Chapter 10
예외 처리

Section 01 예외와 기본 예외 처리(1)

- 예외Exception : 프로그램 실행 중 프로그램이 중단되는 오류
- 예외 처리Exception Handling : 오류를 대처할 수 있게 하는 것
  - 기본 예외 처리, 고급 예외 처리

실행 결과

```
처리되지 않은 예외: System.ArgumentOutOfRangeException: 인덱스가 범위를 벗어났습니다. 인덱스는
음수가 아니어야 하며 컬렉션의 크기보다 작아야 합니다.
매개 변수 이름: index
   위치: System.ThrowHelper.ThrowArgumentOutOfRangeException(ExceptionArgument
argument, ExceptionResource resource)
   위치: System.Collections.Generic.List`1.get_Item(Int32 index)
…생략…
```

- 에러Error : 프로그램이 컴파일조차 안 되게 하는 프로그래밍 언어의 문법적 오류

Section 01 예외와 기본 예외 처리(2)

- **기본예제 10-1** IndexOutOfRangeException 기본 예외 처리 (교재 467p)

/10장/ExceptionBasic

① 예외 상황 확인하기

코드 10-1 예외 상황 확인

```
01 static void Main(string[] args)
02 {
03     string[] array = { "가", "나" };
04     Console.Write("숫자를 입력해주세요: ");
05     int input = int.Parse(Console.ReadLine());
06     Console.WriteLine("입력한 위치의 값은 '" + array[input] + "'입니다.");
07 }
```

실행 결과

```
숫자를 입력해주세요: 100

처리되지 않은 예외: System.IndexOutOfRangeException: 인덱스가 배열 범위를 벗어났습니다.
…생략…
```

Section 01 예외와 기본 예외 처리(4)

② 기본 예외 처리하기

배열 길이 확인해서 입력 값이 배열 길이 넘으면, 잘못되었다고 알려 줌

코드 10-2 기본 예외 처리

```
01 static void Main(string[] args)
02
03 {
04     string[] array = { "가", "나" };
05     Console.Write("숫자를 입력해주세요: ");
06     int input = int.Parse(Console.ReadLine());
07
08     if (input < array.Length)
09     {
10         Console.WriteLine("입력한 위치의 값은 '" + array[input] + "'입니다.");
11     }
12     else
13     {
14         Console.WriteLine("인덱스 범위를 넘었습니다.");
15     }
16 }
```

실행 결과

```
숫자를 입력해주세요: 52273
인덱스 범위를 넘었습니다.
```

Section 02 고급 예외 처리(1)

- try 키워드, catch 키워드, finally 키워드로 예외를 처리하는 방법
- 고급 예외 처리 형식(try catch finally 구문)

```
try
{
    // 예외가 발생하면
}
catch (Exception exception)
{
    // 여기서 처리합니다.
}
finally
{
    // 여기는 무조건 실행합니다.
}
```

Section 02 고급 예외 처리(2)

- catch 구문 또는 finally 구문이 필요 없다면, 해당 부분을 생략하고 사용 가능

```
try
{
    // 예외가 발생하면
}
catch (Exception exception)
{
    // 여기서 처리합니다.
}
```

```
try
{
    // 예외가 발생하면 그냥 넘어갑니다.
}
finally
{
    // 여기는 무조건 실행합니다.
}
```

Section 02 고급 예외 처리(3)

- **기본예제 10-2** Parse( ) 메서드 예외 처리 (교재 470p) /10장/ TryCatchFinallyBasic

① 예외 상황 확인하기

코드 10-3 예외 상황 확인

```
01 static void Main(string[] args)
02 {
03     Console.Write("입력: ");
04     string input = Console.ReadLine();
05
06     int index = int.Parse(input);
07     Console.WriteLine("입력 숫자: " + index);
08 }
```

실행 결과

```
입력: ㅇㅂㅇ

처리되지 않은 예외: System.FormatException: 입력 문자열의 형식이 잘못되었습니다.
    위치: System.Number.StringToNumber(String str, NumberStyles options, NumberBuffer&
number, NumberFormatInfo info, Boolean parseDecimal)
   위치: System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info)
   위치: System.Int32.Parse(String s)
…생략…
```

Section 02 고급 예외 처리(4)

② 고급 예외 처리하기

코드 10-4 고급 예외 처리

```
01 static void Main(string[] args)
02 {
03     Console.Write("입력: ");
04     string input = Console.ReadLine();
05
06     try
07     {
08         int index = int.Parse(input);
09         Console.WriteLine("입력 숫자: " + index);
10     }
11     catch (Exception exception)
12     {
13         Console.WriteLine("예외가 발생했습니다.");
14         Console.WriteLine(exception.GetType());
15     }
16     finally
17     {
18         Console.WriteLine("프로그램이 종료되었습니다.");
19     }
20 }
```

실행 결과

```
입력: ㅇㅂㅇ
예외가 발생했습니다.
System.FormatException
프로그램이 종료되었습니다.
```

NOTE (1)

■ finally 구문

- "프로그램이 종료되었습니다."라는 글자 출력 안됨

코드 10-5 finally 구문을 사용하지 않은 코드 /10장/Exceptions

```
static void Main(string[] args)
{
    Console.Write("입력: ");
    string input = Console.ReadLine();

    try
    {
        int index = int.Parse(input);
        Console.WriteLine("입력 숫자: " + index);
    }
    catch (Exception exception)
    {
        Console.WriteLine("예외가 발생했습니다.");
        Console.WriteLine(exception.GetType());
        return;
    }
    Console.WriteLine("프로그램이 종료되었습니다.");
}
```

NOTE (2)

코드 10-6 finally 구문 사용 /10장/Exceptions

```
static void Main(string[] args)
{
    Console.Write("입력: ");
    string input = Console.ReadLine();

    try
    {
        int index = int.Parse(input);
        Console.WriteLine("입력 숫자: " + index);
    }
    catch (Exception exception)
    {
        Console.WriteLine("예외가 발생했습니다.");
        Console.WriteLine(exception.GetType());
        return;
    }
    finally
    {
        Console.WriteLine("프로그램이 종료되었습니다.");
    }
}
```

return 키워드를 사용해 여기서 코드를 종료합니다.

실행 결과

```
입력: ㅇㅂㅇ
예외가 발생했습니다.
System.FormatException
프로그램이 종료되었습니다.
```

꼭 실행됨

but

- 결과가 달라지는 경우

① catch 구문 내부에서 return 키워드 만날 때

② catch 구문 내부에서 try catch 구문 사용했는데 break 또는 continue 키워드 만날 때

NOTE (3)

■ finally 구문과 return 키워드

- finally 구문 내부는 무조건 실행하고 끝낸다는 규칙
- 중간에 구문을 벗어나는 키워드들은 불가능

```
try
{

}
catch(Exception exception)
{

}
finally
{
  return;
}
```

CS0157: 제어가 finally 절의 본문을 벗어날 수 없습니다.

그림 10-1 finally 구문 내부에서의 return 키워드 오류

Section 03 예외 객체(1)

- 예외 발생 시 어떤 예외가 발생했는지와 관련된 정보 함께 전달해주는 것
- 예

```
try
{

}
catch (Exception exception)
{

}
```

Exception 클래스의 인스턴스로 예외 객체라고 부릅니다.

```
try
{

}
catch(Exception exception)
{
    exception.
}
```

그림 10-2 예외 객체의 속성과 메서드

Section 03 예외 객체(2)

- **기본예제 10-3** 예외 객체에서 정보 추출(교재 475p) /10장/ExceptionObject

```
class Program
{
    static void Main(string[] args)
    {
        Console.Write("입력: ");
        string input = Console.ReadLine();

        try
        {
            int index = int.Parse(input);
            Console.WriteLine("입력 숫자: " + index);
        }
        catch (Exception exception)
        {
            Console.WriteLine("예외가 발생했습니다.");
            Console.WriteLine(exception.GetType());
            Console.WriteLine(exception.Message);
            Console.WriteLine(exception.StackTrace);
        }
    }
}
```

Section 03 예외 객체(2)

- **기본예제 10-3** 예외 객체에서 정보 추출(교재 475p) /10장/ExceptionObject

실행 결과

```
입력: ㅇㅂㅇ
예외가 발생했습니다.
System.FormatException
입력 문자열의 형식이 잘못되었습니다.
    위치: System.Number.StringToNumber(String str, NumberStyles options, NumberBuffer&
number, NumberFormatInfo info, Boolean parseDecimal)
   위치: System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info)
   위치: System.Int32.Parse(String s)
…생략…
```

Section 04 예외 객체를 사용한 예외 구분(1)

- **기본예제 10-4** 예외 객체를 사용한 예외 구분(교재 457p)

/10장/ConditionWithExceptionObject

① 예외 상황 확인하기

코드 10-8 예외 상황 확인

```
01 static void Main(string[] args)
02 {
03     Console.Write("입력: ");
04     string input = Console.ReadLine();
05     int[] array = { 52, 273, 32, 103 };
06
07     int index = int.Parse(input);
08     Console.WriteLine("입력 숫자: " + index);
09     Console.WriteLine("배열 요소: " + array[index]);
10 }
```

실행 결과

```
입력: 10
입력 숫자: 10

처리되지 않은 예외: System.IndexOutOfRangeException: 인덱스가 배열 범위를 벗어났습니다.
…생략…
```

Section 04 예외 객체를 사용한 예외 구분(2)

② 고급 예외 처리하기 : catch 구문을 여러 개 사용 가능 즉, 예외에 따라 서로 다른 처리를 할 수 있다.

코드 10-9 고급 예외 처리

```
static void Main(string[] args)
{
   Console.Write("입력: ");
   try
   {
      string input = Console.ReadLine();
   int[] array = { 52, 273, 32, 103 };

   int index = int.Parse(input);
   Console.WriteLine("입력 숫자: " + index);
   Console.WriteLine("배열 요소: " + array[index]);
}
catch (FormatException exception)
{
```

Section 04 예외 객체를 사용한 예외 구분(3)

```
            Console.WriteLine("FormatException 발생");
            Console.WriteLine(exception.GetType() + "이 발생했습니다.");
        }
        catch (IndexOutOfRangeException exception)
        {
            Console.WriteLine("IndexOutOfRangeException 발생");
            Console.WriteLine(exception.GetType() + "이 발생했습니다.");
        }
    }
```

실행 결과

```
입력: ㅇㅂㅇ
FormatException 발생
System.FormatException이 발생했습니다.
```

실행 결과

```
입력: 52273
입력 숫자: 52273
IndexOutOfRangeException 발생
IndexOutOfRangeException이 발생했습니다.
```

■ catch 구문과 var 키워드

- catch 구문의 괄호 안에는 var 키워드 사용 불가(오류 발생)

```
try
{

}
catch (var exception)
{

}
```

Section 05 예외 강제 발생(1)

- 예외를 강제로 발생시키는 방법으로 throw 키워드 사용

```
throw new Exception();
```

- **기본예제 10-5** 예외 던지기(교재 479p) /10장/ThrowBasic

① 예외 던지기 확인하기

코드 10-10 예외 던지기 확인

```
01 class Program
02 {
03     static void Main(string[] args)
04     {
05         throw new Exception();    → 예외 발생!
06     }
07 }
```

실행 결과

```
처리되지 않은 예외: System.Exception: 'System.Exception' 형식의 예외가 Throw되었습니다.
…생략…
```

Section 05 예외 강제 발생(2)

② 강제로 던진 예외의 예외 처리하기

코드 10-11 강제로 던진 예외의 예외 처리

```
01 class Program
02 {
03     static void Main(string[] args)
04     {
05         try
06         {
07             throw new Exception();
08         }
09         catch (Exception exception)
10         {
11             Console.WriteLine("예외가 발생했습니다.");
12         }
13     }
14 }
```

실행 결과

```
예외가 발생했습니다.
```

Section 05 예외 강제 발생(3)

- **기본예제 10-6** Box 클래스 예외 관련 구현(교재 480p) /10장/ThrowWithBox

```
class Box
{
    // 변수와 속성
    private int width;
    public int Width
    {
        get { return width; }
        set
        {
            if (value > 0) { width = value; }
            else { throw new Exception("너비는 자연수를 입력해주세요"); }
        }
    }

    private int height;
    public int Height
    {
        get { return height; }
        set
        {
            if (value > 0) { height = value; }
            else { throw new Exception("높이는 자연수를 입력해주세요"); }
        }
    }

    // 생성자
    public Box(int width, int height)
    {
        Width = width;
        Height = height;
    }

    // 인스턴스 메서드
    public int Area() { return this.width * this.height; }
}

static void Main(string[] args)
{
    Box box = new Box(-10, -20);
}
```

[실행결과]
Unhandled exception. System.Exception: 너비는 자연수를 입력해주세요

NOTE

■ 사용자 정의 예외 : 다음과 같이 정형적인 형태로 생성!

코드 10-13 사용자 정의 예외 /10장/Exceptions

```
class CustomException : Exception
{
    public CustomException(string message) : base(message)
    {

    }
}

class Program
{
    static void Main(string[] args)
    {
        try
        {
            throw new CustomException("사용자 정의 예외");
        }
        catch (CustomException exception)
        {
            Console.WriteLine(exception.Message);
        }
    }
}
```

부모 생성자를 호출합니다.

예외를 강제로 발생시킵니다.

실행 결과

```
사용자 정의 예외
```

Section 06 윈도 폼: 콤보 박스, 리스트 박스, 데이터 그리드 뷰 사용하기 (1)

- 콤보 박스, 리스트 박스, 그리드는 하나의 글자를 출력하는 요소 아님
- 콤보 박스와 리스트 박스(1차원 리스트 출력 시 사용)
- 그리드(2차원 리스트 출력 시 사용)

■ 콤보 박스와 리스트 박스

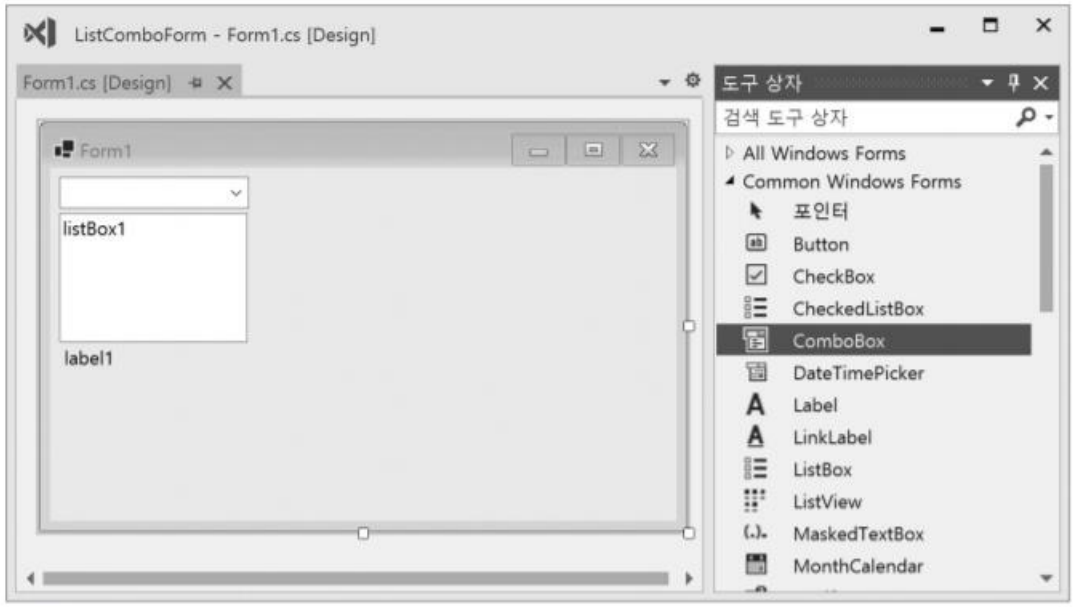

그림 10-3 디자인

- 기본 사용
  - 마우스로 클릭하면 오른쪽 위에 버튼 뜸, 버튼 누르면 결과 출력

코드 10-14 콤보 박스와 리스트 박스 생성

```
01 public Form1()
02 {
03    InitializeComponent();
04
05    var dataSource = new string[] { "고구마", "감자", "토마토" };
06    comboBox1.DataSource = dataSource;
07    listBox1.DataSource = dataSource;
08 }
```

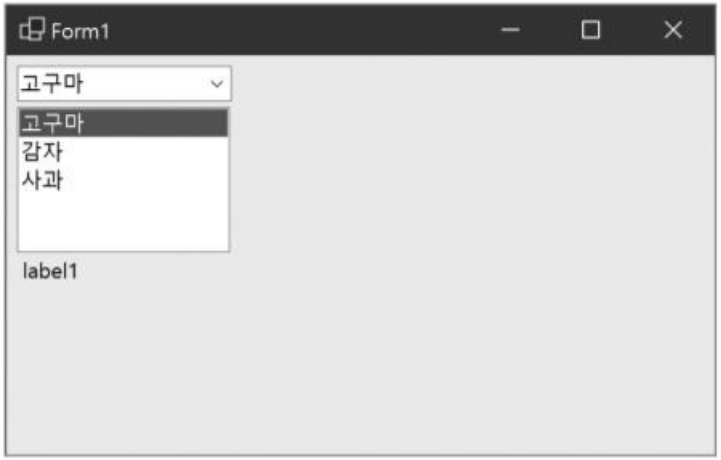

그림 10-4 콤보 박스와 리스트 박스 출력

Section 06 윈도 폼: 콤보 박스, 리스트 박스, 데이터 그리드 뷰 사용하기 (2)

코드 10-15 콤보 박스 선택 이벤트

```
01 public partial class Form1 : Form
02 {
03   public Form1()
04   {
05     InitializeComponent();
06
07     // 데이터 소스를 선택합니다.
08     var dataSource = new string[] { "고구마", "감자", "토마토" };
09     comboBox1.DataSource = dataSource;
10     listBox1.DataSource = dataSource;
11
12     // 콤보박스 선택 이벤트 연결
13     comboBox1.SelectedIndexChanged += ComboBox1_SelectedIndexChanged;
14     listBox1.SelectedIndexChanged += ListBox1_SelectedIndexChanged; ;
15   }
16   private void ComboBox1_SelectedIndexChanged(object sender, EventArgs e)
17   {
18     label1.Text = comboBox1.SelectedIndex + ": " + comboBox1.SelectedItem;
19   }
20
21   private void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
22   {
23     label1.Text = listBox1.SelectedIndex + ": " + listBox1.SelectedItem;
24   }
25 }
```

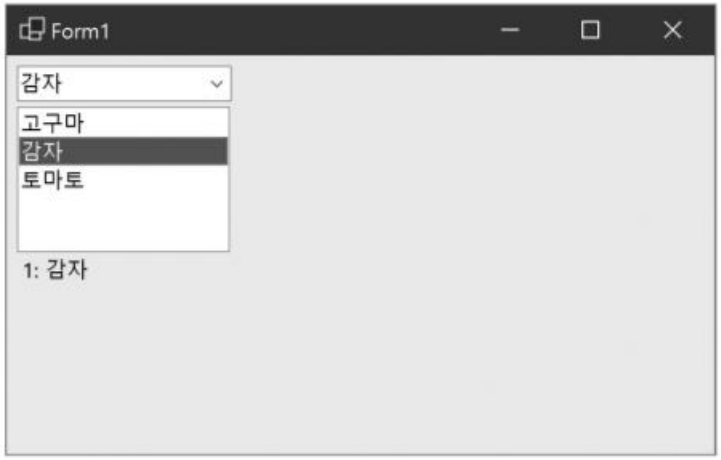

그림 10-5 이벤트 연결

Section 06 윈도 폼: 콤보 박스, 리스트 박스, 데이터 그리드 뷰 사용하기 (3)

코드 10-16 콤보 박스와 이벤트 박스 설정

```
01 public partial class Form1 : Form
02 {
03   class Product
04   {
05     public string Name { get; set; }
06     public int Price { get; set; }
07   }
08
09   public Form1()
10   {
11     InitializeComponent();
12
13     // 데이터 소스를 선택합니다.
14     var dataSource = new List<Product> {
15       new Product() { Name = "고구마", Price = 500  },
16       new Product() { Name = "감자", Price = 600  },
17       new Product() { Name = "사과", Price = 700  }
18     };
19
20     // 콤보박스 설정
21     comboBox1.DisplayMember = "Name";
22     comboBox1.ValueMember = "Price";
23     comboBox1.DataSource = dataSource;
24     comboBox1.SelectedIndexChanged += ComboBox1_SelectedIndexChanged;
25
```

```
26      // 리스트 박스 설정
27      listBox1.DisplayMember = "Name";
28      comboBox1.ValueMember = "Price";
29      listBox1.DataSource = dataSource;
30      listBox1.SelectedIndexChanged += ListBox1_SelectedIndexChanged;
31    }
32
33    private void ComboBox1_SelectedIndexChanged(object sender, EventArgs e)
34    {
35      label1.Text =
36        comboBox1.SelectedIndex
37        + ":" + comboBox1.SelectedValue
38        + ":" + ((Product)comboBox1.SelectedItem).Name;
39    }
40    private void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
41    {
42      label1.Text =
43        comboBox1.SelectedIndex
44        + ":" + comboBox1.SelectedValue
45        + ":" + ((Product)comboBox1.SelectedItem).Name;
46    }
47  }
```

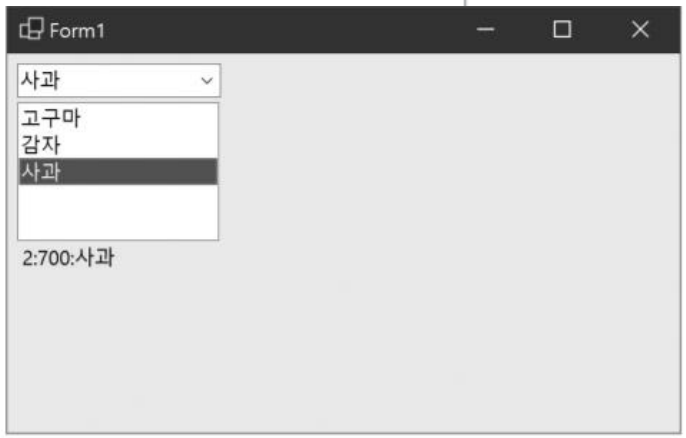

그림 10-6 Index, Value, Item을 추출해서 출력한 결과

Section 06 윈도 폼: 콤보 박스, 리스트 박스, 데이터 그리드 뷰 사용하기 (5)

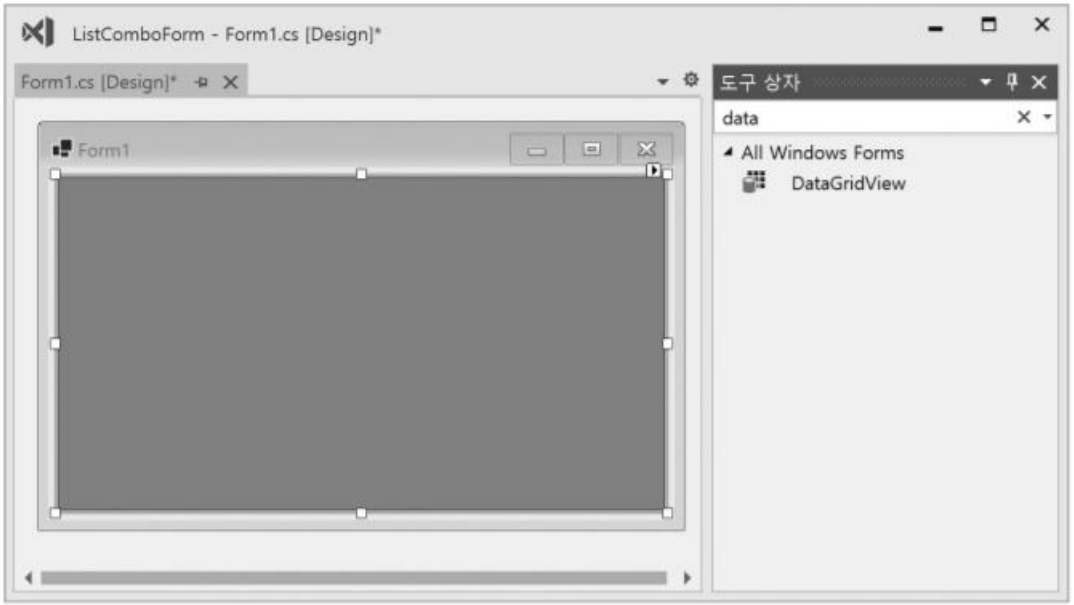

그림 10-7 데이터 그리드 뷰

Section 06 윈도 폼: 콤보 박스, 리스트 박스, 데이터 그리드 뷰 사용하기 (6)

코드 10-17 데이터그리드뷰 사용

```
01 public partial class Form1 : Form
02 {
03   class Product
04   {
05     public string Name { get; set; }
06     public int Price { get; set; }
07   }
08
09   public Form1()
10   {
11     InitializeComponent();
12
13     // 데이터 소스를 선택합니다.
14     var dataSource = new List<Product> {
15       new Product() { Name = "고구마", Price = 500  },
16       new Product() { Name = "감자", Price = 600  },
17       new Product() { Name = "사과", Price = 700  }
18     };
19
20     dataGridView1.DataSource = dataSource;
21   }
22 }
```

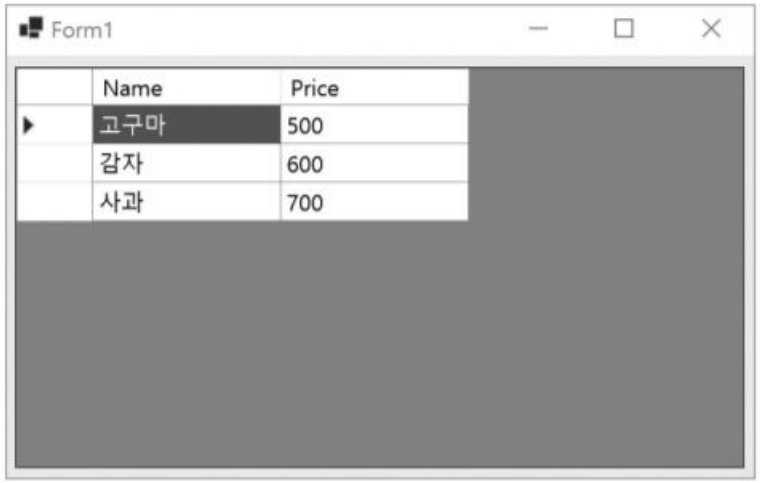

그림 10-8 데이터 그리드 뷰 출력