다형성을 알아봅니다. 클래스를 알아야 C#에서 제공하는 다양한 프레임워크(윈도 폼, WPF, ASP.NET 등)를 활용할 수 있습니다.

3부(9장~12장) : C# 프로그래밍 고급

2부까지 배운 내용만으로도 일반적인 응용 프로그램을 만들 수 있지만 좀 더 세련되고 안정적인 코드를 작성하기 위한 내용을 알아봅니다. 인터페이스, 예외 처리, 델리게이트, 람다, Linq 등을 알아봅니다.

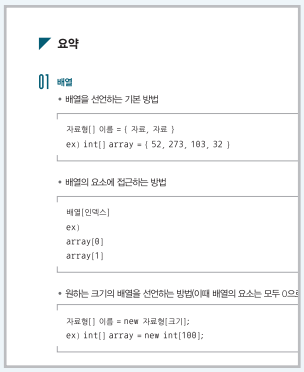
4부(13장) : 프로젝트

배운 내용을 모두 종합해서 도서 관리 프로그램을 만들어 봅니다. 코드가 굉장히 길지만 지금까지 배운 내용을 합쳐 놓은 것뿐입니다.



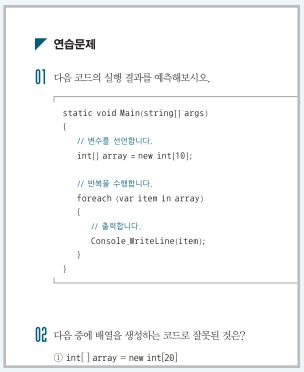
NOTE

공부하다가 궁금하거나 실수할 수 있는 부분, 잘 사용하지는 않지만 기억하면 좋은 내용을 본문과 별도로 정리해 줍니다.



요약

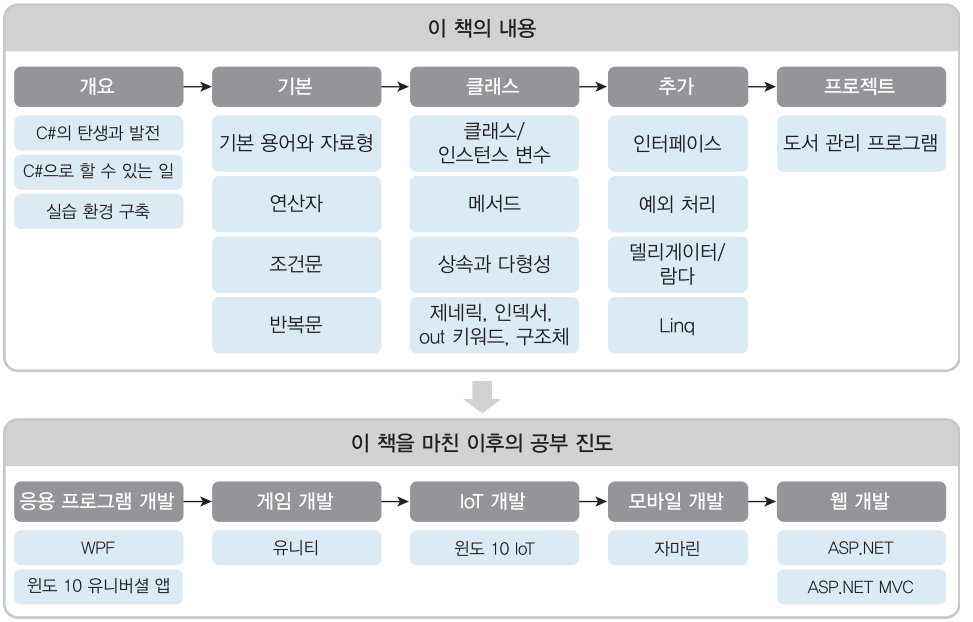
장이 끝날 때마다 핵심 내용을 요약해서 정리합니다.



연습문제

장이 끝날 때마다 문제를 해결하면서 세분화된 지식을 전체적으로 완성해볼 수 있습니다.

학습 로드맵



예제 소스

▶ 실습에 필요한 자료는 아래 주소에서 내려받을 수 있습니다. 기본예제/응용예제/윈도 폼은 개별 프로젝트에, 본문 중간의 코드와 관련 있는 것은 하나의 프로젝트에 넣었습니다.

http://www.hanbit.co.kr/exam/4204

실습 환경

▶ **개발언어** : C# 6.0(단, C#의 가장 기본적인 내용을 다루므로 C# 버전에 무관하게 학습 가능)

▶ **운영체제** : 윈도우 7 이상

▶ **개발도구** : 비주얼 스튜디오 2015(비주얼 스튜디오 2010 이후 버전 모두 가능)

Chapter 01

C# 프로그래밍 첫걸음

학습목표

- 플랫폼과 프로그래밍 언어의 관계를 이해한다.
- 라이브러리와 프레임워크의 차이를 이해한다.
- C#을 사용해 무엇을 할 수 있는지 살펴본다.
- C# 프로그래밍을 위한 실습 환경을 구축한다.

Preview

C# 프로그래밍 언어는 닷넷 플랫폼뿐만 아니라 모노 플랫폼 위에서도 작동합니다.

그리고 닷넷이 제공하는 라이브러리와 프레임워크를 활용해 쉽고 효율적으로 다양한 프로그램을 만들 수 있습니다.

위 문장을 이해하려면 플랫폼, 라이브러리, 프레임워크라는 용어를 알아야 합니다. 이 장에서는 C#을 본격적으로 배우기 앞서 먼저 알고 있어야 할 기본 용어를 알아보고, C#을 사용해 어떤 프로그램을 만들 수 있는지 알아보겠습니다.

04 실습 환경 구축

다음 장부터 C#을 본격적으로 알아볼 텐데 그 전에 C#을 공부할 수 있는 환경을 설치하는 방법을 알아봅시다.

1 개발 환경 설치

이 책에서는 C#이라는 프로그래밍 언어와 윈도 폼을 다룹니다. 그러므로 두 가지를 함께 사용할 수 있는 비주얼 스튜디오를 사용하겠습니다. 마이크로소프트 사의 비주얼 스튜디오 사이트에 들어가서 설치 파일을 다운 받아주세요. 이 책은 비주얼 스튜디오 2015의 무료 제품인 익스프레스를 기준으로 설명하지만 2010 이상 버전이면 어떤 것을 사용해도 상관없습니다. 2015의 경우 다운 받을 때에 국가 이름은 대한민국 또는 South Korea를 선택합니다.

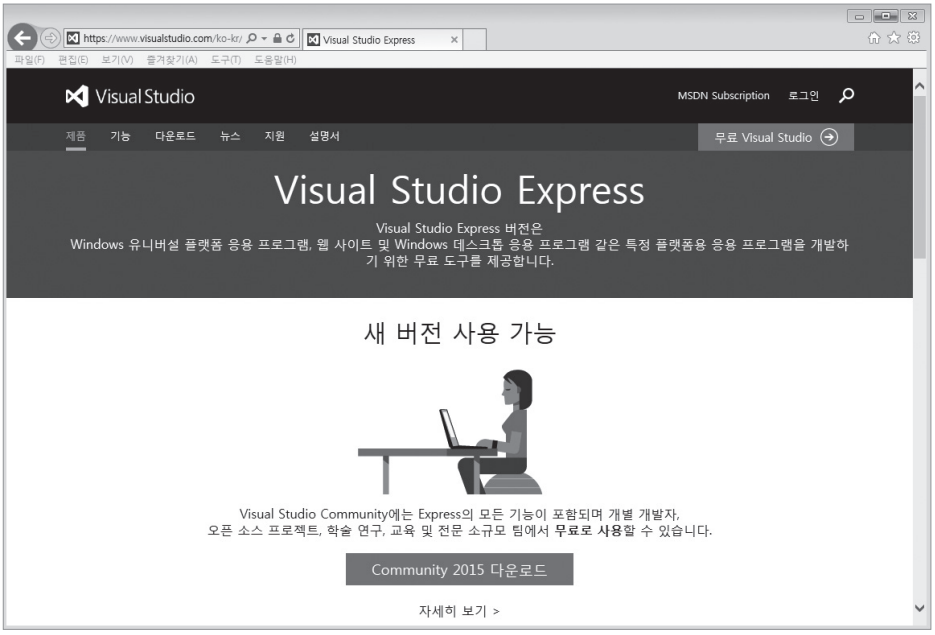


그림 1-14 비주얼 스튜디오 사이트

다운 받은 파일을 실행하면 설치 프로그램이 실행됩니다. 비주얼 스튜디오 익스프레스 2015 for Windows Desktop을 설치하는 데는 약 8기가가 필요합니다. 또한 설치하는 동안에 인터넷에 연결되어 있어야 합니다.

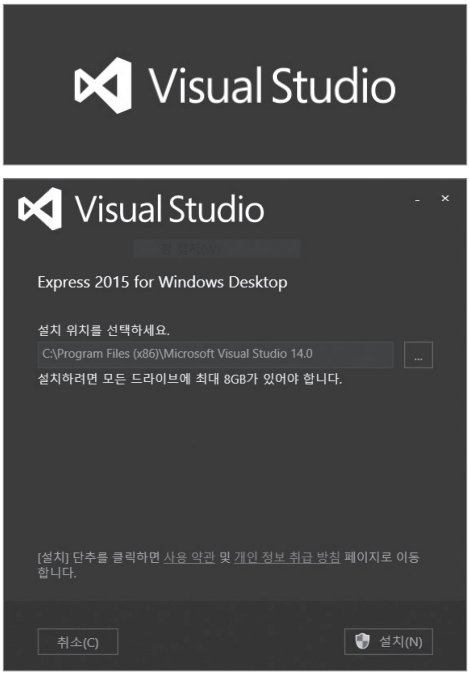


그림 1-15 **비주얼 스튜디오 익스프레스 설치 화면**

NOTE_ 윈도 XP에서는 비주얼 스튜디오 2013은 물론 2012 버전도 설치되지 않습니다. 이 경우에는 비주얼 스튜디오 2010을 설치해주세요. 추가로 컴퓨터 설치 용량에 부담이 없다면 비주얼 스튜디오 2015 커뮤니티를 설치하는 것도 추천합니다. 비주얼 스튜디오 2015 커뮤니티는 C#뿐만 아니라 파이썬, 베이직, F#, 루비, HTML, 자바스크립트, Node.js 등의 다양한 추가 개발 환경을 무료로 제공하여 C# 이외의 것을 공부할 때도 활용할 수 있답니다.

설치 버튼만 누르면 자동으로 설치되므로 설치 방법에 대한 설명은 생략하겠습니다. 개발 환경이 모두 설치되면 프로그램을 실행합니다. 처음 실행하면 개발자 로그인을 하라고 하는데요, 로그인을 안 하면 사용 기간 제한이 걸립니다. 마이크로소프트 사 계정이 없다면 먼저 회원 가입을 한 후 로그인하고, 있다면 바로 로그인해주세요.

그림 1-16 비주얼 스튜디오 익스프레스 로그인 화면

모든 과정을 제대로 수행했으면 [그림 1-17]처럼 비주얼 스튜디오 2015 익스프레스가 실행됩니다.

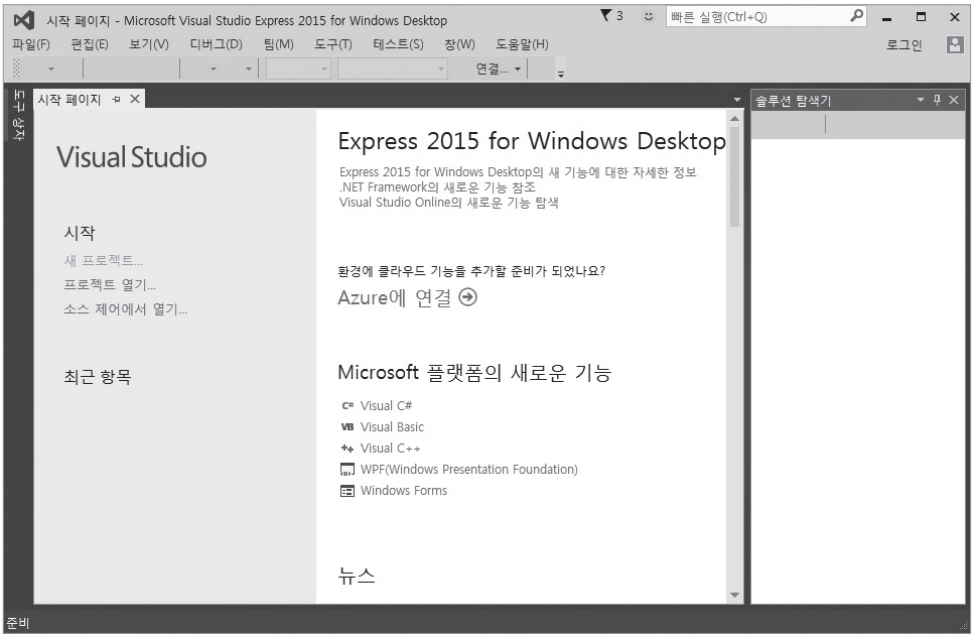


그림 1-17 비주얼 스튜디오 익스프레스 실행 화면

2 프로젝트 생성

이제 프로젝트를 생성하는 방법을 알아보겠습니다. [파일] – [새 프로젝트] 메뉴를 선택합니다. [새 프로젝트] 대화상자가 실행되면 [Visual C#] – [콘솔 응용 프로그램]을 선택하고, 프로젝트 이름에 FirstProgram을 입력한 후 [확인]을 누릅니다.

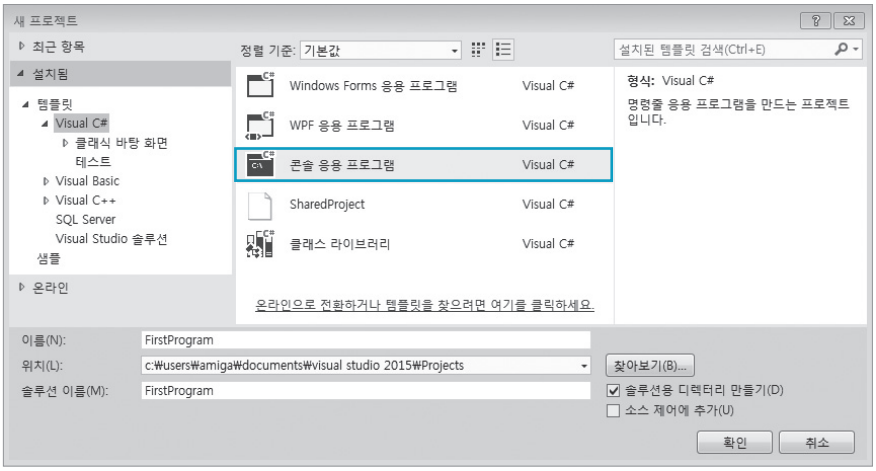


그림 1-18 [새 프로젝트] 대화상자

프로젝트를 생성하면 다음과 같이 프로젝트 폴더가 생성됩니다.

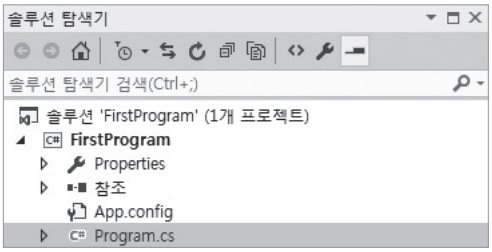


그림 1-19 솔루션 탐색기에서 생성된 프로젝트 확인

여기서 Program.cs가 프로그램의 중심이 되는 파일입니다. Program.cs 파일을 더블 클릭해서 열어보세요. [코드 1-1]과 같이 작성되어 있습니다.

코드 1-1 **기본 코드** /1장/DefaultProject

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgram
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

NOTE_ 한국어 버전은 돋움 글꼴이 기본으로 설정되어 있는데 이는 [도구] – [옵션] 메뉴를 눌러 옵션 대화상자를 실행한 후 [환경] – [글꼴 및 색]에서 변경할 수 있습니다.

3 프로젝트 실행

프로젝트를 실행하는 방법을 알아보겠습니다. 프로젝트를 실행할 때는 상단에 있는 [시작] 버튼 또는 [디버그] – [디버깅 시작] 메뉴를 눌러주세요. F5 단축키도 사용할 수 있습니다.

그림 1-20 **프로젝트 실행**

그런데 프로그램이 한 번 실행되었다가 바로 종료되어 아무것도 보지 못할 것입니다. 방금 전에 실행한 방법은 디버그 모드로 실행되는 것인데요, 이는 오류 등을 자세히 확인하기 위한 목적으로 사용되는 모드입니다. 오류 없이 정상적으로 프로그램이 끝났다고 확인되면 실행된 화면까지 꺼버려 실행 결과를 보기 힘들다는 단점이 있습니다. 그러므로 이 책의 콘솔 응용 프로그램 예제를 진행할 때는 [디버그] – [디버그하지 않고 시작] 메뉴 또는 Ctrl + F5 단축키로 실행해주세요.

4 첫 번째 프로그램

프로젝트까지 생성했으면 간단한 프로그램을 만들어보겠습니다. C#은 대소문자를 구분하므로 대소문자도 정확히 구분해서 입력해주세요.

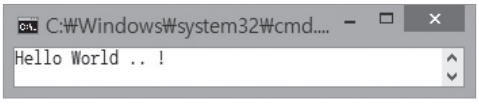
기본예제 1-1 Hello World /1장/FirstProgram

프로그래밍 언어 공부의 시작 프로그램을 대부분 Hello World라고 부릅니다. [코드 1-2]는 문자열 `Hello World .. !`를 출력합니다.

코드 1-2 Hello World 출력

```
Using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FirstProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World .. !");
        }
    }
}
```



5 오류 확인 방법

프로그래밍을 처음 공부하다 보면 사소한 오타로도 오류가 많이 발생합니다. 오류를 확인하는 간단한 방법을 알아보겠습니다. [코드 1-2]의 기본 코드를 다음과 같이 수정해봅시다.

코드 1-3 **오류가 발생하는 코드** /1장/ErrorCheck

```
static void Main(string[] args)
{
    Console.WriteRine("Hello World");
}
```

이상한 코드가 들어갔으므로 개발 환경이 오류를 감지합니다. 그리고 다음과 같이 붉은색 밑줄을 띄워줍니다.

```
static void Main(string[] args)
{
    Console.WriteRine("Hello World");
}
```

그림 1-21 **오류 부분**

코드를 실행하면 다음과 같은 오류가 나옵니다.

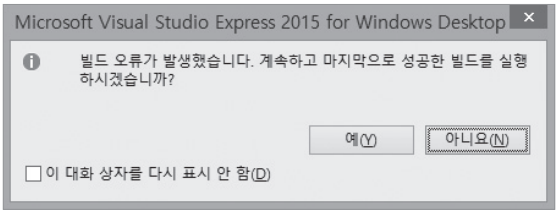


그림 1-22 **오류 발생**

이때 오류 창을 보면 해당 오류가 뜹니다. 어떤 오류인지 설명이 나오며, 오류를 더블 클릭하면 해당 줄로 이동합니다. 오류를 보면 "'WriteRine'에 대한 정의가 포함되어 있지 않습니다"라고 뜹니다. 이 부분에서 오타가 발생했다는 것입니다.

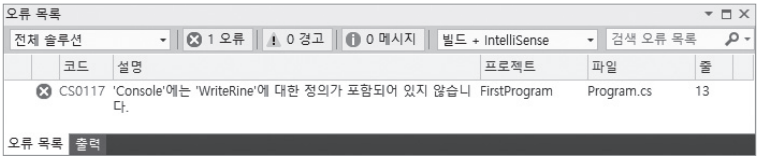


그림 1-23 **오류 확인**

처음에는 오류가 난 이유를 몰라 하루 종일 고민도 할 수 있지만 이는 당연히 거쳐야 하는 과정입니다. 교수님과 친구들에게 묻기 전에 무엇이 문제인지 직접 확인하고 고치는 습관을 길러보세요. 이러한 작업을 반복하다 보면 나중에 어떤 오류가 나와도 쉽게 고칠 수 있을 것입니다.

6 자동 완성 기능과 보조 기능

C#은 정말 방대한 기능을 가지고 있어 사소한 것까지 하나하나 다 외우기는 불가능합니다. 따라서 자동 완성 기능과 보조 기능을 사용하는 방법을 알아보겠습니다.

코드를 입력할 때에 Ctrl+Space 단축키를 눌러보세요. [그림 1-24]처럼 자동 완성 기능이 실행됩니다. 그리고 자동 완성 기능이 실행되면 현재 위치에서 사용할 수 있는 코드가 뜹니다.

그림 1-24 자동 완성 기능

또한 메서드를 사용할 때는 해당 메서드와 관련된 설명이 뜹니다.

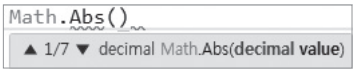


그림 1-25 메서드 설명과 사용 방법

자동 완성 기능을 활용하는 방법은 책의 내용을 진행하면서 하나하나 알아보겠습니다. 참고로 이런 자동 완성 기능을 전문 용어로 인텔리센스Intellisense라고 부릅니다. 뭔가 어렵죠? 그냥 자동 완성 기능이라고 부르겠습니다.

7 프로그래밍을 잘 하는 습관

프로그래밍 언어도 하나의 언어입니다. 말을 많이 해보는 것이 중요합니다. 일반 언어를 공부할 때보다 좋은 점은 원어민 선생님이 우리 앞에 앉아있다는 것이지요. 그 선생님은 컴퓨터라는 의식이 없는 분이라 부끄러워할 것 없이 코드를 마구 입력하고 말이 되는 문장인지 확인해 볼 수 있습니다.

그리고 자동 완성 기능을 활용해도 도움이 됩니다. [코드 1-2]를 입력하다 보면 [그림 1-26]과 같은 모습을 볼 수 있을 것입니다. [코드 1-2]에서는 `WriteLine()` 메서드를 사용해보았는데요. 비슷한 이름의 `Write()` 메서드도 있습니다.

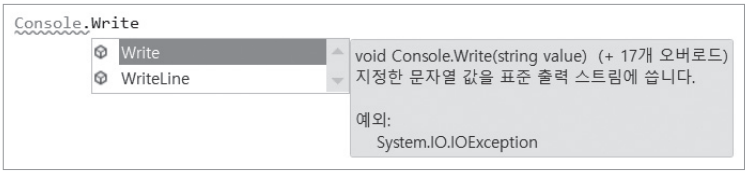


그림 1-26 **`Write()` 메서드**

`Write()` 메서드도 그냥 넘어가지 말고 한 번 사용해보세요. 사용 방법은 [그림 1-27]과 같이 자동 완성 기능이 알려줍니다.

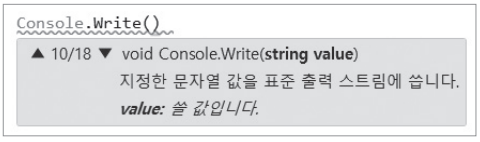


그림 1-27 **`Write()` 메서드 사용 방법**

배운 내용 이상의 것을 자동 완성 기능을 이용해서 스스로 찾아보기 바랍니다. 그리고 직접 예제를 만들어보세요. 어떤 차이가 있는지 알기 위해 시간을 들여서라도 노력하는 과정이 필요합니다.

코드 1-4 **`Write()`와 `WriteLine()` 메서드의 차이** /1장/WriteAndWriteLine

```
static void Main(string[] args)
{
    Console.Write("Write");
    Console.WriteLine("WriteLine");

    Console.WriteLine("WriteLine");
    Console.WriteLine("WriteLine");

    Console.Write("Write");
    Console.Write("Write");
}
```

코드를 실행하면 [그림 1-28]처럼 뜨는데요. Write()와 WriteLine() 메서드가 어떤 차이가 있는지 찾았나요? Write() 메서드는 문자열을 출력하고 다음 줄로 넘어가지 않고, WriteLine() 메서드는 문자열을 출력하고 다음 줄로 넘어갑니다(개행).

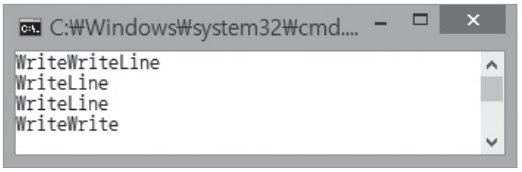


그림 1-28 **실행 결과 확인**

이런 과정들을 직접 해보기 바랍니다. 차이를 잘 모르겠으면 구글에서 검색해보세요. 이러한 과정을 하나하나 거칠 때마다 프로그래밍 실력이 빠르게 늘 것입니다.

Chapter 05

클래스 기본

학습목표

- 클래스와 관련된 기본적인 용어를 이해한다.
- 클래스를 사용하는 방법을 익힌다.
- 클래스를 생성하는 방법을 익힌다.
- 인스턴스 변수와 클래스 변수를 만드는 방법을 익힌다.

Preview

클래스는 사용자 지정 형식Custom Type을 만들 때에 사용하는 기능입니다. 이 장에서는 클래스를 만들고 사용하는 기본 방법을 먼저 알아보고, 클래스의 기본 요소인 인스턴스 변수와 클래스 변수에 대해 알아보겠습니다.

01 클래스 개요

일단 클래스가 무엇인지와 클래스와 관련된 아주 기본적인 용어를 살펴봅시다. 이번 절의 개념이 조금 어려울 수 있을 텐데요, 다음 절로 넘어가서 실제로 사용해보면 그렇게 어렵지 않다는 것을 알 수 있고, 쉽게 사용할 수 있을 것입니다. 쉽게 사용하고자 만들어진 개념이니까요.

1 사용자 정의 자료형

예를 들어 주차 관리 시스템을 지금까지 배운 내용으로 만든다고 생각해봅시다. 자동차를 3대 주차할 수 있는 주차장을 관리하려면 어떤 변수가 필요할까요? 그리고 그러한 정보를 계속 누적하고 싶다면 어떻게 해야 할까요? 일단 차량 번호(carNumber), 입차 시간(inTime), 출차 시간(outTime) 등을 사용해야 할 것입니다. 코드로 나타내면 다음과 같겠죠?

```
int carNumber;
int inTime;
int outTime;
```

하지만 한 대만 주차할 리는 없을 것입니다. 변수를 여러 개 만드는 것이 좋습니다.

```
int carNumberA;
int inTimeA;
int outTimeA;

int carNumberB;
int inTimeB;
int outTimeB;

int carNumberC;
int inTimeC;
int outTimeC;
```

지금은 차량이 3대 정도라 문제없지만 더 많아지면 배열로 관리하는 편이 훨씬 더 깔끔하겠죠?

```
int[] carNumbers = new int[10];
int[] inTimes = new int[10];
int[] outTimes = new int[10];
```

그런데 차량 번호, 입차 시간, 출차 시간은 모두 하나의 자동차(객체)가 가지는 속성입니다. 이러한 속성을 한꺼번에 묶어서 다룰 수는 없을까요? int 자료형 3개를 저장할 수 있는 새로운 자료형을 만들어버릴 수는 없는 걸까요?

클래스 기반의 객체 지향 언어는 클래스라는 기능을 사용해서 우리가 원하는 새로운 자료형을 정의합니다. 코드로 나타낸다면 다음과 같습니다. 아직 배우지 않았으므로 간단히 작성한 코드입니다.

```
class Car
{
   int carNumber;
   int inTime;
   int outTime;
}

static void Main(string[] args)
{
   Car[] cars = new Car[10];
}
```

구문은 이후에 자세히 살펴볼 예정이므로 지금은 이런 것이 있구나, 정도로만 생각해주세요. 어쨌든 이런 코드를 사용하면 Car라는 이름의 사용자 정의 자료형을 정의하게 됩니다. 지금은 int 자료형이 3개라 간단하지만, 실제 개발을 할 때는 훨씬 더 많고 복잡한 자료형들이 엮이게 됩니다.

근데 변수 여러 개 만드는 것과 큰 차이를 못 느끼겠는 걸요?

클래스에는 속성을 나타내는 변수 외에도 행위를 나타내는 메서드를 넣을 수 있습니다. 메서드를 사용하면 코드를 다음과 같이 구성할 수 있게 됩니다. 현재는 콘솔 응용 프로그램이라 큰 의미가 없지만, 이후에 버튼을 누르면 입차 시간이 찍히고, 버튼을 누르면 출차 시간이 찍히게도 만들 수 있겠죠?

```
class Car
{
    int carNumber;
    DateTime inTime;
    DateTime outTime;

    public void SetInTime()
    {
        this.inTime = DateTime.Now;
    }

    public void SetOutTime()
    {
        this.outTime = DateTime.Now;
    }
}
```

```
static void Main(string[] args)
{
    Car car = new Car();
    car.SetInTime();
    car.SetOutTime();
}
```

2 클래스와 인스턴스

그럼 위의 코드를 사용해 간단하게 용어를 정리하겠습니다. 클래스와 관련된 기본 용어는 다음과 같습니다.

그림 5-1 클래스와 관련된 기본 용어

클래스는 사용자 정의 자료형입니다. 그리고 이러한 자료형을 변수로 선언한 것을 인스턴스 또는 객체라고 부릅니다. 그리고 클래스 이름과 같은 메서드(클래스 이름 뒤에 괄호가 붙은 것)를 생성자라고 부릅니다.

꼭 그래야 하는 것은 아니지만, 클래스 이름은 대문자로 시작하는 것이 관례입니다. 현재 [그림 5-1]에서도 클래스 이름이 Car로, 대문자로 시작하고 있습니다.

일단 여기까지만 기억하고 곧바로 클래스를 사용해보겠습니다.

07 윈도 폼: 윈도 폼 기본 익히기

/5장/FirstFormApplication

클래스를 배웠으므로 이번 장부터는 윈도 폼을 사용하는 방법을 배웁니다. 1장에서 윈도 폼과 관련해 이미 설명했으므로 추가 설명은 생략하겠습니다. 이번 절의 윈도 폼은 내용이 조금 많은데요, 어렵지 않지만 중요한 내용이므로 확실하게 이해하고 넘어가도록 합시다.

1 프로젝트 생성

[파일]-[새 프로젝트]를 눌러줍니다. 지금까지는 콘솔 응용 프로그램만을 살펴보았는데요, 이번에는 Windows Forms 응용 프로그램을 선택해줍니다. 프로젝트의 이름은 FirstFormApplication으로 하고 프로젝트를 생성하겠습니다.

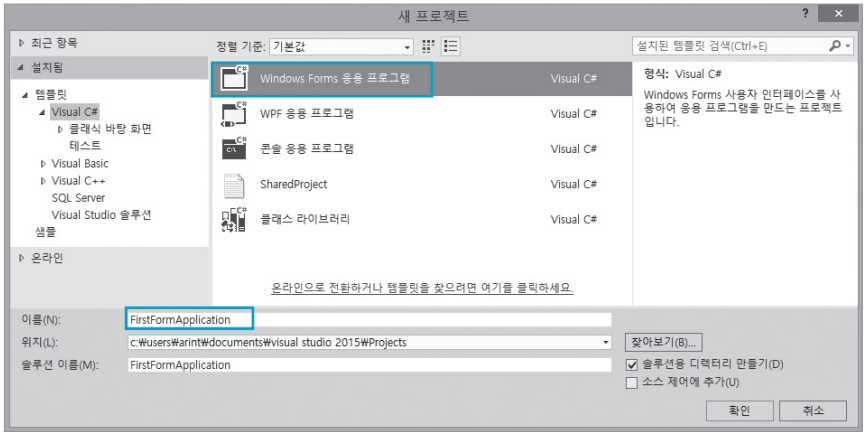


그림 5-26 새 프로젝트 대화상자

프로젝트를 생성하면 [그림 5-27]과 같은 프로젝트가 만들어집니다. 지금까지는 콘솔에서 작업했지만 윈도 폼을 사용하면 손쉽게 GUI 응용 프로그램을 만들 수 있습니다.

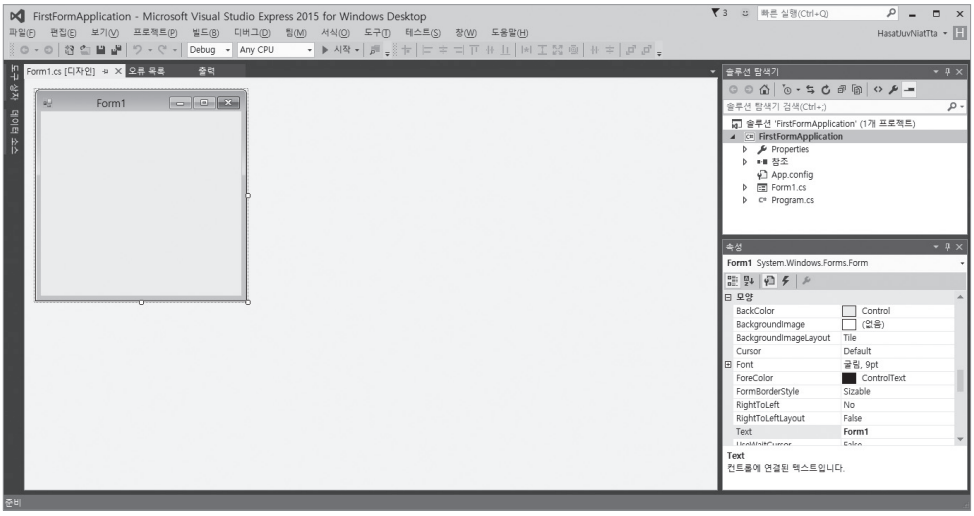


그림 5-27 **생성된 프로젝트**

프로젝트를 실행해봅시다. 프로젝트를 실행하면 다음과 같이 GUI 화면이 함께 있는 응용 프로그램이 실행됩니다. 지금은 아무것도 없지만 이제부터 차근차근 이 화면을 채워나가 보겠습니다.

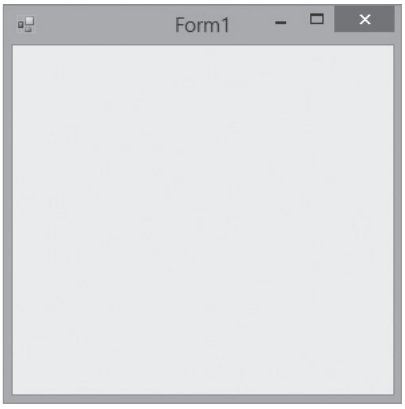


그림 5-28 **실행된 프로젝트**

2 윈도 폼의 기본 구조

솔루션 탐색기를 보면 윈도 폼 응용 프로그램은 [그림 5-29]와 같은 구조로 만들어져 있습니다.

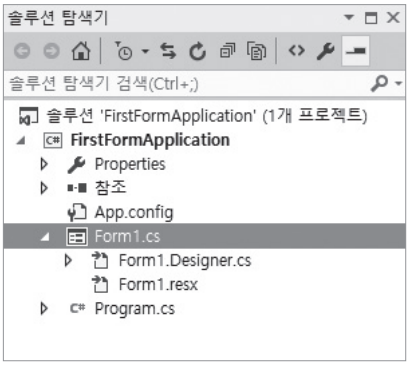


그림 5-29 **윈도 폼 응용 프로그램 프로젝트의 기본 구조**

일단 Program.cs 파일은 지금까지 살펴보았던 것처럼 Main() 메서드가 있는 파일입니다. 내용이 굉장히 단순한데요, 응용 프로그램과 관련된 설정을 수행하고 new Form1()로 Form1 클래스의 인스턴스를 생성하고 실행하는 코드입니다.

코드 5-24 **Program.cs 파일**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace FirstFormApplication
{
    static class Program
    {
        /// <summary>
        /// 해당 응용 프로그램의 주 진입점입니다.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());  ——— Form1 클래스의 인스턴스를 생성하고 실행합니다.
        }
    }
}
```

그럼 Form1 클래스가 무엇인지 살펴봐야겠죠? Form1.cs 파일을 클릭하면 다음과 같은 화면이 나옵니다. 지금까지 보았던 cs 형식의 파일과는 무언가 조금 다릅니다.

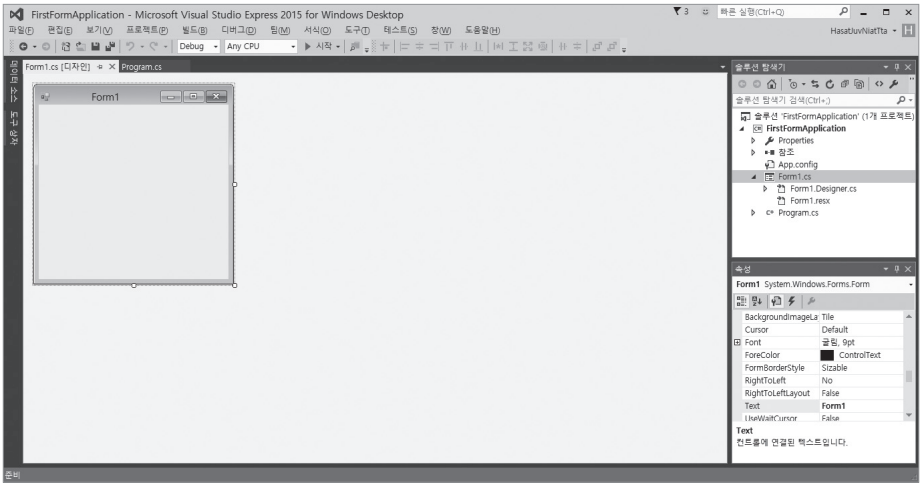


그림 5-30 **Form1.cs 파일을 열었을 때의 화면**

제목을 보면 'Form1.cs [디자인]'이라고 되어 있습니다. 이 화면은 바로 Form1 클래스의 디자인을 지정하는 부분입니다. 여기서 지정된 디자인은 [그림 5-30]에 있는 Form1.Designer.cs 파일에 자동으로 반영됩니다. 간단하게 Form1.Designer.cs 파일을 열어서 내용을 살펴봅시다.

코드 5-25 **Form1.Designer.cs 파일**

```
namespace FirstFormApplication
{
    partial class Form1
    {
        /// <summary>
        /// 필수 디자이너 변수입니다.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// 사용 중인 모든 리소스를 정리합니다.
        /// </summary>
        /// <param name="disposing">관리되는 리소스를 삭제해야 하면 true이고, 그렇지 않으면
false입니다.</param>
```

```
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form 디자이너에서 생성한 코드

        /// <summary>
        /// 디자이너 지원에 필요한 메서드입니다.
        /// 이 메서드의 내용을 코드 편집기로 수정하지 마세요.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.Text = "Form1";
        }

        #endregion
    }
}
```

Dispose라는 이름의 메서드를 기억해두면 이후에 관련된 내용을 다룰 때에 다시 생각해볼 수 있을 것입니다.

써있는 그대로 절대로 수정하지 마세요.

무언가 엄청나게 많이 작성되어 있는데요, 자동으로 작성되는 코드들이므로 우리는 이 코드를 실질적으로 만지지는 않을 것입니다. 주석에 써있는 것처럼 "이 메서드의 내용을 코드 편집기로 수정하지 마세요"라는 것을 지키는 것이지요.

그럼 어떤 코드를 만지나요?

우리가 만져야 하는 코드는 Form1.cs 파일입니다. Form1.cs 파일에서 마우스 오른쪽 버튼을 누르고 [코드 보기]를 눌러주세요.

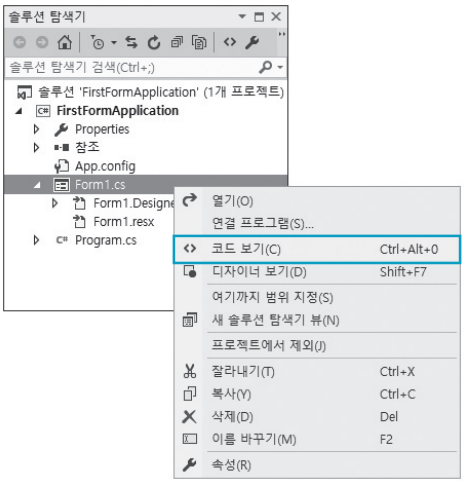


그림 5-31 **Form1.cs 파일의 코드 보기**

버튼을 누르면 다음과 같은 코드가 나옵니다. 이것이 바로 Form1.cs 파일의 본체이며 여기에 우리가 원하는 코드를 작성하게 됩니다.

코드 5-26 **Form1.cs 파일**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

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

NOTE_ partial 클래스

현재 클래스를 보면 partial이라는 키워드가 붙어있습니다. 직접 사용할 일이 거의 없는 키워드이므로 본문에서는 다루지 않았지만 윈도 폼 등의 프레임워크를 살펴볼 때는 이해를 위해 간단하게 알아두는 것이 좋습니다.

partial 키워드를 붙이면 클래스를 여러 곳에서 정의할 수 있습니다. 예를 들어 다음과 같은 형식의 클래스가 있다고 합시다.

코드 5-27 **일반적인 클래스**

```
class Example
{
    public int a;
    public int b;
}
```

지금은 간단한 클래스이므로 상관없지만 이후에 규모가 커지면 별로 보여주고 싶지 않은 지저분한 코드들이 있을 것입니다. 그리고 그러한 코드를 여러 사람이 함께 작업하거나 내가 만든 것을 외부 사람에게 준다면 지저분한 부분을 조금 감추는 것이 좋겠죠?

그럴 때 partial 키워드를 사용합니다. partial 키워드를 사용하면 [코드 5-27]의 클래스를 다음과 같이 분할할 수 있습니다.

코드 5-28 **partial 클래스**

```
partial class Example
{
    public int a;
}

partial class Example
{
    public int b;
}
```

윈도 폼에서도 Form1.cs 파일과 Form1.Designer.cs 파일에서 Form1 클래스를 나눠 작성하고 있습니다. 디자인 파일은 이후에 보면 알겠지만 버튼 하나라도 올리면 굉장히 지저분한 코드가 만들어집니다. 따라서 그런 부분을 분리해서 작성해주는 것이랍니다.

결과적으로 Form1.cs 파일에 있는 Form1 클래스와 Form1.Designer.cs 파일에 있는 Form1 클래스는 같은 것이고, 분할해서 작성하고 있는 것이라고 기억해주세요.

3 디자인 화면에서 요소 생성

화면 위에 요소를 올리는 방법부터 알아보도록 하겠습니다. 화면의 왼쪽에 있는 도구 상자를 눌러서 도구 상자를 열면 모든 Windows Forms에 사용할 수 있는 다양한 요소들이 나옵니다. 만약 도구 상자가 보이지 않는다면 메뉴의 [보기] – [도구 상자]를 눌러서 도구 상자를 열어주세요.

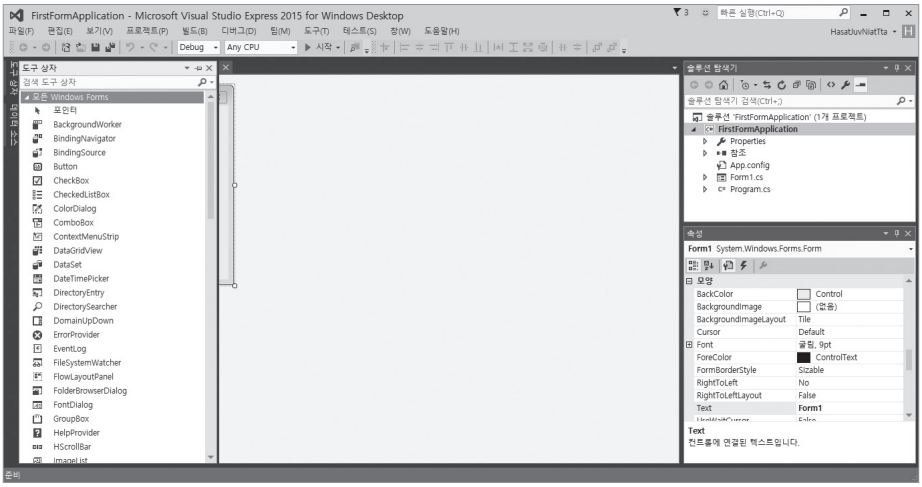


그림 5–32 도구 상자 열기

도구 상자에서 원하는 요소를 드래그해서 폼 위에 올려주면 됩니다. 여기서는 버튼을 사용해 윈도 폼의 기본적인 내용을 알아보도록 하겠습니다. Button을 폼 위로 드래그해주세요.

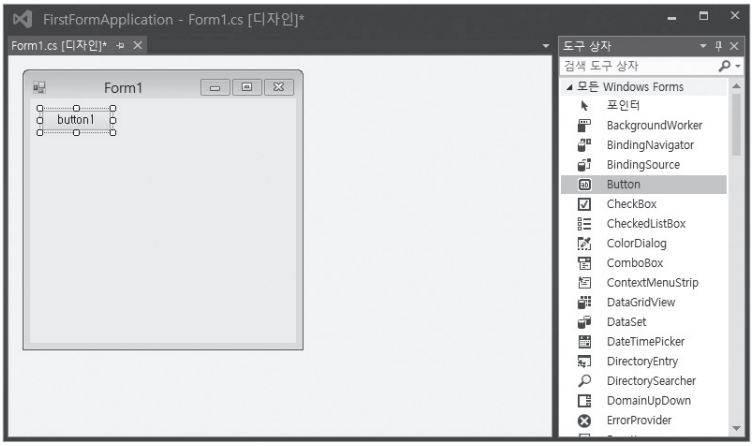


그림 5–33 도구 상자의 사용

버튼을 드래그해서 폼 위에 올리면 [그림 5-34]처럼 출력합니다. 이때 비주얼 스튜디오의 속성 화면(기본적으로는 화면의 오른쪽 아래에 나옵니다)을 보면 해당 버튼과 관련된 다양한 속성을 살펴볼 수 있습니다. 마찬가지로 속성 화면이 보이지 않는다면 메뉴의 [보기] – [속성]을 눌러 메뉴를 열어주세요.

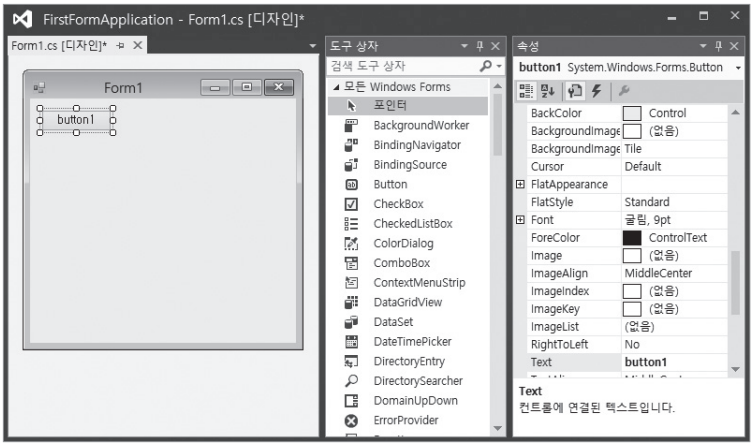


그림 5-34 **속성 메뉴 열기**

속성에서는 특정한 요소와 관련된 다양한 설정을 지정할 수 있습니다. 간단하게 Text라는 이름의 속성을 찾아보고 원하는 글자로 수정해보세요. 지정한 글자로 버튼 내부의 글자가 바뀌는 것을 확인할 수 있습니다.

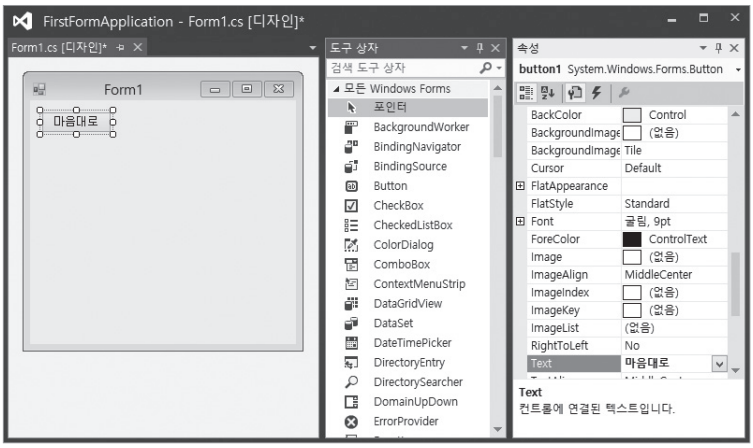


그림 5-35 **속성의 지정**

당연히 이런 코드들이 드래그했다는 이유만으로 화면에 마법처럼 나타나는 것은 아닙니다. 마우스로 드래그해서 화면에 놓으면 비주얼 스튜디오가 C# 코드를 자동으로 작성해주는 것이랍니다. 자동으로 작성된 코드는 Form1.Designer.cs라는 파일에 만들어집니다.

4 디자인 코드

디자인 코드를 살펴보기 전에 버튼을 선택하고 속성 화면에서 Name이라는 이름의 속성을 찾아보세요. 요소에 이름을 붙여주는 속성입니다. 기본적으로는 button1과 같은 이름으로 생성됩니다. 계속해서 버튼을 놓게 되면 button2, button3처럼 이름이 붙는데요, 그런 이름을 그대로 사용했다가는 나중에 "대체 11번째 버튼이 어떤 녀석이지?"하고 생각하게 될 것입니다. 간단하게 myButton이라는 이름으로 바꾸어봅시다.

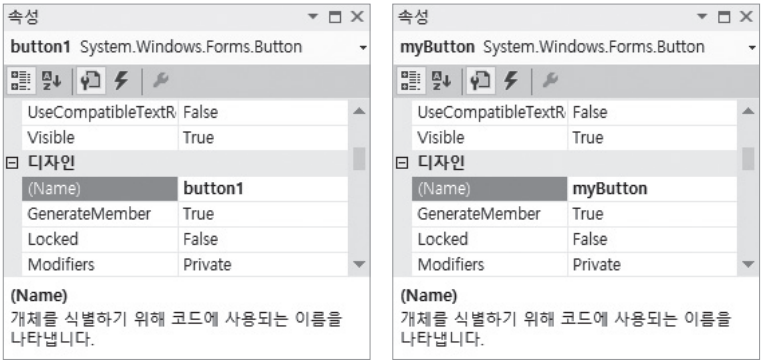


그림 5-36 **Name 속성**

지금 지정한 myButton이라는 이름을 기억하고 Form1.Designer.cs 파일을 다시 살펴봅시다. 버튼과 관련된 코드들이 추가된 것을 확인할 수 있습니다. 갑자기 코드가 많아져서 당황할 수 있는데요, 자동으로 생성되는 코드이므로 이걸 전부 외워야 한다는 생각은 접어두기 바랍니다.

코드 5-29 **버튼이 추가된 Form1.Designer.cs 파일**

```
namespace FirstFormApplication
{
    partial class Form1
    {
        /// <summary>
```

```
/// 필수 디자이너 변수입니다.
/// </summary>
private System.ComponentModel.IContainer components = null;

/// <summary>
/// 사용 중인 모든 리소스를 정리합니다.
/// </summary>
/// <param name="disposing">/* 생략 */</param>
protected override void Dispose(bool disposing)
{
    if (disposing && (components != null))
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}

#region Windows Form 디자이너에서 생성한 코드

/// <summary>
/// 디자이너 지원에 필요한 메서드입니다.
/// 이 메서드의 내용을 코드 편집기로 수정하지 마세요.
/// </summary>
private void InitializeComponent()
{
    this.myButton = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // myButton
    //
    this.myButton.Location = new System.Drawing.Point(12, 12);
    this.myButton.Name = "myButton";
    this.myButton.Size = new System.Drawing.Size(75, 23);
    this.myButton.TabIndex = 0;
    this.myButton.Text = "button1";
    this.myButton.UseVisualStyleBackColor = true;
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 12F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
```

Button 클래스의 인스턴스를 생성해서 myButton에 넣어줍니다.

myButton과 관련된 설정을 수행합니다.

```
            this.ClientSize = new System.Drawing.Size(284, 261);
            this.Controls.Add(this.myButton);
            this.Name = "Form1";
            this.Text = "Form1";
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.Button myButton;
    }
}
```

myButton을 화면에 추가합니다.

변수 myButton이 만들어졌습니다.

여기서 중요한 것은 속성 화면의 Name 속성에서 지정한 myButton이라는 글자로 변수가 만들어졌다는 것입니다. 코드를 보면 이번 장에서 배운 것처럼 인스턴스를 생성하고, 속성을 지정하는 것을 살펴볼 수 있습니다.

5 코드에서 요소의 속성 지정

디자인 화면에서 올려놓은 요소에는 이름이 붙습니다. 그리고 그 이름을 활용하면 변수에 원하는 값을 지정할 수 있습니다. 이러한 사실을 기억하고 이번 절에서는 요소의 속성을 우리가 원하는 대로 바꾸는 방법을 알아보겠습니다.

일단 Form1.cs 파일을 다시 열어봅시다. 디자인 코드(Form1.Designer.cs 파일의 Form1 클래스)에서 myButton 변수를 생성했으므로 Form1 클래스의 메서드 내부에서는 myButton 객체를 활용할 수 있답니다.

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        my
    }
}
```

myButton

그림 5-37 myButton 객체

Form1.cs 파일에서 myButton 객체의 속성을 원하는 대로 변경할 수 있습니다.

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        myButton.
    }
}
```

AccessibilityObject
AccessibleDefaultActionDescripti
AccessibleDescription
AccessibleName
AccessibleRole
AllowDrop
Anchor
AutoEllipsis
AutoScrollOffset

그림 5-38 myButton 객체의 속성

간단하게 코드로도 Text 속성을 바꿔봅시다. [코드 5-29]에서 본 디자인 코드를 확인하면서 속성을 지정해보면 더 좋을 것입니다.

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        myButton.te
    }
```

string ButtonBase.Text

ResetText
Text
TextAlign
TextChanged
TextImageRelation
UseCompatibleTextRendering

그림 5-39 myButton 객체의 Text 속성

[그림 5-39]를 보면 Text 속성에는 문자열을 넣게 되어 있습니다. Text 속성을 다음과 같이 코드에서 변경!으로 바꾸어보았습니다.

코드 5-30 **Text 속성을 코드에서 변경**

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        myButton.Text = "코드에서 변경!";
    }
}
```

이제 프로젝트를 다시 실행해보면 코드에서 변경된 값으로 버튼의 글자가 바뀌는 것을 볼 수 있습니다.

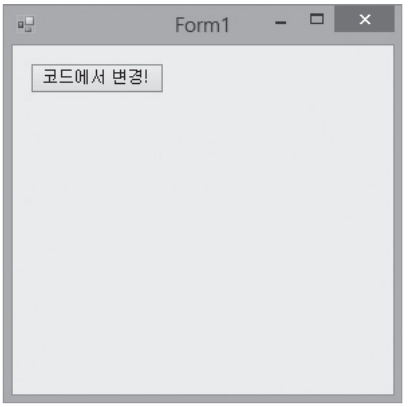


그림 5-40 **코드에서 변경한 글자**

이외의 속성도 코드에서 지정해줄 수 있답니다. 직접 만만해 보이는 이름을 넣어서 다른 속성도 지정해보기 바랍니다.

코드 5-31 **직접 다양한 속성을 지정해보세요.**

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        myButton.Text = "코드에서 변경!";
        myButton.Width = 100;
    }
}
```

NOTE_ 프레임워크를 공부할 때는

프레임워크는 내용이 정말 많습니다. 도구 상자만 보아도 엄청나게 많은 요소들이 있는데요. 그런 것을 모두 하나하나 외운다는 것은 정말 시간 낭비입니다. 또한 지금은 윈도 폼을 배우지만 학생 여러분이 이번 학기가 끝나면 윈도 폼을 더 이상 사용할 일이 없을 수도 있습니다.

이 책에서 윈도 폼을 공부하면서 하나하나 다 외운다는 생각은 절대 하지 마세요. 그냥 GUI 프레임워크에서는 대충 이런 방식으로 요소를 넣고 속성을 변경할 수 있다는 과정 자체를 공부하는 식으로 살펴보기 바랍니다. 그렇게 하면 이후에 C#에 있는 다른 GUI 프레임워크(윈도 10 유니버설 응용 프로그램, WPF 응용 프로그램)는 물론 다른 언어를 사용하는 GUI 프레임워크(아이폰 응용 프로그램, 안드로이드 응용 프로그램 등)를 할 때도 도움이 될 것입니다. 책에서도 하나하나의 요소에 집중하기보다는 대충 큰 그림을 그려나갈 것입니다.

그렇다고 안 외우면 못 만들잖아요?

물론 그렇기는 합니다. 하지만 일단 책에 있는 코드를 읽으며 외우기보다는 코드를 입력하고, 자신이 원하는 응용 프로그램을 만들면서 몸에 익히기 바랍니다. 또한 아주 복잡하고 특이한 코드는 인터넷 검색을 활용하면 됩니다.

추가적으로 자동 생성되는 코드로도 확인할 수 있는데요. 예를 들어 배경 색상 등을 바꾸도 싶을 때 코드로 어떻게 바꿀지 의문이 들면 디자인의 속성 화면에서 속성을 변경하고 디자인 코드를 확인해보세요. 마치 시험 문제에서 모르는 게 나오면 다른 문제에 이 문제에 대한 힌트가 있지 않을까, 하고 시험지를 둘러보는 것과 같습니다.

이런 식으로 무언가를 계속 스스로 시도해보고 결과를 확인하면 프레임워크는 금방 능숙하게 다룰 수 있습니다.

6 코드에서 요소 생성

그럼 우리가 코드에서 직접 버튼을 만들 수는 없을까요? 간단합니다. `Form1.designer.cs` 파일에서 자동 생성되었던 코드를 그냥 직접 쳐서 입력하면 됩니다.

간단하게 버튼을 만들어보겠습니다.

코드 5-32 **Button 클래스의 인스턴스 생성**

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        myButton.Text = "코드에서 변경!";
        myButton.Width = 100;

        Button button = new Button();
    }
}
```

이렇게 생성한 버튼을 화면에 붙이려면 Form1 클래스가 가지고 있는 Controls 속성을 사용합니다.

코드 어디를 보아도 Controls라는 속성은 없는데요?

Controls 속성은 부모로부터 상속받게 됩니다. 상속을 아직 배우지 않았으므로 그냥 요소를 추가할 때는 Controls 속성을 사용한다고 간단하게 기억하기 바랍니다. 아마 대충 지금 상속이라는 단어를 기억해두면 이후에 해당 부분을 공부할 때 이해를 도울 수 있을 것입니다.

어쨌거나 [코드 5-33]과 같이 Controls.Add() 메서드를 사용해주세요. 이후에 이 코드가 기억나지 않는다면 [코드 5-29]에 있던 디자인 코드를 살펴보면서 그때그때 다시 떠올려보기 바랍니다.

코드 5-33 **생성한 버튼 추가**

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        myButton.Text = "코드에서 변경!";
        myButton.Width = 100;

        Button button = new Button();
        Controls.Add(button);
    }
}
```

상속입니다. 이와 관련된 내용은 7장에서 다룹니다. 일단 이때 상속을 사용했었다고 대충 기억해두면 나중에 공부할 때 도움이 될 것입니다.

그럼 이제 새로 만든 button 객체에 속성을 지정해봅시다. Location 속성으로 위치를 지정해볼텐데요. Location 속성은 [그림 5-41]처럼 Point 클래스의 인스턴스입니다.

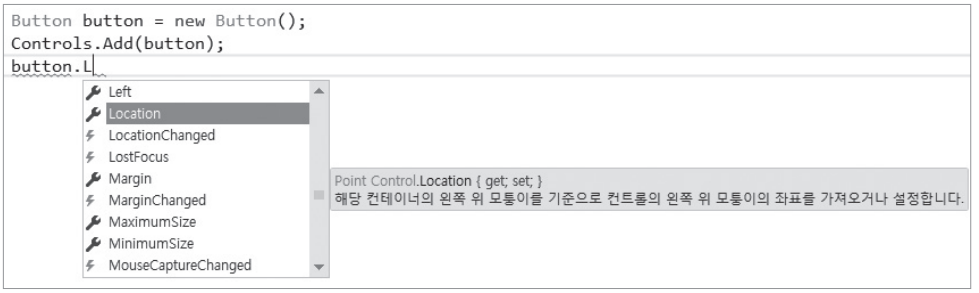


그림 5-41 **Location 속성**

Point 클래스의 인스턴스를 만들어 넣어주면 됩니다. 한 번도 사용해본 적이 없다고 생각하지 말고 일단 입력해보세요. 자동 완성 기능들이 여러분을 도와줄 것입니다.

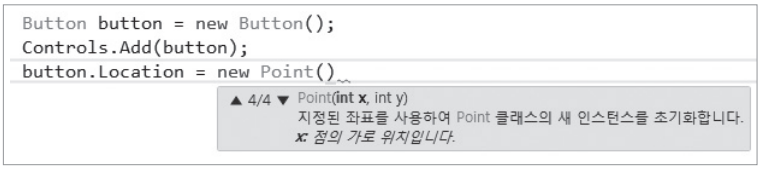


그림 5-42 **Point 클래스**

어쨌거나 이렇게 다음과 같은 코드를 작성하면 화면에 버튼이 추가됩니다.

코드 5-34 **생성한 버튼의 속성 지정**

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        myButton.Text = "코드에서 변경!";
        myButton.Width = 100;

        Button button = new Button();
        Controls.Add(button);
        button.Location = new Point(13, 13 + 23 + 3);
        button.Text = "동적 생성";
    }
}
```

그냥 디자인에서 만들어도 되는 것이 아니냐고 할 수 있는데요, 사실 뭐 맞는 말입니다. 하지만 코드를 직접 입력하면 다음과 같이 반복문 또는 조건문 등을 활용해서 한 번에 일괄 처리를 해줄 수 있답니다.

코드 5-35 **반복문으로 여러 개의 버튼 생성**

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        myButton.Text = "코드에서 변경!";
        myButton.Width = 100;

        for (int i = 0; i < 5; i++)
        {
            Button button = new Button();
            Controls.Add(button);
```

```
            button.Location = new Point(13, 13 + (23 + 3) * i);
            button.Text = "동적 생성 " + i + "번째";
            button.Width = 100;
        }
    }
}
```

모든 GUI 프레임워크는 디자인으로 만드는 방법과 코드로 만드는 방법을 같이 알아야 다양하게 활용할 수 있답니다. 디자인에서 드래그해서 요소를 만드는 방법을 정적으로 요소를 생성한다고 표현하고, 코드에서 만드는 방법을 동적으로 요소를 생성한다고 표현합니다. 물론 내부적으로는 둘 다 코드로 생성하지만요.

어쨌거나 코드를 실행하면 다음과 같이 출력합니다.

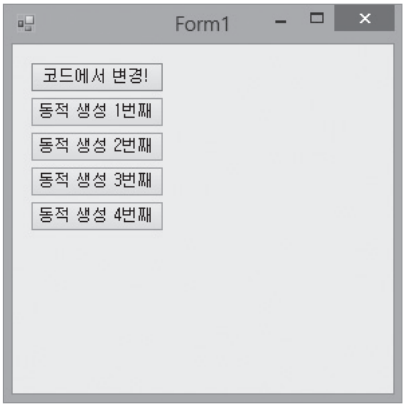


그림 5-43 **동적으로 생성된 버튼**

갑자기 콘솔을 벗어나 굉장히 많은 내용을 다루었는데요, 윈도 폼을 이렇게 많이 다루는 부분은 이번이 처음이자 마지막입니다. 이번 내용을 확실하게 기억한다면 이후에 어렵지 않을 것입니다.

다음 장에서는 메서드와 관련된 내용을 알아보고, 윈도 폼 부분에서 메서드를 활용하는 내용을 살펴보겠습니다. 지금 버튼을 만들었지만 버튼을 클릭해도 무슨 일이 일어나지 않죠? 다음 장에서 버튼을 클릭했을 때 무슨 일이 일어나게 만들어보겠습니다.

다양한 예제와 단계별 학습으로 배우는 C# 프로그래밍의 기본

언어를 "실습 환경 구축 ▸ 기본예제 ▸ 응용예제/윈도 폼 ▸ 프로젝트" 순의
부한 예제와 함께 단계적으로 익힌다.
부분의 기본서는 콘솔창에서 결과를 확인하는 예제를 중심으로 설명한 후
중에 윈도 폼을 다루어 '이런 걸은 화면에서 무엇을 할 수 있는 거지?'라는 생각을 하게 한다.
지만 이 책은 장별 이론을 윈도 폼에 적용하여 비주얼한 실행 결과를 확인할 수 있어
습자가 흥미를 잃지 않도록 도와준다. 또한 객체 지향 언어를 최대한 일반화해서 다루었기 때문에
른 객체 지향 언어를 공부할 때에도 많은 도움이 된다.

엇을 다루는가?

C# 프로그래밍 첫걸음(1장)	• C#의 탄생과 발전 과정 • C#으로 할 수 있는 일 • 실습 환경 구축	
프로그래밍 기초(2-4장)	• 기본 용어와 자료형 • 조건문	• 연산자 • 반복문
클래스와 객체 지향(5-8장)	• 클래스/인스턴스 변수 • 상속과 다형성	• 메서드 • 제네릭, 인덱서, out 키워드, 구조체
C# 프로그래밍 고급(9-12장)	• 인터페이스 • 델리게이트/람다	• 예외 처리 • Linq
프로젝트(13장)	• 도서 관리 프로그램	

프로그래밍/C#
93000
791156 842046