

추상화

팀전목적

#01. 추상 메서드 개요

1) 일반적인 메서드 구조

```
public void hello(파라미터...) // <--- 메서드 선언부 (접근한정자, 리턴타입, 이름, 파라미터)
{
    // <--- 메서드 구현부 ( { ... } )
    ...
}
```

2) 추상 메서드

선언부만을 갖고 구현부를 갖지 않는 형태

리턴타입을 명시하기 전에 `abstract` 예약어를 적용하여 추상 메서드임을 표시한다.

```
public abstract void hello(파라미터...);
```

3) 추상 클래스

추상 메서드를 하나 이상 포함하는 클래스

일반 메서드와 멤버변수를 포함할 수 있다

객체의 선언은 가능하지만 할당은 불가능 `new` 사용불가능

클래스 선언시 `abstract` 예약어를 적용해야 함

추상 메서드를 하나 이상 포함하는 클래스는 무조건 추상 클래스로 지정되어야 한다.

```
public abstract class MyClass {
    ...
}

// 선언은 가능
MyClass my;

// 할당은 할 수 없다. (아래는 에러)
my = new MyClass();
```

4) 추상 클래스의 사용

추상 클래스는 직접적으로 객체를 생성할 수 없지만 상속은 가능.

`new` 를 써서 생성할수는 없지만 상속은 가능함

추상 클래스를 상속받는 자식 클래스는 반드시 부모가 갖고 있는 추상 메서드를 Override 해야 한다.

Override 하지 않을 경우 문법 에러

5) 추상 메서드의 의의

여러 명이 함께 작업하는 프로젝트에서 상위 클래스를 작성한 개발자가 자신이 작성한 소스코드를 Override 하는 개발자에게 특정 메서드의 구현을 강제한다.

#02. Protoss 클래스에 추상화 적용

@startuml Overview

```
abstract class Protoss {
    -name: String
    -hp: int
    -speed: int
    -dps: int
    ----
    +Protoss(name: String, hp: int, speed: int, dps: int)
    ----
    +getName(): String
    +setName(name: String): void
    +getHp(): String
    +setHp(name: String): void
    +getSpeed(): String
    +setSpeed(speed: String): void
    +getDps(): String
    +setDps(dps: String): void
    {abstract} +move(position: String): void
    {abstract} +attack(target: String): void
    +toString(): String
}
```

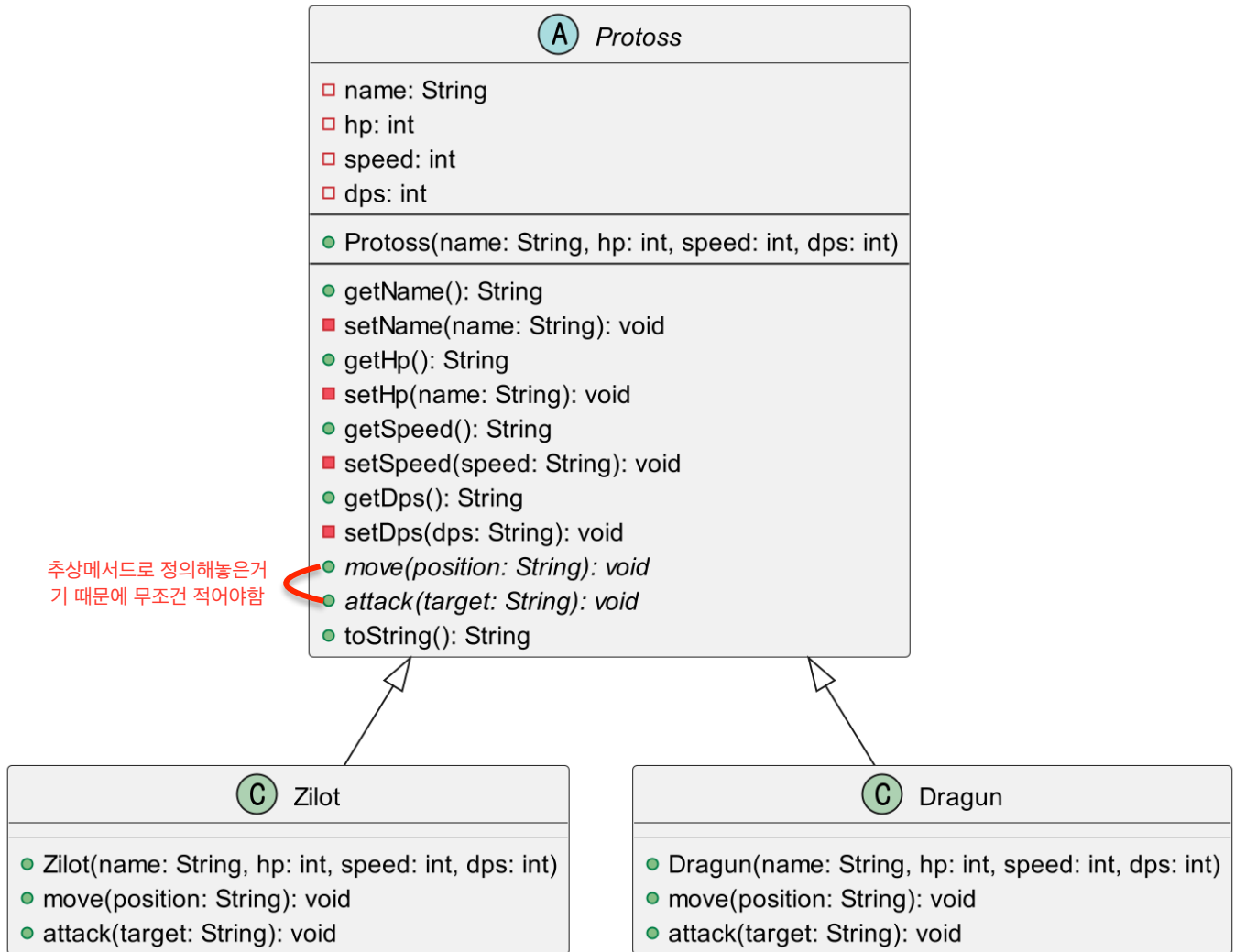
{ } 추상화
된거라는 뜻

```
class Zilot {
    +Zilot(name: String, hp: int, speed: int, dps: int)
    +move(position: String): void
    +attack(target: String): void
}
```

```
class Dragun {
    +Dragun(name: String, hp: int, speed: int, dps: int)
    +move(position: String): void
    +attack(target: String): void
}
```

```
Protoss <|-- Zilot
Protoss <|-- Dragun
```

@enduml



추상화(추상메서드) : 여러명이 동시작업할때 규칙을 정해놓은거임
다르게하면 무조건 에러남
= 팀전목적