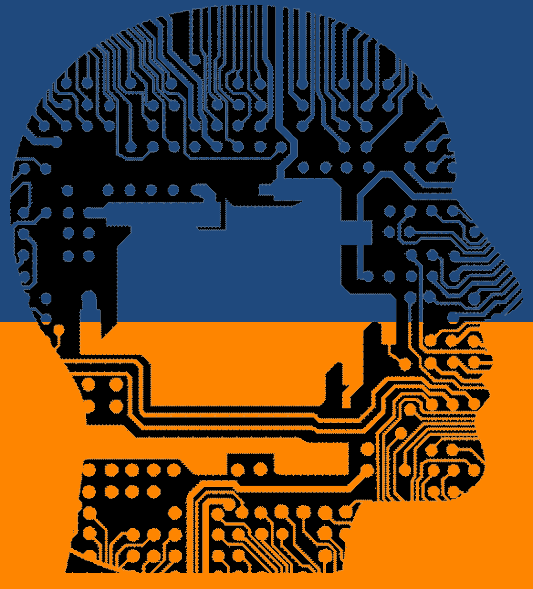
한국IT진흥부설 정보보호교육학원 아이섹

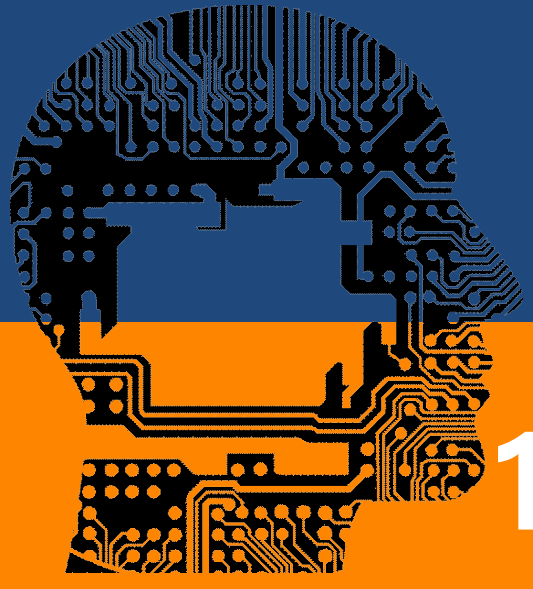


자바 프로그래밍 기초

인터페이스

By SoonGu Hong(Kokono)

한국IT진흥부설 정보보호교육학원 아이섹



파이썬 기초

1. 인터페이스(Interface)

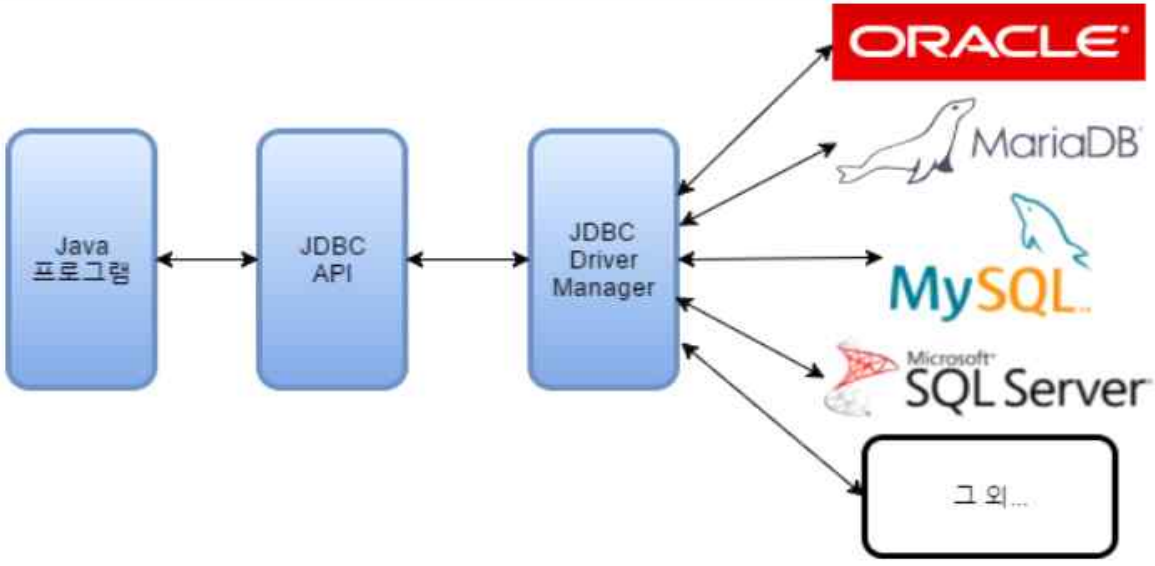
1-1-1. 인터페이스의 필요성

- 데이터베이스 관리시스템을 제어하는 자바 API에 대한 이야기로 인터페이스의 필요성을 말하려 합니다.
- 데이터베이스 회사는 여러 회사가 존재하고 각각의 특수 문법(방언)들을 가지고 있습니다.
- 그래서 과거에는 각 DB벤더회사들이 자기 회사의 DB연결을 할 수 있는 JAVA API를 각각 만들어서 배포했습니다.

1-1-2. 인터페이스의 필요성

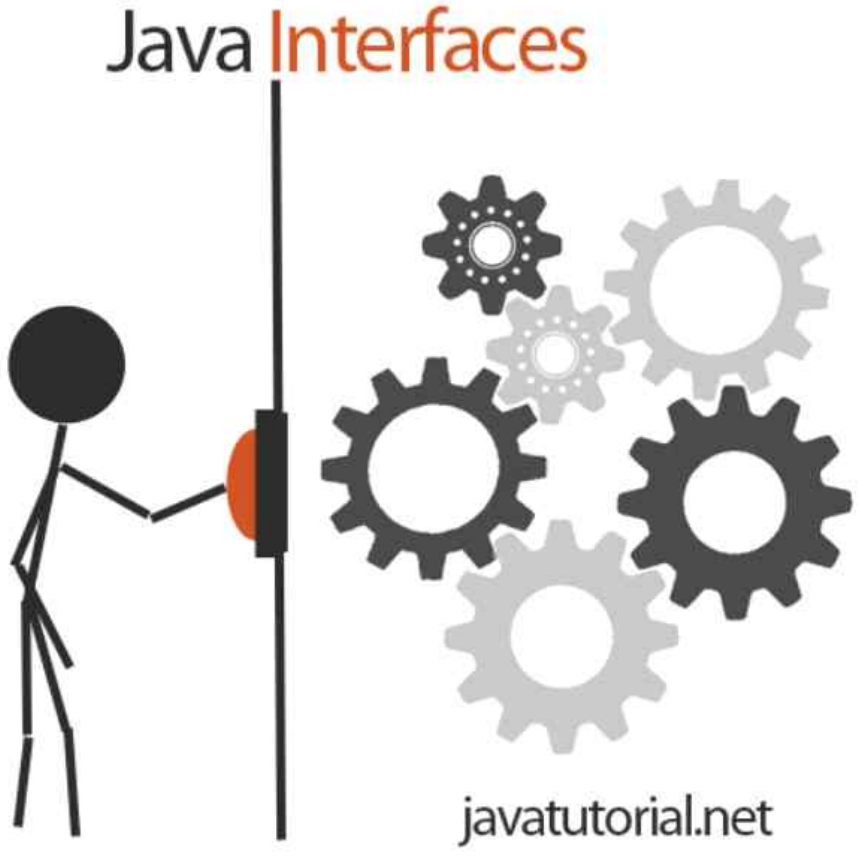
- 그 결과 ORACLE 데이터베이스를 이용하는 자바개발자들은 ORACLE이 만든 API를 학습하여 DB와 통신을 하였고, MYSQL 데이터베이스를 이용하는 개발자들은 MySQL진영에서 만든 API를 학습해서 사용했었습니다.
- 그 결과 사용하던 DB가 바뀔때마다 새로운 API를 학습하는 문제점이 발생했죠.

1-1-3. 인터페이스의 필요성

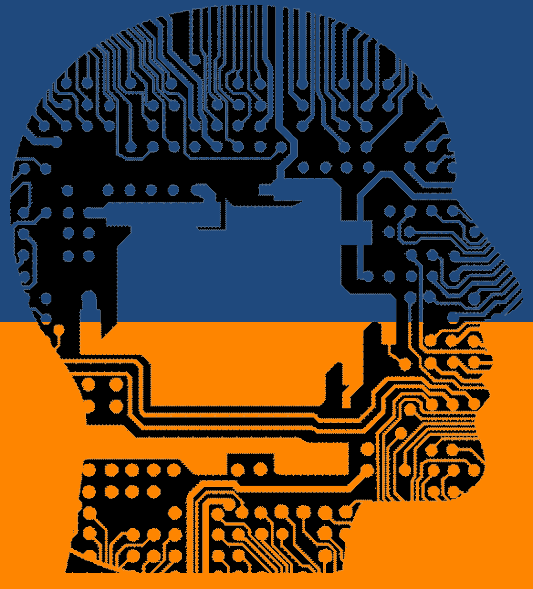


- 이런 번거로운 문제를 해결하기 위해 자바진영에서는 인터페이스를 활용하게 됩니다.
- 바로 JDBC라는 **인터페이스**를 자바진영에서 제시하고 각 데이터베이스 회사들에게 이 인터페이스대로 API를 만들어달라 요구합니다.
- 그 결과 자바개발자들은 **동일한 객체 타입, 메서드 이름을 사용**할 수 있게 되었고, JDBC만 학습하면 모든 DB를 다룰수 있게 되었습니다.

1-2. 결론



- 인터페이스는 **객체들의 표준 사용방법**을 제시합니다.
- 인터페이스는 객체들을 충실하게 **추상화, 캡슐화**할 수 있게 도와줍니다.



파이썬 기초

2. 인터페이스 사용법

2-1. 인터페이스 선언 방법

```
public interface Pet {

    //상수
    String kind = "애완동물";

    //추상 메서드
    Happy eat(Food food);

    //디폴트 메서드
    default void feelHappy() {}

    //정적 메서드
    static void play() {}
}
```

- 인터페이스는 선언시에 class자리에 interface라는 키워드를 넣어서 선언합니다.

- 인터페이스에는 인스턴스 필드를 선언할 수 없으며, 상수만 선언할 수 있습니다. static final을 생략해도 자동으로 붙어서 처리됩니다.

- 인터페이스에는 추상 메서드, 정적 메서드, 디폴트 메서드(자바8이후)를 선언할 수 있습니다.

2-2. 인터페이스 사용 방법 - implements

```
public class BullDog
    implements Pet {

    @Override
    Happy eat(Food food){
        //do something...
    }
}
```

- 인터페이스 자체로는 객체를 생성할 수 없습니다.

- 따라서 인터페이스에서 명세한 기능을 구체화할 클래스가 필요하며 implements라는 키워드를 사용하여 인터페이스를 명시합니다.

- 인터페이스를 구현한 클래스는 인터페이스의 추상메서드들을 모두 오버라이딩해야 합니다.

2-3. 다중 인터페이스 구현

```
public interface Huntable {
    void hunt(Target target);
}
```

```
public class BullDog
    implements Pet, Huntable {

    @Override
    Happy eat(Food food){
        //do something...
    }

    @Override
    void hunt(Target target) {
        //do something...
    }
}
```

- 인터페이스는 클래스간 상속과 달리 여러 개의 인터페이스를 다중 구현할 수 있습니다.

- implements 키워드 뒤에 구현할 인터페이스들을 콤마로 나열합니다.

- 이 때 나열된 인터페이스의 모든 추상메서드를 전부 오버라이딩 해야 합니다.

- 의미상으로 본다면 현재 불독은 애완동물의 기능과 사냥개의 기능을 모두 가진 객체가 되겠죠??

2-4-1. 클래스 상속과 인터페이스 구현

```
public abstract class Animal {
    private String name;
    private double weight;
    private int age;
}
```

```
public class BullDog
    extends Animal
    implements Pet, Huntable {

 //모든 추상메서드 오버라이딩


    .....
}
```

- 인터페이스는 필드를 가질 수 없다고 했습니다.

- 다만 여러 동물들이 공통된 속성이 존재한다면(예: 몸무게, 나이 등) 상속을 통해 클래스를 설계하는 것이 좋겠죠??

- 이럴 때 클래스 상속과 인터페이스 구현을 동시에 할 수 있습니다.

- 상속 키워드 extends를 먼저 써주셔야 합니다! (순서 주의!)

- 만약 상속 부모클래스가 추상 클래스라면 내부의 추상메서드를 반드시 오버라이딩해야 합니다.

2-4-2. 추상 클래스와 인터페이스의 차이

- 여기까지 공부를 잘 따라오셨으면 이제 의문점이 생깁니다.

- 대체 인터페이스는 추상클래스와 다른점이 무엇인가? 에 대한 내용이죠.

- 추상클래스는 인스턴스 필드, 정적 필드, 상수를 모두 가질 수 있고 인스턴스 메서드, 정적 메서드, 파이널 메서드, 추상 메서드도 모두 가질 수 있는데 얼핏 보면 인터페이스보다 뛰어난 것처럼 보입니다.

- 2개의 차이는 사용 목적에 있습니다. **추상 클래스**는 말 그대로 클래스입니다. **상속을 통한 새로운 클래스 확장**을 위한 의도를 가지고 설계하는 것이 추상 클래스입니다.

- **인터페이스**는 **기능을 명세**하여 **객체를 규격화**하는 것에 의의를 둡니다. 그래서 상속에서 막아둔 다중 구현을 인터페이스에서 열어두어 설계의 유연성을 제공하고 있는 것입니다.

감사합니다
THANK YOU