2023-03-27

SELECT → executeQuery() : ResultSet

DML

INSERT, DELETE, UPDATE → executeUpdate() : int

DML명령을 수행할 때는 return값을 사용하기도 한다.

조건에 맞는 데이터가 없어서 0을 return한다는 것은 error가 아니다.

DDL

CREATE TABLE, DROP TABLE, ALTER TABLE → executeUpdate() : int

DDL명령을 수행할 때는 return 값을 사용하는 일이 거의 없다.

MVC

Model View Controller 의 약자로 어플리케이션을 세가지의 역할로 구분한 개발 방법론

비즈니스 처리 로직과 사용자 인터페이스 요소들을 분리시켰기 때문에 서로 영향없이 개발이 수월하고, 유지보수를 독립적으로 수행할 수 있는 장점이 있다.

- Model - 어플리케이션이 “무엇”을 할 것인지 정의한다.
  - Sevice Model : DAO
  - Domain Model :
    - VO : 값 그 자체를 표현하는 객체. 값을 갖는 순수한 도메인
    - DTO : 계층(Layer)간 데이터 교환을 위해 사용하는 객체.
    - Entity : DB 테이블과 매핑하는 객체
- Controller는 모델이 “어떻게” 처리할 지 알려주는 역할을 하고 Model과 Ciew사이에서 연결 역할을 하면서 이 두 그룹의 결합도를 낮춰 확장성을 증가시키도록 설계
- View는 사용자 인터페이스를 구현

제네릭 타입

결정되지 않은 타입을 파라미터로 가지는 클래스와 인터페이스

선언부에’<>’ 부호가 붙고 그 사이에 타입 파라미터들이 위치

```
public class 클래스명<A, B, ...> { ... }
public interface 인터페이스명<A, B, ...> { ... }
```

와일드카드 타입 파라미터

제네릭 타입을 매개값이나 리턴 타입으로 사용할 때 범위에 있는 모든 타입으로 대체할 수 있는 타입 파라미터로 **?** 기호로 표시한다.

```
ArrayList<Friend>
ArrayList<String>
ArrayList<StudentDto>
ArrayList<Integer>

// Student와 Student를 상속한 자식만 올 수 있다.
리턴타입 메서드명(제네릭타입<? extends Student> 변수) { ... }

// Worker와 Worker의 부모만 올 수 있다.
리턴타입 메서드명(제네릭타입<? super Worker> 변수) { ... }

// 어떤 타입이든 올 수 있다.
리턴타입 메서드명(제네릭타입<?> 변수) { ... }
```

제네릭 메서드

타입 파라미터를 가지고 있는 메서드이며 타입파라미너틑 메서드 선언부에 정의됨

리턴 타입 앞에 <>기호와 타입 파라미터를 정의한 후에 리턴타입과 매개변수 타입에서 사용

```
public <A, B, ...> 리턴타입 메서드명(매개변수, ...) { ... }
// < ... > : 타입파라미터를 전달받는 식별자들을 나열한다.
```

타입 파라미터T는 메서드 호출시 전달되는 아규먼트의 타입에 따라 컴파일 과정에서 구체적인 타입으로 대체됨

```
public <T> Box<T> boxing(T t) { ... }
1) Box<Integer> box1 = boxing(100);
2) Box<String> box2 = boxing("택배입니다");
```

제한된 타입 파라미터

모든 타입으로 대체할 수 없고, 특정 타입과 자식 또는 구현 관계에 있는 타입만 대체할 수 있는 타입 파라미터로 extends절을 사용. (상위 타입은 클래스뿐만 아니라 인터페이스도 가능)

```
public <T extends 상위타입> 리턴타입 메서드(매개변수, ...) { ... }
```

```
public <T extends Number> boolean compare(T t1, T t2) {
  double v1 = t1.doubleValue(); // Number의 doubleValue() 메서드 사용
  double v2 = t2.doubleValue();// Number의 doubleValue() 메서드 사용
return (v1 == v2);
}
```

멀티스레드 프로그래밍

- 멀티 프로세스와 멀티 스레드
  - 프로세스 : 실행 중인 프로그램
  - 멀티 태스킹 : 두 가지 이상의 작업을 동시에 처리하는 것
  - 스레드 : 프로세스 내에서 코드의 실행 흐름
  - 멀티 스레드 : 두 개의 코드 실행 흐름. 두 가지 이상의 작업을 처리

    프로그램 내부에서의 멀티 태스킹
  - 멀티 프로세스 : 실행 중인 프로그램이 2개 이상

    프로그램 단위의 멀티 태스킹