12강. 상속

목차

- 의미
- 형식
- 상속 관계에서 생성자와 소멸자 호출
- base, sealed
- override와 overload에 대해서
- 추상 클래스
- 다형성
- 박싱과 언박싱

상속

inheritance

상속

- 의미?

  뒤를 이음

相 서로 상, 빌 양
부수 目 | 총획 9획

1. 서로
2. 바탕
3. 도움, 보조자(補助者)

續 이을 속
부수 糸 | 총획 21 획

1. 잇다
2. 잇닿다(서로 이어져 맞닿다)
3. 계속하다(繼續--)

C# 상속

- 대상?

  클래스

- 목적?

  클래스의 재사용 ➔ **코드량 줄이기**

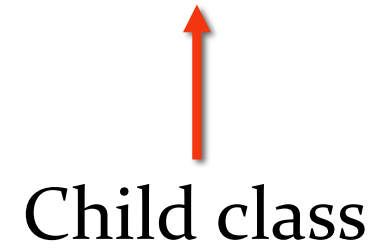
- 상속 클래스의 역할

  부모 클래스 : 상속을 하는 클래스
  (base class, parent class, 상위 클래스)

  자식 클래스 : 상속을 받는 클래스
  (child class, derived class, 파생 클래스)

C# 상속

- 상속 관계 표시 및 형식




```
class A
{
    ……
}

class B : A
{

}
```

C# 상속 접근제한

- 상위 클래스 접근제한

```
class A
{
    private
    protected
    public
}

class B : A
{

}
```

B에서 접근할 수 있는 것?
➔ protected, public

Main과 같은 외부에서 접근 ?
➔ public

소스

상속 관계에서 생성자, 소멸자 호출

- 생성자 호출
  상위 ➔ 하위
- 소멸자 호출
  하위 ➔ 상위

소스

상속과 관련된 base 키워드

- 역할

  상위 클래스의 생성자
  또는 멤버 변수 및 메서드 호출

- 활용?

  - 멤버 이름의 중복
  - 하위에서 상위 설정 등
    (생성자 위주)

소스

상속과 관련된 sealed

- sealed ?

  봉인을 한

- 사용 의미?

  상속 불가에 대한 명시

  (멤버변수, 메서드)

상속과 관련된 sealed

- sealed 사용 형식

(type 1)

```
sealed class  A
{
     .....
}

class B : A
{
```

(type 2)

```
class A
{
     sealed public void Print()
}
```

override와 overload

오버라이드 override

- override

  무시하다

- C#에서의 override 의미?

  상위 메서드를 무시하고
  하위에서 재정의 하는 것

- override의 대상?

  클래스 메서드 > 속성, 인덱서, 이벤트

오버라이드 override

- override 사용 형식

  상위 클래스에는 virtual 명시
  하위 클래스에는 override 명시

  소스

오버로드 overload

- overload ?

  과적하다. 과부하

- 역할?

  하나의 메서드명에 다양한 매개변수 적용

- 장점

  하나의 메서드로 다양한 값을 대입

오버로드 overload

- 형식

  메서드명만 동일

  매개변수는 임의로 적용

- 호출?

  메서드명과 매개변수로 호출

오버로드 overload

- 형식

```
class A
{
        public void Print() { ... }
        public void Print(int number){ .... }
        public void Print(int num, float pi) { ... }
 ......
}

Main()
{
```

추상 클래스

abstract class

추상 클래스 abstract class

- abstract

  추상적인, 관념적인

抽 뽑을 추
부수 手 | 총획 8획

1. 뽑다. 뽑아내다
2. 빼다
3. 없애다. 제거하다(除去--)

象 코끼리 상
부수 豕 | 총획 11획

1. 코끼리
2. 상아(象牙)
3. 꼴. 모양. 형상(形象 · 形像)

**여러 가지 사물이나 개념에서
공통되는 특성이나 속성 따위를
추출하여 파악하는 작용.**

추상 클래스 abstract class

- 의미?

  구현하려는 메서드의 형태만 존재하는 클래스

- 역할

  추상 클래스는 구현 형태만 제공
  실제 구현은 하위에서 구현

- 제한 사항

  - 추상클래스는 상속으로만 사용
  - new를 통해 생성할 수 없다.
  - abstract 가 있는 상위 메서드만
    하위에서 모두 구현

추상 클래스

- 형식

```
abstract class A
{
    public abstract void Print();
     .....
}


class B : A
{
    public override void Print()
    {
                구현....
```

소스

추상 클래스

- 용도

  제공되는 형식으로 메서드를 구성해야 하는 경우

다형성

polymorphism

다형성

- 의미?
  그리스어 ➔ “여러 형태”

  같은 종(*種*)의 생물이면서도
  어떤 형태나 형질이 다양하게
  나타나는 현상

  상속 관계에서 일어남

다형성

- 일반적 형태 : 상위에서 하위 호출

```
class A
{
    public virtual void Print() { ....}
}

class B : A
{
    public override void Print() { ...}
     ........

A Test = new B();
Test.Print();
```

다형성

- type2 : type1 + cast 형을 이용한 하위 참조 호출

소스

박싱 boxing 과 언박싱 unboxing

- 박싱

  값 형식을 object 형 변환

  (int, double, float..)

- 언박싱

  - object 형을 다시 값 형식으로 변환

  - cast를 사용하여 형을 명시

박싱과 언박싱

- 형식 (1)
  int a = 7;
  object obj = a;
  int result = (int)obj;
- 형식 (2)
  구조체도 값 형식이므로
  박싱과 언박싱됨

소스

박싱과 언박싱

- 클래스는 상속 관계에 있으므로

  참조변환이 된다.

  (Upcasting, Downcasting)

  ➔ 박싱과 언박싱과 구별

소스