## 1. 인터페이스의 이해

① 작업명세서(작업지시서) - "앞으로 이렇게 만들어요"라고 표현해 놓은 것

- 실제 구현된 것이 전혀 없는 기본 설계도.

- 객체를 생성할 수 없고, 클래스 작성에 도움을 줄 목적으로 사용된다

- 미리 정해진 규칙에 맞게 구현하도록 표준을 제시하는 데 사용된다

- 추상메서드와 상수 만을 멤버로 가질 수 있다.

② 다형성을 가능하게 한다(하나의 객체를 다양하게 많은 type으로 만들 수 있다).

```
Class S{
        ...
        public void method(){...}
}
Class C extends S {
        ...
        public void method(){...}
}
S s1 = new C();
C c = new C()
```

```
S s2 = new S();
s1.method();
s2.method();
```

③ 객체를 **부속품화** -다양한 객체를 제품의 부속품처럼 개발자 마음대로 변경 할 수 있다.

④ 사용법은 어렵지 않지만, 실제 개발에 적용시키기는 쉽지 않다.

⑤ 인터페이스를 공부하는데 가장 좋은 방법은 패턴이나 프레임워크(ex. Spring)를 통해 습득하는 것

## 2. 인터페이스의 문법

⑴ 'class'대신 'interface' 예약어를 사용한다는 점에서 클래스와 유사

⑵ 실제 구현된 기능 없이 **추상메소드와 상수**만이 존재

```
public interface 인터페이스이름 {
        public static final 타입 상수이름 = 값;
        public abstract 메서드 이름(매개변수 목록);
//구현된 메소드는 가질 수 없다
}
```

☞모든 멤버변수는 public static final이어야하며 static final은 생략할 수 있다.'

모든 메서드는 public abstract 이어야 하며, abstract를 생략할 수 있다.

⑵ private는 불가 - 상수나 메소드를 만들 때 private 접근 제한자는 불가

⑶ 변수 타입 - 인터페이스는 객체를 생성할 수 없다. 다만, 변수 타입으로만 사용 됩니다.

(예외, 익명 구현 객체만이 가능은 하다. 안드로이드에서 주로)

⑷ 구현은 Implement 되는 클래스에서 합니다.

---

오버로딩(overloading) : 컴파일러 입장에서는 기존에 없는 새로운 메서드를 정의하는 것(new)

메소드 다중정의 (같은 class에서 동일한 메소드가 매개변수를 달리 여러 개 존재

오버라이딩(overriding): 상속받은 메서드의 내용을 변경하는 것 (change, modify)

메소드 재정의 : 부모클래스와 자식클래스에 동일한 method 존재(틀만 가져와 재정의)