

3. 객체지향 상속

1. 상속

1). 상속(inheritance)?

상속이란

기본 클래스의 **멤버와 메소드**를 물려받아 새로운 클래스(**파생 클래스**)를 생성할 수 있다.

단, 기본 클래스의 생성자와 종료자는 제외.

그림

Object

Equals()
Finalize()
GetHashCode()
GetType()
MemberwiseClone()
ReferenceEquals()
ToString()

WorkItem : Object

Equals()
Finalize()
GetHashCode()
GetType()
MemberwiseClone()
ReferenceEquals()
ToString() (Overridden)
int ID
string Title
TimeSpan jobLength
string Description
Update()
WorkItem()

ChangeRequest : WorkItem

Equals()
Finalize()
GetHashCode()
GetType()
MemberwiseClone()
ReferenceEquals()
ToString() (Inherits WorkItem implementation)
int ID
string Title
TimeSpan jobLength
string Description
Update()
int originalItemID
ChangeRequest()

Key

Inherited Member

㉔ 상속의 대상이 되는놈은 2가지 있다. : 1. 클래스 & 2. 인터페이스

- 즉 클래스만 상속하는게 아니라 인터페이스도 상속이 된다.

㉕ C#에서는 **단일 상속만 허용**됩니다.

- 클래스 & 추상클래스 : 하나만.

- 인터페이스 : 여러개.

상속	예제
없음	<code>class ClassA { }</code>
단일	<code>class DerivedClass : BaseClass { }</code>
없음. 두 개의 인터페이스 구현	<code>class ImplClass : IFace1, IFace2 { }</code>
단일. 하나의 인터페이스 구현	<code>class ImplDerivedClass : BaseClass, IFace1 { }</code>

- 예시

1. 부모클래스는 없거나, 오직 단 하나만 상속 가능
2. 인터페이스는 여러개도 상속이 된다.

```
Class Foo : _부모클래스_ (또는) _인터페이스1_, _인터페이스2_ , ... , _인터페이스N_
{
}
```

2). 상속의 의의

1. 기존에 작성된 클래스를 **재활용**.
2. 기존 클래스를 **확장**
3. 기존 클래스를 **재정의**

3. 메소드 오버라이딩(method overriding)

1). 특징

클래스의 메소드를 재정의하고 싶을때 사용하는 문법

```
Class 검 {
    int 공격력; int 공속;
    void Attack(){ 대충 3번 치키}
}
Class 광검 : 검 {
```

```

...
    override void Attack() {빛을 내며 3번치기 그리고 마지막 추가타}
    ...
}

```

- 기본 클래스에서 **이미 정의된 메소드**를
파생 클래스에서 **다른 로직으로 메소드를 다시 정의**하는 것이라고 할 수 있습니다.
- private 멤버를 제외한 모든 메소드를 상속받습니다.
이렇게 상속받은 메소드는 그대로 사용해도 되고, 필요한 동작을 위해 재정의하여 사용할 수도 있음.

2). 궁금한데 아니 그럼 멤버변수는 오버라이트 못하나? 🤖

4. 다형성

http://www.tcpschool.com/java/java_polymorphism_concept

1) 특징

"부모클래스 타입"의 참조 변수로 "자식클래스 타입"의 인스턴스를 참조할 수 있다.

그럼 부모클래스는 자식클래스 전용 멤버에 접근할수 있나?? : ✕

- 부모클래스 타입 변수는 오직 부모클래스 멤버만 접근할 수 있다.
- 그말인 즉슨 **비록 자식이 가지고 있다 하더라도 부모에 구현이 안되어 있으면 쓸수**
가 없다는 말이다.

Sophia App.cs

Sophia
Equipment.cs

Sophia App.cs	Sophia Equipment.cs
<pre> 1 using System; 2 3 namespace PROJECT_SoPhIA 4 { 5 using static SophiaUnit; 6 using static Equipment; 7 참조 0개 8 internal class App 9 { 10 참조 0개 11 static void Main(string[] args) 12 { 13 Player sophia = new Player(); 14 sophia.status = new UnitStatus() { hp = 100, energy = -1, defense = 0, invincibleStatus = false }; 15 Weapon 장미칼 = new Sword(); 16 Weapon 플레임샷 = new Gun(); 17 18 sophia.weapon = 장미칼; 19 sophia.weapon = 플레임샷; 20 } 21 } </pre>	<pre> 1 using System; 2 3 namespace PROJECT_SoPhIA 4 { 5 참조 0개 6 internal class Equipment 7 { 8 참조 0개 9 public abstract Health { } 10 참조 7개 11 public abstract class Weapon { } 12 참조 1개 13 public class UndefinedWeapon { } 14 참조 1개 15 public class Sword : Weapon { 16 참조 1개 17 public override void Attack() { } 18 참조 1개 19 public override void WeaponSkill() { } 20 } 21 참조 1개 22 public class Gun : Weapon 23 { 24 참조 1개 25 public override void Attack() { } 26 참조 1개 27 public override void WeaponSkill() { } 28 } 29 } </pre>

📖 5. 추상 클래스

1). 추상 클래스(abstract class)?

㉑ 하나 이상의 추상 메소드를 포함하는 클래스를

1. 클래스는 클래스인데..

추상클래스 그 자체로는 인스턴스를 만들 수 없는 미완성 클래스다.. ☹️

◦ 들어있는 추상 매서드도 미완성이다.(구현부가 없다)

2. 오직 상속용으로, 파생클래스 만 쓸수 있는 미완성 클래스

◦ 오버라이딩하고 나서야 비로소 자식 클래스의 인스턴스를 생성할 수 있게 된다..

3. 추상 클래스는 추상 메소드를 포함하고 있다는 점을 제외하면 일반 클래스와 모든 점이 같은것이, 생성자와 멤버변수, 일반 메소드도 포함할 수 있다.

㉒ 생긴 모습

- 추상메소드가 하나 이상 포함되거나 abstract로 정의된 경우를 말합니다.

```

abstract class 클래스이름 {
    /*...*/
    abstract 반환타입 메소드이름();
}

```

```

        선언부만 있고 구현부가 없다
        /*...*/
    }

```

2). 왜 쓰는건가?

1. 꼭 필요한 매서드 이지만, 자손마다 다르게 구현될 수 있다는 점을 염두해 틀(Form)만 제공한다.

즉, **이름은 똑같은데 내용(돌아가는 알고리즘)이 달라야할때**

- 조상이 사용하던걸 써야 되는게 아니라 클래스 생성마다 거기에 맞는 함수를 만들어야 할때,

2. 확장에 있어서 사용하는 틀(Form)을 만든다

◦ **절차**

1. 이 추상 클래스를 상속한다
2. 확장할때 추상 메소드를 자식 클래스에서 반드시 오버라이딩을 통해 구체적으로 구현한다.

3). 예시

㉠ 분명 Animal로 같고, cry()도 분명 똑같은데 다른 실행모습을 보여준다.

```

abstract class Animal {
    abstract void cry();
}
class Cat : Animal {
    void cry() { System.out.println("냐옹냐옹!"); }
}
class Dog : Animal {
    void cry() { System.out.println("멍멍!"); }
}

public static void main(String[] args) {
    // Animal a = new Animal(); // 추상 클래스는 인스턴스를 생성할 수 없음.
    Animal c = new Cat();
    Animal d = new Dog();
    c.cry();
    d.cry();
}

```

>> 냐옹냐옹!
>> 멍멍!

② 다형성을 응용해, 분명 똑같은 Attack()인데.

[illegible]

6. 인터페이스