01.13 金 _ 추상화

추상화 → 강제性

> 하나의 객체의(사물을 나타내는) 가장 근원적인, 근본적인 부분을 추출 해냄
> (공통적인 속성을 추려내는 것)

⭐ ⇒ 상속 때문에 필요하다 !

⭐ 언제 동적 바인딩을 사용 ?
→ 찍어 내는 것은 최하위에 있는 class들의 객체를 찍어냄
다형성을 가지고 class들의 method나 멤버 변수들에 접근을 해야 할 때

설계

1. 공통적인 기능이 있으면 하나로 모은다
2. 추상화 작업을 한다.
3. 객체를 대표하는(공통 분모) 것을 멤버 (변수, 메서드) 관점에서 도출

abstract

추상화 → 공통적인 부분들을 다 빼서 부모 쪽으로 올리기 -> 동적 바인딩 설계

⭐ 추상은 `class` 앞 , `method` 앞에 만 올 수 있다!

클래스 앞 abstract 붙으면 미완성의 의미를 부여한다.

- class = 틀 인데 **abstract** 를 붙이면 **미완성** 틀이 된다 ⇒ 완성되지 않은 class !
- 만들어지면 안 되는 객체를 다루기 위해 **abstract** 를 사용한다.
- **abstract** 는 미완성 이기 때문에 구현 부분이 없어야 한다!

⇒ 특징을 추려내서 만든 class 이기 때문에 일반 class로 사용해서
객체를 찍는 용도가 아니라는 것을 마킹 해 주는 것! (강제로 막아 버림)

```
  abstract class Player {
    int number;
    String name;

    void shoot() {
        System.out.println("기본 슈팅");
    }

    void pass() {

    }

}

class Center extends Player {
    @Override
```

```
abstract class Player {
    int number;
    String name;

    abstract void shoot() {   // 메서드 앞 추상화 => 미완성 메서드 이기
                              // 자식들이 구현해야 하는 부분 !
        System.out.println("기본 슈팅");
    }

    void pass() {

    }

}

class Center extends Player {
```

```
    void shoot() {
        System.out.println("센터 슈팅");
    }

}
class SJHoon extends Center {
    @Override
    void shoot() {
        System.out.println("서장훈 슈팅");
    }
}

class Scratch {
    public static void main(String[] args) {
//        Player p1 = new Player(); <-- cannot be instantiated
//        p1.shoot();

        Player p2 = new Center();
        p2.shoot();

    }
}
```

```
    @Override
    void shoot() {
        System.out.println("센터 슈팅");
    }

}

class Forward extends Player      / void shoot() 이 없어서 에러 발생
    @Override

  /* void shoot() {
      System.out.println("기본 슈팅");    / 이렇게  void shoot()을 
     } */

    void pass() {
        super.pass();
    }
}

class SJHoon extends Center {
    @Override
    void shoot() {
        System.out.println("서장훈 슈팅");
    }
}

class Scratch {
    public static void main(String[] args) {
//        Player p1 = new Player(); <-- cannot be instantiated
//        p1.shoot();

        Player p2 = new Center();
        p2.shoot();

    }
}
```