\*자료형

int(정수), char(문자), float(소수), string(문자열)

\*var 자료형

숫자, 문자, 문자열 등등 변수의 자료형을 자동으로 알아서 저장한다.

단, 클래스의 멤버변수 또는 전역변수로는 쓸 수 없다.

오직 함수 안에서만 쓰인다.

단점은 자동으로 알아서 처리하기 때문에 메모리가 더 많이 든다.

에러가 났을때, 명시적으로 자료형을 선언한것보다 디버깅이 어려워진다.

\*배열

int[] array = new int[5] { 1, 2, 3, 4, 5 };

int[] array2 = new int[5];

array2[0] = 1;

array2[1]  = 2;

array2[2]  = 3;

array2[3]  = 4;

array2[4]  = 5;

\*반복문

 for(int i=0; i<5; i++)

 {

        array[i] = 3;

        array2[i] = i%2; //%는 나눈 나머지

 }

\*리스트

List<string> list = new List<string>();

            list.Add("사과");

            list.Add("귤");

            list.Add("포도");

            list.Add("딸기");

            list.Add("수박");

int listCount = list.Count();

for(int i=0; i<listCount; i++)

            {

                string row = list[i];

                if("포도".Equals(row)) //if(row.Equals("포도") 이것과 같다

                {

                    list[i] = "샤인머스캣";

                }

            }

//list[2]의 결과값은? 샤인머스캣

\*Dictionary (Java에서의 Map과 같다. 키와 값으로 구성된다. 키만 알면 값을 뽑아쓸수 있다)

Dictionary<string, string> d = new Dictionary<string, string>();

            d.Add("20250001", "홍길동");

            d.Add("20250002", "짱구");

            d.Add("20250003", "맹구");

            d.Add("20250004", "강감찬");

            string person = d["20250003"];

            //person의 결과값은? 맹구

            d["20250001"] = "이순신";

            //d["20250001"]의 결과값은? 이순신

\*배열, 리스트, 딕셔너리 차이점

-배열은 메모리를 미리 고정하여 할당한다.

-리스트는 각 노드마다 이전메모리주소와 다음메모리주소를 가지고 있기 때문에

중간 노드들을 얼마든지 추가하거나 삭제할 수 있다.

-딕셔너리는 키와 값 형태로 이루어져있으므로, 순서와는 상관없이 키만 알면 값을 얼마든지 뽑아쓸 수 있다. (순서가 중요치 않음)

\*화면단

-라벨 텍스트 변경

label1.Text = "클릭함";

-팝업(얼럿)을 띄움

MessageBox.Show("클릭함");

\*객체

클래스를 자료형으로 갖는 변수를 말한다.

\*클래스 생성

TestClass t1 = new TestClass(); //인자값이 없는 기본 생성자 실행

TestClass t2 = new TestClass(1, "하이"); //인자값이 두개인 생성자 실행

\*클래스내의 private, public, protected

-private : 클래스를 객체로 만들었을때 외부에서 접근 불가능

-public :  클래스를 객체로 만들었을때 외부에서 접근 가능

-protected : 클래스를 객체로 만들었을때 외부에서 접근 불가능해서 private와 똑같지만,

그 클래스를 상속받은 자식 클래스 안에서는 접근 가능

\*클래스 상속 ( : 붙이고 뒤에 부모클래스를 넣으면 됨)

public class ChildClass : TestClass

{

}

\*자식클래스의 생성

ChildClass c = new ChildClass();

이때 자식클래스의 객체를 만들면

자식 클래스의 생성자를 실행되지만, 그전에 부모클래스의 생성자가 먼저 실행이 자동으로 됨

(상속받는순간 자식클래스의 생성자를 실행하는순간 사실상 이미 부모클래스의 생성자 실행이 생략되어있음)