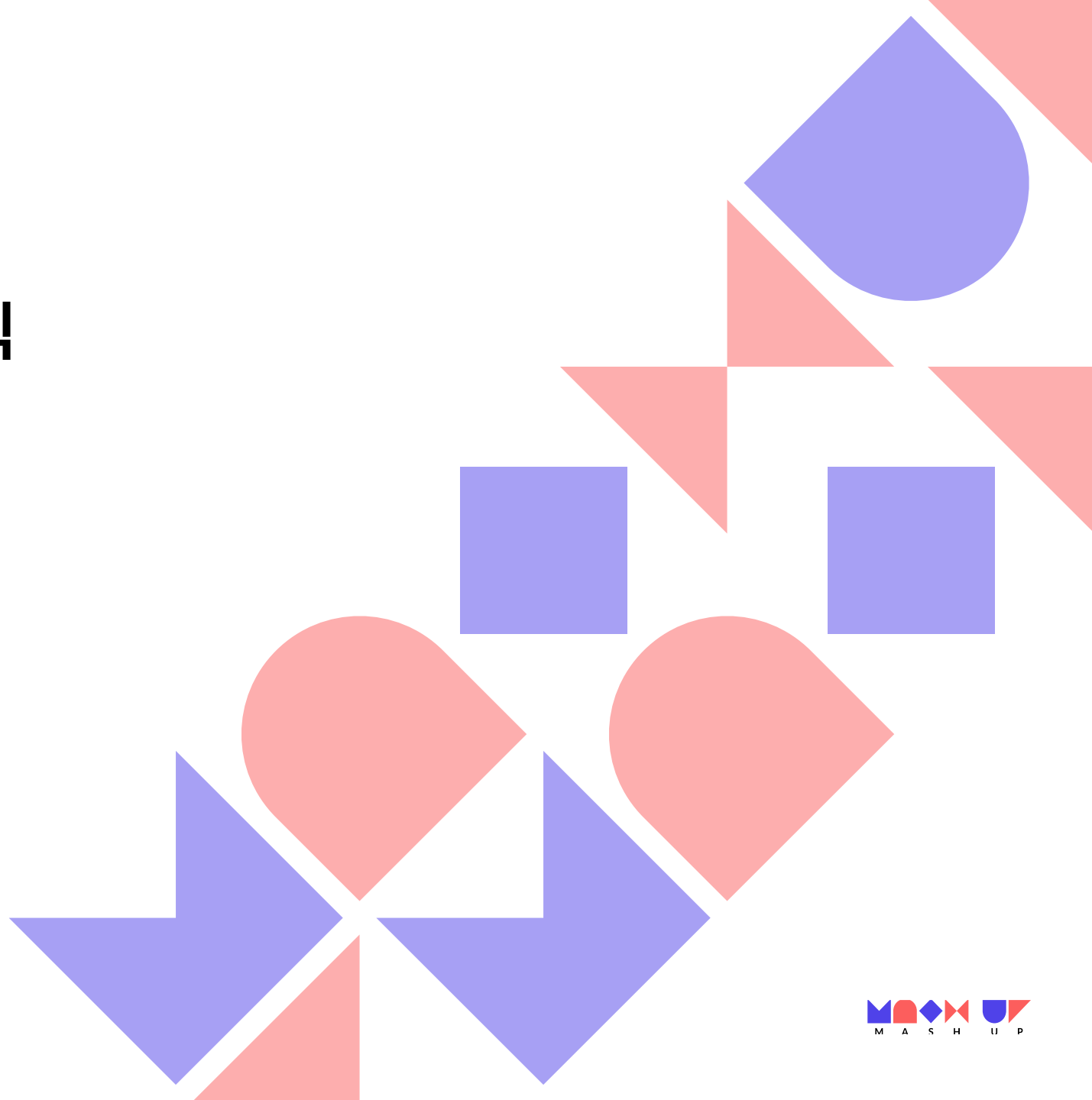
Back-End STUDY

SOLID - 객체지향 설계 원칙

2020. 05. 16

이동준, 김재현



MASH UP

SRP(단일 책임 원칙)

OCP(개방-폐쇄 원칙)

LSP(리스코프 치환 원칙)

DIP(의존 역전 원칙)

ISP(인터페이스 분리 원칙)

01

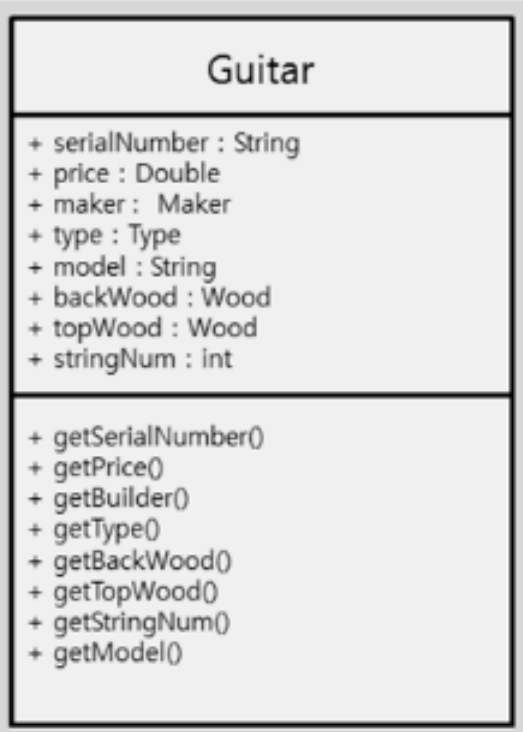
SRP

작성된 클래스는 하나의 기능만 가지며

클래스가 제공하는 모든 서비스는 그 하나의 책임(변화의 축: axis of change)을

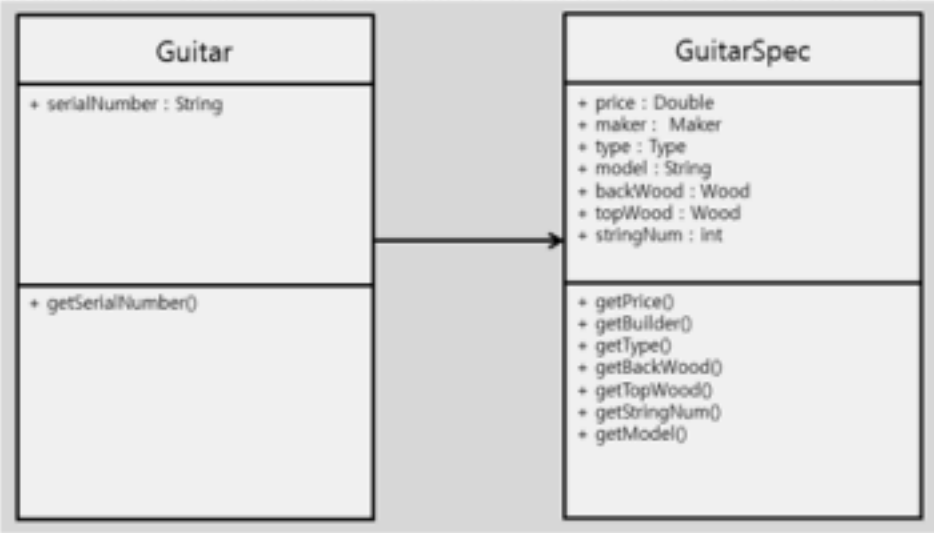
수행하는 데 집중되어 있어야 한다는 원칙.

1.SRP(단일 책임의 원칙) 적용 전




```
class Guitar (){

   public Guitar(String serialNumber, double price, Maker
                 maker, Type type, String model, Wood backWood,
                 Wood topWood, int stringNum){
     this.serivalNumber = serialNumber;
     this.price = price;
     this.maker = maker;
     this.type = type;
     this.model = model;
     this.backWood = backWood;
     this.topWood = topWood;
     this.stringNum = stringNum;

   }

  private String serialNumber;
  private double price;
  private Maker maker;
  private Type type;
  private String model;
  private Wood topWood;
  private Wood backWood;
  private int stringNum;

  ....
}
```

1.SRP(단일 책임의 원칙) 적용 후




```
class Guitar (){

   public Guitar(String serialNumber, GuitarSpec spec){
     this.serivalNumber = serialNumber;
     this.spec = spec;

   }

  private String serialNumber;
  private GuitarSpec spec;

   ....
}

class GuitarSpec(){
   double price;
   Maker maker;
   Type type;
   String model;

   ....
}
```

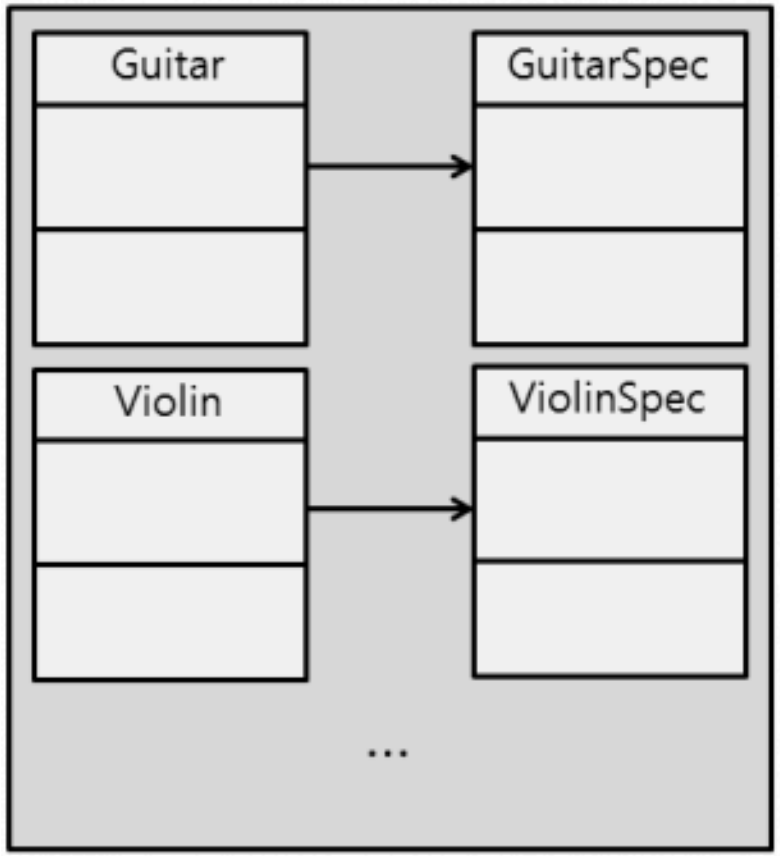
02

OCP

2. OCP

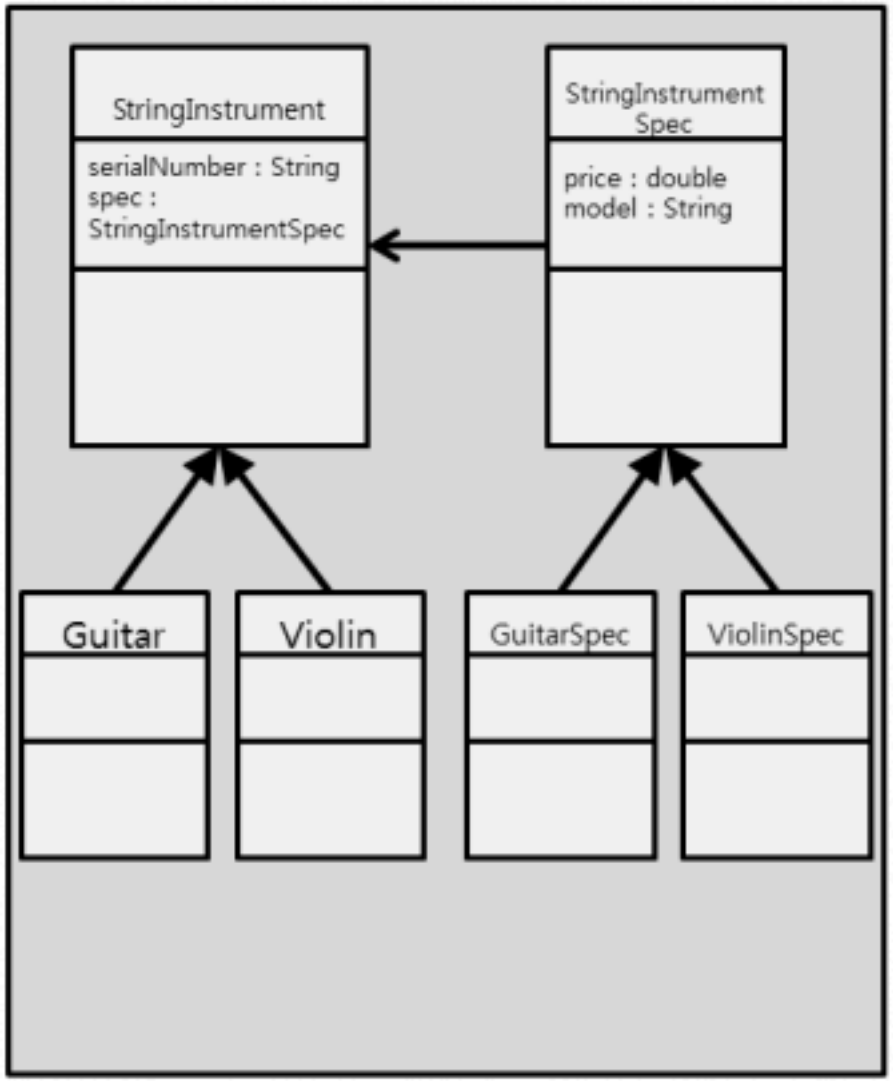
클래스(컴포넌트, 모듈, 함수)는 확장에는 열려있고, 변경에는 닫혀있어야 한다는 원리.

2. OCP적용 전




```
class Guitar (){

 public Guitar(String serialNumber, GuitarSpec spec){
     this.serivalNumber  = serialNumber;
     this.spec = spec;
   }
 private GuitarSpec spec;
}

class GuitarSpec(){
}

class Violin (){

 public Violin (String serialNumber, ViolinSpec spec){
     this.serivalNumber  = serialNumber;
     this.spec = spec;
   }
 private ViolinSpec spec;
}

class ViolinSpec(){
}
                                        ...
```

2. OCP적용 후




```
class Guitar extends StringInstrument(){

 public Guitar(String serialNumber, GuitarSpec spec){
     this.serivalNumber  = serialNumber;
     this.spec = spec;
   }
 private GuitarSpec spec;
}

class GuitarSpec extends StringInstrumentSpec(){
}

class Violin extends StringInstrument(){

 public Violin(String serialNumber, ViolinSpec spec){
     this.serivalNumber  = serialNumber;
     this.spec = spec;
   }
 private ViolinSpec spec;
}

class ViolinSpec extends StringInstrumentSpec(){
}
                                        ...
```

03

LSP

자식이 부모를 대체할 수 있어야한다.(자식 이상속받은 부모를 제대로 구현해야한다.)

3. LSP

```
void f(){
    LinkedList list = new LinkedList();
    // …
    modify(list);
}

void modify(LinkedList list){
    list.add(…);
    doSomethingWith(list);
}
```

3. LSP

```
void f(){
      Collection collection = new HashSet();
      //…
      modify(list);
}

Void modify(Collection collection){
      collection.add(…);
      doSomethingWith(collection);
}
```

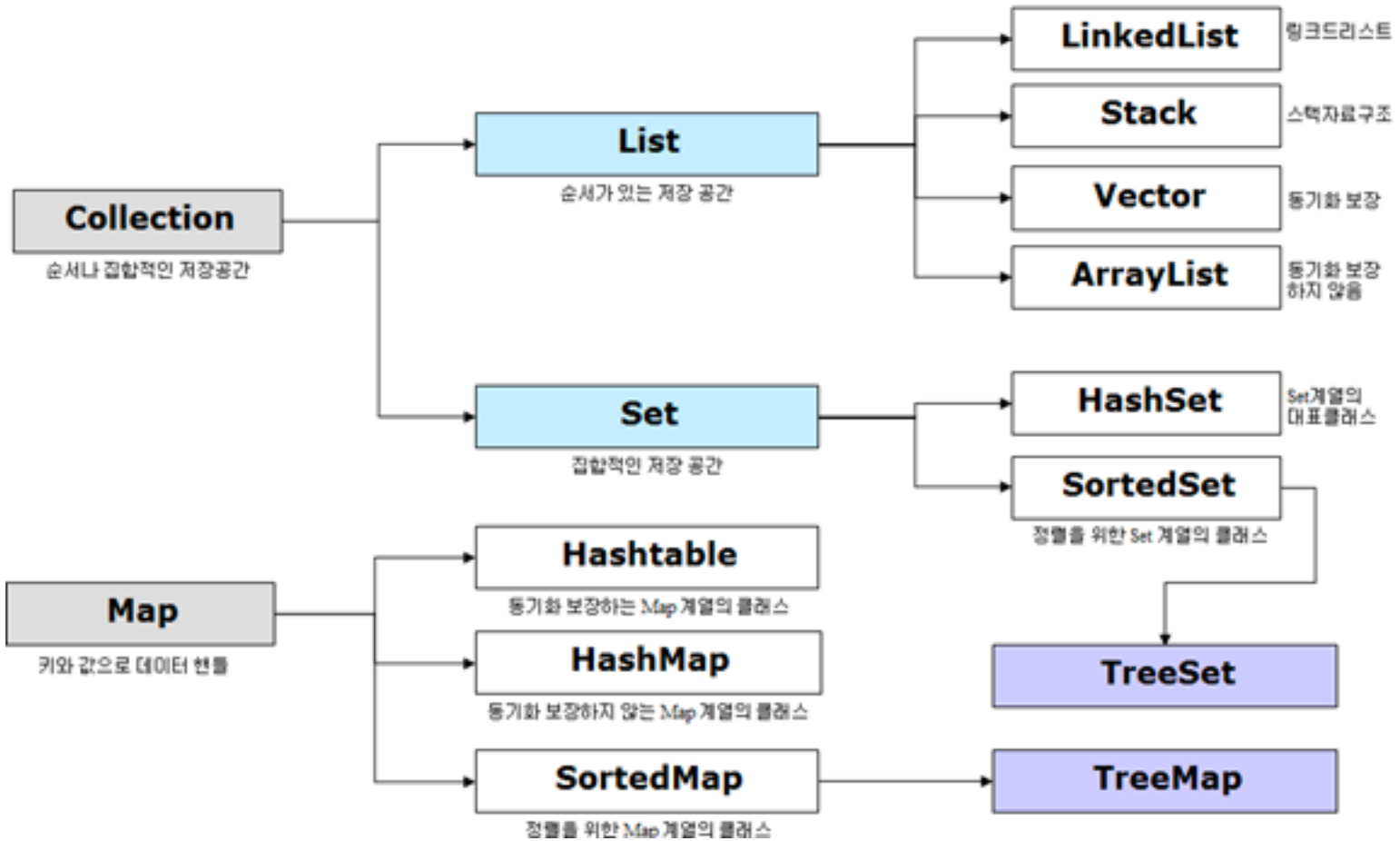
04

ISP

4. DIP

클래스는 자신이 사용하지 않는 인터페이스는 구현하지 말아야 한다는 원리

4. DIP


Collection
순서나 집합적인 저장공간
List
순서가 있는 저장 공간
LinkedList
링크드리스트
Stack
스택자료구조
Vector
동기화 보장
ArrayList
동기화 보장
하지 않음
Set
집합적인 저장 공간
HashSet
Set계열의
대표클래스
SortedSet
정렬을 위한 Set 계열의 클래스
Map
키와 값으로 데이터 핸들
Hashtable
동기화 보장하는 Map 계열의 클래스
HashMap
동기화 보장하지 않는 Map 계열의 클래스
SortedMap
정렬을 위한 Map 계열의 클래스
TreeSet
TreeMap

05

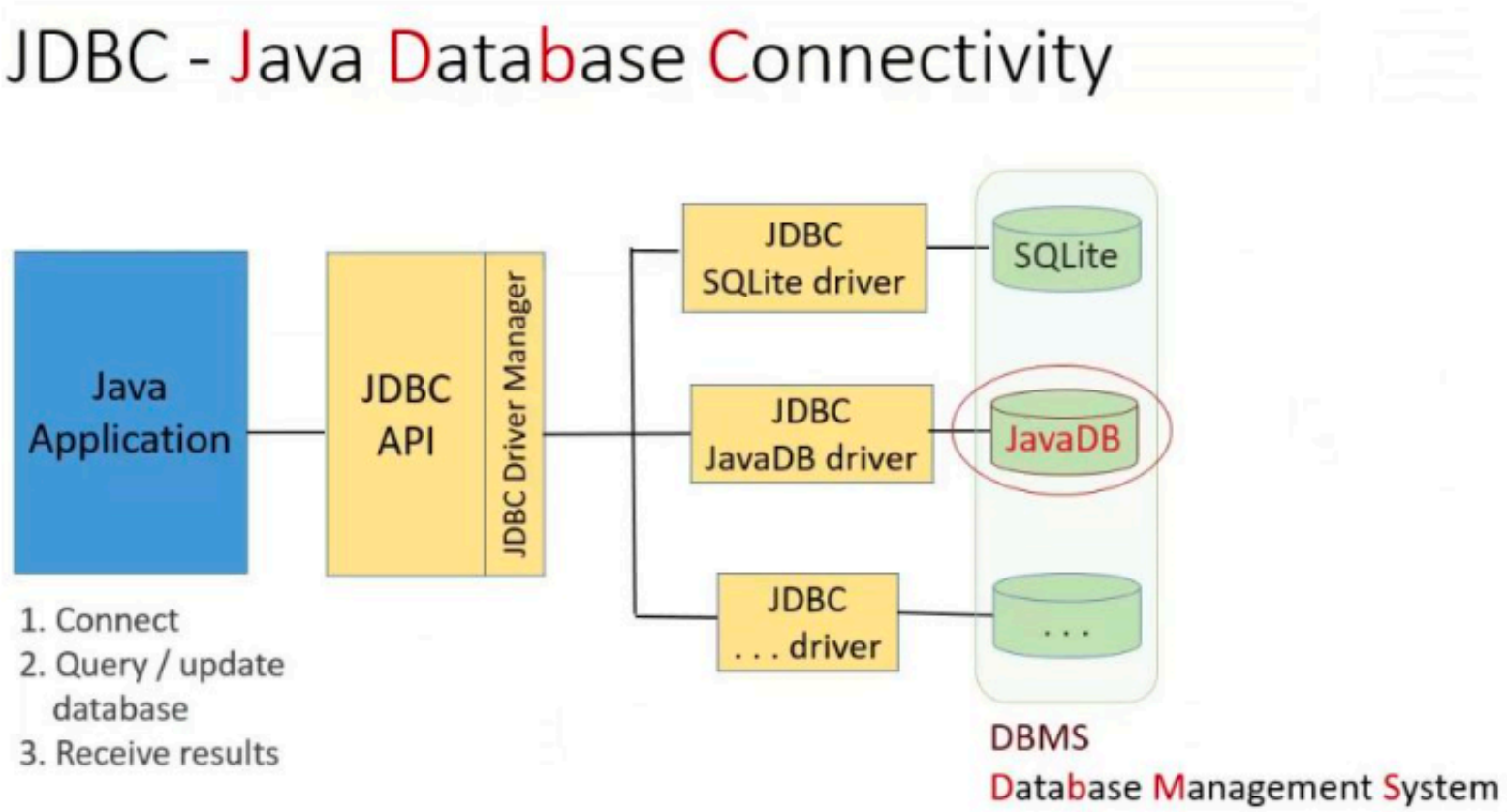
DIP

고수준 모듈이 저수준 모듈을 의존하는 것이 아니라
저수준 모듈이 고수준 모듈을 의존하는 것

고수준 모듈: 컨트롤러, 서비스, 도메인

저수준 모듈: 기술들의 구현체들

5. ISP


JDBC - Java Database Connectivity
Java Application
JDBC API
JDBC Driver Manager
JDBC SQLite driver
JDBC JavaDB driver
JDBC . . . driver
SQLite
JavaDB
. . .
1. Connect
2. Query / update database
3. Receive results
DBMS
Database Management System

THANKS FOR WATCHING

감사합니다

2020.05.16

이동준, 김재현