Contents

SECTION 1

C# Programming

항목 1-1

Hello, C#

☑ 주요 내용

> C# 기본 코드
> 컴파일 하는 방법
> 표준 입출력

1. Hello, C#

C# 기본 코드

C#의 소스 코드는 .cs 확장자를 사용합니다.

```
// hello.cs
using System;

class Program
{
    public static void Main()
    {
        Console.WriteLine("Hello, C#");
    }
}
```

컴파일 하는 방법

C# 컴파일러의 이름은 csc.exe 입니다. /target 옵션을 사용해서 실행파일(.exe) 또는 동적 모듈(.dll) 로 컴파일 할 수 있습니다.

	빌드 방법	출력 파일
실행파일로 빌드	csc hello.cs	Hello.exe
동적 모듈로 빌드	csc hello.cs /target:library	Hello.dll

[참조] /target 옵션은 /t 로 줄여서 사용할 수 있습니다.

2. C# 표준 입출력

System.Console 클래스

C#의 표준 입출력은 System namespace 안에 있는 Console 클래스가 가진 정적 메소드를 사용합니다.

표준 입출력 메소드 - Write, WriteLine, ReadLine, ReadKey

Write 또는 WriteLine 메소드를 사용해서 화면에 출력 합니다.

```
using System;

class IO
{
    public static void Main()
    {
        // 표준 출력
        Console.Write("hello");
        Console.WriteLine("world"); // 문자열 끝에 개행을 포함합니다.

        // 문자열 입력
        string s = Console.ReadLine();

        // 정수 입력
        int n = int.Parse(Console.ReadLine());

        // 입력 버퍼가 아닌 키보드에서 바로 입력합니다.
        Console.ReadKey();
    }
}
```

C# 문자열과 변수값 출력

C# 6.0 부터 추가된 보간문자열(interpolated string)을 사용하면 변수값을 편리하게 출력할 수 있습니다.

또한, 축자 문자열(verbatim string)을 사용하면 정규 표현식 등을 보다 편리하게 출력할 수 있습니다.

```
using System;

class Program
{
    public static void Main()
    {
        int n1 = 10;
        int n2 = 20;
        int n3 = 30;

        // 예전 방식의 변수값 출력
        Console.WriteLine("n1 = {0}, n2 = {1}, n3 = {2}", n1, n2, n3);

        // C# 6.0 보간문자열(interpolated string)
        Console.WriteLine($"n1 = {n1}, n2 = {n2}, n3 = {n3}");

        // 축자 문자열 리터럴(verbatim string literal )
        // "확장문자사용를 나타내는 "\"를 일반 문자열로 인식
        Console.WriteLine(@"\\.\pipe\NamedPipe");

        // 위 코드는 아래 코드와 동일합니다.
        Console.WriteLine("\\\\.\\pipe\\NamedPipe");
    }
}
```

항목 1-2

System.Object

☑ 주요 내용

> 모든 것은 객체이다.
> System.Object
> 타입 조사

1. 모든 것은 객체이다.

▌ 모든 것은 객체이다.

C++언어와 달리 C#에서는 모든 것은 객체입니다. C++ 등의 언어에서는 10, 3.4, "ABC" 등은 객체가 아닌 primitive 타입으로 분류 되지만, C#에서는 10등이 정수 리터럴도 int 타입의 객체 입니다.

```
// 정수형 리터럴 '10' 에도 멤버 함수가 있습니다.
string s1 = 10.ToString();
Console.WriteLine(s1);
```

▌ int 의 멤버

C#에서 int 타입은 .Net Framework 의 System.Int32 와 동일합니다. int 타입 에는 다양한 멤버가 제공됩니다.

instance method	CompareTo
	Equals
	GetHashCode
	GetType
	GetTypeCode
	ToString
static method	Parse
	TryParse

▌ 문자열 을 int 로 변환

문자열을 int로 변환 하려면 int 타입이 가진 정적 메소드인 Parse 함수를 사용하면 됩니다.

```
// 문자열을 int 변환
string s = "10";
int n = int.Parse(s); // int 의 정적 메소드 호출
Console.WriteLine(n);
```

2. System.Object

▌ 모든 타입은 System.Object 로부터 파생 된다.

C#의 모든 타입은 System namespace 의 Object 타입으로부터 파생 됩니다.

Instance method	Equals
	Finalize
	GetHashCode
	GetType
	MemberwiseClone
	ToString
Static method	Equals
	ReferenceEquals

따라서, C#의 모든 타입(값)에는 System.Object 로부터 파생된 멤버가 있습니다.

```
using System;

class Program
{
    // 함수가 object 를 인자로 가지는 경우 모든 값을 인자로 받을 수 있습니다.
    public static void Foo( object o )
    {
        string s = o.ToString();
        Console.WriteLine(s);
    }
    public static void Main()
    {
        Foo(10);
        Foo(3.4);
        Foo("hello");
    }
}
```

3. Reflection

▌ 타입을 조사하는 방법

Type 객체를 사용하면 특정 값(타입)의 정보를 조사할 수 있습니다. Type 객체를 얻어 내려면 typeof 또는 GetType 메소드를 사용하면 됩니다.

```
using System;
using System.Reflection; // MethodInfo 등 리플렉션(타입정보조사)
                         // 관련 클래스 사용

// 타입 정보 얻기

class Program
{
    public static void Main()
    {
        int n = 10;

        // 타입을 조사하는 방법
        // Type 타입의 객체를 사용한다.
        Type t1 = n.GetType(); // 객체을 사용해서 타입 정보 꺼내기
        Type t2 = typeof(int); // 타입을 사용해서 타입 정보 꺼내기

        Console.WriteLine(t1.Name);

        // 모든 메소드의 정보 얻기
        // 여러개를 리턴하는 함수 => 리턴값 배열
        MethodInfo[] mis = t1.GetMethods();

        // 배열로 부터 꺼내기
        foreach( MethodInfo mi in mis)
        {
            Console.WriteLine(mi.ToString());
        }
}
}
```

checked, unchecked

☑ 주요 내용

> 초기화 되지 않은 변수
> checked/unchecked

1. checked/unchecked

▌ 초기화 되지 않은 변수는 사용할 수 없다.

```
int n;
Console.WriteLine(n); // Error. 초기화 되지 않은 변수 사용 안됨.
```

▌ checked, unchecked

```
using System;

class Program
{
    public static void Main()
    {
        int n1 = int.MaxValue;
        int n2 = n1 + 1;         // overflow 발생. 예외 없음.
        int n3 = checked(n1 + 1);// overflow 발생. 예외 발생.

        Console.WriteLine($"{n2}");

        int n4 = int.MaxValue + 1;              // compile error
        int n5 = unchecked(int.MaxValue + 1);   // 에러 없음.
    }
}
```

항목 1-4

const vs readonly

☑ 주요 내용

> const
> readonly

1. const vs readonly

▌ C#과 상수

C# 에서의 상수는 const 또는 readonly 를 사용해서 만들게 됩니다.

const	Compile 시간 상수. 문자 치환 방식으로 사용됩니다.
readonly	실행시간 상수

```
class Program
{
    public static int n = 10;

    public const int c1 = 10;           // 컴파일 시간 상수. 문자 치환방식
    public static readonly int c2 = 10; // 실행시간 상수.

    public const int c1 = n;            // error. 변수값으로 초기화 될수 없습니다.
    public static readonly int c2 = n;  // ok


    public static void Main()
    {
        Console.WriteLine(c1); // Console.WriteLine(10); //
        Console.WriteLine(c2); // Console.WriteLine(c2); //
    }
}
```

▌ 동적 모듈(DLL)과 상수

독립된 어셈블리(또는 동적모듈)에 상수가 있고, 동적 모듈을 update 할 경우, const 상수를 사용하면 실행 파일도 다시 빌드 되어야 합니다.

```
// csc const.cs /target:library
public class Const
{
    public const int const_value = 10;
    public static readonly int readonly_value = 10;
}
```

그리고, 실행파일에서 아래와 같이 모듈에 있는 상수를 사용할 때

```
// csc 4_const.cs /r:Const.dll

class Program
{
    public static void Main()
    {
        Console.WriteLine(Const.const_value); // Console.WriteLine(10); //
        Console.WriteLine(Const.readonly_value); // Console.WriteLine(c2); //
    }
}
```

동적 모듈의 상수를 update 할때 readonly 상수는 update 된 값이 적용되지만 const 상수는 update 된 값이 적용되지 않습니다.

항목 1-5

키워드와 식별자

☑ 주요 내용

> 키워드와 식별자
> var

1. 키워드

▌ C# 키워드

abstract	as	base	bool
break	byte	case	catch
char	checked	class	const
continue	decimal	default	delegate
do	double	else	enum
event	explicit	extern	FALSE
finally	fixed	float	for
foreach	goto	if	implicit
in	int	interface	internal
is	lock	long	namespace
new	null	object	operator
out	override	params	private
protected	public	readonly	ref
return	sbyte	sealed	short
sizeof	stackalloc	static	string
struct	switch	this	throw
TRUE	try	typeof	uint
ulong	unchecked	unsafe	ushort
using	virtual	void	volatile
while			

▌ 문맥적 키워드

Add	alias	ascending
async	await	descending
dynamic	from	get
global	group	into
join	let	nameof
orderby	partial(형식)	partial(메서드)
remove	select	set
value	var	when(필터 조건)
where(제네릭 형식 제약 조건)	where(쿼리 절)	yield

▌ 식별자와 키워드의 충돌

@를 사용하면 키워드를 함수 또는 변수 이름(식별자)으로 사용할 수 있습니다.

문맥적 키워드는 @를 사용하지 않아도 식별자 이름으로 사용할 수 있습니다

```
using System;

// @를 사용하면 키워드를 식별자(클래스 이름)으로 사용할수 있습니다.
class @int
{
    public void show() { Console.WriteLine("int.show()"); }
}

// 문맥적 키워드인 where는 @ 없이도 식별자로 사용가능합니다.
class where
{
}

class Program
{
    public static void Main()
    {
        where w = new where();

        @int n = new @int();
        n.show();

        Type t = n.GetType();
        Console.WriteLine(t.Name);
    }
}
```

2. var

▌ var

var 를 사용하면 우변의 표현식으로부터 변수의 타입을 추론(inference)할 수 있습니다.

```
using System;

class Program
{
    public static void Main()
    {
        int n1 = 10;

        // var : 우변의 표현식을 계산해서 좌변의 타입을 추론.
        var n2 = n1;

    }
}
```

항목 1-6

Value type vs Reference type

☑ 주요 내용

> Value type vs Reference type
> equality
> Boxing/Unboxing

1. Value type / Reference type

▌ C/C++

C/C++ 에서는 객체를 스택 또는 힙에 만들 수 있습니다.

```
class Point
{
    int x, y;
};
int main()
{
    // pt는 스택에 놓이게 됩니다.
    Point pt;

    // Point 객체는 힙에 만들어 지고
    // 포인터 변수 p1은 스택에 놓이게 됩니다.
    Point* p1 = new Point;
}
```

▌ Value type vs Reference type

C#에서는 struct 는 value type 이라고 하고, 스택에 만들어 지게 됩니다.

하지만, class 는 reference type 이라고 하고, 힙에 만들어 지게 됩니다.

```
using System;

struct SPoint
{
    public int x;
    public int y;
}
class CPoint
{
    public int x;
    public int y;
}

class Program
{
    public static void Main()
    {
        SPoint sp1 = new SPoint(); // 한번의 메모리 할당, 스택에 할당
```

```
        SPoint sp2 = sp1;

        sp1.x = 100;
        Console.WriteLine("sp1.x = {0}, sp2.x = {1}", sp1.x, sp2.x);


        CPoint cp1 = new CPoint();      // 2번의 메모리 할당
                                        // cp1이 스택, CPoint 가 힙
        CPoint cp2 = cp1;  // 한번의 메모리 할당. cp2가 스택

        cp1.x = 100;
        Console.WriteLine("cp1.x = {0}, cp2.x = {1}", cp1.x, cp2.x);
    }
}
```

2. Equality

▌ Reference Type 과 Equality

Reference Type 의 동등성을 조사할 때는 == 또는 Equals 함수를 사용할 수 있습니다.

==	2개의 참조가 동일 객체를 가리키는지 조사합니다.
Equals	2개 객체의 상등성을 조사합니다.

```
using System;

class CPoint
{
    public int x = 0;
    public int y = 0;
    public CPoint(int a, int b) { x = a; y = b; }

    // Equals 를 재정의 해서 객체의 동등성에 대한 코드를 제공합니다.
    public override bool Equals(object obj)
    {
        CPoint pt = obj as CPoint;

        return x == pt.x && y == pt.y;
    }
}
class Program
{
     public static void Main()
    {
        CPoint p1 = new CPoint(1, 1);
        CPoint p2 = new CPoint(1, 1);

         Console.WriteLine(p1 == p2);
         Console.WriteLine(p1.Equals(p2));
    }
}
```

▌ Value type 과 Equality

Value type의 경우 == 연산을 사용하려면 == 연산자를 제공해야 합니다.

```
using System;

struct SPoint
{
    public int x;
    public int y;
    public SPoint(int a, int b) { x = a; y = b; }

    public static bool operator ==( SPoint p1, SPoint p2)
    {
        return p1.x == p2.x && p1.y == p2.y;
    }
    public static bool operator !=(SPoint p1, SPoint p2)
    {
        return !(p1 == p2);
    }
}
class Program
{
    public static void Main()
    {
        SPoint p1 = new SPoint(1, 1);
        SPoint p2 = new SPoint(1, 1);
        Console.WriteLine(p1 == p2);

        // Equals 는 object 멤버이다. 모든 객체에 있다
        // 기본적으로는 메모리 내용을 비교, 변경가능.
        Console.WriteLine(p1.Equals(p2));
    }
}
```

3. Boxing / Unboxing

▌ Boxing 과 Unboxing

Value type을 참조 타입으로 캐스팅할 경우 value 타입의 복사본이 힙 에 생성되게 됩니다. 이와 같은 동작 방식을 Boxing 이라고 합니다. 또한, Boxing 되어 있는 참조 타입에서 value type의 값을 꺼내는 것을 Unboxing 이라고 합니다.

Boxing은 암시적으로 변환되지만, Unboxing은 명시적 변환을 사용해야 합니다.

```
class Program
{
    public static void Main()
    {
        int n1 = 10;

        object o = n;   // boxing
        int n2 = (int)o;// unboxing
    }
}
```

4. 타입 조사

▌ Value type 조사

모든 Value Type은 System.ValueType 으로 부터 파생 됩니다.

Type 클래스를 사용하면 특정 객체가 value type 인지, reference type 인지 조사할 수 있습니다.

```
using System;

class Program
{
    public static void Main()
    {
        // C#에서 배열 만들기
        int[] arr = new int[] { 1, 2, 3 };

        // 배열은 값 타입일까 ? 참조 타입 일까 ?
        Type t = arr.GetType();

        if (t.IsValueType)
            Console.WriteLine("value type");
        else
            Console.WriteLine("reference type");

        int[] arr2 = arr;
        int[] arr3 = (int[])arr.Clone();// 복사본 생성..
                                        // prototype 패턴.
        arr[0] = 100;

        Console.WriteLine(arr2[0]); // 100
        Console.WriteLine(arr3[0]); // 1
    }
}
```

casting

☑ 주요 내용

> Casting
> is, as
> explicit/implicit casting operator
> as 와 casting

1. casting

▌ explicit casting vs implicit casting

C/C++에서는 double 값이 int 타입으로 암시적 변환을 허용합니다. 하지만, C# 에서는 double -> int 의 암시적 변환은 허용하지 않습니다. C#에서는 데이터 손실이 없고, 항상 성공할수 있는 경우에만 암시적 형변환을 허용합니다.

```
using System;

class Program
{
    public static void Main()
    {
        int    n1 = 3;
        double d = n1;  // ok.  int 는 double 로 암시적 형변환
                        // 1. 실패하지 않고, 2. data 손실이 없다.
        int n2 = d;      // C/C++ : ok,   C# => error
        int n3 = (int)d; // C#의 명시적(explicit) 캐스팅
    }
}
```

2. is, as

▌ Downcasting

기반 클래스 타입을 파생 클래스 타입으로 캐스팅(Downcasting) 할 때, 잘못된 객체가 전달되면 예외가 발생할 수 있습니다.

```
class Animal{};

class Dog : Animal
{
    public void Cry() { Console.WriteLine("Dog Cry"); }
}

class Program
{
    public static void Main()
    {
        Animal a1 = new Animal();
        Animal a2 = new Dog();

        Dog d1 = (Dog)a2; // a2는 Dog를 가리킵니다. 문제가 없습니다.
        Dog d2 = (Dog)a1; // a1는 Dog를 가리키지 않습니다. 예외발생.
    }
}
```

▌ is 와 as

is 를 사용해서 변수가 실제 어떤 객체를 가리키는지를 조사할 수 있습니다. 또한, as 연산자를 사용하면 실패시 null을 반환 받을수 있습니다.

```
Animal a = new Dog();

// is 사용한 조사
if (a is Dog) // typeid(a) == typeid(Dog)
{
    Dog d = (Dog)a;
    d.Cry();
}
else
    Console.WriteLine("Dog 아님");
```

```
// as를 사용한 캐스팅. 실패시 null 반환
Dog d2 = a as Dog; // 실패시 null 반환

if (d2 == null)
    Console.WriteLine("실패");
else
    Console.WriteLine("성공");
```

3. 변환 연산자

▌ Conversion operator

변환 연산자를 사용하면 객체가 다른 타입으로 암시적(또는 명시적) 변환 되도록 할 수 있습니다.

```
using System;

// 변환 연산자
//
class Point
{
    public int x = 0;
    public int y = 0;
    public Point(int a, int b) { x = a; y = b; }

    public void Dump() { Console.WriteLine("x = {0}, y = {0}", x, y); }

    // 변환 연산자 : 다른 타입이 객체로 변환 되게 한다.
    // int => Point 변환을 허용한다.
    // explicit : 명시적 변환만 허용
    // implicit : 암시적 변환도 허용
    // int => Point : operator Point(int n)
    // Point => int : operator int(Point n)
    public static explicit operator Point(int n)
    {
        Console.WriteLine("operator Point");
        Point p = new Point(n, n); // 다양한 정책 사용
        return p;
    }
}
class Program
{
    public static void Main()
    {
        int n = 10;
        Point p = (Point)n; // 이 순간 변환 연산자(operator Point)가 사용됩니다.

        p.Dump();
     }
}
```

▌ 캐스팅 vs as

캐스팅은 사용자 정의 변환 함수를 호출할수 있지만, as는 사용자 정의 변환 함수를 호출할 수 없습니다.

```
Point p = (Point)n;   // ok. 사용자 정의 변환 함수가 호출됩니다.
Point p = n as Point; // error. 사용자 정의 변환 함수를 호출할 수 없습니다.
```

항목 1-8

null

☑ 주요 내용

> Nullable Type
> ? and ??

1. Nullable Type

▌ Nullable<T>

Nullable<T> 를 사용하면 reference 타입 뿐 아니라 value type 도 null 상태를 가질 수 있습니다.

```
using System;

class Program
{
    // int 는 null이 될수 없으므로 실패를 표현하려면
    //특정 값(0 또는 -1등)을 사용해야 합니다.
    public static int F1()   {  return -1;    }

    // Nullable<int> 는 null이 될수 있습니다.
    public static Nullable<int> F2() { return null; }

    // Nullable<T> 는 T? 로 축약 표기할수 있습니다.
    public static int? F3() { return null; }

    public static void Main()
    {
        var ret = F3();
        if (ret == null) { }
    }
}
```

2. ?, ?? operator

▌ ? - 널 조건부 연산자 - (null conditional operator, 'Elvis' operator)

? 연산자를 사용하면 변수가 null이 아닌 경우만 멤버에 접근할 수 있습니다.

```
using System;

class Car
{
    public void Go() { Console.WriteLine("Car Go"); }
}
class Program
{
    public static Car CreateCar(int speed)
    {
        if (speed < 200)
            return new Car();
        return null;
    }
    public static void Main()
    {
        Car c = CreateCar(300);
        if (c != null)
            c.Go();
        // 아래 한줄은 위 2줄과 동일합니다.
        c?.Go();
    }
}
```

▌ ?? - 널 접합 연산자 (null - coalescing operator)

??를 사용하면 변수가 null 인 경우에 null 대신 디폴트 값을 반환 받을 수 있습니다.

```
string s1 = null;
string s2 = s1 ?? "이름없음";
Console.WriteLine(s2);
```

항목 1-9

method

☑ 주요 내용

> 인자 전달 방식
> 인자 전달 방식과 참조 타입
> Params
> Optional parameter
> Expression-bodied method
> Extension method

1. 인자 전달 방식

▌ Call by value, reference

C# 에서는 method 에 인자를 전달할 때 기본적으로 call by value 를 사용합니다. Call by reference 로 전달 하려면 ref 또는 out 를 사용해야 합니다.

```
using System;

class Program
{
    public static void f1(int n)     { n = 20; }
    public static void f2(ref int n) { n = 20; }
    public static void f3(out int n) { n = 20; }

    public static void Main()
    {
        int n1 = 10;
        int n2 = 10;
        int n3 = 10;

        f1(n1);
        f2(ref n2);
        f3(out n3);
        Console.WriteLine($"{n1}, {n2}, {n3}");

        int n;     // 초기화 되지 않은 변수
        f2(ref n); // error.
        f3(out n); // ok
    }
}
```

2. Reference type 과 ref 전달

▌ Reference 타입과 call by value

Reference type의 경우 call by value로 전달해도 함수 안에서 객체의 상태를 변경할 수 있습니다.

```
using System;

class Point
{
    public int x = 0;
    public int y = 0;
    public Point(int a, int b) { x = a; y = b; }
}

class Program
{
    public static void f1(Point p)     { p.x = 10; }
    public static void f2(ref Point p) { p.x = 20; }

    public static void Main()
    {
        Point p1 = new Point(0, 0);

        f1(p1);
        Console.WriteLine($"{p1.x}"); // 10

        f2(ref p1);
        Console.WriteLine($"{p1.x}"); // 20
    }
}
```

▌ Reference 타입과 call by reference

함수에서 새로운 객체를 생성할 경우에는 reference type이라고 ref 로 인자를 전달해야 합니다.

```
class Program
{
    public static void f3(Point p)     { p = new Point(1, 1); }
    public static void f4(ref Point p) { p = new Point(1, 1); }
    public static void Main()
    {
```

```
        Point p1 = new Point(0, 0);

        f3(p1);
        Console.WriteLine($"{p1.x}"); // 0

        f4(ref p1);
        Console.WriteLine($"{p1.x}"); //  1
    }
}
```

3. params

▌ params 와 배열

params 를 사용하면 동일 타입의 값을 임의의 개수 만큼 받을 수 있습니다.

```
using System;

class Program
{
    public static void F1( int[] arr ) {}

    public static void F2( params int[] arr)
    {
        foreach (var n in arr)
            Console.WriteLine(n);
    }
    public static void Main()
    {
        F1(new int[] { 10, 20, 30 });
        F2(new int[] { 10, 20, 30 });   // 배열 전달
        F2( 10, 20, 30 );               // 배열 대신, 요소를 직접 보낼수도 있습니다.
    }
}
```

▌ params 는 마지막 인자로만 사용할 수 있다.

```
public static void f1(int a, params int[] arr) { }  // ok
public static void f1(params int[] arr, int n) { }  // error
```

4. Optional Parameter

▌ Optional Parameter

C# 4.0 부터는 선택적 파라미터(Optional Parameter, default parameter)를 사용할 수 있습니다.

```
class Program
{
    public static void goo(int x = 0, int y = 0, int z = 0) { }

    public static void Main()
    {
        goo(1, 2, 3);
        goo(1, 2 );
        goo(1);
     }
}
```

▌ Parameter name

함수 호출시 인자의 이름을 사용할 수 있습니다. 이경우 인자의 순서를 변경 할 수도 있습니다.

```
goo(x:1, y:2, z:3);
goo(1, z: 3, y: 2);
goo(y: 2);
goo(1, x: 1, z: 2); // error
```

▌ 가상 함수와 Optional Parameter

가상 함수 호출은 실행 시간에 이루어 지지만, Optional Parameter 는 컴파일 시간에 결정 됩니다. 가상함수에서 Optional Parameter를 사용할 경우 예상치 못한 결과가 나올수 있습니다.

```
using System;

class Base
{
    public virtual void foo(int n = 10) { Console.WriteLine($"Base : {n}"); }
}
class Derived : Base
{
```

```
    public override void foo(int n = 20) { Console.WriteLine($"Derived : {n}"); }
}
class Program
{
    public static void Main()
    {
        Base b = new Derived();
        b.foo(); // Derived : 10
    }
}
```

5. 식-본문 메소드 (expression-bodied method)

함수 구현부가 간단한 경우 expression-bodied method를 사용하면 편리합니다.

```
using System;

class Program
{
    public static int square1(int x) { return x * x; }
    public static int square2(int x) => x * x;

    public static void Main()
    {
        Console.WriteLine(square1(3));
        Console.WriteLine(square2(3));
    }
}
```

6. 확장 메소드 (extension method)

▌ Extension method

기존에 존재하는 클래스에 새로운 메소드를 추가할 수 있습니다.

```
public class Car
{
    public void Go() { Console.WriteLine("Car Go"); }
}
// Car 안에 Stop 이라는 instance method를 추가하려면
// 1. static class에  static method로 추가한다.
// 2. 함수에 Car 를 인자로 받아야 한다.
public static class CarExtension
{
    public static void Stop(this Car c) { Console.WriteLine("Car Stop"); }
}

class Program
{
    public static void Main()
    {
        Car c = new Car();
        c.Go();
        c.Stop();  // CarExtension.Stop(c);
    }
}
```

항목 1-10

constructor

☑ 주요 내용

> constructor
> static constructor
> 초기화 순서

1. this를 사용한 다른 생성자 호출

하나의 생성자에서 다른 생성자를 호출하려면 this를 사용합니다.

```
using System;

class Point
{
    public int x;
    public int y;
    public Point(int a, int b) { x = a; y = b; Console.WriteLine("Point(int, int)"); }

    // 하나의 생성자에서 다른 생성자를 호출하는 방법.
    public Point() : this(0,0) {  Console.WriteLine("Point()"); }
}
class Program
{
    public static void Main()
    {
        Point p1 = new Point(0, 0);
        Point p2 = new Point();

        Console.WriteLine($"{p2.x}, {p2.y}");
    }
}
```

2. static constructor

▌ static constructor

static 멤버를 초기화 하기 위해 static 생성자를 사용할 수 있습니다.

static 생성자는 인자를 가질수 없고, 오직 한 개만 만들수 있고, 단 한번만 호출됩니다.

static 생성자는 객체를 생성하거나 static 멤버에 접근하는 경우에만 호출됩니다.

```
using System;

class Car
{
    public int speed = 0;
    public static int count = 10;

    // static 생성자 : static 멤버를 초기화하기 위해 사용
    static Car()
    {
        count = 20;
        Console.WriteLine($"static Car() : {count}");
    }
}

class Program
{
    public static void Main()
    {
        // Console.WriteLine(Car.count); // static 생성자가 먼저 호출된다.
        Car c = new Car();  // static 생성자가 먼저 호출, 일반 생성자 호출
    }
}
```

3. 상속과 생성자

▌ 상속과 생성자

객체를 생성한 경우 기반 클래스의 생성자가 먼저 호출됩니다. 정확히는 파생 클래스의 생성자에서 기반 클래스의 생성자를 호출하게 됩니다.

기반 클래스에 디폴트 생성자가 없거나, 디폴트 생성자가 아닌 다른 생성자를 호출하려면 파생 클래스에서 기반 클래스의 생성자를 명시적으로 호출해야 합니다.

```
using System;

class Base
{
    public Base()      { Console.WriteLine("Base()"); }
    public Base(int a) { Console.WriteLine("Base(int)"); }
}
class Derived : Base
{
    public Derived()  //  Derived() : base()
    {
        Console.WriteLine("Derived()");
    }
    public Derived(int a) : base(a) { Console.WriteLine("Derived(int)"); }
}

class Program
{
    public static void Main()
    {
        Derived d1 = new Derived();
        Derived d2 = new Derived(5);
    }
}
```

4. 초기화 순서

▌ 필드가 초기화 되는 순서

필드 초기화를 사용하고, 생성자에서도 필드를 초기화 한 경우 다음의 순서로 초기화 됩니다.

① 자신 멤버의 필드 초기화

② 기반 클래스 멤버의 필드 초기화

③ 기반 클래스의 생성자

④ 자신의 생성자

```
using System;

// 생성자 호출 순서
class BM { public BM() { Console.WriteLine("BM()"); } }
class DM { public DM() { Console.WriteLine("DM()"); } }

class Base
{
    public int data = BaseFieldInit();
    public BM bm = new BM();
    protected Base() { Console.WriteLine("Base()"); }

    public static int BaseFieldInit() { Console.WriteLine("BaseFieldInit"); return 0; }
}
class Derived : Base
{
    public int data = DerivedFieldInit();
    public DM dm = new DM();
    public Derived() { Console.WriteLine("Derived()"); }

    public static int DerivedFieldInit()
    { Console.WriteLine("DerivedFieldInit"); return 0; }
}
class Program
{
    public static void Main()
    {
        Derived d = new Derived();
        // 화면 출력 결과
        // DerivedFieldInit
```

```
        // DM()
        // BaseFieldInit
        // BM()
        // Base()
        // Derived()
    }
}
```

5. 가상 함수와 생성자

▌ 가상함수와 생성자

생성자에서 가상함수를 호출하는 경우, C++에서는 기반클래스의 함수가 호출되지만, C#에서는 파생클래스가 override 한 함수가 호출됩니다.

그러나, 이 경우 초기화 되지 않은 객체를 사용하는 문제가 발생할 수 있습니다.

```
using System;

class DM
{
    public DM()         { Console.WriteLine("DM()"); }
    public void UseDM() { Console.WriteLine("Use DM"); }
}
class Base
{
    public Base() { foo(); } // Derived::foo()가 호출됩니다.
    public virtual void foo() { Console.WriteLine("Base foo()"); }
}
class Derived : Base
{
    public DM dm = null;
    public Derived() { dm = new DM(); }
    public override void foo()
    {
        Console.WriteLine("Derived foo()");
        dm.UseDM();     // Derived()생성자가 호출되지 않았습니다.
                        // dm은 null 입니다.
    }
}

class Program
{
    public static void Main()
    {
        Derived d = new Derived();
    }
}
```

항목 1-11

property

☑ 주요 내용

> property 개념
> automatic property

1. property

▌ property

사용시에는 필드 처럼 보이지만 만들때는 함수(method)처럼 만드는 문법.

getter 와 setter.

```
using System;

class Bike
{
    public int gear = 0;

    public int Gear
    {
        get { return gear; }
        set { gear = value; } // value는 약속된 키워드.
    }
}
class Program
{
    public static void Main()
    {
        Bike b = new Bike();

        // property 접근.
        b.Gear = 10;                    // set {} 호출
        Console.WriteLine(b.Gear);      // get {} 호출
    }
}
```

▌ automatic property

getter/setter 뿐 아니라 field 까지 자동으로 생성

```
using System;

class Bike
{
    // field(멤버 데이타), getter, setter등 3가지 요소를 자동생성
    public int Gear { get; set; } = 0;
```

```
}
class Program
{
    public static void Main()
    {
        Bike b = new Bike();

        b.Gear = 10;
        Console.WriteLine(b.Gear);
    }
}
```

indexer

☑ 주요 내용

> indexer

1. indexer

▌ indexer

인덱서 문법을 사용하면 객체를 배열처럼 보이게 할 수 있습니다.

```
using System;

class Sentense
{
    protected string[] words;
    public Sentense(string s) { words = s.Split(); }

    // indexer : 객체를 배열처럼 보이게 하는 문법 - [] 연산자 재정의
    public string this[int idx]
    {
        set { words[idx] = value; }
        get { return words[idx]; }
    }
}
class Program
{
    public static void Main()
    {
        Sentense s = new Sentense("we are the world");

        Console.WriteLine(s[3]);

        s[3] = "frield";

    }
}
```

항목 1-13

inheritance

☑ 주요 내용

> virtual
> new vs override

1. Field name 과 new

▌ Member field name

기반 클래스가 가진 필드 이름과 동일한 이름을 가진 필드가 파생 클래스에 있을 경우 에러는 없지만 경고가 발생합니다. 이 경우 new를 사용하면 경고가 나타나지 않습니다.

```
using System;

// 주제 : 멤버 이름 충돌과 new
class Base
{
    public int value = 10;
}
class Derived : Base
{
    //public int value = 20;  // 아무 문제 없지만 경고 발생
    public new int value = 20;// 경고가 없다.
}
class Program
{
    public static void Main()
    {
        Derived d = new Derived();

        Console.WriteLine(d.value); // 20
        Console.WriteLine(((Base)d).value); // 10
    }
}
```

2. virtual function 과 new, override

▌ new, override

가상 함수를 재정의 하려면 override 키워드를 사용합니다. new 를 사용하면 가상 함수 재정의가 아닌 새로운 함수를 만들게 됩니다.

```
using System;

// 함수 재정의(override)
class Base
{
    public virtual void Foo() { Console.WriteLine("Base Foo"); }
    public virtual void Goo() { Console.WriteLine("Base Goo"); }
}
class Derived : Base
{
    public new      void Foo() { Console.WriteLine("Derived Foo"); }
    public override void Goo() { Console.WriteLine("Derived Goo"); }
}

class Program
{
    public static void Main()
    {
        Base    b = new Derived();
        Derived d = new Derived();

        b.Foo(); // Base Foo
        d.Foo(); // Derived Foo
        b.Goo(); // Derived Goo
        d.Goo(); // Derived Goo
    }
}
```

항목 1-14

abstract class / interface

☑ 주요 내용

> abstract method
> abstract class
> interface

1. abstract class

▌ abstract method

method 앞에 abstract 가 붙고, method의 구현만 있고, 선언이 없는 method를 추상 메소드 라고 합니다. 또한, 추상 메소드가 한 개 이상인 클래스를 추상 클래스라고 합니다.

추상 클래스는 객체를 만들 수 없습니다.

추상 클래스의 파생 클래스도 추상 메소드의 구현을 제공하지 않은 경우는 추상 클래스가 됩니다.

```
using System;

abstract class Shape
{
    public int color = 0;
    public abstract void DrawImp(); //
    public void Draw() { DrawImp(); }
}
class Rect : Shape
{
}
class Program
{
    public static void Main()
    {
        Shape p1 = new Shape(); // error
        Shape p2 = new Rect(); // error
    }
}
```

▌ override abstract method

추상 메소드를 재정의 할 때는 override 키워드를 사용합니다.

```
class Rect : Shape
{
    public override void DrawImp() { Console.WriteLine("DrawImp"); }
}
```

2. interface

▌ abstract class vs interface

추상 클래스는 추상 메소드 뿐 아니라 일반 메소드, 필드 등도 가질수 있습니다. 하지만, 인터페이스는 추상 메소드 또는 프라퍼티만 가질수 있습니다.

```
interface IShape
{
//    public int color = 10; // error. interface는 필드를 가질 수 없습니다.
    void DrawImp();
}
```

▌ interface의 메소드를 재정의 하는 방법1 – non overridable

```
using System;

interface IShape
{
    void DrawImp();
}
class Rect : IShape
{
    public void DrawImp() { Console.WriteLine("Rect DrawImp"); }// 1
}
class NewRect : Rect
{
    public void DrawImp() { Console.WriteLine("NewRect DrawImp"); } // 2
}
class Program
{
    public static void Main()
    {
        IShape p = new NewRect();
        p.DrawImp(); //1
    }
}
```

▌ interface의 메소드를 재정의 하는 방법 2 –overridable

```
using System;

interface IShape
{
    void DrawImp();
}
class Rect : IShape
{
    public virtual void DrawImp() { Console.WriteLine("Rect DrawImp"); }// 1
}
class NewRect : Rect
{
    public override void DrawImp() { Console.WriteLine("NewRect DrawImp"); } // 2
}
class Program
{
    public static void Main()
    {
        IShape p = new NewRect();
        p.DrawImp(); // 2
    }
}
```

항목 1-15

generic

☑ 주요 내용

> Generic class
> Generic method
> Type parameter 제약
> CRTP

1. Generic class

Generic vs Non-Generic

Non-generic

```
class Point
{
    public int x;
    public int y;
    public Point(int a = 0, int b = 0) { x = a; y = b; }
}
class Generic1
{
    public static void Main()
    {
        Point p1 = new Point(1, 1.5);
    }
}
```

Using generic

```
class Point<T>
{
    public T x;
    public T y;
    // default(T) : 참조타입일때 null, primitive 타입일때  0
    // C++ : T a = T()
    public Point(T a = default(T), T b = default(T) ) { x = a; y = b; }
}
class Generic1
{
    public static void Main()
    {
        Point<int> p1 = new Point<int>(1, 1);
    }
}
```

2. Generic method

▌ Generic method

클래스 뿐 아니라 메소드 도 generic 으로 만들 수 있습니다.

```
using System;

class Generic1
{
    // 클래스 뿐아니라 method도 generic(template) 이 될수 있다.
    public static void Foo<T>(T  n) { }

    public static void Main()
    {
        // template method 사용.
        Foo<int>(10); // 타입을 명시적으로 지정
        Foo(10);    // 컴파일러가 타입을 결정
        Foo("aaa"); // 컴파일러가 타입을 결정
    }
}
```

3. Generic constraint(제약)

▌ T는 object 로 간주 된다.

Generic method 에서 T는 object 타입으로 간주 됩니다.

```
class Generic3
{
    public static T Max<T>(T a, T b)
    {
        // error. 컴파일러는 a를 object 타입으로 생각합니다.
        // object 타입에는 Compare가 없습니다.
        return a.CompareTo(b) < 0 ? b : a;
    }
    public static void Main()
    {
        Console.WriteLine(Max(10, 3));
        Console.WriteLine(Max("AAA", "BBB"));
    }
}
```

▌ 해결책 1. 캐스팅

Generic 메소드에서 특정 메소드(CompareTo)를 사용하려면 해당 인터페이스 타입으로 캐스팅하면 됩니다.

```
class Generic3
{
    public static T Max<T>(T a, T b)
    {
        // CompareTo() 메소드는 IComparable interace에 있습니다.
        IComparable x = a as IComparable;
        IComparable y = a as IComparable;

        return x.CompareTo(y) < 0 ? a : a;
    }
    public static void Main()
    {
        Console.WriteLine(Max(10, 3));
        Console.WriteLine(Max("AAA", "BBB"));
    }
}
```

▌ 해결책 2. Type parameter constraint(제약)

Generic 메소드를 만들 때 제약을 사용하면 원하는 기능을 사용할 수 있습니다.

```
class Generic3
{
    public static T Max<T>(T a, T b) where : IComparable
    {
        return a.CompareTo(b) < 0 ? b : a;
    }
    public static void Main()
    {
        Console.WriteLine(Max(10, 3));
        Console.WriteLine(Max("AAA", "BBB"));
    }
}
```

▌ 다양한 형태의 제약

Generic 메소드를 만들 때 제약을 사용하면 원하는 기능을 사용할 수 있습니다.

```
where T : class
where T : struct
where T : new()
where T : interface이름
where T : base class 이름
```

4. CRTP

▌ Curiously Recurring Template Pattern

기반 클래스가 Generic 인 경우 기반 클래스의 generic 인자로 파생 클래스의 이름을 전달 할 수 있습니다.

```
using System;

class Singleton<T> where T : new()
{
    protected static T instance = null;
    public static T getInstance()
    {
        if (instance == null)
            instance = new T();
        return instance;
    }
}
class Mouse  : Singleton<Mouse>  {}
class Cursor : Singleton<Cursor> {}

class Generic4
{
    public static void Main()
    {
        Mouse s1 = Mouse.getInstance();
        Mouse s2 = Mouse.getInstance();
    }
}
```

항목 1-16

delegate

☑ 주요 내용

> Delegate 개념
> Callback using delegate
> event
> Func, Action
> covariance, contra-variance

1. Delegate 개념

▌ 함수를 담는 타입

Delegate를 사용하면 함수를 담는 데이터 타입을 만들 수 있습니다.

```
delegate void FP(int a);

class Delegate1
{
    public static void Foo(int a) { }

    public static void Main()
    {
        int    n = 10;  // 정수를 담는 타입 - int
        double d = 3.4; // 실수를 담는 타입 - double
        FP f = Foo;     // 함수를 담는 타입 - FP
    }
}
```

▌ System.MulticastDelegate

delegate 키워드를 사용해서 만들어진 타입은 System.MulticastDelegate 로부터 파생된 타입 입니다.

```
delegate void FP(int a);

// 위 코드에서 FP는 System.MulticastDelegate 로 부터 파생된 클래스입니다.
class FP : System.MulticastDelegate
{
}
```

▌ () vs invoke

Delegate 타입의 변수에 등록된 메소드를 호출할 때 () 또는 invoke 멤버 함수를 사용할 수 있습니다.

```
FP f = Foo;
f(10);
f.Invoke(10); // System.MulticastDelegate 로 부터 상속받은 멤버
```

2. Multicast Delegate

▌ Instance method, static method

delegate 타입의 변수에는 static method 뿐 아니라 instance method 도 담을 수 있습니다. 또한, 한 개 이상의 method를 담을 수 있습니다.

```
using System;

// Multicast Delegate
delegate void FP(int a);

class Program
{
    public static void Foo(int a) { Console.WriteLine($"Foo : {a}"); }
    public static void Goo(int a) { Console.WriteLine($"Goo : {a}"); }
    public void Hoo(int a)        { Console.WriteLine($"Hoo : {a}"); }

    public static void Main()
    {
        Program p = new Program();

        FP f = Foo;       // static method
        f += Program.Goo; // 클래스이름.static method
        f += p.Hoo;       // instance method는 객체도 필요합니다.

        f(10);
    }
}
```

▌ System.MulticastDelegate

delegate 키워드를 사용해서 만들어진 타입은 System.MulticastDelegate 로부터 파생된 타입 입니다.

```
delegate void FP(int a);

// 위 코드에서 FP는 System.MulticastDelegate 로 부터 파생된 클래스입니다.
class FP : System.MulticastDelegate
{
}
```

▌ () vs invoke

Delegate 타입의 변수에 등록된 메소드를 호출할 때 () 또는 invoke 멤버 함수를 사용할 수 있습니다.

```
FP f = Foo;
f(10);
f.Invoke(10); // System.MulticastDelegate 로 부터 상속받은 멤버
```

3. Event handling using delegate

▌ event handling

C#에서는 delegate를 사용해서 다양한 event를 처리합니다.

```
using System;

delegate void Handler();

class Button
{
    public Handler Click = null;

    public void Press()
    {
        Click?.Invoke();
    }
}
class Delegate3
{
    public static void Foo() { Console.WriteLine("Button Click"); }

    public static void Main()
    {
        Button b = new Button();
        b.Click += Foo;

        b.Press();
    }
}
```

4. Func, Action

▌ Func, Action

C#에는 미리 정의된 Generic 형태의 delegate 타입인, Func, Action을 제공합니다.

```
using System;

delegate TResult Func<TResult>();
delegate TResult Func<TResult, T1>(T1 arg1);
delegate TResult Func<TResult, T1, T2>(T1 arg1, T2 arg2);

delegate void Action();
delegate void Action<T1>(T1 arg1);
delegate void Action<T1, T2>(T1 arg1, T2 arg2);

class Delegate6
{
    public static int    F1()         { Console.WriteLine("F1"); return 0; }
    public static double F2()         { Console.WriteLine("F2"); return 0; }
    public static double F3(string s) { Console.WriteLine("F3"); return 0; }

    public static void Main()
    {
        Func<int>    f1 = F1;
        Func<double> f2 = F2;
        Func<double, string> f3 = F3;

        Action      a1 = F4;
        Action<int> a2 = F5;
        Action<double> a3 = F6;
    }

    public static void F4()      { Console.WriteLine("F4");  }
    public static void F5(int a) { Console.WriteLine("F5");  }
    public static void F6(double a) { Console.WriteLine("F6"); }
}
```

5. 공변성(covariance), 반변성(contravariance)

▌ 공변성과 가변성

X가 Y로 변환 가능할 때 G<X>가 G<Y>로 변환 가능한 것을 공변성(covariance)이라고 합니다.

반대로, X가 Y로 변환 가능할 때 G<Y> 가 G<X> 로 변환 가능한 것을 반변성(contravariance)라고 합니다.

Covariance를 위해서는 out, contravariance 를 위해서는 in 을 사용합니다.

```
using System;

class Animal { }
class Dog : Animal { }

// 공변성 ( Covariance ), 반변성 ( contravariance )

delegate TResult Func<out TResult>();
delegate void Action<in T>(T arg);

class Delegate6
{
    public static Animal F1() { return null; }
    public static Dog    F2() { return null; }

    public static void Main()
    {
        Func<Animal> d1 = F1;
        Func<Dog>    d2 = F2;
        Func<Animal> d3 = d2; // 공변성(covariance)

        Action<Animal> a1 = F3;
        Action<Dog>    a2 = F4;

        Action<Dog>    a3 = a1; // 반변성(contravariance)
    }
    public static void F3(Animal a) { }
    public static void F4(Dog a)    { }
}
```

6. Lambda Expression

▌ 익명의 함수

람다 표현식을 사용하면 익명의 함수를 만들수 있습니다. C#의 람다 표현식은 다양한 형태로 만들수 있습니다.

```
using System;

class Labmda1
{
    public static void Main()
    {
        int v1 = 10;

        // 람다 표현식의 다양한 형태
        Func<int, int, int> plus1 = (int x, int y) => { return x + y; };
        Func<int, int, int> plus2 = (x, y) => { return x + y; };
        Func<int, int, int> plus3 = (x, y) => x + y;
        Func<int, int, int> plus4 = (x, y) => x + y + v1;

        Func<int, int, int> plus5 = (x, y) => { v1 = 100; return x + y + v1; };

        Console.WriteLine(plus1(1, 2));

        Console.WriteLine(v1);
    }
}
```

항목 1-17

LINQ

☑ 주요 내용

> LINQ 개념
> Fluent Syntax
> 지연된 실행
> LINQ의 원리

1. LINQ 개념

▌ 기본 개념

LINQ(Language-Integrated Query)는 C# 언어에 직접 쿼리 기능을 통합하는 방식을 기반으로 하는 기술 집합 이름입니다.

다음 코드는 컬렉션(배열)에 있는 요소 중에서 홀수만 출력하는 예제 입니다.

```
public static void Main()
{
    int[] arr = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

        // 방법 1. LINQ 를 사용하지 않고, foreach 를 사용한 코드
    foreach (var n in arr)
    {
        if (n % 2 == 1)
            Console.WriteLine(n);
    }

    // 방법 2. LINQ 의 “Where” 사용
    IEnumerable<int> c = arr.Where(n => n % 2 == 1);

    foreach (var n2 in c)
        Console.WriteLine(n2);
}
```

Fluent Syntax

```
class Program
{
    public static void Main()
    {
        string[] arr = { "kim", "park", "choi", "lee", "jung" };

        // Fluent Syntax : Where, OrderBy, Select
        var names = arr.Where(s => s.Contains('i'))
                       .OrderBy(s => s.Length)
                       .Select(s => s.ToUpper());

        foreach (var n in names)
        {
            Console.WriteLine(n);
        }
    }
}
```

Linq 반환 값

LINQ의 반환 타입은 컬렉션 또는 값일 수 있습니다.

```
class Program
{
    public static void Main()
    {
        int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

        // 핵심 1. 컬렉션을 반환하는 질의
        // => result는 컬렉션임. IEnumerable<int>
        var result = numbers.Take(3);

        foreach (int n in result)
            Console.WriteLine(n);

        // 핵심 2. 순차열이 아닌 값을 반환하는 질의 연산자
        int ret = numbers.First();
        Console.WriteLine(ret);

        // 핵심 3. 순차열 결합
        int[] arr1 = { 1, 2, 3 };
```

```
        int[] arr2 = { 3, 4, 5 };

        var arr3 = arr1.Concat(arr2);

        foreach (int n in arr3)
            Console.WriteLine(n);
    }
}
```

▌ 지연된 실행, 캡쳐된 변수, ToList()

Linq 는 Collection 에 대한 참조와, 적용할 함수를 만들어서 내부적으로 보관합니다.

따라서, 실제 적용은 반환된 컬렉션을 열거 할 때 적용됩니다.

```
class Program
{
    public static void Main()
    {
        IList<int> list = new List<int> { 1, 2, 3 };

        //var result = list.Select(n => n * 10);

        // 1. Select 절을 가지고 각 요소에 적용할 함수를 만듭니다.
        // 2. result 안에는 list에 대한 참조와 1에서 만든 함수가 보관됩니다.
        // 핵심은 함수가 아직 적용되지는 않은 상태 입니다.

        IEnumerable<int> result = list.Select(n => n * 10);

        list.Clear();

        foreach (int n in result)
            Console.WriteLine(n); // 결과 어떻게 나올까요 ?
    }
}
```

▌ LINQ 의 원리

Select 는 확장 메소드와 Coroutine 을 사용해서 만들 수 있습니다.

```
class Program
{
    public static void Main()
    {
        int[] arr = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

        IEnumerable<int> result  = arr.MySelect(n => n * 2);

        foreach (int n in result)
            Console.WriteLine(n);
    }
}
// 배열에 새로운 멤버 함수(MySelect)를 추가해야 한다.
// 확장 메소드 문법
static class ArrayExtension
{
    public static IEnumerable<int> MySelect( this Array ar,
                                             Func<int, int> f)
    {
        foreach (int elem in ar)
            yield return f(elem);

    }
}
```

항목 1-18

Collection

☑ 주요 내용

> Object Collection vs Generic Collection
> IEnumerator
> Find & Sort

1. Object Collection vs Generic Collection

▌ 기본 개념

Generic Collection 을 사용하면 타입 안정성이 뛰어납니다.

```
using System;
using System.Collections;
using System.Collections.Specialized;
using System.Collections.Generic;
using System.Collections.Concurrent;

class Program
{
    static void Main()
    {
        ArrayList c1 = new ArrayList();

        c1.Add(1);
        c1.Add(2);
        //c1.Add("aaa"); // c2대신 c1 이라고 오타 발생

        int n = (int)c1[0];
        StringCollection c2 = new StringCollection();

        c2.Add("aa");
        //c2.Add(1); // error. 타입 안정성이 뛰어 나다.
        string s2 = c2[0];

        List<int>        c3 = new List<int>();
        c3.Add(1);
        //c3.Add("aa"); // error. 타입 안정성이 뛰어나다.
        int n2 = c3[0];

        ConcurrentStack<int> c4 = new ConcurrentStack<int>();
        c4.Push(10);
    }
}
```

2. 열거자 - IEnumerator

▌ 기본 개념

열거자(반복자)를 사용하면 컬렉션 내의 모든 요소를 열거 할 수 있습니다.

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
        int[] arr = { 1, 2, 3, 4, 5 };

        List<int>       c1 = new List<int>(arr);
        LinkedList<int> c2 = new LinkedList<int>(arr);

        IEnumerator<int> e1 = c1.GetEnumerator();
        IEnumerator<int> e2 = c2.GetEnumerator();
        // var  e1 =

        //e1.MoveNext(); // 최초 호출 - 초기화
        //Console.WriteLine(e1.Current); // 1

        while( e2.MoveNext())  // 다음으로 이동
            Console.WriteLine(e2.Current); // 2

        e2.Reset(); //초기상태로 reset

        while (e2.MoveNext())  // 다음으로 이동
            Console.WriteLine(e2.Current); // 2
    }
}
```

3. Collection Method

▌ 검색

```
using System;
using System.Collections.Generic;

class Program
{
    public static bool Divide3(int n) { return n % 3 == 0; }

    static void Main()
    {
        List<int> c1 = new List<int>() { 1, 2, 3, 1, 2, 3, 1, 2, 3, 9 };

        // 값 검색
        Console.WriteLine( c1.IndexOf(3) ); // 2
        Console.WriteLine( c1.IndexOf(3, 5)); // 5
        Console.WriteLine( c1.IndexOf(3, 6, 2)); // -1

        // 조건 검색 : 3의 배수 찾기
        Console.WriteLine(c1.FindIndex(Divide3) ); // 2
        Console.WriteLine(c1.FindIndex(n => n % 3 == 0)); // 2

        // 조건을 만족하는 모든 요소 찾기
        List<int> c2 = c1.FindAll(n => n % 3 == 0);

        foreach (int n in c2)
            Console.WriteLine(n);

    }
}
```

▌ 메소드의 정책 변경

Sort 메소드를 사용하면 Collection 내의 모든 요소를 정렬할 수 있습니다.

이때, Sort의 비교 정책을 변경하려면 Delegate 또는 IComparer<T> 인터페이스를 구현한 객체를 전달하면 됩니다.

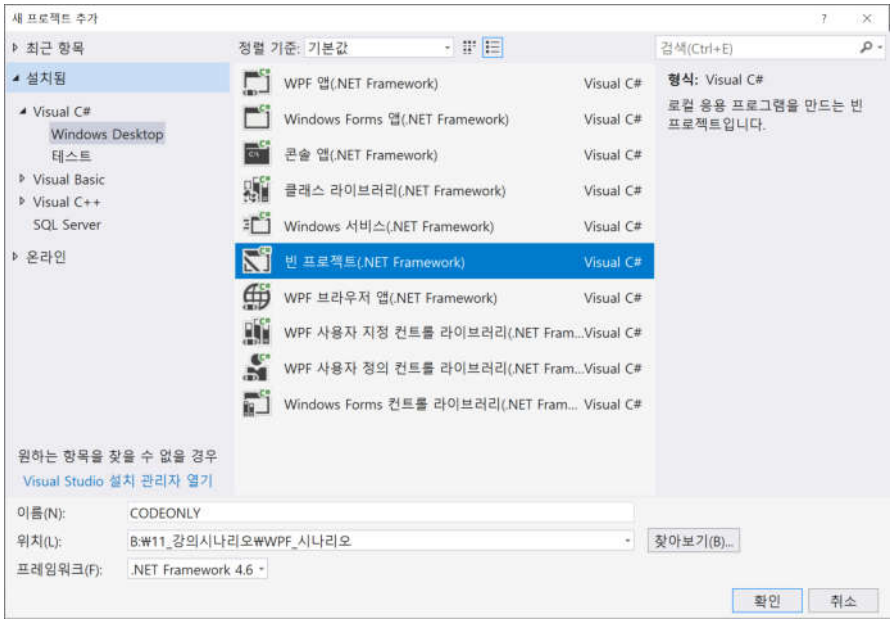
```
using System;
using System.Collections.Generic;

class Program
{
    public static bool Divide3(int n) { return n % 3 == 0; }

    static void Main()
    {
        List<int> c1 = new List<int>() { 1, 2, 3, 1, 2, 3, 1, 2, 3, 9 };

        // 값 검색
        Console.WriteLine( c1.IndexOf(3) ); // 2
        Console.WriteLine( c1.IndexOf(3, 5)); // 5
        Console.WriteLine( c1.IndexOf(3, 6, 2)); // -1

        // 조건 검색 : 3의 배수 찾기
        Console.WriteLine(c1.FindIndex(Divide3) ); // 2
        Console.WriteLine(c1.FindIndex(n => n % 3 == 0)); // 2

        // 조건을 만족하는 모든 요소 찾기
        List<int> c2 = c1.FindAll(n => n % 3 == 0);

        foreach (int n in c2)
            Console.WriteLine(n);
    }
}
```

SECTION 2

WPF 프로그래밍 구조

1. 프로젝트 만들기

▌ 빈 프로젝트(.NET Framework)



▌ 소스 코드

```
// Step01.cs
using System;

namespace CODEONLY
{
    public class Entry
    {
        public static void Main()
        {
            Console.WriteLine("Hello, WPF");
        }
    }
}
```

2. Application 객체와 Window

참조 추가

- PresentationFramwork
- WindowsBase

소스 코드

```
// Step02.cs
using System;
using System.Windows;

namespace CODEONLY
{
    public class Entry
    {
        [STAThread]
        public static void Main()
        {
            Application app = new Application();

            Console.WriteLine("Hello, WPF");

            app.Run(new Window());
        }
    }
}
```

3. Application Event 처리

▌ 소스 코드

```
// Step03.cs
using System;
using System.Windows;

namespace CODEONLY
{
    public class Entry
    {
        public static void AppStartup(object sender, StartupEventArgs e)
        {
            Console.WriteLine("AppStartup");
        }
        public static void AppExit(object sender, ExitEventArgs e)
        {
            Console.WriteLine("AppExit");
        }
        public static void WindowActivate(object sender, EventArgs e)
        {
            Console.WriteLine("WindowActivate");
        }
        [STAThread]
        public static void Main()
        {
            Application app = new Application();
            app.Startup += AppStartup;
            app.Exit += AppExit;
            app.Activated += WindowActivate;

            app.Run(new Window());
        }
    }
}
```

4. Application 파생 클래스와 가상 메소드 재정의

▌Delegate 방식 vs 가상 메소드 방식

- Delegate 방식의 callback 메소드 : 인자 2개
- 가상 메소드 : 인자 1개

▌소스 코드

```
// Step04.cs
using System;
using System.Windows;

namespace CODENONLY
{
    public class App : System.Windows.Application
    {
        protected override void OnStartup(StartupEventArgs e)
        {
            base.OnStartup(e);
            Console.WriteLine("OnStartup");
        }

        [STAThread]
        public static void Main()
        {
            App app = new App();

            app.Run(new Window());
        }
    }
}
```

5. 마우스, 키보드 이벤트 처리

▌ 참조 추가

- PresentationCore
- System
- System.Xaml

▌ 소스 코드

```
// Step05.cs
using System;
using System.Windows;
using System.Windows.Input;

namespace CODEONLY
{
    public class App : System.Windows.Application
    {
        public void Window_MouseDown(object sender, MouseButtonEventArgs e)
        {
            Point p = e.GetPosition(this.MainWindow);

            switch (e.ChangedButton)
            {
                case MouseButton.Left:
                                        Console.WriteLine($"Left {p.X} {p.Y}");
                                        break;

                case MouseButton.Right:
                                        Console.WriteLine($"Right {p.X} {p.Y}");
                                        break;
            }
        }

        [STAThread]
        public static void Main()
        {
            App app = new App();
```

```
            Window win = new Window();

            win.MouseDown += app.Window_MouseDown;
            //win.KeyDown += app.Window_KeyDown;

            win.Show();  // PresentationCore 필요

            // Main Window 를 등록하는 방법
            // 1. app.MainWindow 에 등록
            // app.MainWindow = win;
            // app.Run();

            // 2. Run 메소드에 인자로 전달
            app.Run(win);
        }
    }
}
```

6. Window 파생 클래스, 가상 메소드 재정의

▌ 소스 코드

```
// Step06.cs
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace CODEONLY
{
    public class MainWindow : System.Windows.Window
    {
        protected override void OnMouseDown(MouseButtonEventArgs e)
        {
            base.OnMouseDown(e);
            Console.WriteLine("MouseDown");
        }
        public MainWindow()
        {
        }
    }
    public class App : System.Windows.Application
    {
        [STAThread]
        public static void Main()
        {
            App app = new App();
            MainWindow win = new MainWindow();
            win.Show();
            app.Run(win);
        }
    }
}
```

7. Child Control, Grid

▌ 소스 코드

```
// Step07.cs
using System;
using System.Windows;
using System.Windows.Controls;

namespace CODEONLY
{
    public class MainWindow : System.Windows.Window
    {
        private Button btn1 = null;
        private Button btn2 = null;
        private Grid grid = null;

        public MainWindow()
        {
            grid = new Grid();
            grid.RowDefinitions.Add(new RowDefinition());
            grid.RowDefinitions.Add(new RowDefinition());

            this.Content = grid;

            btn1 = new Button();
            btn1.Content = "확인1";
            Grid.SetRow(btn1, 0);

            btn2 = new Button();
            btn2.Content = "확인2";
            Grid.SetRow(btn2, 1);

            grid.Children.Add(btn1);
            grid.Children.Add(btn2);
        }
    }
    public class App : System.Windows.Application
    {
        [STAThread]
        public static void Main()
        {
```

```
            App app = new App();
            MainWindow win = new MainWindow();
            win.Show();
            app.Run(win);
        }
    }
}
```

8. Child Control event

▌ 소스 코드

```
// Step08.cs
using System;
using System.Windows;
using System.Windows.Controls;

// 버튼 이벤트 처리

namespace CODEONLY
{
    public class MainWindow : System.Windows.Window
    {
                // ……
        protected void InitializeComponent()
        {
                        // ……

            // button event 처리
            btn1.Click += OnButton1_Click;
        }

        private void OnButton1_Click(object sender, RoutedEventArgs e)
        {
            Button btn = sender as Button;
            string s1 = btn.Content as string;
            Console.WriteLine($"Click , {s1}");
        }
    }
}
```

9. XAML Without Window

참조 추가

- System.xml

Step09.xaml

```
<Grid xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
<Button Name = "button1"> 확인1 </Button>
</Grid>
```

소스 코드

```
// Step09.cs
using System;
using System.IO;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Markup;

namespace CODEONLY
{
    public class MainWindow : System.Windows.Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
        protected void InitializeComponent()
        {
            DependencyObject rootElement;

            using (FileStream fs = new FileStream("Step9.xaml", FileMode.Open))
            {
                rootElement = (DependencyObject)XamlReader.Load(fs);
            }
            // Windows 에 grid 연결
            this.Content = rootElement;
```

```
                        FrameworkElement frameworkElement =
(FrameworkElement)rootElement;
            Button btn1 = (Button)frameworkElement.FindName("button1");
                        btn1.Click += OnButton1_Click;
        }

        private void OnButton1_Click(object sender, RoutedEventArgs e)
        {
            Button btn = sender as Button;
            string s1 = btn.Content as string;
            Console.WriteLine($"Click , {s1}");
        }
    }
    public class App : System.Windows.Application
    {
        [STAThread]
        public static void Main()
        {
            App app = new App();
            MainWindow win = new MainWindow();
            win.Show();
            app.Run(win);
        }
    }
}
```

10. XAML With Window

Step10.xaml

```
<local:MainWindow xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                  xmlns:local="clr-namespace:CODEONLY;assembly=CODEONLY"
                  Title="Step10" Height="450" Width="800">
        <Grid>
                <Button Name = "button1" Click="OnButton1_Click"> 확인1 </Button>
        </Grid>
</local:MainWindow>
```

소스 코드

```
// Step10.cs
using System;
using System.IO;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Markup;

namespace CODEONLY
{
    public class MainWindow : System.Windows.Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
        protected void InitializeComponent()
        {
        }
        private void OnButton1_Click(object sender, RoutedEventArgs e)
        {
            Button btn = sender as Button;
            string s1 = btn.Content as string;
            Console.WriteLine($"Click , {s1}");
        }
    }
```

```
    public class App : System.Windows.Application
    {
        [STAThread]
        public static void Main()
        {
            App app = new App();
            MainWindow win = null;
            using (FileStream fs = new FileStream("Step10.xaml", FileMode.Open))
            {
                win = (MainWindow)XamlReader.Load(fs);
            }
            win.Show();
            app.Run(win);
        }
    }
}
```

11. XAML With User Define Type

Step11.xaml

```
<local:MainWindow xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                                xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                        xmlns:local="clr-namespace:CODEONLY;assembly=CODEONLY"
                        Title="Step11" Height="450" Width="800">

<Grid>
        <Button Name = "button1" Click="OnButton1_Click">
                <local:People>
                        <x:Arguments>
                                <x:String>Kim</x:String>
                        </x:Arguments>
                </local:People>
        </Button>
</Grid>

</local:MainWindow>
```

소스 코드

```
// Step11.cs
using System;
using System.IO;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Markup;

namespace CODEONLY
{
    public class People
    {
        public string Name { set; get; } = "UNKNOWN";

        public People(string name) { Name = name; Console.WriteLine("People()"); }

        public override string ToString()
        {
```

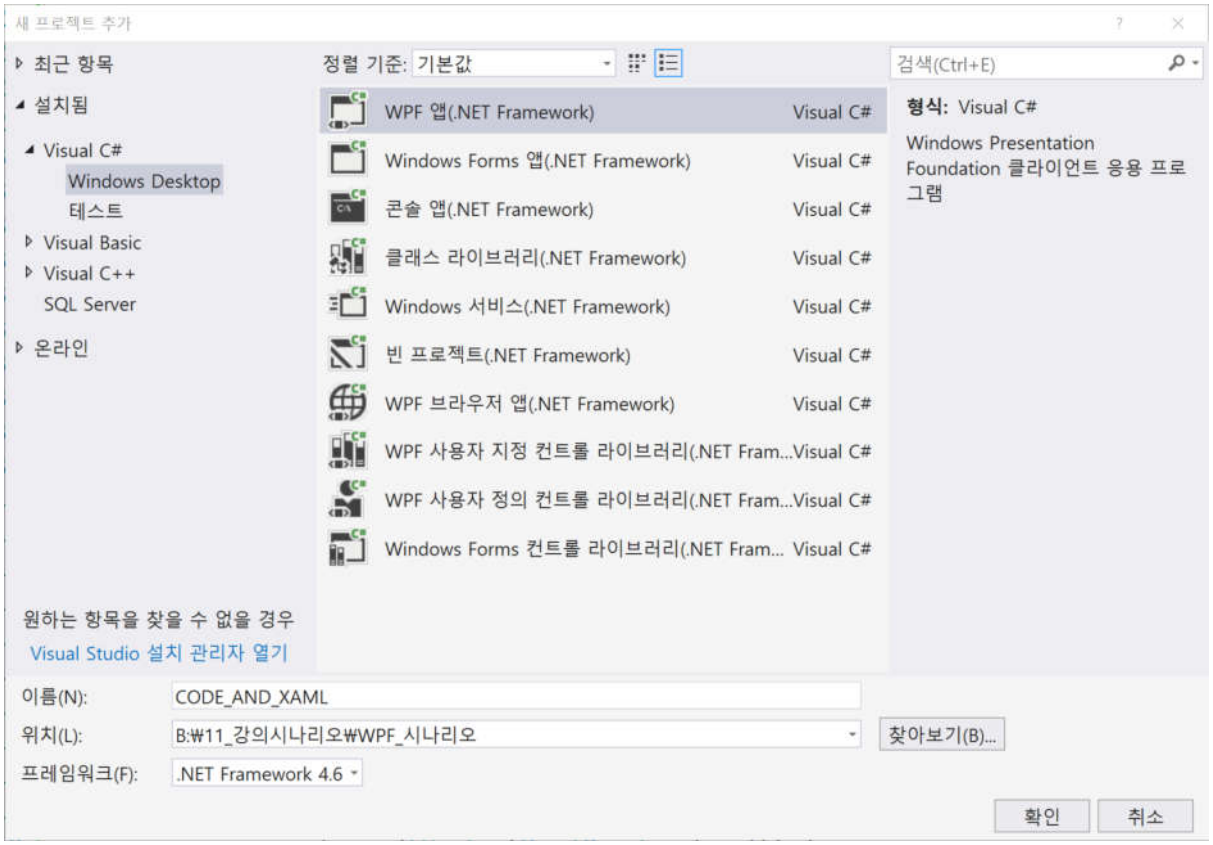
```
            return Name;
        }
    }

    public class MainWindow : System.Windows.Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        protected void InitializeComponent()
        {
        }
        public void OnButton1_Click(object sender, RoutedEventArgs e)
        {
            Button btn = sender as Button;
            string s1 = btn.Content as string;
            Console.WriteLine($"Click , {s1}");
        }
    }

    public class App : System.Windows.Application
    {
        [STAThread]
        public static void Main()
        {
            App app = new App();
            Window win = null;

            using (FileStream fs = new FileStream("Step11.xaml", FileMode.Open))
            {
                win = (Window)XamlReader.Load(fs);
            }
            win.Show();
            app.Run(win);
        }
    }
}
```

SECTION 3

CODE + XAML

1. 프로젝트 만들기

▌ WPF 앱(.NET Framework)



▌ 생성된 소스

Application	App.xaml App.Xaml.cs App.g.i.cs
Window	MainWindow.xaml MainWindow.xaml.cs MainWindow.g.i.cs

▌ Window 창 변경

App.xaml 에서 StartupUri 변경

```
<Application x:Class="CODE_AND_XAML.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:CODE_AND_XAML"
             StartupUri="Step1.xaml">
    <Application.Resources>

    </Application.Resources>
</Application>
```

2. XAML 과 컨트롤

컨트롤의 속성을 변경하는 방법

- XAML에서 attribute 변경
- 속성창 이용
- 코드로 변경 – Window_Load에서 처리.

Step1.xaml

```
//……
<Grid>
<Button Name="button1" Content="Button" HorizontalAlignment="Left" Margin="148,129,0,0"
VerticalAlignment="Top" Width="240" Height="46" Background="#FFCA4444"/>
</Grid>
```

Step1.xaml.cs

```
//……
<Grid>
<Button Name="button1" Content="Button" HorizontalAlignment="Left" Margin="148,129,0,0"
VerticalAlignment="Top" Width="240" Height="46" Background="#FFCA4444"/>
</Grid>
```

버튼 이벤트 처리

- Xaml 에서 Click attribute 작성
- 코드에서 핸들러 작성

3.새로운 창을 나타내는 방법

- WPF 창(xaml)을 추가
- 창을 나타내고 싶을 때 객체 생성

▌Step1.xaml.cs

```
private void Button1_Click(object sender, RoutedEventArgs e)
{
        Step2 win = new Step2();

        //win.Show();
        win.ShowDialog();
}
```

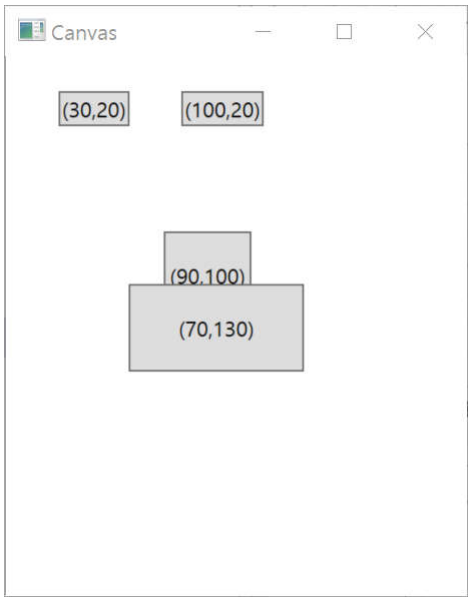
SECTION 4

Layout

1. Canvas

Canvas.xaml

```
<Canvas>
    <Button Canvas.Left="30" Canvas.Top="20">(30,20)</Button>
    <Button Canvas.Left="100" Canvas.Top="20">(100,20)</Button>
    <Button Canvas.Left="90" Canvas.Top="100" Width="50" Height="50">(90,100)</Button>
    <Button Canvas.Left="70" Canvas.Top="130" Width="100" Height="50">(70,130)</Button>
</Canvas>
```



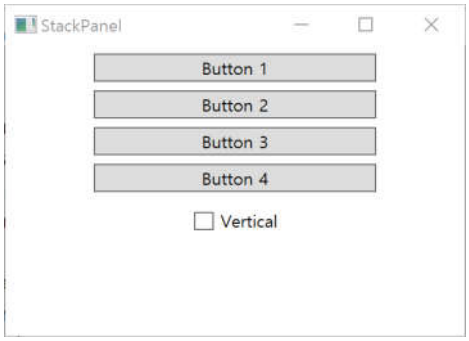
2. StackPanel

StackPanel.xaml

```
<StackPanel Margin="3" Name="stackPanel1">
        <Button Margin="3" MaxWidth="200" MinWidth="100">Button 1</Button>
        <Button Margin="3" MaxWidth="200" MinWidth="100">Button 2</Button>
        <Button Margin="3" MaxWidth="200" MinWidth="100">Button 3</Button>
        <Button Margin="3" MaxWidth="200" MinWidth="100">Button 4</Button>
        <CheckBox Name="chkVertical" Margin="10" HorizontalAlignment="Center"
                  Checked="ChkVertical_Checked" Unchecked="ChkVertical_Unchecked">
                  Vertical</CheckBox>
    </StackPanel>
```

StackPanel.xaml.cs

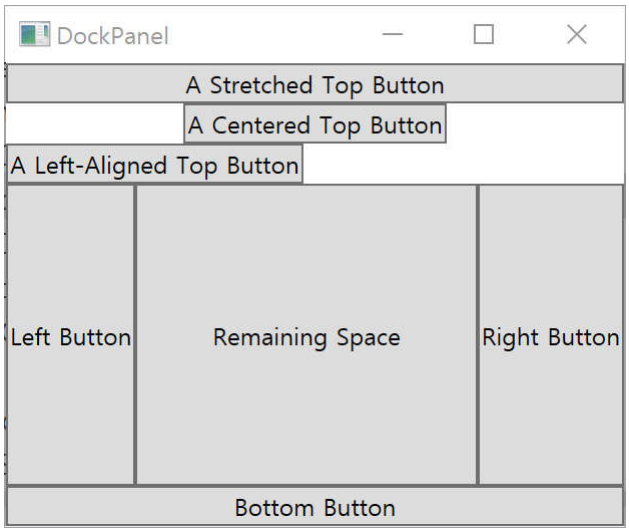
```
private void ChkVertical_Checked(object sender, RoutedEventArgs e)
{
    stackPanel1.Orientation = Orientation.Horizontal;
}
private void ChkVertical_Unchecked(object sender, RoutedEventArgs e)
{
    stackPanel1.Orientation = Orientation.Vertical;
}
```



3. DockPanel

▌ DockPanel.xaml

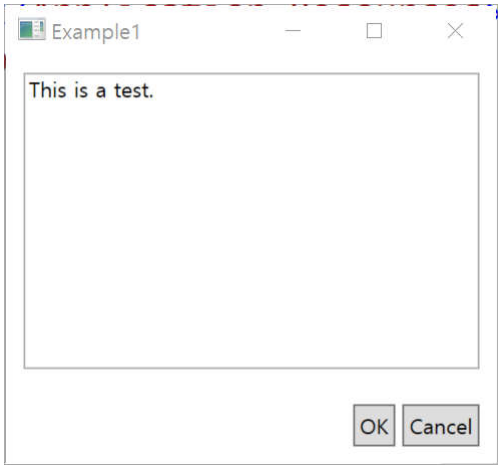
```
<DockPanel LastChildFill="True">
        <Button DockPanel.Dock="Top">A Stretched Top Button</Button>
        <Button DockPanel.Dock="Top" HorizontalAlignment="Center">A Centered Top
Button</Button>
        <Button DockPanel.Dock="Top" HorizontalAlignment="Left">A Left-Aligned Top
Button</Button>
        <Button DockPanel.Dock="Bottom">Bottom Button</Button>
        <Button DockPanel.Dock="Left">Left Button</Button>
        <Button DockPanel.Dock="Right">Right Button</Button>
        <Button >Remaining Space</Button>
</DockPanel>
```



4. Example 1. DockPanel + StackPanel

▌ Example1.xaml

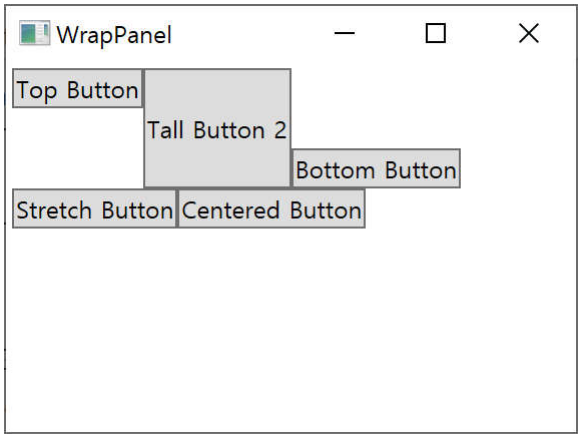
```
<DockPanel LastChildFill="True">
    <StackPanel DockPanel.Dock="Bottom" HorizontalAlignment="Right"
Orientation="Horizontal">
        <Button Margin="10,10,2,10" Padding="3,3,3,3">OK</Button>
        <Button Margin="2,10,10,10" Padding="3,3,3,3">Cancel</Button>
    </StackPanel>
    <TextBox DockPanel.Dock="Top" Margin="10">This is a test.</TextBox>
</DockPanel>
```



5. WrapPanel

WrapPanel.xaml

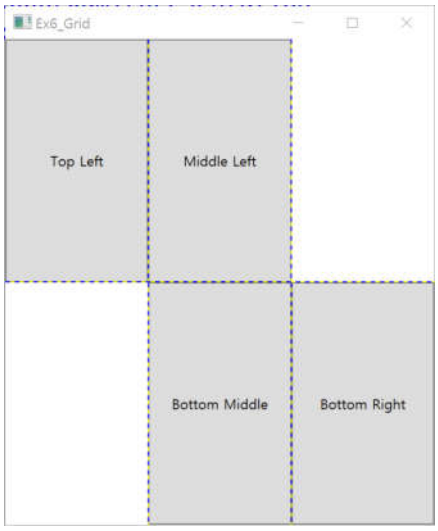
```
<WrapPanel Margin="3">
        <Button VerticalAlignment="Top">Top Button</Button>
        <Button MinHeight="60">Tall Button 2</Button>
        <Button VerticalAlignment="Bottom">Bottom Button</Button>
        <Button>Stretch Button</Button>
        <Button VerticalAlignment="Center">Centered Button</Button>
    </WrapPanel>
```



6. Grid

Grid.xaml

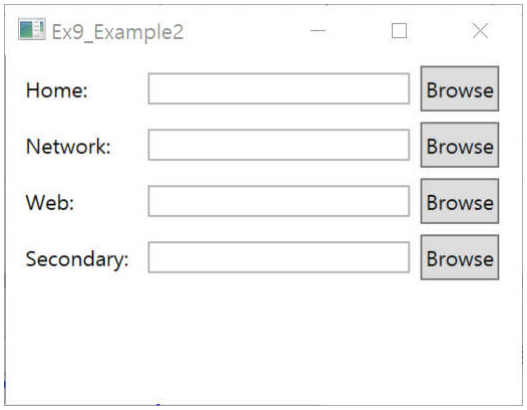
```
<Grid ShowGridLines="True">
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Button Grid.Row="0" Grid.Column="0">Top Left</Button>
        <Button Grid.Row="0" Grid.Column="1">Middle Left</Button>
        <Button Grid.Row="1" Grid.Column="2">Bottom Right</Button>
        <Button Grid.Row="1" Grid.Column="1">Bottom Middle</Button>
</Grid>
```

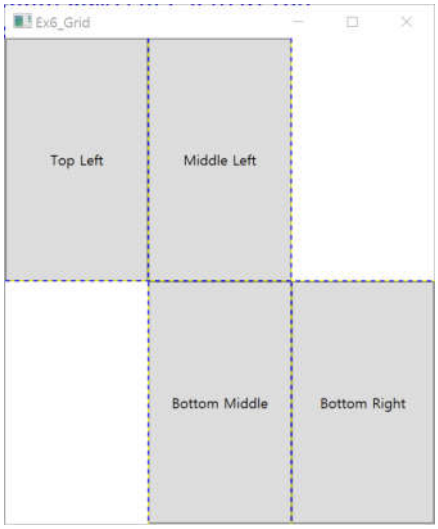


7. Grid - Example

Example2.xaml

```
<Grid Margin="3,3,10,3">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"></ColumnDefinition>
        <ColumnDefinition Width="*" MinWidth="50" MaxWidth="800"></ColumnDefinition>
        <ColumnDefinition Width="Auto"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Label Grid.Row="0" Grid.Column="0" Margin="3"
         VerticalAlignment="Center">Home:</Label>
    <TextBox Grid.Row="0" Grid.Column="1" Margin="3"
         Height="Auto"  VerticalAlignment="Center"></TextBox>
    <Button Grid.Row="0" Grid.Column="2" Margin="3" Padding="2">Browse</Button>
    <Label Grid.Row="1" Grid.Column="0" Margin="3"
         VerticalAlignment="Center">Network:</Label>
    <TextBox Grid.Row="1" Grid.Column="1" Margin="3"
         Height="Auto"  VerticalAlignment="Center"></TextBox>
    <Button Grid.Row="1" Grid.Column="2" Margin="3" Padding="2">Browse</Button>
    <Label Grid.Row="2" Grid.Column="0" Margin="3"
             VerticalAlignment="Center">Web:</Label>
    <TextBox Grid.Row="2" Grid.Column="1" Margin="3"
         Height="Auto"  VerticalAlignment="Center"></TextBox>
    <Button Grid.Row="2" Grid.Column="2" Margin="3" Padding="2">Browse</Button>
    <Label Grid.Row="3" Grid.Column="0" Margin="3"
             VerticalAlignment="Center">Secondary:</Label>
    <TextBox Grid.Row="3" Grid.Column="1" Margin="3" Height="Auto"
                          VerticalAlignment="Center"></TextBox>
    <Button Grid.Row="3" Grid.Column="2" Margin="3" Padding="2">Browse</Button>
</Grid>
```


Ex9_Example2
Home:
Browse
Network:
Browse
Web:
Browse
Secondary:
Browse


Ex6_Grid
Top Left
Middle Left
Bottom Middle
Bottom Right

8. GridSplitter

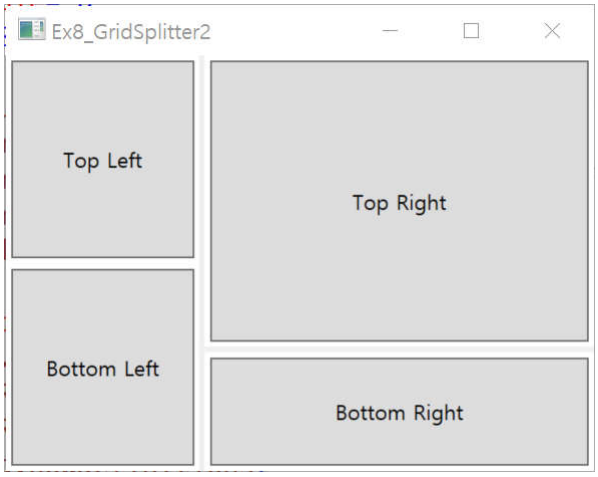
▌ GridSplitter.xaml

```
<Grid>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition MinWidth="100"></ColumnDefinition>
            <ColumnDefinition Width="Auto"></ColumnDefinition>
            <ColumnDefinition MinWidth="50"></ColumnDefinition>
        </Grid.ColumnDefinitions>

        <Button Grid.Row="0" Grid.Column="0" Margin="3">Left</Button>
        <Button Grid.Row="0" Grid.Column="2" Margin="3">Right</Button>
        <Button Grid.Row="1" Grid.Column="0" Margin="3">Left</Button>
        <Button Grid.Row="1" Grid.Column="2" Margin="3">Right</Button>

        <GridSplitter Grid.Row="0" Grid.Column="1" Grid.RowSpan="2"
                  Width="3" VerticalAlignment="Stretch" HorizontalAlignment="Center"
                  ShowsPreview="False"></GridSplitter>
</Grid>
```

9. GridSplitter - Double

▌ GridSplitter2.xaml

```
<Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition Width="Auto"></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>

        <Grid Grid.Column="0" VerticalAlignment="Stretch">
            <Grid.RowDefinitions>
                <RowDefinition></RowDefinition>
                <RowDefinition></RowDefinition>
            </Grid.RowDefinitions>
            <Button Margin="3" Grid.Row="0">Top Left</Button>
            <Button Margin="3" Grid.Row="1">Bottom Left</Button>
        </Grid>

        <GridSplitter Grid.Column="1" Width="3" VerticalAlignment="Stretch"
                      HorizontalAlignment="Center"
ShowsPreview="False"></GridSplitter>

        <Grid Grid.Column="2">
            <Grid.RowDefinitions>
                <RowDefinition></RowDefinition>
                <RowDefinition Height="Auto"></RowDefinition>
                <RowDefinition></RowDefinition>
            </Grid.RowDefinitions>

            <Button Grid.Row="0" Margin="3">Top Right</Button>
            <Button Grid.Row="2" Margin="3">Bottom Right</Button>

            <GridSplitter Grid.Row="1" Height="3" VerticalAlignment="Center"
                  HorizontalAlignment="Stretch" ShowsPreview="False"></GridSplitter>
        </Grid>
    </Grid>
```


Ex8_GridSplitter2
Top Left
Top Right
Bottom Left
Bottom Right

SECTION 5

Routed Event

1. Bubbled Event & Tunelling Event

RautedEvent.xaml

```
<Window x:Class="RAUTEDEVENT.Ex1_BubbledEvent"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:RAUTEDEVENT"
        mc:Ignorable="d"
        Title="Ex1_BubbledEvent" Height="450" Width="800" MouseDown="Window_MouseDown"
PreviewMouseDown="Window_PreviewMouseDown">
    <Grid Background="Aqua" MouseDown="Grid_MouseDown"
PreviewMouseDown="Grid_PreviewMouseDown">
        <Label Content="Label" MouseDown="Label_MouseDown"
PreviewMouseDown="Label_PreviewMouseDown"  HorizontalAlignment="Left"
Margin="250,274,0,0" VerticalAlignment="Top"/>
    </Grid>
</Window>
```

RautedEvent.xaml.cs

```
private void Label_MouseDown(object sender, MouseButtonEventArgs e)
{
    Console.WriteLine("Label_MouseDown");
}
private void Grid_MouseDown(object sender, MouseButtonEventArgs e)
{
    Console.WriteLine("Grid_MouseDown");
}
private void Window_MouseDown(object sender, MouseButtonEventArgs e)
{
    Console.WriteLine("Window_MouseDown");
}
private void Window_PreviewMouseDown(object sender, MouseButtonEventArgs e)
{
    Console.WriteLine("Window_PreviewMouseDown");
```

```
}
private void Grid_PreviewMouseDown(object sender, MouseButtonEventArgs e)
{
    Console.WriteLine("Grid_PreviewMouseDown");
}
private void Label_PreviewMouseDown(object sender, MouseButtonEventArgs e)
{
    Console.WriteLine("Label_PreviewMouseDown");
}
```

2. KeyPressEvent

KeyPressEvent.xaml

```
<StackPanel>
    <TextBox PreviewKeyDown = "KeyEvent" KeyDown="KeyEvent"
                    PreviewKeyUp="KeyEvent" KeyUp="KeyEvent"
                    PreviewTextInput="textInput"
                    TextChanged="TextChanged">
    </TextBox>
</StackPanel>
```

KeyPressEvent.xaml.cs

```
private void KeyEvent(object sender, KeyEventArgs e)
{
    string message = "(KeyEvent)Event: " + e.RoutedEvent + " " + " Key: " + e.Key;
    Console.WriteLine(message);
}
private void textInput(object sender, TextCompositionEventArgs e)
{
    string message = "(textInput)Event: " + e.RoutedEvent + " " + " Text: " + e.Text;
    Console.WriteLine(message);
}
private void TextChanged(object sender, TextChangedEventArgs e)
{
    string message = "(TextChanged)Event: " + e.RoutedEvent;
    Console.WriteLine(message);
}
```

3. Example - NumericTextBox

NumericTextBox.xaml

```
<StackPanel Margin = "5" PreviewTextInput="pnl_PreviewTextInput"
PreviewKeyDown="pnl_PreviewKeyDown">
    <TextBox Margin = "3" AcceptsTab="False"></TextBox>
    <TextBox Margin = "3" ></ TextBox >
    < TextBox Margin="3"></TextBox>
</StackPanel>
```

NumericTextBox.xaml.cs

```
private void pnl_PreviewTextInput(object sender, TextCompositionEventArgs e)
{
    short val;
    if (!Int16.TryParse(e.Text, out val))
    {
        e.Handled = true;
    }
}

private void pnl_PreviewKeyDown(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Space)
    {
        e.Handled = true;
    }
}
```

4. AddHandler

AddHandler.xaml

```
<Grid>
    <Button Name = "cmd" Content="Button" MouseDown="Button_MouseDown"
MouseUp="Button_MouseUp" HorizontalAlignment="Left" Margin="178,134,0,0"
VerticalAlignment="Top" Width="212" Height="58"/>
    <Label Content = "Label" MouseDown="Label_MouseDown" MouseUp="Label_MouseUp"
HorizontalAlignment="Left" Margin="183,231,0,0" VerticalAlignment="Top" Width="219"/>
</Grid>
```

AddHandler.xaml.cs

```
private void Button_MouseDown(object sender, MouseButtonEventArgs e)
{
    Console.WriteLine("MouseDown");
}
private void Button_MouseUp(object sender, MouseButtonEventArgs e)
{
    Console.WriteLine("MouseUp");
}
private void Label_MouseDown(object sender, MouseButtonEventArgs e)
{
    Console.WriteLine("label MouseDown");
}
private void Label_MouseUp(object sender, MouseButtonEventArgs e)
{
    Console.WriteLine("label MouseUp");
}
private void Backdoor(object sender, RoutedEventArgs e)
{
    Console.WriteLine("button mouseup");
}
```

SECTION 6

Control

1. Control Style

ControlStyle.xaml

```
<Grid>
        <Button Name ="button1" Background="Yellow" Content="Button"
HorizontalAlignment="Left" Margin="42,33,0,0" VerticalAlignment="Top" Width="75"/>
        <Button Name ="button2" Content="Button" HorizontalAlignment="Left"
Margin="41,74,0,0" VerticalAlignment="Top" Width="75">
            <Button.Background>
                <SolidColorBrush Color="Red"/>
            </Button.Background>
        </Button>
        <Button Name ="button3" Content="Button" HorizontalAlignment="Left"
Margin="41,115,0,0" VerticalAlignment="Top" Width="75">
            <Button.Background>
                <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                    <GradientStop Color="Black" Offset="0"/>
                    <GradientStop Color="#FFEDDADA" Offset="1"/>
                    <GradientStop Color="#FF040404" Offset="0.491"/>
                </LinearGradientBrush>
            </Button.Background>
        </Button>
        <Button Name ="button4" Content="Button" HorizontalAlignment="Left"
Margin="40,157,0,0" VerticalAlignment="Top" Width="75"/>
        <Button Name ="button5" HorizontalAlignment="Left" Margin="40,207,0,0"
VerticalAlignment="Top" Width="145">
            <StackPanel>
                <CheckBox Content="동의합니다."></CheckBox>
            </StackPanel>
        </Button>
    </Grid>
```

ControlStyle.xaml.cs

```
public partial class Ex1_ControlStyle : Window
{
    public Ex1_ControlStyle()
    {
        InitializeComponent();
        button4.Background = new SolidColorBrush(Colors.Yellow);
    }
}
```

2. ControlEvent

▌ ControlEvent.xaml

```
< StackPanel ButtonBase.Click = "StackPanel_Click" >
    < Button Name = "button1" Click = "Button1_Click" Content="Button1" Margin= "10" />
    < Button Name = "button2" Click = "Button_Click" Content="Button2" Margin = "10" />
    < Button Name = "button3" Click = "Button_Click" Content="Button3" Margin = "10" />
    < Button Name = "button4" Content = "Button4" Margin = "10" />
    < Button Name = "button5" Content = "Button5" Margin = "10" >
        < Button.Triggers >
            < EventTrigger RoutedEvent = "Button.Click" >
                < EventTrigger.Actions >
                </ EventTrigger.Actions >
            </ EventTrigger >
        </ Button.Triggers >
    </ Button >


    < Button Name = "button6" Click = "Clicked" > Click Me! </ Button >
    < x:Code >
        < ![CDATA[
            void Clicked(object sender, RoutedEventArgs e)
            {
                button1.Content = "Hello World";
            }]]>
    </ x:Code >
</ StackPanel >
```

▌ ControlEvent.xaml.cs

```
public partial class Ex2_ControlEvent : Window
{
    public Ex2_ControlEvent()
    {
        InitializeComponent();

        button4.AddHandler(Button.ClickEvent, new RoutedEventHandler(Button4_Click),
```

```
true);
    }
    private void Button4_Click(object sender, RoutedEventArgs e)
    {
        MessageBox.Show("Click4");
    }
    private void Button1_Click(object sender, RoutedEventArgs e)
    {

    }
    private void Button_Click(object sender, RoutedEventArgs e)
    {
        Button btn = sender as Button;

        string title = (string)btn.Content;

        MessageBox.Show(title);
    }

    private void StackPanel_Click(object sender, RoutedEventArgs e)
    {

        MessageBox.Show("StackPanel_Click");

        Button btn = e.OriginalSource as Button;
        string title = (string)btn.Content;
        MessageBox.Show(title);
    }
}
```

3. ControlContent

▌ ControlContent.xaml

```
public partial class Ex3_ControlContent : Window
{
    private Button btn1 = null;   // Content 만 가짐
    private GroupBox gbox = null;   // Header + Content

    public Ex3_ControlContent()
    {
        InitializeComponent();

        btn1 = new Button();
        btn1.Content = "OK";

        gbox = new GroupBox();
        gbox.Header = "Header";
        gbox.Content = "AAA";

        this.Content = gbox;
    }
}
```

4. ControlExample

▌ ControlExample.xaml

```
< Grid >
    < Grid.RowDefinitions >
        < RowDefinition Height = "114*" />
        < RowDefinition Height = "201*" />
        < RowDefinition Height = "107*" />
    </ Grid.RowDefinitions >
    < Grid.ColumnDefinitions >
        < ColumnDefinition Width = "128*" />
        < ColumnDefinition Width = "145*" />
        < ColumnDefinition Width = "161*" />
        < ColumnDefinition Width = "179*" />
        < ColumnDefinition Width = "181*" />
    </ Grid.ColumnDefinitions >
    < TextBox Name = "textbox" Margin = "20" Grid.Row = "1" TextWrapping = "Wrap" Text
= "TextBox" />
    < Button Name = "button1" Click = "Button1_Click"  Margin = "20" Content = "Button"
Grid.Column = "1" Grid.Row = "1" />
    < ListBox Name = "listbox" Margin = "20" Grid.Column = "2" Grid.Row = "1" />
    < Button Name = "button2" Click = "Button2_Click" Margin = "20" Content = "Button"
Grid.Column = "3" Grid.Row = "1" />
    < ComboBox Name = "combobox" Margin = "20" Grid.Column = "4" Grid.Row = "1" Height
= "50" />

</ Grid >
```

▌ ControlExample.xaml.cs

```
public partial class Ex4_ControlExample1 : Window
{
    public Ex4_ControlExample1()
    {
        InitializeComponent();
    }
```

```
    private void Button1_Click(object sender, RoutedEventArgs e)
    {
        string s = textbox.Text;

        listbox.Items.Add(s);

        textbox.Clear();
    }

    private void Button2_Click(object sender, RoutedEventArgs e)
    {
        string s = listbox.SelectedItem.ToString();

        combobox.Items.Add(s);

        int idx = listbox.SelectedIndex;

        if (idx != -1)
            listbox.Items.RemoveAt(idx);
    }
}
```

5. VisualTree

VisualTree.xaml

```
< Canvas >
    < Button Click = "Button_Click" Content = "Button" Canvas.Left = "143" Canvas.Top =
"64" Width = "75" />
    < ComboBox Canvas.Left = "331" Canvas.Top = "72" Width = "120" />
    < TextBlock Canvas.Left = "553" TextWrapping = "Wrap" Text = "TextBlock" Canvas.Top
= "88" />
    < TreeView Height = "100" Canvas.Left = "140" Canvas.Top = "172" Width = "100" >
        < TreeViewItem Header = "Item1" />
        < TreeViewItem Header = "Item2" />
        < TreeViewItem Header = "Item3" >
            < TreeViewItem Header = "Child1" />
            < TreeViewItem Header = "Child2" />
            < TreeViewItem Header = "Child3" />
        </ TreeViewItem >
    </ TreeView >

</ Canvas >
```

VisualTree.xaml.cs

```
public partial class Ex6_VisualTree : Window
{
    public Ex6_VisualTree()
    {
        InitializeComponent();
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        ShowVisualTree svt = new ShowVisualTree();
        svt.Process(this);
        svt.Show();
    }
}
```

ShowVisualTree.xaml

```
<Grid>
        <TreeView Name="treeView"></TreeView>
</Grid>
```

ShowVisualTree.xaml.cs

```
public partial class ShowVisualTree : Window
{
    public ShowVisualTree()
    {
        InitializeComponent();
    }

    public void Process(DependencyObject root)
    {
        // Clear the tree.
        treeView.Items.Clear();

        // Start processing elements, begin at the root.
        ProcessElement(root, null);
    }

    private void ProcessElement(DependencyObject element, TreeViewItem previousItem)
    {
        // Create a TreeViewItem for the current element.
        TreeViewItem item = new TreeViewItem();
        item.Header = element.GetType().Name;

        item.IsExpanded = true;

        // Check whether this item should be added to the root of the tree
        //(if it's the first item), or nested under another item.
        if (previousItem == null)
        {
            treeView.Items.Add(item);
        }
        else
        {
            previousItem.Items.Add(item);
        }
```

```
        // Check if this element contains other elements.
        for (int i = 0; i < VisualTreeHelper.GetChildrenCount(element); i++)
        {
            // Process each contained element recursively.
            ProcessElement(VisualTreeHelper.GetChild(element, i), item);
        }
    }
}
```

6. CustomControl

▌ CustomControl.xaml

```
< Grid >
    < Button Content = "Button" HorizontalAlignment = "Left" Margin = "51,85,0,0"
VerticalAlignment = "Top" Width = "75" />
    < Button HorizontalAlignment = "Left" Margin = "202,86,0,0" VerticalAlignment =
"Top" Width = "75" >
        < StackPanel >
            < CheckBox Content = "동의합니다." ></ CheckBox >
            < ListBox >
                < ListBoxItem > Item1 </ ListBoxItem >
                < ListBoxItem > Item2 </ ListBoxItem >
                < ListBoxItem > Item3 </ ListBoxItem >
            </ ListBox >
        </ StackPanel >
    </ Button >

    < Button HorizontalAlignment = "Left" Margin = "352,82,0,0" VerticalAlignment =
"Top" Width = "75" >
        < Ellipse Fill = "Red" Width = "30" Height = "30" />
    </ Button >

    < Button Content = "Button4" Background = "blue" HorizontalAlignment = "Left"
Margin = "510,91,0,0" VerticalAlignment = "Top" Width = "75" >
        < Button.Template >
            < ControlTemplate TargetType = "{x:Type Button}" >
                < Grid >
                    < Ellipse Fill = "Red" Width = "30" Height = "30" />
                    < Label Content = "{TemplateBinding Content}" HorizontalAlignment =
"Center" VerticalAlignment = "Center" ></ Label >
                </ Grid >
            </ ControlTemplate >
        </ Button.Template >
    </ Button >
    < Button Content = "Button" HorizontalAlignment = "Left" Margin = "620,83,0,0"
VerticalAlignment = "Top" Width = "75" Click = "Button_Click" />
</ Grid >
```

SECTION 7

Command & Menu

1. Button Enable & Disable

▌ 기본 개념

아래 코드는 TextBox 의 입력된 값이 없을 때 버튼을 Disable 하는 코드입니다.

```
<StackPanel>
        <TextBox TextChanged="txtBox_TextChanged"       x:Name="txtBox" Margin="10"

                        FontSize="30"/>
        <Button  Click="button_Click" x:Name="button"   Margin="10" FontSize="30"

                                Content="button1"/>
</StackPanel>
```

```
private void txtBox_TextChanged(object sender, TextChangedEventArgs e)
{
        button.IsEnabled = !string.IsNullOrEmpty(txtBox.Text);
}
```

2. ICmmand 인터페이스

▌ 기본 개념

버튼을 눌렀을 때 코드를 연결하는 방법은 2가지가 있습니다.

① Click="메소드" 연결

② Command="ICommand로부터 파생된 타입의 객체" 연결

ICommand 로 부터 파생된 객체를 사용하면 UI 의 enable/disable 기능을 편리하게 작성할수 있습니다.

```
<StackPanel>
        <TextBox x:Name="txtBox" Margin="10" FontSize="30"/>
        <Button  Command="local:MyCommand3.cmdAction"
                                x:Name="button1" Margin="10" FontSize="30"
Content="button1"/>
</StackPanel>
```

```
public class ActionCommand3 : ICommand
{
    public event EventHandler CanExecuteChanged
    {
        add
        {
            CommandManager.RequerySuggested += value;
        }
        remove
        {
            CommandManager.RequerySuggested -= value;
        }
    }
    public TextBox txtBox = null;

    public bool CanExecute(object parameter)
    {
        Console.WriteLine("CanExecute");

        if (txtBox == null)
            txtBox = ((Command4Window)Application.Current.MainWindow).txtBox;

        return !string.IsNullOrEmpty(txtBox.Text);
```

```
    }
    public void Execute(object parameter)
    {
        MessageBox.Show(txtBox?.Text);
    }
}
static class MyCommand3
{
    public static ActionCommand3 cmdAction = new ActionCommand3();
}
```

3. RoutedCommand

▌ 기본 개념

WPF는 ICommand 로 부터 파생된 타입인 RoutedCommand 클래스를 제공합니다.

```
<StackPanel>
        <TextBox x:Name="txtBox" Margin="10" FontSize="30"/>
        <Button Command="local:MyCommand.cmdAction"
                                x:Name="button" Margin="10" FontSize="30"
Content="button1"/>
</StackPanel>
```

```
public static class MyCommand
{
    public static RoutedCommand cmdAction = new RoutedCommand();
}
public partial class RoutedCommand1 : Window
{
    public RoutedCommand1()
    {
        InitializeComponent();

        CommandBinding cmd = new CommandBinding(MyCommand.cmdAction);

        cmd.Executed += Cmd_Executed;
        cmd.CanExecute += Cmd_CanExecute;

        this.CommandBindings.Add(cmd);

        InputBinding ibFind = new InputBinding(MyCommand.cmdAction,
                                            new KeyGesture(Key.R,
ModifierKeys.Control));
        this.InputBindings.Add(ibFind);
    }

    private void Cmd_CanExecute(object sender, CanExecuteRoutedEventArgs e)
    {
        e.CanExecute = !string.IsNullOrEmpty(txtBox.Text);
    }
```

```
    private void Cmd_Executed(object sender, ExecutedRoutedEventArgs e)
    {
        MessageBox.Show(txtBox.Text);
    }
}
```

Using Xaml

아래 코드는 Xaml 을 사용해서 RoutedCommand 를 사용하는 코드 입니다.

```
<Window.Resources>
        <RoutedCommand x:Key="cmdAction" />
</Window.Resources>

<Window.CommandBindings>
        <CommandBinding Command="{StaticResource ResourceKey=cmdAction}"
                                                Executed="Cmd_Executed"
CanExecute="Cmd_CanExecute"/>
</Window.CommandBindings>

<Window.InputBindings>
        <KeyBinding Key="R" Modifiers="Control"
                                                Command="{StaticResource
ResourceKey=cmdAction}"/>
</Window.InputBindings>

<StackPanel>
        <TextBox x:Name="txtBox" Margin="10" FontSize="30"/>
    <Button  Command="{StaticResource ResourceKey=cmdAction}"
                        x:Name="button" Margin="10" FontSize="30" Content="button1"/>
</StackPanel>
```

4. Menu1

Menu1.xaml

```
< Grid >
    < Menu HorizontalAlignment = "Left" Height = "100" Margin = "163,71,0,0"
VerticalAlignment = "Top" Width = "387" >
        < MenuItem Header = "File" >
            < MenuItem Header = "Open" Click = "File_Open" ></ MenuItem >
            < MenuItem Header = "Open" Command = "ApplicationCommands.Open" ></
MenuItem >
        </ MenuItem >
        < MenuItem Header = "File" ></ MenuItem >
    </ Menu >
</ Grid >
```

Menu1.xaml.cs

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

    private void File_Open(object sender, RoutedEventArgs e)
    {
        MessageBox.Show("New");
    }

    private void MenuItem_Click(object sender, RoutedEventArgs e)
    {
        MessageBox.Show("Open");
    }
}
```

5. Menu2

Menu2.xaml

```
< Window.CommandBindings >
    < CommandBinding Command = "ApplicationCommands.New" Executed = "NewCommand" />
</ Window.CommandBindings >
< StackPanel >
    < Menu >
        < MenuItem Header = "File" >
            < MenuItem Command = "New" ></ MenuItem >
        </ MenuItem >
    </ Menu >
    < Button Margin = "5" Padding = "5" Command = "ApplicationCommands.New" ToolTip =
"{x:Static ApplicationCommands.New}" > New </ Button >
    < Button Margin = "5" Padding = "5" Visibility = "Hidden" Command =
"ApplicationCommands.Open" > Open(unwired) </ Button >
    < Button Margin = "5" Padding = "5" Visibility = "Hidden" Click =
"cmdDoCommand_Click" > DoCommand </ Button >
</ StackPanel >
```

Menu2.xaml.cs

```
public partial class Step1 : Window
{
    public Step1()
    {
        //ApplicationCommands.New.Text = "Completely New";

        InitializeComponent();

        //CommandBinding bindingNew = new CommandBinding(ApplicationCommands.New);
        //bindingNew.Executed += NewCommand;

        //this.CommandBindings.Add(bindingNew);
    }

    private void NewCommand(object sender, ExecutedRoutedEventArgs e)
```

```
    {
        MessageBox.Show("New");
    }

    private void cmdDoCommand_Click(object sender, RoutedEventArgs e)
    {
        this.CommandBindings[0].Command.Execute(null);
    }
}
```

SECTION 8

Resource & Style

1. MarkupExtension

▌ CustomMarkup.xaml.cs

```
public class MyFont : MarkupExtension
{
    private string key;
    public MyFont(string k)
    {
        key = k;
    }
    public override object ProvideValue(IServiceProvider serviceProvider)
    {
        switch(key)
        {
            case "Size": return (double)18;
            case "Weight": return FontWeights.Bold;
        }
        return null;
    }
}
```

▌ CustomMarkup.xaml

```
<Button Name="button1" FontSize="{local:MyFont Size}" FontWeight="{local:MyFont Weight}"> Hello</Button>
```

2. Assembly Resource

▌ AssemblyResource.xaml

```
<StackPanel>
    <Button Click="cmdPlay_Click" Margin="5" Padding="5">Play</Button>
    <Image Name="img" Margin="5" Source="Images/first.jpg"></Image>
    <MediaElement Name="Sound" Source="Sounds/start.wav"
                                        LoadedBehavior="Manual"></MediaElement>
</StackPanel>
```

▌ AssemblyResource.xaml.cs

```
private void cmdPlay_Click(object sender, RoutedEventArgs e)
{
    img.Source = new BitmapImage(new Uri("images/second.jpg", UriKind.Relative));
    Sound.Stop();
    Sound.Play();
}
```

3. Object Resource

ObjectResource.xaml

```
<Window.Resources>
    <ImageBrush x:Key="TileBrush" TileMode="Tile" ViewportUnits="Absolute" Viewport="0
0 32 32" ImageSource="image/first.png" Opacity="0.3"></ImageBrush>
</Window.Resources>

<StackPanel Margin="5">
    <Button Padding="5" Margin="5" FontWeight="Bold" FontSize="14">Button1</Button>
    <Button Background="yellow" Padding="5" Margin="5" FontWeight="Bold"
FontSize="14">Button2</Button>
    <Button Padding="5" Margin="5" FontWeight="Bold" FontSize="14">
        <Button.Background>
            <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                <GradientStop Color="Black" Offset="0"/>
                <GradientStop Color="White" Offset="1"/>
            </LinearGradientBrush>
        </Button.Background> Button1
    </Button>
    <Button Background="{StaticResource TileBrush}" Padding="5" FontWeight="Bold"
FontSize="14" Margin="5">A Tiled Button</Button>
    <Button Background="{DynamicResource TileBrush}" Padding="5" FontWeight="Bold"
FontSize="14" Margin="5">A Tiled Button</Button>
    <Button Background="{DynamicResource RedBrush}" Padding="5" FontWeight="Bold"
FontSize="14" Margin="5">A Tiled Button</Button>
    <Button Padding="5" Margin="5" FontWeight="Bold" FontSize="14"
Click="change_brush">A Normal Button</Button>
</StackPanel>
```

ObjectResource.xaml.cs

```
private void change_brush(object sender, RoutedEventArgs e)
{
    //ImageBrush brush = (ImageBrush)this.Resources["TileBrush"];
    //brush.Viewport = new Rect(0, 0, 5, 5);

    this.Resources["TileBrush"] = new SolidColorBrush(Colors.LightBlue);
```

```
}
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    this.Resources["RedBrush"] = new SolidColorBrush(Colors.Red);
}
```

4. Style

▌ Style.xaml

```
<Window.Resources>
    <!-- Step 1. 각 속성을 별도로 지정 -->
    <FontFamily x:Key="ButtonFontFamily">Times New Roman</FontFamily>
    <s:Double x:Key="ButtonFontSize">18</s:Double>
    <FontWeight x:Key="ButtonFontWeight">Bold</FontWeight>

    <!-- Step 2. 키 값을 가지는 Style 만들기 -->
    <Style x:Key="BigFontButtonStyle">
        <Setter Property = "Control.FontFamily" Value="Times New Roman" />
        <Setter Property = "Control.FontSize" Value="18" />
        <Setter Property = "Control.FontWeight" Value="Bold" />
    </Style>

    <!-- Step 3. Style 상속 -->
    <Style x:Key="EmphasizedBigFontButtonStyle" BasedOn="{StaticResource
BigFontButtonStyle}">
        <Setter Property = "Control.Foreground" Value="White" />
        <Setter Property = "Control.Background" Value="DarkBlue" />
    </Style>

    <!-- Step 4. Automatic Style.컨트롤에 Style -->
    <Style TargetType = "Button" >
        < Setter Property= "FontFamily" Value= "Times New Roman" />
        < Setter Property= "FontSize" Value= "18" />
        < Setter Property= "FontWeight" Value= "Bold" />
    </ Style >

    < !--Step 5. Style 에 이벤트 넣기 -->
    <Style x:Key= "MouseOverHighlightStyle" >
        < Setter Property= "TextBlock.Padding" Value= "5" />
        < EventSetter Event= "FrameworkElement.MouseEnter" Handler=
"element_MouseEnter" />
        < EventSetter Event= "FrameworkElement.MouseLeave" Handler=
"element_MouseLeave" />
    </ Style >

</ Window.Resources >
```

```
< StackPanel Margin= "5" >
    < Button Padding= "5" Margin= "5" > A Normal Button</Button>
    <Button Padding = "5" Margin= "5" FontFamily= "Times New Roman" FontWeight= "Bold"
FontSize= "18" > A Customized Button</Button>
    <Button Padding = "5" Margin= "5" FontFamily= "{StaticResource ButtonFontFamily}"
FontWeight= "{StaticResource ButtonFontWeight}"  FontSize= "{StaticResource
ButtonFontSize}" > A Customized Button</Button>
    <Button Padding = "5" Margin= "5" Style= "{StaticResource BigFontButtonStyle}" >
Customized Button</Button>
    <Button Padding = "5" Margin= "5" Style= "{StaticResource
EmphasizedBigFontButtonStyle}" > Customized Button</Button>

    <TextBlock Padding = "5" MouseEnter= "element_MouseEnter" MouseLeave=
"element_MouseLeave" > TextBlock1 </ TextBlock >
    < TextBlock Padding= "5" > TextBlock2 </ TextBlock >
    < TextBlock Style= "{StaticResource MouseOverHighlightStyle}" > TextBlock3 </
TextBlock >

</ StackPanel >
```

Style.xaml.cs

```
private void element_MouseEnter(object sender, MouseEventArgs e)
{
    ((TextBlock)sender).Background = new SolidColorBrush(Colors.LightGoldenrodYellow);
}
private void element_MouseLeave(object sender, MouseEventArgs e)
{
    ((TextBlock)sender).Background = null;
}
```

5. Trigger

Trigger.xaml

```
<Window.Resources>
    <Style x:Key="BigFontButton">
        <Style.Setters>
            <Setter Property = "Control.FontFamily" Value="Times New Roman" />
            <Setter Property = "Control.FontSize" Value="18" />
        </Style.Setters>

        <Style.Triggers>
            <Trigger Property = "Control.IsMouseOver" Value="True">
                <Setter Property = "Control.Foreground" Value="Red" />
            </Trigger>
            <Trigger Property = "Control.IsFocused" Value="True">
                <Setter Property = "Control.Foreground" Value="Yellow" />
            </Trigger>

            <Trigger Property = "Button.IsPressed" Value="True">
                <Setter Property = "Control.Foreground" Value="Blue" />
            </Trigger>

            <EventTrigger RoutedEvent = "Mouse.MouseEnter" >
                < EventTrigger.Actions >
                    < BeginStoryboard >
                        < Storyboard >
                            < DoubleAnimation Duration="0:0:0.2"
Storyboard.TargetProperty="FontSize" To="22"/>
                        </Storyboard>
                    </BeginStoryboard>
                </EventTrigger.Actions>
            </EventTrigger>

            <EventTrigger RoutedEvent = "Mouse.MouseLeave" >
                < EventTrigger.Actions >
                    < BeginStoryboard >
                        < Storyboard >
                            < DoubleAnimation  Duration="0:0:1"
Storyboard.TargetProperty="FontSize"  />
                        </Storyboard>
```

```
                    </BeginStoryboard>
                </EventTrigger.Actions>
            </EventTrigger>
        </Style.Triggers>
    </Style>
</Window.Resources>

<StackPanel Margin = "5" >
    < Button Padding="5" Margin="5">Normal Button</Button>
        <Button Padding = "5" Margin= "5"  Style= "{StaticResource BigFontButton}" > A
Customized Button</Button>
</StackPanel>
```

SECTION 9

Drawing & Animation

1. Sketch

▌ Sketch.xaml

```
<Canvas  Name="paintSurface" MouseDown="Canvas_MouseDown_1"
MouseMove="Canvas_MouseMove_1" >
    <Canvas.Background>
        <SolidColorBrush Color="White" Opacity="0"/>
    </Canvas.Background>
</Canvas>
```

▌ Sketch.xaml.cs

```
public partial class Ex1_Sketch : Window
{
    public Ex1_Sketch()
    {
        InitializeComponent();
    }
    Point currentPoint = new Point();

    private void Canvas_MouseDown_1(object sender,
System.Windows.Input.MouseButtonEventArgs e)
    {
        if (e.LeftButton == MouseButtonState.Pressed)
            currentPoint = e.GetPosition(this);
        else if (e.RightButton == MouseButtonState.Pressed)
        {
            paintSurface.Children.RemoveAt(paintSurface.Children.Count);
            paintSurface.InvalidateVisual();
        }
    }

    private void Canvas_MouseMove_1(object sender, System.Windows.Input.MouseEventArgs
e)
    {
        if (e.LeftButton == MouseButtonState.Pressed)
        {
```

```
            Line line = new Line();

            line.Stroke = SystemColors.WindowFrameBrush;
            line.X1 = currentPoint.X;
            line.Y1 = currentPoint.Y;
            line.X2 = e.GetPosition(this).X;
            line.Y2 = e.GetPosition(this).Y;

            currentPoint = e.GetPosition(this);

            paintSurface.Children.Add(line);
        }

    }
}
```

2. Draw Image

ImageDraw.xaml.cs

```
public partial class MainWindow : Window
{
    private Image[,] img = new Image[5, 5];

    private double bx, by;

    public MainWindow()
    {
        InitializeComponent();

        BitmapImage bitmap = new BitmapImage();
        bitmap.BeginInit();
        bitmap.UriSource = new Uri("B:\\totoro.jpg");
        bitmap.EndInit();

        this.Width = bitmap.Width;
        this.Height = bitmap.Height + 30;


        Console.WriteLine(bitmap.Width);
        Console.WriteLine(bitmap.Height);

        bx = ((bitmap.Width) / 5);
        by = ((bitmap.Height) / 5);

        // bx = bx - 1;
        // bx = by - 1;
        Console.WriteLine(bx);
        Console.WriteLine(by);


        for (int y = 0; y < img.GetLength(1); y++)
        {
            for (int x = 0; x < img.GetLength(0); x++)
            {
                //Console.WriteLine("{0}, {1}", (x + 1) * bx, (y + 1) * by);
                CroppedBitmap cb = new CroppedBitmap((BitmapSource)bitmap, new
```

```
Int32Rect((int)(x * bx), (int)(y * by), (int)bx - 2, (int)by - 2));

                img[y, x] = new Image();
                img[y, x].Source = cb;


                Grid.SetRow(img[y, x], y);
                Grid.SetColumn(img[y, x], x);

                grid.Children.Add(img[y, x]);
            }
        }
    }
}
```

3. Code Animation

CodeAnimation.xaml

```
<Canvas>
        <Ellipse Name="RedBall" Fill="Red" Width="100" Height="100" Canvas.Left="132"
Canvas.Top="62"/>
        <Button Name="button1" Click="Button1_Click" Content="Button" Canvas.Left="144"
Canvas.Top="200" Width="75"/>
        <Button Name="button2" Click="Button2_Click" Content="Button" Canvas.Left="144"
Canvas.Top="241" Width="75"/>
        <Button Name="button3" Click="Button3_Click" Content="Button" Canvas.Left="148"
Canvas.Top="280" Width="75"/>
        <Button Name="button4" Click="Button4_Click" Content="Button" Canvas.Left="148"
Canvas.Top="323" Width="75"/>
        <Button Name="button5" Click="Button5_Click" Content="Button" Canvas.Left="148"
Canvas.Top="353" Width="75"/>
        <Button Name="button6" Click="Button6_Click" Content="Button" Canvas.Left="248"
Canvas.Top="353" Width="75"/>
        <Button Name="button7" Click="Button5_Click" Content="Button" Canvas.Left="148"
Canvas.Top="400" Width="75"/>
        <Button Name="button8" Click="Button6_Click" Content="Button" Canvas.Left="248"
Canvas.Top="400" Width="75"/>
    </Canvas>
```

CodeAnimation.xaml.cs

```
public partial class Ex1_CodeAnimation : Window
{
    public Ex1_CodeAnimation()
    {
        InitializeComponent();
    }

    private void Animation_Completed(object sender, EventArgs e)
    {
        Console.WriteLine("Animation Completed");
    }
```

```
private void Button1_Click(object sender, RoutedEventArgs e)
{
    // Width 속성 변경
    DoubleAnimation anim1 = new DoubleAnimation();
    anim1.From = RedBall.Width;
    anim1.To = RedBall.Width + 100;
    anim1.Duration = new TimeSpan(0, 0, 3);

    //   anim1.Completed += Animation_Completed;

    RedBall.BeginAnimation(Ellipse.WidthProperty, anim1);
}

private void Button2_Click(object sender, RoutedEventArgs e)
{
    DoubleAnimation anim1 = new DoubleAnimation();
    anim1.Duration = TimeSpan.FromSeconds(3);
    RedBall.BeginAnimation(Ellipse.WidthProperty, anim1);
}

private void Button3_Click(object sender, RoutedEventArgs e)
{
    DoubleAnimation anim1 = new DoubleAnimation();
    anim1.From = (double)RedBall.GetValue(Canvas.LeftProperty);
    anim1.To = (double)RedBall.GetValue(Canvas.LeftProperty) + 100;
    anim1.Duration = TimeSpan.FromSeconds(3);

    RedBall.BeginAnimation(Canvas.LeftProperty, anim1);
}
private void Button4_Click(object sender, RoutedEventArgs e)
{
    DoubleAnimation anim1 = new DoubleAnimation();
    anim1.By = -30;
    anim1.Duration = TimeSpan.FromSeconds(1);

    RedBall.BeginAnimation(Canvas.LeftProperty, anim1);
}

// DispatcherTimer 사용
private DispatcherTimer MyTimer = new DispatcherTimer();

private void Button5_Click(object sender, RoutedEventArgs e)
{
    MyTimer.Interval = new TimeSpan(30); // 30 nano
```

```
        MyTimer.Tick += UpdateRedBall;
        MyTimer.Start();
    }

    private double xPos = 0;

    void UpdateRedBall(object sender, EventArgs e)
    {
        Canvas.SetLeft(RedBall, xPos);
        if (xPos > this.Width)
            xPos = 0;
        else
            xPos += 0.1;
        Thread.Sleep(1);
    }
    private void Button6_Click(object sender, RoutedEventArgs e)
    {
        MyTimer.Stop();
    }

    private void Button7_Click(object sender, RoutedEventArgs e)
    {
        CompositionTarget.Rendering += new EventHandler(CompositionTarget_Rendering);
    }

    private void Button8_Click(object sender, RoutedEventArgs e)
    {
        CompositionTarget.Rendering -= new EventHandler(CompositionTarget_Rendering);
    }
    void CompositionTarget_Rendering(object sender, EventArgs e)
    {
        Canvas.SetLeft(RedBall, xPos);
        if (xPos > this.Width)
            xPos = 0;
        else
            xPos += 0.5;
    }
}
```

4. Xaml Animation

▌ XamlAnimation.xaml

```
<Canvas>
        <Ellipse Name="RedBall" Fill="Red" Width="100" Height="100" Canvas.Left="132"
Canvas.Top="62"/>
        <Button  Name="button1" Click="Button1_Click" Content="Button"
Canvas.Left="144" Canvas.Top="200" Width="75">
        <Button.Triggers>
            <EventTrigger RoutedEvent="Button.Click">
                    <BeginStoryboard Name="MyStoryBoard">
                        <Storyboard>
                            <!-- Width 프라퍼티는 직접, Canvas.Left 는 ()로 묶어야
한다. -->
                            <DoubleAnimation Storyboard.TargetName ="RedBall"
Storyboard.TargetProperty="(Canvas.Left)"
                                From="0" To="300"  Duration="0:0:15"></DoubleAnimation>
                        </Storyboard>
                    </BeginStoryboard>
                </EventTrigger>
            </Button.Triggers>
        </Button>

        <Button  Name="button2" Content="Button" Canvas.Left="144" Canvas.Top="250"
Width="75">
            <Button.Triggers>
                <EventTrigger RoutedEvent="Button.Click">
                    <StopStoryboard  BeginStoryboardName="MyStoryBoard"/>
                </EventTrigger>
            </Button.Triggers>
        </Button>
    </Canvas>
```

SECTION 10

Binding

Model – View - ViewModel

1. Element Binding

ElementBinding.xaml

```
<StackPanel>

    <Label Name = "label1" Content="Label" FontSize="{Binding ElementName=slider3,
Path=Value, Mode=TwoWay}"/>

    <Slider Name = "slider1" Maximum="100" Minimum="30" Margin="5"
ValueChanged="Slider1_ValueChanged"/>
    <Button Name = "button1" Click="Button1_Click" Margin="5">button1</Button>
    <Slider Name = "slider2" Maximum="100" Minimum="30" Margin="5"/>
    <Button Name = "button2" Click="Button2_Click" Margin="5">button2</Button>
    <Slider Name = "slider3" Maximum="100" Minimum="30" Margin="5"/>
    <Slider Name = "slider4" Maximum="100" Minimum="30" Margin="5"
            Value="{Binding ElementName=label1, Path=FontSize,
UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}"/>

    <TextBox Name = "textbox1" Height="23" TextWrapping="Wrap" Text="TextBox"/>
</StackPanel>
```

ElementBinding.xaml.cs

```
private void Slider1_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double>
e)
{
    label1.FontSize = slider1.Value;
}
private void Button1_Click(object sender, RoutedEventArgs e)
{
    Binding binding = new Binding();
    binding.Source = slider2;
    binding.Path = new PropertyPath("Value");
    binding.Mode = BindingMode.TwoWay;
    label1.SetBinding(TextBlock.FontSizeProperty, binding);

}
private void Button2_Click(object sender, RoutedEventArgs e)
```

```
{
    BindingOperations.ClearAllBindings(label1);
}
```

2. Data Binding 1

App.xaml.cs

```
public class People : INotifyPropertyChanged
{
    //public string Name { get; set; } = "UNKNOWN";
    public string Address { get; set; } = "SEOUL";

    private string name;
    public string Name
    {
        get { return name; }
        set
        {
            name = value;
            OnPropertyChanged(new PropertyChangedEventArgs("Name"));
        }
    }

    public event PropertyChangedEventHandler PropertyChanged;

    public void OnPropertyChanged(PropertyChangedEventArgs e)
    {
        if (PropertyChanged != null)
            PropertyChanged(this, e);
    }
}
```

DataBinding1.xaml

```
<Grid Name = "mainGrid" >
    < Button Content="Button" HorizontalAlignment="Left" Margin="107,70,0,0"
VerticalAlignment="Top" Width="75" Click="Button_Click"/>
    <TextBox Text = "{Binding Path=Name}" HorizontalAlignment="Left"
Margin="107,126,0,0" VerticalAlignment="Top" Width="201"/>
    <TextBox Text = "{Binding Path=Address}" HorizontalAlignment="Left"
Margin="107,175,0,0" VerticalAlignment="Top" Width="201"/>
```

```
</Grid>
```

DataBinding1.xaml.cs

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    People p = new People();
    p.Name = "kim";
    p.Address = "seoul";
    mainGrid.DataContext = p;
}
```

3. DataBinding2

DataBindng2.xaml

```
<Grid>
    <ListBox Name = "listPeople" DisplayMemberPath="Name" HorizontalAlignment="Left"
Height="196" Margin="77,33,0,0" VerticalAlignment="Top" Width="442"/>
    <Button Content = "Button" HorizontalAlignment="Left" Margin="83,263,0,0"
VerticalAlignment="Top" Width="75" Click="Button_Click"/>
    <Button Content = "Button" HorizontalAlignment="Left" Margin="229,283,0,0"
VerticalAlignment="Top" Width="75" Click="Button_Click_1"/>
    <Button Content = "Button" HorizontalAlignment="Left" Margin="368,303,0,0"
VerticalAlignment="Top" Width="75" Click="Button_Click_2"/>
</Grid>
```

DataBindig2.xaml.cs

```
public DataBinding2()
{
    InitializeComponent();

    st.Add(new People { Name = "kim1", Address = "seoul1" });
    st.Add(new People { Name = "kim2", Address = "seoul2" });
    st.Add(new People { Name = "kim3", Address = "seoul3" });
    st.Add(new People { Name = "kim4", Address = "seoul4" });
}

private void Button_Click(object sender, RoutedEventArgs e)
{
    listPeople.ItemsSource = st;
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
    listPeople.ItemsSource = st;
}

private void Button_Click_1(object sender, RoutedEventArgs e)
{
```

```
    st.Add(new People { Name = "kim3", Address = "seoul3" });
}

private void Button_Click_2(object sender, RoutedEventArgs e)
{
    st[0].Name = "XXXX";
}
```

SECTION 11

Examples

1. PuzzleGame

▌ PuzzleGame.xaml

```
< Grid >
    < Grid.RowDefinitions >
        < RowDefinition Height = "50" />
        < RowDefinition Height = "50" />
        < RowDefinition />
    </ Grid.RowDefinitions >
    < TextBlock HorizontalAlignment = "Center" VerticalAlignment = "Center" FontSize =
"30" > 숫자 퍼즐 </ TextBlock >
    < Button Name = "btnShuffle" Grid.Row = "1" Width = "100" Height = "30" Click =
"btnShuffle_Click" > Shuffle </ Button >
    < Grid Margin = "0,1,0,0" Name = "Board" Grid.Row = "2" >
        < Line Stroke = "LightSteelBlue" Grid.Row = "0" VerticalAlignment = "Bottom" X2
= "300" ></ Line >
        < Line Stroke = "LightSteelBlue" Grid.Row = "1" VerticalAlignment = "Bottom" X2
= "300" ></ Line >
    </ Grid >
</ Grid >
```

PuzzleGame.xaml.cs

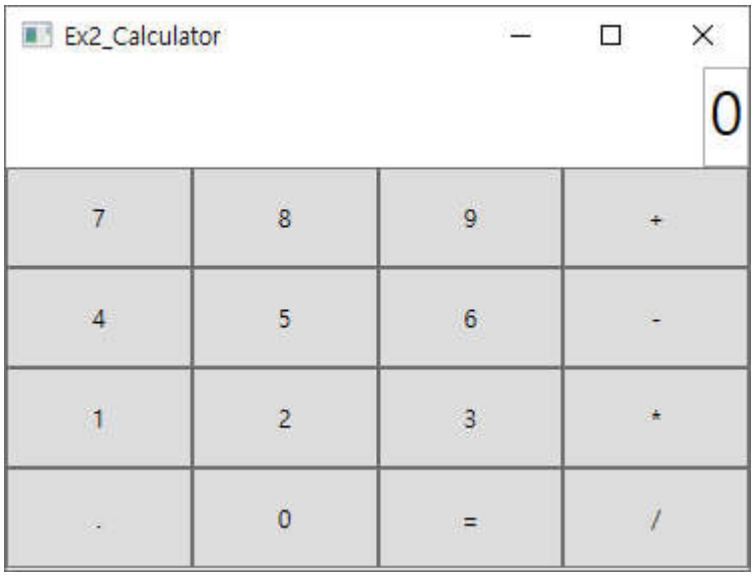
```
public partial class Ex1_PUZZLE : Window
{
    private const int count = 4;
    private const int EMPTY = count * count - 1;
    private int[,] placement = new int[count, count];
    private bool isShuffled = false;

    public Ex1_PUZZLE()
    {
        InitializeComponent();
        BoardSet();     // Board setting
        InitBoard();    // 초기화면처럼 순서대로 배치
        DrawBoard();    // placement[,] 배열에 따라 버튼 그리기
    }

    // Grid Board를 4x4로 분할한다
    private void BoardSet()
    {
        for (int i = 0; i < count; i++)
        {
            RowDefinition row = new RowDefinition();
            Board.RowDefinitions.Add(row);
        }
        for (int i = 0; i < count; i++)
        {
            ColumnDefinition col = new ColumnDefinition();
            Board.ColumnDefinitions.Add(col);
        }
    }

    // 1~15까지의 숫자를 순서대로 배치한다
    private void InitBoard()
    {
        for (int i = 0; i < count; i++)
            for (int j = 0; j < count; j++)
                placement[i, j] = i * count + j;
    }

    // placement[,] 배열의 내용을 읽어서 Board에 그려준다
    private void DrawBoard()
    {
        for (int row = 0; row < count; row++)
```

```
        {
            for (int col = 0; col < count; col++)
            {
                if (placement[row, col] != EMPTY)
                {
                    Button btn = new Button();
                    btn.Margin = new Thickness(1);
                    btn.FontSize = 15;
                    btn.FontWeight = FontWeights.Bold;
                    btn.Content = placement[row, col].ToString();
                    btn.Name = "btn" + row.ToString() + col.ToString();
                    btn.Click += btn_Click;
                    Board.Children.Add(btn);
                    Grid.SetRow(btn, row);
                    Grid.SetColumn(btn, col);
                }
            }
        }
    }
    // 겹치지 않는 0~15까지의 랜덤 생성
    private void btnShuffle_Click(object sender, RoutedEventArgs e)
    {
        isShuffled = true;
        Board.Children.Clear();
        RandomPlacement();
        DrawBoard();
    }

    // placement[] 배열에 서로 겹치지 않는 0~15까지의 숫자를 할당한다
    private void RandomPlacement()
    {
        int[] flag = new int[count * count];   // 0으로 초기화된다
        Random r = new Random();

        for (int row = 0; row < count; row++)
        {
            for (int col = 0; col < count; col++)
            {
                int num = r.Next(count * count);
                while (flag[num] != 0)
                    num = r.Next(count * count);
                flag[num] = 1;
                placement[row, col] = num;
            }
```

```
        }
    }
    private void btn_Click(object sender, EventArgs e)
    {
        if (isShuffled == false)
        {
            MessageBox.Show("Press Shuffle Button First, before you Start the Game!");
            return;
        }

        Button btn = sender as Button;
        int eRow = -1, eCol = -1;   // Empty 좌표

        int row = Convert.ToInt32(btn.Name.Remove(0, 3)) / 10;
        int col = Convert.ToInt32(btn.Name.Remove(0, 3)) % 10;

        // 이웃한 공간을 검색
        if (row - 1 >= 0 && placement[row - 1, col] == 0)    // North
        {
            eRow = row - 1; eCol = col;
        }

        else if (row + 1 <= 3 && placement[row + 1, col] == 0) // South
        {
            eRow = row + 1; eCol = col;
        }

        else if (col + 1 <= 3 && placement[row, col + 1] == 0)   // East
        {
            eRow = row; eCol = col + 1;
        }
        else if (col - 1 >= 0 && placement[row, col - 1] == 0)    // West
        { eRow = row; eCol = col - 1; }

        // 이웃 공간에 빈칸이 없다. 끝
        if (eRow == -1) return;

        // 이웃 공간에 빈칸이 있다 -> 클릭된 버튼을 빈칸으로 이동
        // 이 버튼의 좌표를 (x, y)에서 (emptyX, emptyY)로 이동
        btn.Name = "btn" + eRow.ToString() + eCol.ToString();
        placement[eRow, eCol] = placement[row, col];
        placement[row, col] = 0;

        Board.Children.Clear();
```

```
            DrawBoard();

            checkComplete();
        }

        private void checkComplete()
        {
            if (isShuffled == false)
                return;

            for (int i = 0; i < count; i++)
                for (int j = 0; j < count; j++)
                    if (placement[i, j] != i * count + j + 1)
                    {
                        if (i == count - 1 && j == count - 1)
                            MessageBox.Show("Congraturation!! You Completed!");
                        return;
                    }
        }
    }
```

2. Calculator



Calculator.xaml

```
< StackPanel >
    < TextBox Height = "50" Name = "txtResult" FontSize = "30" HorizontalAlignment =
"Right" > 0 </ TextBox >
    < Grid Height = "200" >
        < Grid.ColumnDefinitions >
            < ColumnDefinition Width = "25*" />
            < ColumnDefinition Width = "25*" />
            < ColumnDefinition Width = "25*" />
            < ColumnDefinition Width = "25*" />
        </ Grid.ColumnDefinitions >
        < Grid.RowDefinitions >
            < RowDefinition Height = "25*" />
            < RowDefinition Height = "25*" />
            < RowDefinition Height = "25*" />
            < RowDefinition Height = "25*" />
        </ Grid.RowDefinitions >
        < Button Content = "7" Grid.Column = "0" Grid.Row = "0" Click = "num_click" />
        < Button Content = "4" Grid.Column = "0" Grid.Row = "1" Click = "num_click" />
        < Button Content = "1" Grid.Column = "0" Grid.Row = "2" Click = "num_click" />
        < Button Content = "." Grid.Column = "0" Grid.Row = "3" Click = "dot_click" />
```

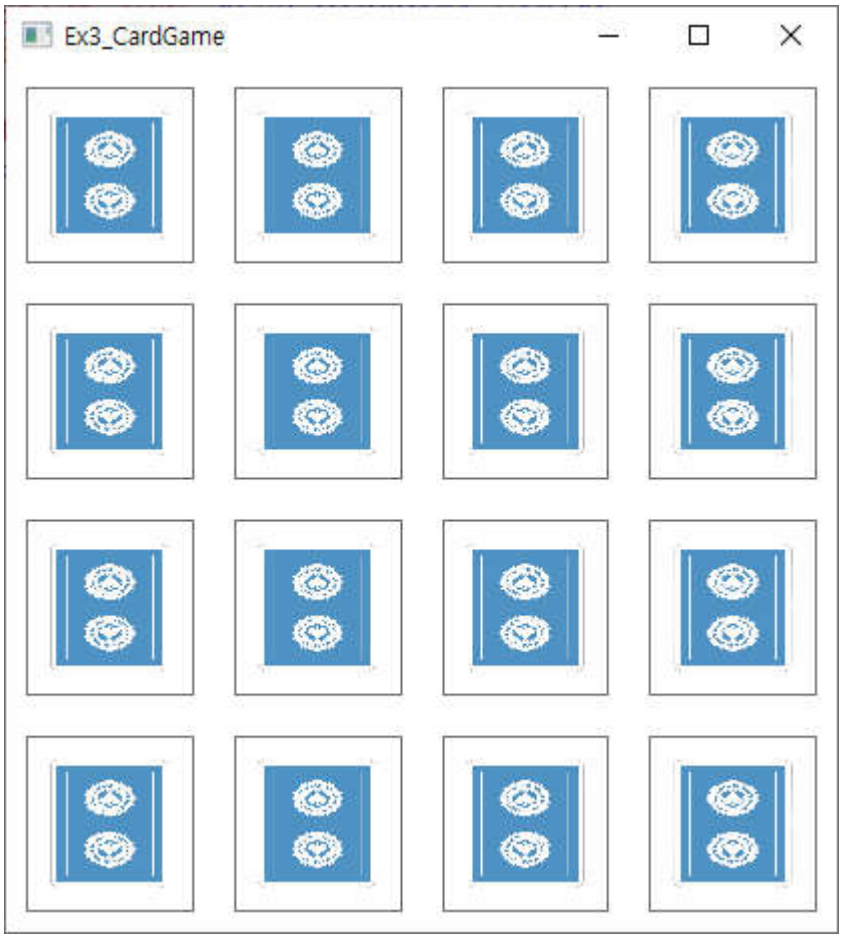
```
        < Button Content = "8" Grid.Column = "1" Grid.Row = "0" Click = "num_click" />
        < Button Content = "5" Grid.Column = "1" Grid.Row = "1" Click = "num_click" />
        < Button Content = "2" Grid.Column = "1" Grid.Row = "2" Click = "num_click" />
        < Button Content = "0" Grid.Column = "1" Grid.Row = "3" Click = "num_click" />
        < Button Content = "9" Grid.Column = "2" Grid.Row = "0" Click = "num_click" />
        < Button Content = "6" Grid.Column = "2" Grid.Row = "1" Click = "num_click" />
        < Button Content = "3" Grid.Column = "2" Grid.Row = "2" Click = "num_click" />
        < Button Content = "=" Grid.Column = "2" Grid.Row = "3" Click ="equal_click" />
        < Button Content = "+" Grid.Column = "3" Grid.Row = "0" Click = "op_click" />
        < Button Content = "-" Grid.Column = "3" Grid.Row = "1" Click = "op_click" />
        < Button Content = "*" Grid.Column = "3" Grid.Row = "2" Click = "op_click" />
        < Button Content = "/" Grid.Column = "3" Grid.Row = "3" Click = "op_click" />
    </ Grid >
</ StackPanel >
```

Calculator.xaml.cs

```
public partial class Ex2_Calculator : Window
{
    public Ex2_Calculator()
    {
        InitializeComponent();
    }

    private double savedValue = 0;
    private char op;
    private bool newButton = false;

    private void num_click(object sender, RoutedEventArgs e)
    {
        Button btn = sender as Button;

        string number = btn.Content.ToString();

        if (txtResult.Text == "0" || newButton == true)
        {
            txtResult.Text = number;
            newButton = false;
        }
        else
            txtResult.Text = txtResult.Text + number;
```

```
    }

    private void dot_click(object sender, RoutedEventArgs e)
    {
        if (txtResult.Text.Contains(".") == false)
            txtResult.Text += ".";
    }

    private void op_click(object sender, RoutedEventArgs e)
    {
        Button btn = sender as Button;

        savedValue = double.Parse(txtResult.Text);// string의 첫번째 요소 값
        op = btn.Content.ToString()[0];
        newButton = true;
    }

    private void equal_click(object sender, RoutedEventArgs e)
    {
        if (op == '+')
            txtResult.Text = (savedValue + double.Parse(txtResult.Text)).ToString();
        else if (op == '-')
            txtResult.Text = (savedValue - double.Parse(txtResult.Text)).ToString();
        else if (op == '*')
            txtResult.Text = (savedValue * double.Parse(txtResult.Text)).ToString();
        else if (op == '/')
            txtResult.Text = (savedValue / double.Parse(txtResult.Text)).ToString();
    }
}
```

3. PuzzleGame



▌ PuzzleGame.xaml

```
<UniformGrid Name="board">
</UniformGrid>
```

▌ PuzzleGame.xaml.cs

```
public partial class Ex3_CardGame : Window
{
    Button first;
```

```
Button second;
string[] strImageName = { "AH", "2H", "3H", "4H", "5H", "6H", "7H", "8H" };
DispatcherTimer myTimer = new DispatcherTimer();
int matched = 0;

Image[] imgCard = new Image[8];
Image imgBack = null;

public Ex3_CardGame()
{
    InitializeComponent();

    boardSet();
    LoadImage();
    // 타이머 객체 생성
    myTimer.Interval = new TimeSpan(0, 0, 0, 0, 750); // 0.75초
    myTimer.Tick += MyTimer_Tick;
}

private void LoadImage()
{
    imgBack = MakeImage("/Image/blue_back.jpg");

    for (int i = 0; i < 8; i++)
    {
        imgCard[i] = MakeImage("/Image/" + strImageName[i] + ".jpg");
    }
}

private void MyTimer_Tick(object sender, EventArgs e)
{
    myTimer.Stop();

    first.Content = imgBack;
    second.Content = imgBack;
    first = null;
    second = null;
}
//
private void boardSet()
{
    for (int i = 0; i < 16; i++)
    {
        Button c = new Button();
```

```
            c.Background = Brushes.White;
            c.Margin = new Thickness(10);
            c.Content = MakeImage("/Image/blue_back.jpg");
            c.Tag = TagSet();   //  그림의 인덱스
            c.Click += Card_Click;
            board.Children.Add(c);
        }
    }

    private Image MakeImage(string v)
    {
        BitmapImage bi = new BitmapImage();
        bi.BeginInit();
        bi.UriSource = new Uri(v, UriKind.Relative);
        bi.EndInit();

        Image myImage = new Image();
        myImage.Margin = new Thickness(10);
        myImage.Stretch = Stretch.Fill;
        myImage.Source = bi;

        return myImage;
    }
    //--------------------------------
    // 0~7사이의 정수를 만들어 리턴하는 함수
    // 0~15사이의 랜덤값이 중복되지 않게 만들어지고
    // 이를 8로 나눈 나머지 값을 리턴합니다.

    int[] rnd = new int[16];  // 랜덤 숫자가 중복되는지 체크

    private int TagSet()
    {
        int i;

        Random r = new Random();

        while (true)
        {
            i = r.Next(16); // 0~15까지

            if (rnd[i] == 0)
            {
                rnd[i] = 1;
                break;
```

```
            }
        }
        return i % 8; // 태그는 0~7까지, 8개의 그림을 표시
    }

    //-----------------------------------------------

    private void Card_Click(object sender, RoutedEventArgs e)
    {
        Button btn = sender as Button;

        // 버튼에 따른 그림 Load
        string imgNo = strImageName[(int)btn.Tag];

        if (first == null) // 이 버튼은 첫번째 버튼
        {
            first = btn;
            btn.Content = MakeImage("/Image/" + imgNo + ".jpg");
            return;
        }
        else if (second == null)
        {
            second = btn;
            btn.Content = MakeImage("/Image/" + imgNo + ".jpg");
        }
        else   // 이미 두개의 버튼이 열렸는데 3번째 버튼을 눌렀을 경우, 아무 일 안하고
리턴
            return;

        // 매치가 되었을 때 (2)
        if ((int)first.Tag == (int)second.Tag)  // (3) 같다면
        {
            first = null;
            second = null;
            matched += 2;
            if (matched >= 16)
            {
                MessageBoxResult res = MessageBox.Show(
                        "성공! 다시 하시겠습니까?", "Success", MessageBoxButton.YesNo);
                if (res == MessageBoxResult.Yes)
                {
                    resetRnd();
                    boardReset();
                    boardSet();
```

```
                    matched = 0;
                }
            }
        }
        // 매치가 안되었을 때는 다시 덮어주어야 한다 (4)
        else
        {
            myTimer.Start();
        }
    }
    // 게임 초기화
    private void boardReset()
    {
        board.Children.Clear();
    }

    // rnd[] 배열 초기화
    private void resetRnd()
    {
        for (int i = 0; i < 16; i++)
            rnd[i] = 0;
    }
}
```

SECTION 12

ETC

1. FileDialog1

▌ FileDialog1.xaml

```
< Grid >
    < Button Content = "Button" HorizontalAlignment = "Left" Margin = "145,114,0,0"
VerticalAlignment = "Top" Width = "75" Click = "Button_Click" />
</ Grid >
```

▌ FileDialog1.xaml.cs

```
public partial class Ex1_FileDialog : Window
{
    public Ex1_FileDialog()
    {
        InitializeComponent();
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        // Configure open file dialog box
        Microsoft.Win32.OpenFileDialog dlg = new Microsoft.Win32.OpenFileDialog();
        dlg.FileName = "Document"; // Default file name
        dlg.DefaultExt = ".txt"; // Default file extension
        dlg.Filter = "Text documents (.txt)¦*.txt"; // Filter files by extension

        // Show open file dialog box
        Nullable<bool> result = dlg.ShowDialog();

        // Process open file dialog box results
        if (result == true)
        {
            // Open document
            string filename = dlg.FileName;
        }
    }
}
```

2. FileDialog2

FileDialog2.xaml

```
< DockPanel Margin = "10" >
    < WrapPanel HorizontalAlignment = "Center" DockPanel.Dock = "Top" Margin =
"0,0,0,10" >
        < Button Name = "btnSaveFile" Click = "btnSaveFile_Click" > Save file </ Button
>
    </ WrapPanel >
    < TextBox Name = "txtEditor" TextWrapping = "Wrap" AcceptsReturn = "True"
ScrollViewer.VerticalScrollBarVisibility = "Auto" />
</ DockPanel >
```

FileDialog2.xaml.cs

```
public partial class SaveFileDialogSample : Window
{
    public SaveFileDialogSample()
    {
        InitializeComponent();
    }
    private void btnSaveFile_Click(object sender, RoutedEventArgs e)
    {
        SaveFileDialog saveFileDialog = new SaveFileDialog();

        saveFileDialog.Filter = "Text file (*.txt)¦*.txt¦C# file (*.cs)¦*.cs";
        saveFileDialog.InitialDirectory = @"c:\temp\";
        saveFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
        if (saveFileDialog.ShowDialog() == true)
            File.WriteAllText(saveFileDialog.FileName, txtEditor.Text);
    }
}
```

3. Threading

▌ Threading.xaml

```
< Grid >
    < Grid.RowDefinitions >
        < RowDefinition Height = "50*" />
        < RowDefinition Height = "50*" />
    </ Grid.RowDefinitions >
    < Button x: Name = "button" Grid.Row = "0"
        Width = "100"
        Height = "30"
        Content = "테스트" />
    < TextBox x: Name = "textBox" Grid.Row = "1"
        Width = "100"
        Height = "25"
        HorizontalContentAlignment = "Center"
        VerticalContentAlignment = "Center"
        BorderBrush = "Black"
        BorderThickness = "1" />
</ Grid >
```

▌ Threading.xaml.cx

```
public partial class MainWindow : Window
{

    public MainWindow()
    {
        InitializeComponent();

        this.button.Click += button_Click;
    }

    private void button_Click(object sender, RoutedEventArgs e)
    {
        IsEnabled = false;
```

```
        Thread thread = new Thread(new ThreadStart(ProcessTest));

        thread.Start();
    }
    private void ProcessTest()
    {
        for (int i = 1; i <= 10; i++)
        {
            Thread.Sleep(100);

            DispatcherOperation dispatcherOperation = Dispatcher.BeginInvoke
            (
                DispatcherPriority.Normal, new Action<string, int>(
                    (message, value) => {
                        this.textBox.Text = string.Format("{0} : {1}", message, value);
                    }
                ),
                "Test",
                i
            );

            DispatcherOperationStatus dispatcherOperationStatus =
dispatcherOperation.Status;

            while (dispatcherOperationStatus != DispatcherOperationStatus.Completed)
            {
                dispatcherOperationStatus =
dispatcherOperation.Wait(TimeSpan.FromMilliseconds(1000));

                if (dispatcherOperationStatus == DispatcherOperationStatus.Aborted)
                {
                }
            }
        }
        Dispatcher.BeginInvoke(DispatcherPriority.Normal, new Action(() => { IsEnabled
= true; }));
    }
}
```

4. Play Audio

PlayAudio.xaml

```
< Window x: Class = "WpfTutorialSamples.Audio_and_Video.SystemSoundsSample"
        xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns: x = "http://schemas.microsoft.com/winfx/2006/xaml"
        Title = "SystemSoundsSample" Height = "200" Width = "150" >
    < StackPanel Margin = "10" HorizontalAlignment = "Center" VerticalAlignment =
"Center" >
        < Button Name = "btnAsterisk" Click = "btnAsterisk_Click" > Asterisk </ Button
>
        < Button Name = "btnBeep" Margin = "0,5" Click = "btnBeep_Click" > Beep </
Button >
        < Button Name = "btnExclamation" Click = "btnExclamation_Click" > Exclamation
</ Button >
        < Button Name = "btnHand" Margin = "0,5" Click = "btnHand_Click" > Hand </
Button >
        < Button Name = "btnQuestion" Click = "btnQuestion_Click" > Question </ Button
>
    </ StackPanel >
</ Window >
```

PlayAudio.xaml.cs

```
public partial class SystemSoundsSample : Window
{
    public SystemSoundsSample()
    {
        InitializeComponent();
    }

    private void btnAsterisk_Click(object sender, RoutedEventArgs e)
    {
        SystemSounds.Asterisk.Play();
    }

    private void btnBeep_Click(object sender, RoutedEventArgs e)
```

```
    {
        SystemSounds.Beep.Play();
    }

    private void btnExclamation_Click(object sender, RoutedEventArgs e)
    {
        SystemSounds.Exclamation.Play();
    }

    private void btnHand_Click(object sender, RoutedEventArgs e)
    {
        SystemSounds.Hand.Play();
    }

    private void btnQuestion_Click(object sender, RoutedEventArgs e)
    {
        SystemSounds.Question.Play();
    }
}
```

5. Play Video

PlayVideo.xaml

```
< Grid Margin = "10" >
    < Grid.RowDefinitions >
        < RowDefinition Height = "*" />
        < RowDefinition Height = "Auto" />
    </ Grid.RowDefinitions >
    < MediaElement Source = "test.mpg" LoadedBehavior = "Manual" Name = "mePlayer" />
    < StackPanel Grid.Row = "1" >
        < Label Name = "lblStatus" Content = "Not playing..."
HorizontalContentAlignment = "Center" Margin = "5" />
        < WrapPanel HorizontalAlignment = "Center" >
            < Button Name = "btnPlay" Click = "btnPlay_Click" > Play </ Button >
            < Button Name = "btnPause" Margin = "5,0" Click = "btnPause_Click" > Pause
</ Button >
            < Button Name = "btnStop" Click = "btnStop_Click" > Stop </ Button >
        </ WrapPanel >
    </ StackPanel >
</ Grid >
```

PlayVideo.xaml.cs

```
public partial class MediaPlayerVideoControlSample : Window
{
    public MediaPlayerVideoControlSample()
    {
        InitializeComponent();

        DispatcherTimer timer = new DispatcherTimer();
        timer.Interval = TimeSpan.FromSeconds(1);
        timer.Tick += timer_Tick;
        timer.Start();
    }

    void timer_Tick(object sender, EventArgs e)
    {
```

```
        if (mePlayer.Source != null)
        {
            if (mePlayer.NaturalDuration.HasTimeSpan)
                lblStatus.Content = String.Format("{0} / {1}",
mePlayer.Position.ToString(@"mm\:ss"),
mePlayer.NaturalDuration.TimeSpan.ToString(@"mm\:ss"));
        }
        else
            lblStatus.Content = "No file selected...";
    }

    private void btnPlay_Click(object sender, RoutedEventArgs e)
    {
        mePlayer.Play();
    }

    private void btnPause_Click(object sender, RoutedEventArgs e)
    {
        mePlayer.Pause();
    }

    private void btnStop_Click(object sender, RoutedEventArgs e)
    {
        mePlayer.Stop();
    }
}
```

6. Culture

Culture.xaml

```
< StackPanel Margin = "20" >
    < Grid >
        < Grid.RowDefinitions >
            < RowDefinition Height = "Auto" />
            < RowDefinition Height = "Auto" />
        </ Grid.RowDefinitions >
        < Grid.ColumnDefinitions >
            < ColumnDefinition Width = "*" />
            < ColumnDefinition Width = "*" />
        </ Grid.ColumnDefinitions >
        < Label > Number:</ Label >
        < Label Name = "lblNumber" Grid.Column = "1" />
        < Label Grid.Row = "1" > Date:</ Label >
        < Label Name = "lblDate" Grid.Row = "1" Grid.Column = "1" />
    </ Grid >
    < StackPanel Orientation = "Horizontal" HorizontalAlignment = "Center" Margin =
"0,20" >
        < Button Tag = "en-US" Click = "CultureInfoSwitchButton_Click"
HorizontalContentAlignment = "Stretch" > English(US) </ Button >
        < Button Tag = "de-DE" Click = "CultureInfoSwitchButton_Click"
HorizontalContentAlignment = "Stretch" Margin = "10,0" > German(DE) </ Button >
        < Button Tag = "sv-SE" Click = "CultureInfoSwitchButton_Click"
HorizontalContentAlignment = "Stretch" > Swedish(SE) </ Button >
    </ StackPanel >
</ StackPanel >
```

Culture.xaml.cs

```
public partial class ApplicationCultureSwitchSample : Window
{
    public ApplicationCultureSwitchSample()
    {
        InitializeComponent();
    }
```

```
    private void CultureInfoSwitchButton_Click(object sender, RoutedEventArgs e)
    {
        Thread.CurrentThread.CurrentCulture = new CultureInfo((sender as
Button).Tag.ToString());
        lblNumber.Content = (123456789.42d).ToString("N2");
        lblDate.Content = DateTime.Now.ToString();
    }
}
```