DI(Dependency Injection)

- 어떤 객체가 사용하는 의존 객체를 직접 만들어 사용하는게 아니라, 외부에서 주입 받아 사용하는 방법
- 프로그램의 소스를 변경하지 않고 부분적인 전환(교체)를 쉽게 만드는 것
- 인터페이스구현 방식과 ApplicationContext 이용

실습1 클래스를 교체

```
public class SamsungTV {

    public void powerOn() {
        System.out.println("삼성 TV--전원 on");
    }
    public void powerOff() {
        System.out.println("삼성 TV--전원 off");
    }
    public void volumeUp() {
        System.out.println("삼성 TV--볼륨 up");
    }
    public void volumeDown() {
        System.out.println("삼성 TV--볼륨 down");
    }
}

public class LgTV {

    public void powerOn() {
        System.out.println("LG TV--전원 on");
    }
    public void powerOff() {
        System.out.println("LG TV--전원 off");
    }
    public void soundUp() {
        System.out.println("LG TV--볼륨 up");
    }
    public void soundDown() {
        System.out.println("LG TV--볼륨 down");
    }
}

@SpringBootTest
public class DITest {

    @Test
    public void test() {

        SamsungTV tv = new SamsungTV();    //LgTV로 교체해보세요

        tv.powerOn();
        tv.volumeUp();
        tv.volumeDown();
```

```
        tv.powerOff();

    }
}
```

실습2 인터페이스를 이용한 클래스 구현

- 기본적으로 구현해야 할 메소드에 대한 규격을 지정하고 강제적으로 구현

인터페이스 선언

- Refactor -> Extract Interface... -> 인터페이스 이름 입력하고 members는 모두 선택

```
public interface TV {
    void powerOn();
    void powerOff();
    void volumeUp();
    void volumeDown();
}
```

TV 인터페이스 상속받도록 SamsungTV, LgTV 수정

```
public class SamsungTV implements TV { ...  }
public class LgTV implements TV { ...  }
```

실습3 DI 실습

`@Component` 애노테이션을 이용하여 applicationContext에 bean 등록

```
@Component
public class SamsungTV
```

`@Autowired` 애노테이션을 이용하여 객체 주입

```
public class DITest {

    @Autowired TV tv;

    @Test
    public void test() {
        tv.powerOn();
    }
}
```

```
  @Autowired  ApplicationContext context;

  @Test
  public void test() {
    TV tv = context.getBean(TV.class);
```

실습4 의존관계에 있는 객체 주입

Speaker 인터페이스 선언

```
public interface Speaker {
    void volumeUp();
    void volumeDown();
}
```

SonySpeaker 구현 클래스

```
public class SonySpeaker implements Speaker {

    public SonySpeaker(){       System.out.println("Sony Speaker 생성");     }

    @Override
    public void volumeUp(){
        System.out.println("Sony Speaker 소리 올림");
    }

    @Override
    public void volumeDown(){
        System.out.println("Sony Speaker 소리 내림");
    }
}
```

AppleSpeaker 구현 클래스

```
public class AppleSpeaker implements Speaker {

    public AppleSpeaker() {
        System.out.println("Apple Speaker 생성");
    }

    @Override
    public void volumeUp() {
        System.out.println("Apple Speaker 소리 올림");
    }

    @Override
```

```
    public void volumeDown() {
        System.out.println("Apple Speaker 소리 내림");
    }
}
```

DI

1. 필드주입

```
    @Autowired
    private Speaker speaker;
```

2. setter 주입

```
    @Autowired
    public void setSpeaker(Speaker speaker) {
      this.speaker = speaker;
    }
```

3. **생성자 주입**

```
    private Speaker speaker;

  //  @Autowired    Unnecessary `@Autowired` annotation
  [JAVA_AUTOWIRED_CONSTRUCTOR]
    public void setSpeaker(Speaker speaker) {
      this.speaker = speaker;
    }
```

DI : 롬복

1. @Setter 애노테이션

```
    @Setter(onMethod_ = {@Autowired})
    private Speaker speaker;
```

2. **`final` 과 `@RequiredArgsConstructor` 애노테이션**

```
  @RequiredArgsConstructor
  public class LGTV implements TV {

    private final Speaker speaker;
```

DI : java config

- `@Configuration`이 붙으면, AnnotationConfigApplicationContext 에 파라미터로 넘긴 값은 스프링 빈으로 등록된다.
- @Bean만 사용해도 스프링 빈으로 등록되지만, 싱글톤을 보장하지 않는다.

```
@Configuration
public class JavaConfig {
    @Bean
    Speaker speaker() {
        return new AppleSpeaker();
    }

    @Bean
    TV tv() {
        return new LgTV(speaker());
    }
}
```

실습5 di test : Restaurant 클래스, Chift 클래스

```
public class Restaurant {
    private Chef chef;
}
```

```
public class Chef {

}
```

```
public class DITest {

    @Autowired
    ApplicationContext context;

    @Autowired TV tv;



    @Test
    public void test() {

    @Autowired Restaurant restaurant;

  @Test
```

```
    public void Restauranttest() {
        restaurant.getChef();
    }
}
```