12.인터페이스

1)인터페이스

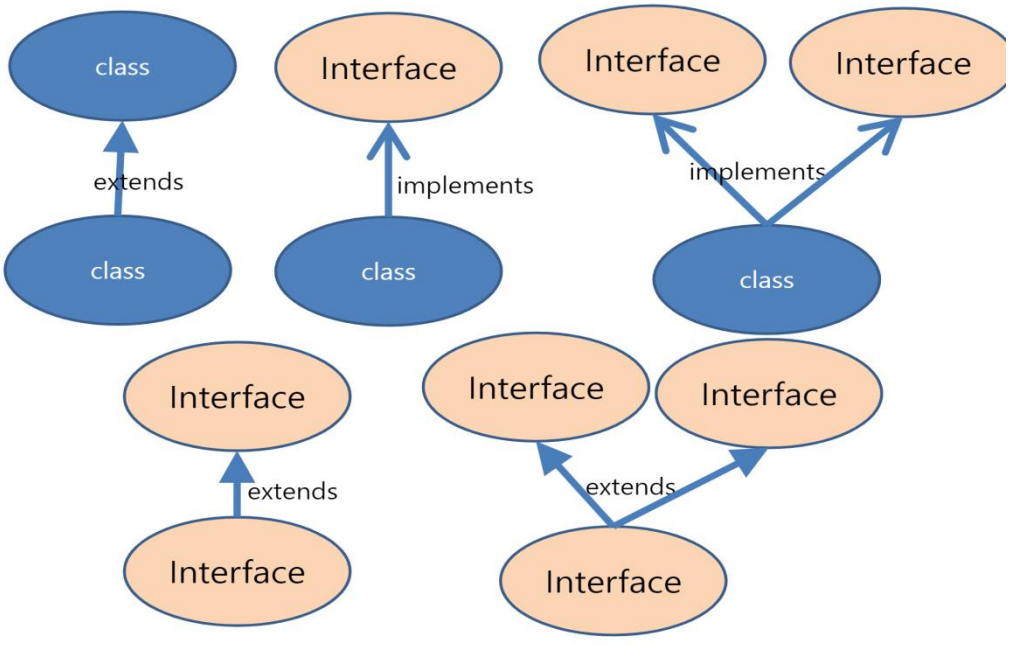
작업명세서 용도, 프로그램 생성 지침, 극단적인 코드 강제 구현 명령어로 미리 정해놓은 구성에 맞게 구현하도록 표준을 제시하는 데 사용된다

개발코드 수정을 줄이고 프로그램 유지보수성을 높이기 위해 인터페이스를 사용하기 위한 목적이다

인터페이스는 다음과 같은 특징이 있다.

- 하나의 클래스나 인터페이스가 여러 인터페이스를 구현가능하다.
- 생성자를 생성할수 없다.(매개변수 입력값을 받지 않는다.)
- 인터페이스 내 생성된 모든 메소드는 abstract 상속과는 다르게 구현을 강제한다.

Ex)상속과 구현 관계



인터페이스와 추상클래스

- 추상클래스나 인터페이스 모두 객체 생성이 불가능하다.
- 인터페이스의 모든 변수는 기본적으로 public static final이며 생략해도 무방하다.
- 인터페이스의 모든 매소드 또한 public abstract 를 기본적으로 함유하고있고 생략해도 무방하다.
- 반면 추상클래스는 static이나 final이 아닌 변수를 지정할수 있고 매소드 역시 abstract 이외에 여러 일반 메소드를 구현할수 있다.

- 인터페이스를 구현하는 클래스는 다중의 인터페이스를 구현할수 있고 다른 클래스를 상속받을수 있다.
- 자바는 다중상속을 지원하지 않으므로 추상클래스는 상속받은 다른 클래스는 다른 상위클래스를 상속받을수 없다.

3)기타

- Static변수는 일반 메소드 호출이 불가능하다. 일반메소드로 선언된 함수는 해당 클래스의 객체로 생성이 되어야 스택 메모리 부분에서 실행가능하기 때문이다.
- 특정 매소드가 값을 반환하고 그 매소드가 반복 사용된다면 변수를 만들어 매소드 값을 할당하고 변수를 반복시키는 것이 큰차 이는 아니지만 프로그램 구현의 시간을 줄여준다.

Ex)인터페이스 정의

```java
package com.lec.ex01;

public interface InterfaceEx1 { // 접근제어자 인터페이스 "인터페이스명"
    public /* static final */ int CONSTANT_NUM = 10;
   //변수에 static과 final을 생략해도 인터페이스에선 해당 명령어가 있는것으로 간주한다
   //따라서 생략해도 무방

    public /* abstract */ void method1()
   //메소드 역시 abstract를 생략해도 있는것으로 간주한다.
```

```java
public interface InterfaceEx2 { // 다른 인터페이스
    public String CONSTANT_STRING = "Hello, World!";
    public String method2();
}
```

Ex)인터페이스를 구현한 클래스

```java
public class InterfaceClass implements InterfaceEx1, InterfaceEx2{
    //클래스는 인터페이스를 다중으로 구현할수있다.
    @Override
    public String method2() { //InterfaceEx1의 오버라이딩 메소드
        System.out.println("실제 구현은  implements한 클래스에서 해요. method2");
        return null;
    }
    @Override
    public void method1() { //InterfaceEx2의 오버라이딩 메소드
        System.out.println("실제 구현은  implements한 클래스에서 해요. method1");
    }
```

Ex)실행

```java
public class TestMain {
    public static void main(String[] args) {
        InterfaceClass i = new InterfaceClass();
        InterfaceEx1 x = new InterfaceClass();
        InterfaceEx2 y = new InterfaceEx1();

        i.method1(); //실행
        i.method2(); //실행
        System.out.println(i.CONSTANT_NUM + i.CONSTANT_STRING);//객체 변수로 static 호출은 지양
        System.out.println(InterfaceEx1.CONSTANT_NUM + i.CONSTANT_STRING);
        //static변수나 매서드는 보통 클래스명.으로 호출하는것을 지향한다.
        x.method1();  //실행
        //x.method2() //x객체는 두인터페이스를 구현했지만
                      //자료형이 InterfaceEx1이기때문에 InterfaceEx2의 메소드를 호출할수없다.
                      //명시적 형변환으로 호출가능
        if(x instanceof InterfaceClass) { //객체의 자료형변환시에는 instanceof 명령어로 클래스 상속관계를
            ((InterfaceClass)x).method2(); //확인하여 에러를 체크해야한다.
        }
        System.out.println(((InterfaceClass)x).CONSTANT_STRING);//변수 역시 명시적 형변환으로 호출
    }

```