Lombok 정리

개념
- Java에서 자주 반복되는 코드(Getter, Setter, 생성자 등등)들을 어노테이션을 통해 자동으로 생성해주는 라이브러리

Lombok 어노테이션은 한마디로 말해서, 지저분해지는 코드들은 전부 간편하게 자동 생성해준다.
기존에 썼던 것들은 간편화 해주는 것이기 때문에 lombok을 사용해도 어떻게 사용되는지 알아야 한다.

@Getter, @Setter

⚠️ 사용하기 전 코드

```
public class Post {
    private String title;
    private String content;
    private String author;

    public void setTitle(String title) {
        this.title = title;
    }

    public void setContent(String content) {
        this.content = content;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getTitle() {
        return title;
    }

    public String getContent() {
        return content;
    }

    public String getAuthor() {
```

```
        return author;
    }
}
```

✅ 사용하고 난 후 코드

```
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class Post {
    private String title;
    private String content;
    private String author;
}
```

위와 같이 @Getter, @Setter 어노테이션들을 붙여주면 명클래스 멤버 변수들의 getter, setter 메소드를 이용이 가능하다.
(제일 유용하고 제일 많이 쓰이는 어노테이션이다.)

@NoArgsConstructor

```
public Post() {}
```

@NoArgsConstructor 어노테이션을 붙이면 위와같은 클래스의 기본 생성자를 생성해준다.

@AllArgsConstucto

클래스의 모든 멤버변수를 받는 생성자를 만들어준다.

@RequiredArgsConstructor

클래스의 멤버변수중 ***final*** 키워드, 혹은 lombok 어노테이션인 ***@NonNull*** 이

붙은 멤버 변수만을 받는 생성자를 만들어 준다.

@Builder

빌더패턴을 적용 시킨 방법으로 객체를 생성 할 수 있다. 빌더 패턴을 적용 시킨 객체 생성 방법은 다음과 같다.

```
-----------------------------------------------------
post = Post.builder()
        .title(title)
        .content(content)
        .author(author)
        .build();
-----------------------------------------------------
```

@ToString

클래스의 멤버 변수들에 toString을 적용시켜 변수들을 출력 할 수 있게 해 준다.

```
-----------------------------------------------------
// 위의 빌더 패턴으로 생성한 Post 객체가 있다고 가정.
// @ToString annotation이 있으면

System.out.println(post); // 결과 - > Post(title = title, content =
content, author = author)
-----------------------------------------------------
```

⚠️ ToString 사용 시 멤버 변수 중 **객체타입**이 있고, 순환 참조가 있다면 **무한루프**가 발생한다.
(A class의 멤버 변수로 B class를 갖고 있고, B class의 멤버 변수로 A class를 갖고 있다면 toString 무한 반복이 발생한다.)
이럴 때는 exclude 키워드를 이용해서 명시적으로 해당 필드를 제외해야 한다.
-> @ToString(exclude="classA")

@EqualsAndHashCode

equals, hashCode를 자동 생성해주는 어노테이션이다.

- equals : 두 객체의 내용이 같은지, 동등성(equality)를 비교하는 연산자
- hashCode : 두 객체가 같은 객체인지. 동일성(identity)를 비교하는 연산자

자바 bean에서 동등성 비교를 위해 equals와 hashcode 메소드를 오버라이딩

해서 사용하는데, 이 어노테이션을 사용하면 자동으로 메서도가 생성된다.

@Data

위에 언급 했던 모든 어노테이션을 포함하는 어노테이션이다.