Chapter 09 인터페이스

01 인터페이스의 선언

* 인터페이스 선언 형식

Interface 인터페이스이름{

반환형식 메소드이름1 ( 매개변수 목록 );

반환형식 메소드이름2 ( 매개변수 목록 );

반환형식 메소드이름3 ( 매개변수 목록 );

// ..

}

* 인터페이스 선언 예

Interface ILogger{

Void WriteLog( string log );

}

02 인터페이스는 약속이다.

* 인터페이스로부터 파생될 클래스가 어떤 메소드를 구현해야 할지를 정하는 것이다.
* 인터페이스도 소프트웨어 내에서 USB 같은 역할을 함
  + 인터페이스에 선언되어 있는 메소드를 구현하기만 한다면 해당 인터페이스를 지원하는 코드에는 그 인터페이스의 모든 파생 클래스를 사용할 수 있음
  + 앞 페이지에서 선언한 ILogger 인터페이스를 상속받는 클래스는 WriteLog( ) 메소드를 구현할 때 로그를 파일에 기록하도록 할 수도 있고 모니터로 보내도록 할 수도 있음.
* 특정 클래스 만들 때 사용하는 규약(실수하지 않게 도와주는 기능)

01 인터페이스 소개

* IComparable 인터페이스
  + 인터페이스는 모두 대문자 I 로 시작
  + 비교할 때 사용하는 규약
* class Program
* {
* class Product
* {
* public string Name { get; set; }
* public int Price { get; set; }
* public override string ToString()
* {
* return Name + " : " + Price + "원";
* }
* }
* static void Main(string[] args)
* {
* List<Product> list = new List<Product>()
* {
* new Product() { Name = "고구마", Price = 1500 },
* new Product() { Name = "사과", Price = 2400 },
* new Product() { Name = "바나나", Price = 1000 },
* new Product() { Name = "배", Price = 3000 }
* };
* list.Sort();
* foreach (var item in list)
* {
* Console.WriteLine(item);
* }
* }
* }

1. 기본적인 클래스와 자료를 생성한다.
2. IComparable 인터페이스를 상속한다.

* Product 클래스의 정렬 기준을 정해주기 위해 IComparable 인터페이스 구현

1. IComparable 인터페이스의 메서드

* IComparable 클래스에 커서 놓아 생기는 파란색 상자 클릭, 또는 커서를 놓고 Ctrl + .단축키

1. CompareTo( ) 메서드 구현하기

* IComparable 인터페이스는 CompareTo( ) 메서드 구현해야됨

class Program

{

class Product:IComparable

{

public string Name { get; set; }

public int Price { get; set; }

public int CompareTo(object obj)

{

return this.Price.CompareTo((obj as Product).Price);

}

public override string ToString()

{

return Name + " : " + Price + "원";

}

}

static void Main(string[] args)

{

List<Product> list = new List<Product>()

{

new Product() { Name = "고구마", Price = 1500 },

new Product() { Name = "사과", Price = 2400 },

new Product() { Name = "바나나", Price = 1000 },

new Product() { Name = "배", Price = 3000 }

};

list.Sort();

foreach (var item in list)

{

Console.WriteLine(item);

}

}

* }
* 인터페이스 인스턴스화
  + 인터페이스는 실체 없는 규칙이기 때문에 인스턴스화 할 수 없음(오류 발생)

IComparable comparable = new IComparable( );

* IDisposable 인터페이스
  + Using 블록 사용할 때 자동으로 호출되는 규약

1. IDisposable 인터페이스 상속하기
2. IDisposable 인터페이스의 메서드 생성하기

* - IDisposable 클래스에 커서를 놓아 생기는 파란 상자 클릭!

1. Dispose( ) 메서드 구현 및 호출하기
2. Using 키워드와 IDisposable 인터페이스

class Program

{

class Dummy : IDisposable

{

public void Dispose()

{

Console.WriteLine("Dispose() 메서드를 호출합니다.");

}

}

static void Main(string[] args)

{

Dummy dummyA = new Dummy();

dummyA.Dispose();

using (Dummy dummyB = new Dummy())

{

}

}

* }
* 인터페이스 생성 방법

Interface [인터페이스 이름] { }

* 같은 파일에 인터페이스 생성
  + 예



* 인터페이스 구현

1. 인터페이스 생성하기

Interface IBasic {

Int TestInstanceMethod( ); /// 메서드에 내부 구현을 입력할 수 없다.

Int TestProperty { get; set;} /// 속성에도 마찬가지로 내부 구현을 입력할 수 없습니다.

}

1. 인터페이스 상속하기

Class Program{

Class TestClass : IBasic { }

Static void Main(string[ ] args) { }

}

1. 인터페이스 구현하기

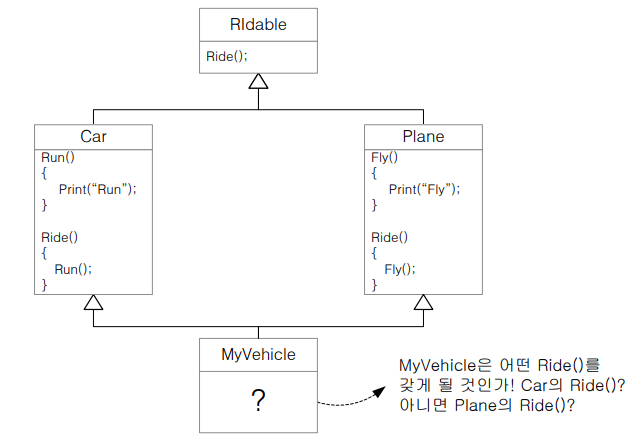
인터페이스 주변에 마우스 커서 올려 뜨는 파란색 커서로 인터페이스 구현



* 인터페이스의 다형성
  + 인터페이스 구현한 클래스는 인터페이스 상속

IBasic basi = new TestClass( ); //// 이런식으로 나타낼 수 있다.

* 
* 죽음의 다이아몬드
  + 하나의 할아버지 클래스를 두 개의 파생 클래스가 상속하고, 이 두개의 파생 클래스를 다시 하나의 자식 클래스가 상속하는 것



* 하나의 클래스가 여러 부모 클래스 가질 수 있게 되는 것
* 클래스 상속과 인터페이스 상속 함께 활용, 하나의 클래스가 여러 다형성 가지게 됨

Class Child : Parent, IDisposable, IComparable 이렇게 쓴다면 한 개의 클래스와 두 개의 인터페이스를 상속 받은 것이 된다.

Class Child : Parent, IDisposable, IComparable {

Public void Dispose( ){

///…. } //// IDispose 인터페이스 구현

Public int CompareTo(object obj){

//….} //// IComparable 인터페이스 구현

}

* 자료형 변환

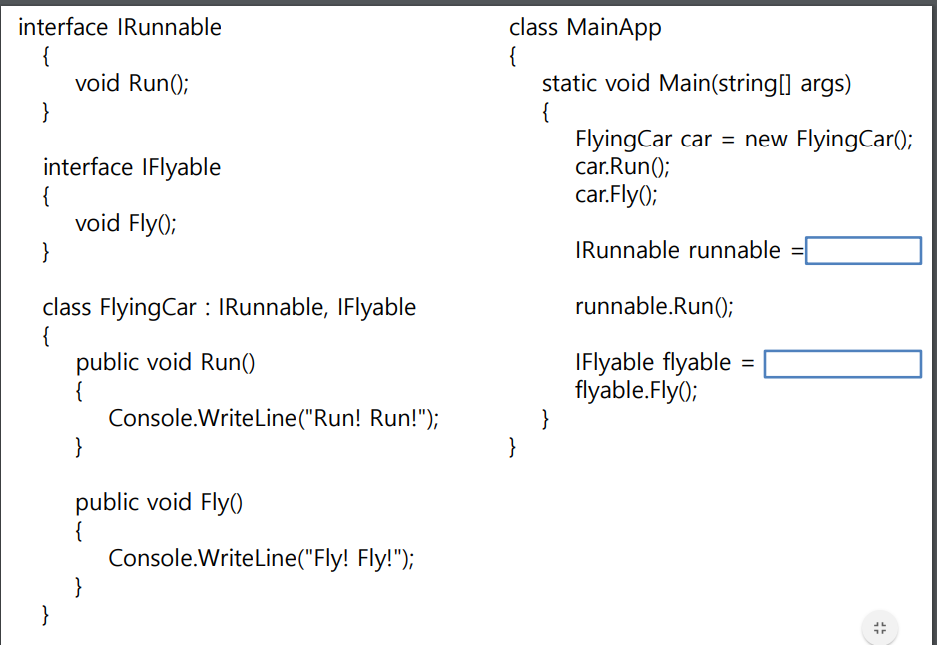
Child child = new Child( );

Parent childAsParent = new Child( );

IDisposable childAsDisposable = new Child( );

IComparable childAsComparable = new Child( );

* 인터페이스를 활용하면 코드에 규약 부여해, 여러 사람과 작업 시 안정성 높임



* 위의 빈칸에는 new IRunnable( ) 과 new FlyingCar( ) 이 올 수 있다.
* 그 아래 빈칸에는 new IFlyable( ) 과 new FlyingCar( ) 이 올 수 있다. 🡺 인터페이스의 다형성
* 파일 처리

1. 한번에 일고 쓰기

🡺 파일에 문자열 쓰기

File.WriteAllText (@ “C:\test\test.txt”,”문자열을 파일에 씁니다.”);

* @ 기호를 사용한 문자열

🡺 파일 읽기

Console.WriteLine(File.ReadAllText (@ “C:\test\test.txt”));

1. 스트림으로 쓰기

* 한 줄씩 쓸 때 StreamWriter 클래스 사용, 참조 추가 이후 [정의로 이동]

StreamWriter writer = new StreamWriter( );

🡺using 구문에서 StreamWriter 클래스의 인스턴스 생성

Using (StreamWriter writer = new StreamWriter()){ }

🡺StreamWriter 클래스로 파일에 문자열 쓰기

using(StreamWriter writer = new StreamWriter(@”C:\test\test.txt”))

{

Writer.WriterLine(“안녕하세요”);

}

Console.WriteLine(File.ReadAllText(@C:\test\test.txt”));

* streamWriter 를 이용하여 글씨를 쓰고 ReadAllText 를 이용하여 텍스트 파일에 적힌 것들을 전부 읽어온다.

1. 스트림으로 읽기

* 읽을 때 StreamReader 클래스를 사용

Using (StreamReader reader = new StreamReader(@”C:\test\test.text”)){

String line = reader.ReadLine();

Console.WriteLine(line);

}

🡺파일을 한 줄씩 읽으려면

Using (StreamReader reader new = StreamReader(@”C:\test\test.txt”)){

String line;

While((line = reader.ReadLine())!=null){

Console.WriteLine(line);

}

}