

클래스란 무엇인가?

-포켓몬을 예시로 알아보자!-

작성자 : 비공개

클래스란?

사용자 정의 자료형

인스턴스(객체)를 생성하기 위한 틀

객체 지향 언어 특징



추상화

원빈(or김태희)와 닮았어요 > 눈 두 개, 코 하나, 입 하나, 귀 두개 -> 사람의 특징 적인 것들을 뽑아내서 정의 > 속성(프로퍼티)

사람이 할 수 있는 행동 > 걷기, 잠자기 , 먹기 > 메소드(함수)

특징과 행동을 정의하는 것이 클래스

당신과 원빈(or 김태희)는 사람이라는 같은 클래스인 것이다.

같은 클래스인데 왜 다르냐고? > 당신과 원빈(or 김태희)는 다른 객체(개체?)이기 때문

포켓몬의 특징? 추상화를 해보자!

포켓몬은 종류가 있다.(ex) 피카츄, 파이리, 꼬부기, 이상해씨

포켓몬은 능력치를 공격, 체력, 스피드로 수치로 가지고 있다.
(객체별 능력)

포켓몬은 종족별로 가질 수 있는 최대/최소능력치가 정해져 있다.
(종족값)

포켓몬은 각자의 타입이 존재한다.(ex) 불, 물, 전기

포켓몬은 자신의 타입에 따라 스킬을 사용할 수 있다.

포켓몬은 레벨이 있고 특정 레벨에 도달 하면 진화를 할 수 있다.

```

public class Pokemon
{
    string name;

    protected int myStrong;
    protected int myHealth;
    protected int mySpeed;

    protected int maxStrong = 10;
    protected int minStrong = 1;
    protected int maxHealth = 10;
    protected int minHealth = 1;
    protected int maxSpeed = 10;
    protected int minSpeed = 1;

    protected PokemonType type = 0;
    protected int level;

    참조 0개
    public Pokemon()
    {
    }

    참조 0개
    public void ShowUp()
    {
        Console.WriteLine("포켓몬이 나타났다!");
    }

    참조 0개
    public void UseSkill()
    {
        Console.WriteLine("포켓몬은 스킬을 사용했다.");
    }
}

```

- 풀숲을 지나가다가 당신은 포켓몬을 조우하였습니다.
당신이 만난 포켓몬은?

```

class Program
{
    참조 0개
    static void Main(string[] args)
    {
        Pokemon pokemonA = new Pokemon("A");
        pokemonA.ShowUp();

        Pokemon pokemonB = new Pokemon("B");
        pokemonB.ShowUp();

        Thread.Sleep(5000);
    }
}

```

```

A 포켓몬이 나타났다!
B 포켓몬이 나타났다!

```

클래스를 정의하여 인스턴스를 생성하여
우리는 다양한 포켓몬을 만날 수 있습니다.

포켓몬이라는 같은 클래스에 속하지만 A와
B는 다른 포켓몬입니다.

```
class Program
{
    참조 0개
    static void Main(string[] args)
    {
        Pokemon poketmonA = new Pokemon("A");
        poketmonA.ShowUp();

        Pokemon poketmonB = new Pokemon("B");
        poketmonB.ShowUp();

        Thread.Sleep(5000);
    }
}
```

상속

- 포켓몬에는 피카츄, 파이리, 꼬부기, 이상해씨 등 다양한 포켓몬이 있는데, 클래스 별로 속성을 다시 정의해야 하나요? > 서로 생긴 모습은 달라도 같은 포켓몬이기 때문에 공통적인 속성들은 포켓몬 클래스에서 정의 하고 상속을 하면 반복하여 다시 정의할 필요가 없어집니다.
-

참조 1개

```
public class Pikachu : Pokemon
{
    참조 0개
    public Pikachu()
    {
        name = "피카츄";
        maxStrong = 5;
        minStrong = 2;
        maxHealth = 3;
        minHealth = 1;
        maxSpeed = 10;
        minSpeed = 7;

        type = PokemonType.Electric;

        SetMyAbility();
    }
}
```

참조 1개

```
protected void SetMyAbility()
{
    Random random = new Random();
    myStrong = random.Next(minStrong, maxStrong);
    myHealth = random.Next(minHealth, maxHealth);
    mySpeed = random.Next(minSpeed, maxSpeed);
}
```

참조 0개

```
public void ShowAbility()
{
    Console.WriteLine($"{name}이 가질 수 있는 공격 최대 수치 : {maxStrong}/ 최소 수치 {minStrong}/ 해당 객체 수치 {myStrong}");
    Console.WriteLine($"{name}이 가질 수 있는 체력 최대 수치 : {maxHealth}/ 최소 수치 {minHealth}/ 해당 객체 수치 {myHealth}");
    Console.WriteLine($"{name}이 가질 수 있는 스피드 최대 수치 : {maxSpeed}/ 최소 수치 {minSpeed}/ 해당 객체 수치 {mySpeed}");
}
```

포켓몬별 능력치를 설정하는 메소드를 Pokemon 클래스에
서 추가 정의
능력치를 확인하는 메소드 추가 정의
상속 받은 피카츄 클래스에서는 다시 정의하는 거 없이 사용
만 함

참조 0개

```
static void Main(string[] args)
```

```
{
```

```
    Pokemon pokemonA = new Pikachu();
```

```
    pokemonA.ShowUp();
```

```
    pokemonA.ShowAbility();
```

```
    Thread.Sleep(100);
```

```
    Pokemon pokemonB = new Pikachu();
```

```
    pokemonB.ShowUp();
```

```
    pokemonB.ShowAbility();
```

```
    Thread.Sleep(5000);
```

```
}
```

```
}
```

```
피카츄 포켓몬이 나타났다!  
피카츄이 가질 수 있는 공격 최대 수치 : 5/ 최소 수치2/ 해당 객체 수치2  
피카츄이 가질 수 있는 체력 최대 수치 : 3/ 최소 수치1/ 해당 객체 수치1  
피카츄이 가질 수 있는 스피드 최대 수치 : 10/ 최소 수치7/ 해당 객체 수치9  
피카츄 포켓몬이 나타났다!  
피카츄이 가질 수 있는 공격 최대 수치 : 5/ 최소 수치2/ 해당 객체 수치4  
피카츄이 가질 수 있는 체력 최대 수치 : 3/ 최소 수치1/ 해당 객체 수치1  
피카츄이 가질 수 있는 스피드 최대 수치 : 10/ 최소 수치7/ 해당 객체 수치7
```

```

public class Charmander : Pokemon
{
    참조 1개
    public Charmander()
    {
        name = "파이리";
        maxStrong = 10;
        minStrong = 6;
        maxHealth = 8;
        minHealth = 1;
        maxSpeed = 6;
        minSpeed = 3;

        type = PokemonType.Fire;

        SetMyAbility();
    }
}

```

```

static void Main(string[] args)
{
    Pokemon pokemonA = new Pikachu();
    pokemonA.ShowUp();
    pokemonA.ShowAbility();

    Thread.Sleep(100);

    Pokemon pokemonB = new Charmander();
    pokemonB.ShowUp();
    pokemonB.ShowAbility();

    Thread.Sleep(5000);
}

```

C:\Users\USER\source\repos\Pokemon\bin\Debug\Pokemon.exe

```

피카츄 포켓몬이 나타났다!
피카츄이 가질 수 있는 공격 최대 수치 : 5/ 최소 수치2/ 해당 객체 수치4
피카츄이 가질 수 있는 체력 최대 수치 : 3/ 최소 수치1/ 해당 객체 수치1
피카츄이 가질 수 있는 스피드 최대 수치 : 10/ 최소 수치7/ 해당 객체 수치8
파이리 포켓몬이 나타났다!
파이리이 가질 수 있는 공격 최대 수치 : 10/ 최소 수치6/ 해당 객체 수치9
파이리이 가질 수 있는 체력 최대 수치 : 8/ 최소 수치1/ 해당 객체 수치2
파이리이 가질 수 있는 스피드 최대 수치 : 6/ 최소 수치3/ 해당 객체 수치5

```

참조 2개

```
public virtual void UseSkill()  
{  
    Console.WriteLine("포켓몬은 스킬을 사용했다.");  
}
```

참조 0개

참조 2개

```
public override void UseSkill()  
{  
    Console.WriteLine("백만 볼트");  
}
```

참조 2개

```
public override void UseSkill()  
{  
    Console.WriteLine("화염방사");  
}
```

메소드를 재정의 하여 다르게 동작할 수 있습니다.

```
static void Main(string[] args)  
{  
    Pokemon pokemonA = new Pikachu();  
    pokemonA.ShowUp();  
    pokemonA.UseSkill();  
  
    Thread.Sleep(100);  
  
    Pokemon pokemonB = new Charmander();  
    pokemonB.ShowUp();  
    pokemonB.UseSkill();  
  
    Thread.Sleep(5000);  
}
```

```
01 > C:\Users\user>dotnet run  
02  
03 피카츄 포켓몬이 나타났다!  
04 백만 볼트  
05 파이리 포켓몬이 나타났다!  
06 화염방사  
07
```

다형성

- 하나의 클래스가 여러 형태로 변환될 수 있는 성질
 - 포켓몬을 조우할 때, 포켓몬이 모습이 나오기 전까지는 어떤 포켓몬인지 알 수 없다. (포켓몬을 확인하기 전까지는 피카츄인지? 파이리인지? 말할 수 없는 것이다.)
 - 여태 보여준 예시를 보면 부모 클래스로 정의하고 자식 클래스 생성자를 사용하였다. 따로 각각 피카츄, 파이리 리스트(or배열)을 선언할 필요 없이 Pokemon 클래스로 선언하여 피카츄 파이리를 담을 수 있게 된다.
-

참조 1개

```
private static Pokemon MeetPokemon()  
{  
    Random random = new Random();  
    Pokemon pokemon;  
  
    switch (random.Next(0, 3))  
    {  
        case 0:  
            pokemon = new Pikachu();  
            break;  
        case 1:  
            pokemon = new Charmander();  
            break;  
        default: pokemon = null;  
            break;  
    }  
  
    return pokemon;  
}
```

랜덤으로 포켓몬을 만나는 메소드 구현 후
사용

```
static void Main(string[] args)  
{  
    List<Pokemon> meetPokemonList = new List<Pokemon>();  
  
    for(int i = 0; i < 5; i++)  
    {  
        meetPokemonList.Add(MeetPokemon());  
  
        Thread.Sleep(500);  
    }  
  
    foreach(var pokemon in meetPokemonList)  
    {  
        if (pokemon == null)  
            Console.WriteLine("포켓몬을 조우하지 못했습니다.");  
        else  
        {  
            pokemon.ShowUp();  
            pokemon.ShowAbility();  
            pokemon.UseSkill();  
        }  
  
        Console.WriteLine("-----");  
    }  
  
    Thread.Sleep(10 * 1000);  
}
```

```
C:\Users\USER\source\repos\Poketmon\bin\Debug\Poketmon.exe

파이리 포켓몬이 나타났다!
파이리 가질 수 있는 공격 최대 수치 : 10/ 최소 수치6/ 해당 객체 수치8
파이리 가질 수 있는 체력 최대 수치 : 8/ 최소 수치1/ 해당 객체 수치2
파이리 가질 수 있는 스피드 최대 수치 : 6/ 최소 수치3/ 해당 객체 수치4
화염방사

-----
포켓몬을 조우하지 못했습니다.

-----
피카츄 포켓몬이 나타났다!
피카츄 가질 수 있는 공격 최대 수치 : 5/ 최소 수치2/ 해당 객체 수치2
피카츄 가질 수 있는 체력 최대 수치 : 3/ 최소 수치1/ 해당 객체 수치1
피카츄 가질 수 있는 스피드 최대 수치 : 10/ 최소 수치7/ 해당 객체 수치8
백만 볼트

-----
파이리 포켓몬이 나타났다!
파이리 가질 수 있는 공격 최대 수치 : 10/ 최소 수치6/ 해당 객체 수치8
파이리 가질 수 있는 체력 최대 수치 : 8/ 최소 수치1/ 해당 객체 수치1
파이리 가질 수 있는 스피드 최대 수치 : 6/ 최소 수치3/ 해당 객체 수치3
화염방사

-----
파이리 포켓몬이 나타났다!
파이리 가질 수 있는 공격 최대 수치 : 10/ 최소 수치6/ 해당 객체 수치8
파이리 가질 수 있는 체력 최대 수치 : 8/ 최소 수치1/ 해당 객체 수치2
파이리 가질 수 있는 스피드 최대 수치 : 6/ 최소 수치3/ 해당 객체 수치3
화염방사

-----
```

*관련하여 디자인 패턴 팩토리/추상 팩토리 패턴도 학습해 보세요

추가) 인터페이스

- 특정 클래스를 만들 때, 사용하는 규약
 - 인터페이스를 상속 받으면 정의된 메소드는 반드시 구현되어야 합니다.
 - 동작은 다르지만, 호환이 되어야 할 때, 사용합니다.
 - 포켓몬은 보통 레벨이 오르면 진화를 하게 됩니다. 하지만 특정 포켓몬은 특정 조건을 달성해야 진화할 수 있습니다. 진화 방법과 진화 조건을 만족하는지 확인하고 진화 조건을 달성시키는 인터페이스를 구현하겠습니다.
-
- 인터페이스도 다형성을 사용할 수 있습니다.
-

참조 5개

```
public interface IEvolve
{
    참조 4개
    bool CheckCanEvolve();

    참조 4개
    void NotifyHowToEvolve();

    참조 4개
    void MeetEvovleCondition();
}
```

bool canEvolve = false;

참조 4개

```
public bool CheckCanEvolve()
```

```
{
    return canEvolve;
}
```

참조 4개

```
public void MeetEvovleCondition()
```

```
{
    Random random = new Random();
    for(int i = 0; i < 5; i++)
    {
        Console.WriteLine("진화의 돌을 찾습니다.");

        if(random.Next(0, 10) <= 3)
        {
            canEvolve = true;
            break;
        }
    }
}
```

참조 4개

```
public void NotifyHowToEvolve()
```

```
{
    Console.WriteLine("진화의 돌이 필요합니다.");
}
```

참조 4개

```
public bool CheckCanEvolve()
```

```
{
    return level >= 16;
}
```

참조 4개

```
public void MeetEvovleCondition()
```

```
{
    while(level < 16)
    {
        Console.WriteLine("아무 키 입력 Enter 입력 시 중단");
        ConsoleKeyInfo keyInfo = Console.ReadKey();

        if (keyInfo.Key == ConsoleKey.Enter)
            break;
        else
            level++;
    }

    Console.WriteLine("중단");
}
```

참조 4개

```
public void NotifyHowToEvolve()
{
    Console.WriteLine("레벨 16을 달성해야 진화 할 수 있습니다.");
}
```

```

public class InspectEvolveMachine
{
    참조 0개
    public void CheckCanEvolve(IEvolve evolve)
    {
        Console.WriteLine($"진화 가능 여부 : {evolve.CheckCanEvolve()}");
    }

    참조 0개
    void NotifyHowToEvolve(IEvolve evolve)
    {
        evolve.NotifyHowToEvolve();
    }

    참조 0개
    void MeetEvovleCondition(IEvolve evolve)
    {
        evolve.MeetEvovleCondition();
    }
}

```

진화 검사 장비를 만들어 진화 할 수 있는지 확인 해보겠습니다.

```

static void Main(string[] args)
{
    InspectEvolveMachine machine = new InspectEvolveMachine();

    IEvolve poketmon = new Pikachu();

    machine.NotifyHowToEvolve(poketmon);
    machine.CheckCanEvolve(poketmon);
    machine.MeetEvolveCondition(poketmon);
    machine.CheckCanEvolve(poketmon);

    Thread.Sleep(10 * 1000);
}

```

```

피카츄 : 진화의 돌이 필요합니다.
진화 가 이 여부 : False
진화 의 돌을 찾습니다.
진화 의 돌을 찾지 못했습니다.
진화 의 돌을 찾습니다.
진화 의 돌을 찾았습니다.
진화 가 이 여부 : True

```

피카츄 진화 조건을 만족시키기 위해 반복문을 5번 돌아 진화의 돌을 찾도록 구현하였습니다.

```
static void Main(string[] args)
{
    InspectEvolveMachine machine = new InspectEvolveMachine();

    //IEvolve poketmon = new Pikachu();
    IEvolve poketmon = new Charmander();

    machine.NotifyHowToEvolve(poketmon);
    machine.CheckCanEvolve(poketmon);
    machine.MeetEvolveCondition(poketmon);
    machine.CheckCanEvolve(poketmon);

    Thread.Sleep(10 * 1000);
}
```

```
파이리 : 레벨 16을 달성해야 진화 할 수 있습니다.
진화 가능 여부 : False
아무 키 입력 Enter 입력 시 중단
a
현재 레벨 : 1
아무 키 입력 Enter 입력 시 중단
s
현재 레벨 : 2
아무 키 입력 Enter 입력 시 중단
중단
진화 가능 여부 : False
-
```

```
아무 키 입력 Enter 입력 시 중단
s
현재 레벨 : 9
아무 키 입력 Enter 입력 시 중단
s
현재 레벨 : 10
아무 키 입력 Enter 입력 시 중단
s
현재 레벨 : 11
아무 키 입력 Enter 입력 시 중단
s
현재 레벨 : 12
아무 키 입력 Enter 입력 시 중단
s
현재 레벨 : 13
아무 키 입력 Enter 입력 시 중단
s
현재 레벨 : 14
아무 키 입력 Enter 입력 시 중단
s
현재 레벨 : 15
아무 키 입력 Enter 입력 시 중단
s
현재 레벨 : 16
중단
진화 가능 여부 : True
```

파이리는 진화조건을 달성하기 위해서는 Enter를 제외한 아무 키를 눌러 레벨을 16으로 올려야 합니다.

게임으로 만든다면 미니 게임을 만들어 진화조건을 달성하도록 하면 재미있을 것 같네요

클래스 상속 받아서 다르게 구현하면?

- 메소드를 재정의 하여 다르게 동작하도록 할 수 있는데 왜 인터페이스를 구현하는지 의문이 들것입니다.
 - 만약에 진화검사장비에 사람이 들어갈 수 있다면 어떻게 될까요?
 - 만약에 포켓몬 클래스를 상속받아 재정의하여 구현하게 된다면 사람 클래스가 포켓몬을 상속을 받아야 합니다. 하지만 사람은 포켓몬이 아닙니다.
-

```

public class Human : IEvolve
{
    참조 4개
    public bool CheckCanEvolve()
    {
        Console.WriteLine("사람은 진화할 수 없어요");
        return false;
    }

    참조 4개
    public void MeetEvovleCondition()
    {
        Console.WriteLine("사람은 진화할 수 없어요");
    }

    참조 4개
    public void NotifyHowToEvolve()
    {
        Console.WriteLine("사람은 진화할 수 없어요");
    }
}

```

사람 클래스를 생성하여 진화 검사장비에 넣어보겠습니다.

```

static void Main(string[] args)
{
    InspectEvolveMachine machine = new InspectEvolveMachine();

    //IEvolve poketmon = new Pikachu();
    //IEvolve poketmon = new Charmander();
    //machine.NotifyHowToEvolve(poketmon);
    //machine.CheckCanEvolve(poketmon);
    //machine.MeetEvovleCondition(poketmon);
    //machine.CheckCanEvolve(poketmon);

    IEvolve human = new Human();
    machine.NotifyHowToEvolve(human);
    machine.CheckCanEvolve(human);
    machine.MeetEvovleCondition(human);
    machine.CheckCanEvolve(human);

    Thread.Sleep(10 * 1000);
}

```

```

사람이 진화할 수 없어요
사람이 진화할 수 없어요
진화 가능 여부 : False
사람이 진화할 수 없어요
사람이 진화할 수 없어요
진화 가능 여부 : False

```

포켓몬을 상속받지 않아도 사람도 진화 검사 장비에 넣을 수 있게 되었습니다.

보시는 것과 같이 인터페이스는 좀 더 유연성을 제공할 수 있습니다.

*추상 클래스와 인터페이스의 차이점도 나중에 비교해보세요

참고 서적

- C# 프로그래밍 프로그래밍 기초부터 객체 지향 핵심까지
윤인성 지음 - 한빛아카데미
-

끝으로

- 해당 문서를 무단 배포 및 수정하는 것을 금지합니다.
 - 작성자 뇌피셜이 많으므로 틀린 내용이 **많이** 있을 수 있습니다. 해당 내용을 100% 신뢰하지 마세요. 이로 인한 잘못된 지식 습득에 피해는 책임지지 않습니다.
 - 소스코드 깃 허브 : <https://github.com/Jinuk-Noh/StudyClass>
-