HOL : Spring DI

Task 1. Non-DI Java Project

1. Project 유형 : Java Project
2. Project Name : BeforeSpring
3. Package Name : com.example

4. Calculator Class

```
com.example.Calculator.java
package com.example;

public class Calculator {
    public void addAction(int a, int b){
        System.out.println("Called addAction()");
        System.out.printf("%d + %d = %d\n", a, b, (a + b));
    }
    public void subAction(int a, int b){
        System.out.println("Called subAction()");
        System.out.printf("%d - %d = %d\n", a, b, (a - b));
    }
    public void multiAction(int a, int b){
        System.out.println("Called multiAction()");
        System.out.printf("%d x %d = %d\n", a, b, (a * b));
    }
    public void divAction(int a, int b){
        System.out.println("Called divAction()");
        System.out.printf("%d / %d = %d\n", a, b, (a / b));
    }
}
```

5. MyCalculator Class

```
com.example.MyCalculator.java
package com.example;

public class MyCalculator {
    private Calculator calculator;
    private int firstNum;
    private int secondNum;

    public void setFirstNum(int firstNum) {
        this.firstNum = firstNum;
    }
    public void setSecondNum(int secondNum) {
        this.secondNum = secondNum;
    }
    public void setCalculator(Calculator calculator){
        this.calculator = calculator;
    }

    public void add(){
        this.calculator.addAction(firstNum, secondNum);
    }
    public void sub(){
        this.calculator.subAction(firstNum, secondNum);
    }
    public void multi(){
        this.calculator.multiAction(firstNum, secondNum);
    }
    public void div(){
        this.calculator.divAction(firstNum, secondNum);
    }
}
```

6. MainClass Class

```
    com.example.MainClass
    package com.example;

    public class MainClass {
        public static void main(String[] args) {
            MyCalculator myCalculator = new MyCalculator();
            myCalculator.setCalculator(new Calculator());

            myCalculator.setFirstNum(10);
            myCalculator.setSecondNum(2);

            myCalculator.add();
            myCalculator.sub();
            myCalculator.multi();
            myCalculator.div();
        }
    }
```

7. Result

```
    Called addAction()
    10 + 2 = 12
    Called subAction()
    10 - 2 = 8
    Called multiAction()
    10 x 2 = 20
    Called divAction()
    10 / 2 = 5
```

Task 2. DI Demo in Spring

1. New > Java Project
    1)Project Name : StartSpring
    2)JRE : Use default JRE 'jdk-11.0.12' and workspace compiler preferences
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

2. Create package to src : com.example

3. Copy MyCalculator.java, Calculator.java from BeforeSpring project to StartSpring's package

4. Create class : com.example.MainClass.java

```
    package com.example;

    public class MainClass {
        public static void main(String[] args) {

        }
    }
```

5. Java Project를 Spring Project로 변환
    1)StartSpring Project > right-click > Configure > Convert to Maven Project
        -Project : /StartSpring
        -Group Id : StartSpring
        -Artifact Id : StartSpring
        -version : 0.0.1-SNAPSHOT
        -Packaging : jar
        -Finish
        -Package Explorer에서 보이는 Project icon에 Maven의 'M'자가 보임.

    2)StartSpring Project > right-click > Spring > Add Spring Project Nature
        -Package Explorer에서 보이는 Project icon에 'M'자와 Spring의 'S'가 보임.

3)pom.xml file에 Spring Context Dependency 추가하기
-https://mvnrepository.com에서 'spring context'로 검색
-[Spring Context] click
-현재 Spring 5.x의 현재 version인 5.3.10 click
-Copy하여 pom.xml에 paste

```
  <version>0.0.1-SNAPSHOT</version>
   <dependencies>   <--- dependencies element 추가
      <dependency>   <---여기에 paste
         <groupId>org.springframework</groupId>
         <artifactId>spring-context</artifactId>
         <version>5.3.10</version>
      </dependency>
   </dependencies>
```

4)pom.xml Save

5)pom.xml > right-click > Run As > Maven install
[INFO] BUILD SUCCESS 확인

6. config folder 생성
1)StartSpring project > right-click > New > Source Folder
-Folder name : config
-Finish

7. Bean Configuration XML 작성
1)config > right-click > New > Other > Spring > Spring Bean Configuration File > Next
2)Name : applicationContext.xml > Finish

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans.xsd">

   <bean id="calculator" class="com.example.Calculator" />

   <bean id="myCalculator" class="com.example.MyCalculator">
      <property name="calculator">
         <ref bean="calculator" />
      </property>
      <property name="firstNum" value="10" />
      <property name="secondNum" value="2" />
   </bean>
</beans>
```

8. MainClass.java

```
package com.example;

import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

public class MainClass {
   public static void main(String[] args) {
      String configFile = "classpath:applicationContext.xml";
      AbstractApplicationContext ctx = new GenericXmlApplicationContext(configFile);
      MyCalculator myCalculator = ctx.getBean("myCalculator", MyCalculator.class);

      myCalculator.add();
      myCalculator.sub();
      myCalculator.multi();
      myCalculator.div();

      ctx.close();
   }
```

```
    }
```

9. Result

```
    BeforeSpring과 같음.
    Called addAction()
    10 + 2 = 12
    Called subAction()
    10 - 2 = 8
    Called multiAction()
    10 x 2 = 20
    Called divAction()
    10 / 2 = 5
```

Task 3. 간단한 DI Project

1. In Package Explorer > right-click > New > Java Project
    1)Project name : DIDemo
    2)JRE : Use default JRE 'jdk-11.0.12' and workspace compiler preferences
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

2. src > right-click > New > Package
    1)Package name : com.example
    2)Finish

3. Interface 작성
    1)com.example > right-click > New > Interface
    2)Name : Printer

    3)Printer.java

```
        package com.example;

        public interface Printer{
            void print(String message);
        }
```

4. POJO class 작성
    1)com.example > right-click > New > Class
    2)Name : Hello
    3)Finish
    4)Hello.java

```
        package com.example;

        public class Hello{
            private String name;
            private Printer printer;

            public Hello(){}

            public void setName(String name){
                this.name = name;
            }

            public void setPrinter(Printer printer){
                this.printer = printer;
            }

            public String sayHello(){
                return "Hello " + name;
            }
            public void print(){
                this.printer.print(sayHello());
            }
        }
```

5. Printer interface의 child class 작성하기

1)com.example > right-click > New > Class
- -Name : StringPrinter
- -Interfaces : com.example.Printer
- -Finish

2)StringPrinter.java

```
package com.example;

public class StringPrinter implements Printer{
    private StringBuffer buffer = new StringBuffer();

    @Override
    public void print(String message){
        this.buffer.append(message);
    }

    public String toString(){
        return this.buffer.toString();
    }
}
```

3)com.example > right-click > New > Class
- -Name : ConsolePrinter
- -Interface : com.example.Printer
- -Finish

4)ConsolePrinter.java

```
package com.example;

public class ConsolePrinter implements Printer{

    @Override
    public void print(String message){
        System.out.println(message);
    }
}
```

6. Java Project를 Spring Project로 변환

1)DIDemo Project > right-click > Configure > Convert to Maven Project
- -Project : /DIDemo
- -Group Id : DIDemo
- -Artifact Id : DIDemo
- -version : 0.0.1-SNAPSHOT
- -Packaging : jar
- -Finish

2)DIDemo Project > right-click > Spring > Add Spring Project Nature

3)pom.xml file에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.10</version>
    </dependency>
</dependencies>
```

4)pom.xml > right-click > Run As > Maven install
[INFO] BUILD SUCCESS 확인

7. config folder 생성
1)StartSpring project > right-click > New > Source Folder
-Folder name : config
-Finish

8. Bean Configuration XML 작성
1)config > right-click > New > Other > Spring > Spring Bean Configuration File > Next
2)File name : beans.xml
3)Finish

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="hello" class="com.example.Hello">
        <property name="name" value="Spring" />
        <property name="printer" ref="printer" />
    </bean>
    <bean id="printer" class="com.example.StringPrinter" />
    <bean id="consolePrinter" class="com.example.ConsolePrinter" />

</beans>
```

9. Beans Graph 사용하기
1)Window menu > Show View > Other > Spring > Spring Explorer > Open
2)Spring Explorer
-DIDemo > Beans > beans.xml > right-click > Open Beans Graph

10. DI Test class 작성
1)src/com.example > right-click > New > Package
-Name : com.example.test
-Finish

2)/src/com.example.test > New > Class
-Name : HelloBeanTest.java

```
package com.example.test;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

import com.example.Hello;
import com.example.Printer;

public class HelloBeanTest {
    public static void main(String [] args){
        //1. IoC Container 생성
        ApplicationContext context =
                new GenericXmlApplicationContext("classpath:beans.xml");

        //2. Hello Beans 가져오기
        Hello hello = (Hello)context.getBean("hello");
        System.out.println(hello.sayHello());
        hello.print();

        //3. SpringPrinter 가져오기
        Printer printer = (Printer)context.getBean("printer");
        System.out.println(printer.toString());
```

```java
            Hello hello2 = context.getBean("hello", Hello.class);
            hello2.print();

            System.out.println(hello == hello2);  //Singleton Pattern
        }
    }
```

11. Result

```
Hello Spring
Hello Spring
true
```

Task 4. JUnit을 사용한 DI test class 작성하기
1. JUnit을 사용한 DI test class(HelloBeanJunitTest.java) 작성
    1)pom.xml에 아래 코드 붙여넣기

```xml
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
</dependency>
```

    2)pom.xml > right-click > Run As > Maven install
        [INFO] BUILD SUCCESS 확인

    3)src/com.example.test > New > Class
        -Name : HelloBeanJUnitTest.java

```java
package com.example.test;

import org.junit.Before;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

import com.example.Hello;
import com.example.Printer;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertSame;

public class HelloBeanJUnitTest {
    private ApplicationContext context;

    @Before
    public void init(){
        //항상 먼저 ApplicationContext를 생성해야 하기 때문에
        //1. IoC Container 생성
        context = new GenericXmlApplicationContext("classpath:beans.xml");
    }

    @Test
    public void test1(){
        //2. Hello Beans 가져오기
        Hello hello = (Hello)context.getBean("hello");
        assertEquals("Hello Spring", hello.sayHello());
        hello.print();

        //3. SpringPrinter 가져오기
        Printer printer = (Printer)context.getBean("printer");
        assertEquals("Hello Spring", printer.toString());
    }
```

```java
        @Test
        public void test2(){
            Hello hello = (Hello)context.getBean("hello");

            Hello hello2 = context.getBean("hello", Hello.class);
            assertSame(hello, hello2);
        }
    }
```

2. @Before에 mouse를 올려놓으면 Fix project setup... click
    1)Add archive 'junit-4.12.jar ... > OK
        -import org.junit...에 mouse를 올려놓으면 Fix project setup... click
        -Add JUnit 4 library to the build path > OK

3. right-click > Run As > JUnit Test
    1)결과 -> JUnit View에 초록색 bar
    2)만일, test1() method를 jUnit에서 제외하고 싶을 때에는 @Test 옆에 @Ignore를 선언한다.

```java
import import org.junit.Ignore;
...
@Test @Ignore
public void test1(){
...
```

    3)right-click > Run As > Junit Test
        -JUnit Test 목록에서 test1()는 실행되지 않는다.

Task 5. Spring TestContext Framework
1. Spring-Test library 설치
    1)http://mvnrepository.com에서 'spring test'로 검색
    2)검색 결과 목록에서 'Spring TestContext Framework' Click
    3)version 목록에서 5.3.10 Click

2. dependency 복사해서 pom.xml에 붙여넣기

```xml
<!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>5.3.10</version>
    <scope>test</scope>
</dependency>
```

3. pom.xml > right-click > Maven Install
    [INFO] BUILD SUCCESS

4. Spring-Test를 사용할 DI test class-HelloBeanJunitSpringTest.java 작성하기
    1)src/com.example.test > New > Class
    2)Name : HelloBeanJunitSpringTest

```java
package com.example.test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertSame;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.test.context.ContextConfiguration;
```

```
      import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

      import com.example.Hello;
      import com.example.Printer;

      @RunWith(SpringJUnit4ClassRunner.class)
      //JUnit 4.x에서 사용
      @ContextConfiguration(locations="classpath:beans.xml")
      public class HelloBeanJunitSpringTest {
        @Autowired
        ApplicationContext context;

        @Test
        public void test1(){
          Hello hello = (Hello)context.getBean("hello");
          assertEquals("Hello Spring", hello.sayHello());
          hello.print();

          Printer printer = (Printer)context.getBean("printer");
          assertEquals("Hello Spring", printer.toString());
        }

        @Test
        public void test2(){
          Hello hello = (Hello)context.getBean("hello");

          Hello hello2 = context.getBean("hello", Hello.class);
          assertSame(hello, hello2);
        }
      }
```

    -right-click > Run As > JUnit Test
    -결과 -> JUnit View에 초록색 bar

  4)만일 해당 객체를 찾을 수 없다는 오류가 계속 발생하면
    -해당 Project > right-click > Build Path > Configure Build Path > Libraries tab
    -spring-test-5.3.10.jar 선택 후 [Remove] 로 삭제
    -Classpath 선택
    -[Add External JARs...] Click
    -Local M2 Repository(e.g C:\Users\instructor\.m2\repository\org\springframework\spring-test\5.3.10\spring-test-5.3.10.jar)에서 직접 jar(spring-test-5.3.10.jar)를 선택할 것
    -[Order and Export] tab에서 spring-test-5.3.10.jar 선택 후 [Up] button을 클릭
    -해당 DIDemo/src 바로 아래까지 올리고 [Apply and Close] Click

Task 6. Java Annotation을 이용하여 setter를 이용한 의존주입하기 실습

1. In Package Explorer > right-click > New > Java Project
  1)Project name : DIDemo1
  2)JRE
    -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
  3)Uncheck [Create module-info.java file]
  4)Next
  5)Finish

2. src > right-click > New > Package
  1)Package name : com.example
  2)Finish

3. POJO class 작성
  1)com.example > right-click > New > Class
  2)Name : Hello

```
    package com.example;

    public class Hello {
        private String name;
        private Printer printer;

        public Hello(){}

        public void setName(String name){
            this.name = name;
        }

        public void setPrinter(Printer printer){
            this.printer = printer;
        }

        public String sayHello(){
            return "Hello " + name;
        }

        public void print(){
            this.printer.print(sayHello());
        }
    }
```

3)com.example > right-click > New > Interface
4)Name : Printer

```
    package com.example;

    public interface Printer{
        void print(String message);
    }
```

5)com.example > right-click > New > Class
6)Name : StringPrinter
7)Interfaces : com.example.Printer

```
    package com.example;

    public class StringPrinter implements Printer{
        private StringBuffer buffer = new StringBuffer();

        @Override
        public void print(String message){
            this.buffer.append(message);
        }

        public String toString(){
            return this.buffer.toString();
        }
    }
```

8)com.example > right-click > New > Class
9)Name : ConsolePrinter
10)Interfaces : com.example.Printer

```
    package com.example;

    public class ConsolePrinter implements Printer{

        @Override
        public void print(String message){
            System.out.println(message);
        }
    }
```

4. Java Project를 Spring Project로 변환
   1)DIDemo1 Project > right-click > Configure > Convert to Maven Project
      -Project : /DIDemo1
      -Group Id : DIDemo1
      -Artifact Id : DIDemo1
      -version : 0.0.1-SNAPSHOT
      -Packaging : jar
      -Finish

   2)DIDemo1 Project > right-click > Spring > Add Spring Project Nature

   3)pom.xml file에 Spring Context Dependency 추가하기

```
  <version>0.0.1-SNAPSHOT</version>
   <dependencies>
      <dependency>
         <groupId>org.springframework</groupId>
         <artifactId>spring-context</artifactId>
         <version>5.3.10</version>
      </dependency>
   </dependencies>
```

   4)pom.xml > right-click > Run As > Maven install

```
      [INFO] BUILD SUCCESS
```

5. config package 생성
   1)com.example > right-click > New > Package > com.example.config
   2)Finish

6. AppCtx Class 생성
   1)com.example.config > right-click > New > Class
   2)Name : AppCtx.java

```java
package com.example.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import com.example.ConsolePrinter;
import com.example.Hello;
import com.example.StringPrinter;

@Configuration
public class AppCtx {

    @Bean
    public Hello hello() {
        Hello hello = new Hello();
        hello.setName("Spring");
        hello.setPrinter(this.printer());
        return hello;
    }

    @Bean
    public StringPrinter printer() {
        return new StringPrinter();
    }

    @Bean
    public ConsolePrinter consolePrinter() {
        return new ConsolePrinter();
    }
}
```

7. DI Test class 작성
    1)src > right-click > New > Package
    2)Package Name : com.example.test
    3)Finish
    4)com.example.test > right-click > New > Class
    5)Name : HelloBeanTest

```
package com.example.test;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import com.example.Hello;
import com.example.Printer;
import com.example.config.AppCtx;

public class HelloBeanTest {
    public static void main(String[] args) {
        // 1. IoC Container 생성
        ApplicationContext ctx = new AnnotationConfigApplicationContext(AppCtx.class);

        // 2. Hello Beans 가져오기
        Hello hello = (Hello)ctx.getBean("hello");
        System.out.println(hello.sayHello());
        hello.print();

        // 3. SpringPrinter 가져오기
        Printer printer = (Printer) ctx.getBean("printer");
        System.out.println(printer.toString());
        Hello hello2 = ctx.getBean("hello", Hello.class);
        hello2.print();
        System.out.println(hello == hello2);
    }
}
```

8. Test
    1)/src/com.example.test/HelloBeanTest.java > right-click > Run As > Java Application

```
Hello Spring
Hello Spring
true
```

Task 7. setter를 이용한 의존주입하기 실습
1. In Package Explorer > right-click > New > Java Project
    1)Project Name : SpringDemo
    2)JRE
        -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

2. src > right-click > New > Package
    1)Package name : com.example

3. POJO class 작성
    1)com.example > right-click > New > Class
    2)Class Name : BmiCalculator

```
package com.example;

public class BmiCalculator {
    private double lowWeight;
    private double normal;
    private double overWeight;
    private double obesity;

    public void setLowWeight(double lowWeight) {
        this.lowWeight = lowWeight;
    }

    public void setNormal(double normal) {
        this.normal = normal;
    }

    public void setOverWeight(double overWeight) {
        this.overWeight = overWeight;
    }

    public void setObesity(double obesity) {
        this.obesity = obesity;
    }
    public void bmiCalcu(double weight, double height){
        double h = height * 0.01;
        double result = weight / (h * h);

        System.out.println("BMI 지수 : " + (int)result);

        if(result > obesity)
            System.out.println("비만입니다.");
        else if(result > overWeight)
            System.out.println("과체중입니다.");
        else if(result > normal)
            System.out.println("정상입니다.");
        else
            System.out.println("저체중입니다.");
    }
}
```

3)com.example > right-click > New > Class
4)Class Name : MyInfo.java

```
package com.example;

import java.util.ArrayList;

public class MyInfo {
    private String name;
    private double height;
    private double weight;
    private ArrayList<String> hobby;
    private BmiCalculator bmiCalculator;

    public void setBmiCalculator(BmiCalculator bmiCalculator) {
        this.bmiCalculator = bmiCalculator;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setHeight(double height) {
        this.height = height;
    }
    public void setWeight(double weight) {
        this.weight = weight;
    }
    public void setHobby(ArrayList<String> hobby) {
        this.hobby = hobby;
    }
    public void getInfo(){
        System.out.println("Name : " + this.name);
```

```
            System.out.println("Height : " + this.height);
            System.out.println("Weight : " + this.weight);
            System.out.println("Hobby : " + this.hobby);
            this.bmiCalcu();
        }
        public void bmiCalcu(){
            this.bmiCalculator.bmiCalcu(this.weight, this.height);
        }
    }
```

4. Java Project를 Spring Project로 변환

   1)SpringDemo Project > right-click > Configue > Convert to Maven Project
      -Project : /SpringDemo
      -Group Id : SpringDemo
      -Artifact Id : SpringDemo
      -version : 0.0.1-SNAPSHOT
      -Packaging : jar
      -Finish

   2)SpringDemo Project > right-click > Spring > Add Spring Project Nature

   3)pom.xml file에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.10</version>
    </dependency>
</dependencies>
```

   4)pom.xml > right-click > Run As > Maven install
      [INFO] BUILD SUCCESS 확인

5. SpringDemo/resources folder 생성

   1)SpringDemo project > right-click > Build Path > Configure Build Path
   2)Source Tab > Add Folder
   3)SpringDemo 선택 확인
   4)Create New Folder > Folder name : resources > Finish > OK
   5)SpringDemo/resources(new) 확인
   6)Apply and Close

6. Bean Configuration XML 작성

   1)SpringDemo/resources > right-click > New > Other > Spring > Spring Bean Configuration File
   2)File name : applicationContext.xml
   3)Finish

```
<bean id="bmiCalculator" class="com.example.BmiCalculator">
    <property name="lowWeight" value="18.5" />
    <property name="normal" value="23" />
    <property name="overWeight" value="25" />
    <property name="obesity">
        <value>30</value>
    </property>
</bean>
<bean id="myInfo" class="com.example.MyInfo">
    <property name="name" value="백두산" />
    <property name="height" value="170.5" />
    <property name="weight" value="67" />
    <property name="hobby">
        <list>
            <value>수영</value>
            <value>요리</value>
            <value>독서</value>
```

```
                </list>
            </property>
            <property name="bmiCalculator">
                <ref bean="bmiCalculator" />
            </property>
        </bean>
```

7. MainClass 생성하기
    1)com.example.MainClass.java

```
        package com.example;

        import org.springframework.context.AbstractApplicationContext;
        import org.springframework.context.support.GenericXmlApplicationContext;

        public class MainClass {
            public static void main(String[] args) {
                String configFile = "classpath:applicationContext.xml";

                //Spring Container 생성
                AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);

                //Spring Container 에서 객체를 가져옴
                MyInfo myInfo = context.getBean("myInfo", MyInfo.class);

                myInfo.getInfo();
                context.close();
            }
        }
```

8. Java Application 실행

```
    Name : 백두산
    Height : 170.5
    Weight : 67.0
    Hobby : [수영, 요리, 독서]
    BMI 지수 : 23
    정상입니다.
```

[추가 lab] : PropertyEditor 실습
1. In Package Explorer > right-click > New > Java Project
    1)Project Name : PropertyEditorDemo
    2)JRE
        -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

2. src > right-click > New > Package
    1)Package name : com.example

3. POJO class 작성
    1)com.example > right-click > New > Class
    2)Class Name : SimpleBean

```
        package com.example;

        import java.io.File;
        import java.io.InputStream;
        import java.net.URL;
        import java.util.Date;
        import java.util.List;
        import java.util.Locale;
```

```java
import java.util.Properties;
import java.util.regex.Pattern;

public class SimpleBean {
    private byte[] bytes; // ByteArrayPropertyEditor
    private Class cls; // ClassEditor
    private Boolean trueOrFalse; // CustomBooleanEditor
    private List<String> stringList; // CustomCollectionEditor
    private Float floatValue; // CustomNumberEditor
    private File file; // CustomFileEditor
    private InputStream stream; // InputStreamEditor
    private Locale locale; // LocaleEditor
    private Pattern pattern; // PatternEditor
    private Properties properties; // PropertiesEditor
    private URL url; // URLEditor

    public void setBytes(byte[] bytes) {
        System.out.println("Adding " + bytes.length + "bytes");
        this.bytes = bytes;
    }

    public void setCls(Class cls) {
        System.out.println("Setting class : " + cls.getName());
        this.cls = cls;
    }

    public void setTrueOrFalse(Boolean trueOrFalse) {
        System.out.println("Settting Boolean : " + trueOrFalse);
        this.trueOrFalse = trueOrFalse;
    }

    public void setStringList(List<String> stringList) {
        System.out.println("Setting string list with size : " + stringList.size());
        for (String s : stringList) {
            System.out.println("String member : " + s);
        }
        this.stringList = stringList;
    }

    public void setFloatValue(Float floatValue) {
        System.out.println("Setting float value : " + floatValue);
        this.floatValue = floatValue;
    }

    public void setFile(File file) {
        System.out.println("Setting file : " + file.getName());
        this.file = file;
    }

    public void setStream(InputStream stream) {
        System.out.println("Setting stream : " + stream);
        this.stream = stream;
    }

    public void setLocale(Locale locale) {
        System.out.println("Setting Locale : " + locale.getDisplayName());
        this.locale = locale;
    }

    public void setPattern(Pattern pattern) {
        System.out.println("Setting pattern : " + pattern);
        this.pattern = pattern;
    }

    public void setProperties(Properties properties) {
        System.out.println("Loaded : " + properties.size() + "properties");
        this.properties = properties;
    }

    public void setUrl(URL url) {
        System.out.println("Setting URL : " + url.toExternalForm());
        this.url = url;
    }

}
```

4. Java Project를 Spring Project로 변환
    1)PropertyEditorDemo Project > right-click > Configue > Convert to Maven Project
        -Project : /PropertyEditorDemo
        -Group Id : PropertyEditorDemo
        -Artifact Id : PropertyEditorDemo
        -version : 0.0.1-SNAPSHOT
        -Packaging : jar
        -Finish

    2)PropertyEditorDemo Project > right-click > Spring > Add Spring Project Nature

    3)pom.xml file에 Spring Context Dependency 추가하기

```
        <version>0.0.1-SNAPSHOT</version>
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.10</version>
            </dependency>
        </dependencies>
```

    4)pom.xml > right-click > Run As > Maven install
        [INFO] BUILD SUCCESS 확인

5. PropertyEditorDemo/resources folder 생성
    1)PropertyEditorDemo project > right-click > Build Path > Configure Build Path
    2)Source Tab > Add Folder
    3)PropertyEditorDemo 선택 확인
    4)Create New Folder > Folder name : resources > Finish > OK
    5)PropertyEditorDemo/resources(new) 확인
    6)Apply and Close

6. Bean Configuration XML 작성
    1)PropertyEditorDemo/resources > right-click > New > Other > Spring > Spring Bean Configuration File
    2)File name : applicationContext.xml
    3)Finish

```
        <!-- 실제 bean에 대한 정의 -->
        <bean id="simpleBean" class="com.example.SimpleBean">
            <!--  property type에 맞게 알아서 PropertyEditor가 동작한다.  -->
            <property name="bytes">
                <value>Hello, World</value>
            </property>
            <property name="cls">
                <value>java.lang.String</value>
            </property>
            <property name="trueOrFalse">
                <value>true</value>
            </property>
            <property name="stringList">
                <util:list>
                    <value>String member 1</value>
                    <value>String member 2</value>
                </util:list>
            </property>
            <property name="floatValue">
                <value>123.45678</value>
            </property>
            <property name="file">
                <value>classpath:applicationContext.xml</value>
            </property>
            <property name="stream">
                <value>classpath:applicationContext.xml</value>
            </property>
            <property name="locale">
                <value>en_US</value>
            </property>
            <property name="pattern">
                <value>a*b</value>
            </property>
            <property name="properties">
                <value>
                    name=foo
                    age=19
                </value>
            </property>
            <property name="url">
                <value>http://java.sun.com</value>
            </property>
        </bean>
```

7. MainClass 생성하기
    1)com.example.MainClass.java

```
        import org.springframework.context.support.GenericXmlApplicationContext;

        public class MainClass {
            public static void main(String[] args) {
                GenericXmlApplicationContext ctx = new GenericXmlApplicationContext();
                ctx.load("classpath:applicationContext.xml");
                ctx.registerShutdownHook();
                ctx.refresh();
            }
        }
```

8. Java Application 실행

Task 8. 생성자 이용하여 의존 주입하기 실습

1. In Package Explorer > right-click > New > Java Project
    1)Project name : DIDemo2
    2)JRE
        -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

2. src > right-click > New > Package
    1)Package name : com.example
    2)Finish

3. POJO class 작성
    1)com.example > right-click > New > Class
    2)Class Name : Hello

```
        package com.example;

        public class Hello{
            private String name;
            private Printer printer;

            public Hello(){}

            public void setName(String name){
                this.name = name;
            }

            public void setPrinter(Printer printer){
                this.printer = printer;
            }

            public String sayHello(){
                return "Hello " + name;
            }

            public void print(){
                this.printer.print(sayHello());
            }
        }
```

3)com.example > right-click > New > Interface
4)interface name : Printer

```
    package com.example;

    public interface Printer{
        void print(String message);
    }
```

5)com.example > right-click > New > Class
6)Class Name : StringPrinter
7)Interfaces : com.example.Printer

```
    package com.example;

    public class StringPrinter implements Printer{
        private StringBuffer buffer = new StringBuffer();

        @Override
        public void print(String message){
            this.buffer.append(message);
        }

        public String toString(){
            return this.buffer.toString();
        }
    }
```

8)com.example > right-click > New > Class
9)Class Name : ConsolePrinter
10)Intefaces : com.example.Printer

```
    package com.example;

    public class ConsolePrinter implements Printer{

        @Override
        public void print(String message){
            System.out.println(message);
        }
    }
```

4. Java Project를 Spring Project로 변환
   1)DIDemo2 Project > right-click > Configure > Convert to Maven Project
      -Project : /DIDemo2

        -Group Id : DIDemo2
        -Artifact Id : DIDemo2
        -version : 0.0.1-SNAPSHOT
        -Packaging : jar
        -Finish

    2)DIDemo2 Project > right-click > Spring > Add Spring Project Nature

    3)pom.xml file에 Spring Context Dependency 추가하기

```xml
        <version>0.0.1-SNAPSHOT</version>
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.10</version>
            </dependency>
        </dependencies>
```

    4)pom.xml > right-click > Run As > Maven install
        [INFO] BUILD SUCCESS 확인

5. DIDemo2/resources folder 생성
    1)DIDemo2 project > right-click > Build Path > Configure Build Path
    2)Source Tab > Add Folder
    3)DIDemo2 선택확인
    4)Create New Folder > Folder name : resources > Finish > OK
    5)DIDemo2/resources(new) 확인
    6)Apply and Close

6. Bean Configuration XML 작성
    1)DIDemo2/resources > right-click > New > Other > Spring > Spring Bean Configuration File
    -File name : beans.xml > Finish

```xml
        <?xml version="1.0" encoding="UTF-8"?>
        <beans xmlns="http://www.springframework.org/schema/beans"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans.xsd">

            <bean id="hello" class="com.example.Hello">
                <property name="name" value="Spring" />
                <property name="printer" ref="printer" />
            </bean>
            <bean id="printer" class="com.example.StringPrinter" />
            <bean id="consolePrinter" class="com.example.ConsolePrinter" />

        </beans>
```

7. Test class 작성
    1)/src > right-click > New > Package
    2)Package Name : com.example.test
    3)/src/com.example/test/HelloBeanTest.java

```java
        package com.example.test;

        import org.springframework.context.ApplicationContext;
        import org.springframework.context.support.GenericXmlApplicationContext;

        import com.example.Hello;
        import com.example.Printer;

        public class HelloBeanTest {
            public static void main(String [] args){
                //1. IoC Container 생성
                ApplicationContext context =
                        new GenericXmlApplicationContext("classpath:beans.xml");

                //2. Hello Beans 가져오기
                Hello hello = (Hello)context.getBean("hello");
                System.out.println(hello.sayHello());
                hello.print();

                //3. SpringPrinter 가져오기
                Printer printer = (Printer)context.getBean("printer");
                System.out.println(printer.toString());

                Hello hello2 = context.getBean("hello", Hello.class);
                hello2.print();

                System.out.println(hello == hello2);  //Singleton Pattern
            }
        }
```

8. Test
    1)/src/com.example.test/HelloBeanTest.java > right-click > Run As > Java Application

```
        Hello Spring
        Hello Spring
        true
```

9. /src/com.example.Hello 생성자 추가

```
    public Hello(String name, Printer printer) {
        this.name = name;
        this.printer = printer;
    }
```

10. /resources/beans.xml에 아래 Code 추가

```
    <bean id="hello2" class="com.example.Hello">
        <constructor-arg index="0" value="Spring" />
        <constructor-arg index="1" ref="printer" />
    </bean>
```

11. /src/com.example.test/HelloBeanTest.java 수정

```
        ...
        //2. Hello Beans 가져오기
        Hello hello = (Hello)context.getBean("hello2");
        ...
        Hello hello2 = context.getBean("hello2", Hello.class);
        ...
```

12. Test
    1)/src/com.example.test/HelloBeanTest.java > right-click > Run As > Java Application

```
        Hello Spring
        Hello Spring
        true
```

Task 9. Java Annotation을 이용한 생성자 이용하여 의존 주입하기 실습
1. In Package Explorer > right-click > New > Java Project
    1)Project Name : SpringDemo1
    2)JRE > Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]

    4)Next
    5)Finish

2. src > right-click > New > Package
    1)Package name : com.example
    2)Finish

3. POJO Class 생성
    1)com.example.Student.java

```
package com.example;

public class Student {
    private String name;
    private int age;
    private int grade;
    private int classNum;
}
```

    2)com.example.StudentInfo.java

```
package com.example;

public class StudentInfo {
    private Student student;
}
```

4. Java Project를 Spring Project로 변환
    1)SpringDemo1 Project > right-click > Configure > Convert to Maven Project
        -Project : /SpringDemo1
        -Group Id : SpringDemo1
        -Artifact Id : SpringDemo1
        -version : 0.0.1-SNAPSHOT
        -Packaging : jar
        -Finish

    2)SpringDemo1 Project > right-click > Spring > Add Spring Project Nature

    3)pom.xml file에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.10</version>
    </dependency>
</dependencies>
```

    4)pom.xml > right-click > Run As > Maven install
        [INFO] BUILD SUCCESS 확인

5. Lombok library 추가
    1)https://mvnrepository.com/에서 'lombok'으로 검색
    2)'Project Lombok' click
    3)1.18.20 click
    4)depency copy해서 pom.xml에 붙여넣기

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.10</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
    <dependency>
```

```
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.20</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
```

5)pom.xml > right-click > Run As > Maven install
    [INFO] BUILD SUCCESS 확인

6. Student.java와 StudentInfo.java 수정
    1)Student.java

```
package com.example;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;
import lombok.AllArgsConstructor;

@Getter
@Setter
@ToString
@AllArgsConstructor
public class Student {
    private String name;
    private int age;
    private int grade;
    private int classNum;
}
```

    2)StudentInfo.java

```
package com.example;

import lombok.Setter;
import lombok.AllArgsConstructor;

@Setter
@AllArgsConstructor
public class StudentInfo {
    private Student student;

    public void printInfo(){
        if(this.student != null){
            System.out.println("Name : " + this.student.getName());
            System.out.println("Age : " + this.student.getAge());
            System.out.println("Grade : " + this.student.getGrade());
            System.out.println("Class : " + this.student.getClassNum());
            System.out.println("-------------------------");
        }
    }
}
```

7. 환경설정을 위해 config package 생성
    1)com.example package > right-click > New > Package
    2)Name : com.example.config
    3)Finish

8. ApplicationContext.java 생성
    1)com.example.config > right-click > New > Class
    2)Name : ApplicationCtx
    3)Finish

```
    package com.example.config;

    import org.springframework.context.annotation.Bean;
    import org.springframework.context.annotation.Configuration;

    import com.example.Student;
    import com.example.StudentInfo;

    @Configuration
    public class ApplicationCtx {
        @Bean
        public Student student1() {
            return new Student("백두산", 15, 2, 5);
        }

        @Bean
        public Student student2() {
            return new Student("한라산", 16, 3,7);
        }

        @Bean
        public StudentInfo studentInfo() {
            return new StudentInfo(this.student1());
        }
    }
```

9. com.example.MainClass.java

```
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import com.example.config.ApplicationCtx;

public class MainClass {
    public static void main(String[] args) {
        ApplicationContext ctx = new AnnotationConfigApplicationContext(ApplicationCtx.class);

        StudentInfo studentInfo = ctx.getBean("studentInfo", StudentInfo.class);
        studentInfo.printInfo();

        Student student2 = ctx.getBean("student2", Student.class);
        studentInfo.setStudent(student2);
        studentInfo.printInfo();
    }
}
```

10. Java Application 실행

```
Name : 백두산
Age : 15
Grade : 2
Class : 5
-------------------------
Name : 한라산
Age : 16
Grade : 3
Class : 7
-------------------------
```

Task 10. Context file 여러개 사용하기
1. In Package Explorer > right-click > New > Java Project

    1)Project Name : SpringDemo2
    2)JRE
        -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

2)src > right-click > New > Package
    2)Package name : com.example

3. POJO Class 생성
    1)com.example.Student.java

```
        package com.example;

        import java.util.ArrayList;

        public class Student {
            private String name;
            private int age;
            private ArrayList<String> hobbys;
            private double height;
            private double weight;
        }
```

    2)com.example.StudentInfo.java

```
        package com.example;
        public class StudentInfo {
            private Student student;
        }
```

    3)com.example.Product.java

```
        package com.example;
        public class Product {
            private String pName;
            private int pPrice;
            private String maker;
            private String color;
        }
```

4. Java Project를 Spring Project로 변환
    1)SpringDemo2 Project > right-click > Configure > Convert to Maven Project
        -Project : /SpringDemo2
        -Group Id : SpringDemo2
        -Artifact Id : SpringDemo2
        -version : 0.0.1-SNAPSHOT
        -Packaging : jar
        -Finish

    2)SpringDemo2 Project > right-click > Spring > Add Spring Project Nature

    3)pom.xml file에 Spring Context Dependency 추가하기

```
        <version>0.0.1-SNAPSHOT</version>
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.10</version>
            </dependency>
        </dependencies>
```

    4)pom.xml > right-click > Run As > Maven install

[INFO] BUILD SUCCESS 확인

5. Lombok library 추가
    1)https://mvnrepository.com/에서 'lombok'으로 검색
    2)'Project Lombok' click
    3)1.18.20 click
    4)depency copy해서 pom.xml에 붙여넣기

```xml
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.10</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.20</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

    5)pom.xml > right-click > Run As > Maven install
        [INFO] BUILD SUCCESS 확인

6. SpringDemo2/resources folder 생성
    1)SpringDemo2 project > right-click > new > Source Folder
    2)Folder Name : resources
    3)Finish

7. Bean Configuration XML 작성
    1)resources Folder > right-click > New > Spring Bean Configuration File
    2)File name : applicationContext.xml > Finish
    3)resources Folder > right-click > New > Spring Bean Configuration File
    4)File name : applicationContext2.xml > Finish

8. Student.java, StudentInfo.java 그리고 Product.java에 lombok Annotation 붙이기
    1)Student.java

```java
package com.example;

import java.util.ArrayList;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

@Data
@RequiredArgsConstructor
@AllArgsConstructor
public class Student {
    private @NonNull String name;
    private @NonNull int age;
    private @NonNull ArrayList<String> hobbys;
    private double height;
    private double weight;
}
```

    2)StudentInfo.java

```java
package com.example;
```

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

@Setter
@Getter
public class StudentInfo {
    private Student student;
}
```

3)Product.java

```java
package com.example;

import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@NoArgsConstructor
@AllArgsConstructor
@RequiredArgsConstructor
@Setter
@ToString
public class Product {
    private @NonNull String pName;
    private @NonNull int pPrice;
    private String maker;
    private String color;
}
```

9. applicationContext.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="student1" class="com.example.Student">
        <constructor-arg value="백두산" />
        <constructor-arg value="25" />
        <constructor-arg>
            <list>
                <value>독서</value>
                <value>영화감상</value>
                <value>요리</value>
            </list>
        </constructor-arg>
        <property name="height" value="165" />
        <property name="weight">
            <value>45</value>
        </property>
    </bean>

    <bean id="studentInfo1" class="com.example.StudentInfo">
        <property name="student">
            <ref bean="student1" />
        </property>
    </bean>
</beans>
```

10. applicationContext2.xml

1)Namespace tab을 선택하여 c, p를 선택한다.

```
    <?xml version="1.0" encoding="UTF-8"?>
    <beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:c="http://www.springframework.org/schema/c"
      xmlns:p="http://www.springframework.org/schema/p"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">

      <bean id="student3" class="com.example.Student">
        <constructor-arg value="한라산" />
        <constructor-arg value="50" />
        <constructor-arg>
          <list>
            <value>노래부르기</value>
            <value>게임</value>
          </list>
        </constructor-arg>
        <property name="height" value="175" />
        <property name="weight">
          <value>75</value>
        </property>
      </bean>

      <bean id="product" class="com.example.Product" c:pName="Computer"
      c:pPrice="2000000" p:maker="Samsung">
        <property name="color" value="Yellow" />
      </bean>
    </beans>
```

11. com.example.MainClass

```
  package com.example;

  import org.springframework.context.support.AbstractApplicationContext;
  import org.springframework.context.support.GenericXmlApplicationContext;

  public class MainClass {
    public static void main(String[] args) {
      String configFile = "classpath:applicationContext.xml";
      String configFile1 = "classpath:applicationContext2.xml";
      AbstractApplicationContext context = new GenericXmlApplicationContext(configFile,
      configFile1);
      Student student1 = context.getBean("student1", Student.class);
      System.out.println(student1);

      StudentInfo studentInfo = context.getBean("studentInfo1", StudentInfo.class);
      Student student2 = studentInfo.getStudent();
      System.out.println(student2);
      if(student1.equals(student2)) System.out.println("Equals");
      else System.out.println("Different");

      Student student3 = context.getBean("student3", Student.class);
      System.out.println(student3);

      if(student1.equals(student3)) System.out.println("Equals");
      else System.out.println("Different");

      Product product = context.getBean("product", Product.class);
      System.out.println(product);
      context.close();
    }
  }
```

12. Java Application 실행

```
  Student [name=백두산, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
  Student [name=백두산, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
```

```
Equals
Student [name=한라산, age=50, hobbys=[노래부르기, 게임], height=175.0,weight=75.0]
Different
Product [pName=Computer, pPrice=2000000, maker=Samsung, color=Yellow]
```

Task 11. Java Annotation을 이용하여 두 개 이상의 설정 파일로 DI 설정하기

1. In Package Explorer > right-click > New > Java Project
   1)Project Name : SpringDemo3
   2)JRE
      -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
   3)Uncheck [Create module-info.java file]
   4)Next
   5)Finish

2)src > right-click > New > Package
   1)Package name : com.example
   2)Finish

3. POJO Class 생성
   1)com.example.Student.java

```java
package com.example;

import java.util.List;

public class Student {
    private String name;
    private int age;
    private List<String> hobbys;
    private double height;
    private double weight;
}
```

   2)com.example.StudentInfo.java

```java
package com.example;
public class StudentInfo {
    private Student student;
}
```

   3)com.example.Product.java

```java
package com.example;
public class Product {
    private String pName;
    private int pPrice;
    private String maker;
    private String color;
}
```

4. Java Project를 Spring Project로 변환
   1)SpringDemo3 Project > right-click > Configure > Convert to Maven Project
      -Project : /SpringDemo3
      -Group Id : SpringDemo3
      -Artifact Id : SpringDemo3
      -version : 0.0.1-SNAPSHOT
      -Packaging : jar
      -Finish

   2)SpringDemo3 Project > right-click > Spring > Add Spring Project Nature

3)pom.xml file에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.10</version>
    </dependency>
</dependencies>
```

4)pom.xml > right-click > Run As > Maven install
  [INFO] BUILD SUCCESS 확인

5. Lombok library 추가
  1)https://mvnrepository.com/에서 'lombok'으로 검색
  2)'Project Lombok' click
  3)1.18.20 click
  4)depency copy해서 pom.xml에 붙여넣기

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.10</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.20</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

  5)pom.xml > right-click > Run As > Maven install
    [INFO] BUILD SUCCESS 확인

6. com.example.config package 생성
  1)com.example > right-click > new > Package
  2)Name : com.example.config
  3)Finish

7. 2개의 Config Class 작성
  1)com.example.config > right-click > New > Class
  2)Name : AppConfig1
  3)Finish
  4)com.example.config > right-click > New > Class
  5)Name : AppConfig2
  6)Finish

8. Student.java, StudentInfo.java 그리고 Product.java에 lombok Annotation 붙이기
  1)Student.java

```java
package com.example;

import java.util.List;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

@Data
```

```
        @RequiredArgsConstructor
        @AllArgsConstructor
        public class Student {
            private @NonNull String name;
            private @NonNull int age;
            private @NonNull List<String> hobbys;
            private double height;
            private double weight;
        }
```

    2)StudentInfo.java

```
        package com.example;

        import lombok.Setter;
        import lombok.Getter;

        @Setter
        @Getter
        public class StudentInfo {
            private Student student;
        }
```

    3)Product.java

```
        package com.example;

        import lombok.AllArgsConstructor;
        import lombok.NoArgsConstructor;
        import lombok.NonNull;
        import lombok.RequiredArgsConstructor;
        import lombok.Setter;
        import lombok.ToString;

        @NoArgsConstructor
        @AllArgsConstructor
        @RequiredArgsConstructor
        @Setter
        @ToString
        public class Product {
            private @NonNull String pName;
            private @NonNull int pPrice;
            private @NonNull String maker;
            private String color;
        }
```

9. AppConfig1.java

```
    package com.example.config;

    import java.util.Arrays;
    import java.util.List;

    import org.springframework.context.annotation.Bean;
    import org.springframework.context.annotation.Configuration;

    import com.example.Student;
    import com.example.StudentInfo;

    @Configuration
    public class AppConfig1 {
        @Bean
        public Student student1() {
            List<String> list = Arrays.asList("독서", "영화감상", "요리");
            Student student1 = new Student("백두산", 25, list);
            student1.setHeight(165);
            student1.setWeight(45);
            return student1;
        }

        @Bean
        public StudentInfo studentInfo1() {
            StudentInfo studentInfo1 = new StudentInfo();
            studentInfo1.setStudent(this.student1());
            return studentInfo1;
        }
    }
```

10. AppConfig2.java

```
    package com.example.config;

    import java.util.Arrays;
    import java.util.List;

    import org.springframework.context.annotation.Bean;
    import org.springframework.context.annotation.Configuration;

    import com.example.Product;
    import com.example.Student;

    @Configuration
    public class AppConfig2 {
        @Bean
        public Student student3() {
            List<String> list = Arrays.asList("노래부르기", "게임");
            Student student3 = new Student("한라산", 50, list);
            student3.setHeight(175);
            student3.setWeight(75);
            return student3;
        }

        @Bean
        public Product product() {
            Product product = new Product("Computer", 2000000, "Samsung");
            product.setColor("Yellow");
            return product;
        }
    }
```

11. com.example.MainClass

```
    package com.example;

    import org.springframework.context.ApplicationContext;
    import org.springframework.context.annotation.AnnotationConfigApplicationContext;

    import com.example.config.AppConfig1;
    import com.example.config.AppConfig2;

    public class MainClass {
        public static void main(String[] args) {
            ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig1.class,
            AppConfig2.class);
            Student student1 = context.getBean("student1", Student.class);
            System.out.println(student1);

            StudentInfo studentInfo = context.getBean("studentInfo1", StudentInfo.class);
            Student student2 = studentInfo.getStudent();
            System.out.println(student2);
            if(student1.equals(student2)) System.out.println("Equals");
            else System.out.println("Different");

            Student student3 = context.getBean("student3", Student.class);
            System.out.println(student3);

            if(student1.equals(student3)) System.out.println("Equals");
            else System.out.println("Different");

            Product product = context.getBean("product", Product.class);
            System.out.println(product);
        }
    }
```

12. Java Application 실행

```
    Student [name=백두산, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
    Student [name=백두산, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
    Equals
    Student [name=한라산, age=50, hobbys=[노래부르기, 게임], height=175.0,weight=75.0]
    Different
    Product [pName=Computer, pPrice=2000000, maker=Samsung, color=Yellow]
```

Task 12. Java Annotation과 XML 을 이용한 DI 설정 방법 : XML file에 Java file을 포함시켜 사용하는 방법

1. In Package Explorer > right-click > New > Java Project
    1)Project Name : SpringDemo4
    2)JRE
        -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

2. src > right-click > New > Package
    1)Package name : com.example

3. POJO 생성
    1)com.example.Student.java

```
        package com.example;

        import java.util.ArrayList;

        public class Student {
            private String name;
            private int age;
            private ArrayList<String> hobbys;
            private double height;
            private double weight;
        }
```

4. Java Project를 Spring Project로 변환
    1)SpringDemo4 Project > right-click > Configure > Convert to Maven Project
        -Project : /SpringDemo4
        -Group Id : SpringDemo4
        -Artifact Id : SpringDemo4
        -version : 0.0.1-SNAPSHOT
        -Packaging : jar
        -Finish

    2)SpringDemo4 Project > right-click > Spring > Add Spring Project Nature

    3)pom.xml file에 Spring Context Dependency 추가하기

```
      <version>0.0.1-SNAPSHOT</version>
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
```

```xml
                <artifactId>spring-context</artifactId>
                <version>5.3.10</version>
            </dependency>
        </dependencies>
```

4)pom.xml > right-click > Run As > Maven install
    [INFO] BUILD SUCCESS 확인

5. Lombok library 추가
    1)https://mvnrepository.com/에서 'lombok'으로 검색
    2)'Project Lombok' click
    3)1.18.20 click
    4)depency copy해서 pom.xml에 붙여넣기

```xml
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.10</version>
            </dependency>
            <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
            <dependency>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <version>1.18.20</version>
                <scope>provided</scope>
            </dependency>
        </dependencies>
```

5)pom.xml > right-click > Run As > Maven install
    [INFO] BUILD SUCCESS 확인

6. Student.java lombok Annotation 붙이기
    1)Student.java

```java
        package com.example;

        import java.util.List;

        import lombok.AllArgsConstructor;
        import lombok.Data;
        import lombok.NonNull;
        import lombok.RequiredArgsConstructor;

        @Data
        @RequiredArgsConstructor
        @AllArgsConstructor
        public class Student {
            private @NonNull String name;
            private @NonNull int age;
            private @NonNull List<String> hobbys;
            private double height;
            private double weight;
        }
```

7. com.example.ApplicationConfig.java

```java
    package com.example;

    import java.util.Arrays;
    import java.util.List;

    import org.springframework.context.annotation.Bean;
    import org.springframework.context.annotation.Configuration;

    @Configuration
    public class ApplicationConfig {
        @Bean
        public Student student1(){
            List<String> list = Arrays.asList("독서", "영화감상", "요리");

            Student student1 = new Student("백두산", 25, list);
            student1.setHeight(165);
            student1.setWeight(45);

            return student1;
        }
    }
```

8. SpringDemo4/resources folder 생성
    1)SpringDemo4 project > right-click > Build Path > Configure Build Path
    2)Source Tab > Add Folder
    3)SpringDemo4 선택 확인
    4)Create New Folder > Folder name : resources > Finish > OK
    5)SpringDemo4/resources(new) 확인
    6)Apply and Close

9. Bean Configuration XML 작성
    1)SpringDemo4/resources > right-click > New > Spring Bean Configuration File
    2)File name : applicationContext.xml > Finish

10. /resources/applicationContext.xml

```
    <?xml version="1.0" encoding="UTF-8"?>
    <beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

        <bean class="org.springframework.context.annotation.ConfigurationClassPostProcessor" />
        <bean class="com.example.ApplicationConfig" />
        <bean id="student3" class="com.example.Student">
            <constructor-arg value="북한산" />
            <constructor-arg value="50" />
            <constructor-arg>
                <list>
                    <value>노래부르기</value>
                    <value>게임</value>
                </list>
            </constructor-arg>
            <property name="height" value="175" />
            <property name="weight">
                <value>75</value>
            </property>
        </bean>
    </beans>
```

11. com.example.MainClass.java

```
    package com.example;

    import org.springframework.context.support.AbstractApplicationContext;
    import org.springframework.context.support.GenericXmlApplicationContext;

    public class MainClass {
        public static void main(String[] args) {
            String configFile = "classpath:applicationContext.xml";
            AbstractApplicationContext context = new GenericXmlApplicationContext(configFile);
            Student student1 = context.getBean("student1", Student.class);
            System.out.println(student1);

            Student student3 = context.getBean("student3", Student.class);
            System.out.println(student3);
        }
    }
```

12. Java Application 실행

```
Student [name=백두산, age=25, hobbys=[독서, 영화감상, 요리], height=165.0,weight=45.0]
Student [name=북한산, age=50, hobbys=[노래부르기, 게임], height=175.0,weight=75.0]
```

13. JUnit을 사용한 DI test class 작성하기
    1)com.example > right-click > New > JUnit Test Case
    2)Select [New JUnit 4 test]
    3)Name : HelloBeanJUnitTest
    4)Finish
    5)[New JUnit Test Case] 창에서 Select [Perform the follwing action:] > Add JUnit 4 library to the build path
    6)OK

14. JUnit을 사용한 Test
    1)src/com.example > New > Class
        -Name : HelloBeanJUnitTest.java

```
package com.example;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertSame;

import org.junit.Before;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

public class HelloBeanJUnitTest {
    private ApplicationContext context;

    @Before
    public void init(){
        context = new GenericXmlApplicationContext("classpath:applicationContext.xml");
    }

    @Test
    public void test1(){
        Student student1 = (Student)context.getBean("student1");
        assertEquals("백두산", student1.getName());
        System.out.println(student1);
    }

    @Test
    public void test2(){
        Student student3 = context.getBean("student3", Student.class);
        System.out.println(student3);

        Student student4 = (Student)context.getBean("student3");
        assertSame(student3, student4);
    }
}
```

    2)HelloBeanJUnitTest.java > right-click > Run As > JUnit Test
        -JUnit 창에 Green Bar

```
Student(name=백두산, age=25, hobbys=[독서, 영화감상, 요리], height=165.0, weight=45.0)
Student(name=북한산, age=50, hobbys=[노래부르기, 게임], height=175.0, weight=75.0)
```

15. Spring TestContext Framework을 이용한 Test

1)Spring-Test library 설치
- -http://mvnrepository.com에서 'spring test'로 검색
- -검색 결과 목록에서 'Spring TestContext Framework' Click
- -version 목록에서 5.3.10 Click

2)dependency 복사해서 pom.xml에 붙여넣기

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>5.3.10</version>
    <scope>test</scope>
</dependency>
```

3)pom.xml > right-click > Maven Install

```
[INFO] BUILD SUCCESS
```

4)Spring-Test를 사용할 HelloBeanJunitSpringTest.java 작성
- -src/com.example > New > Class
- -Name : HelloBeanJunitSpringTest
- -Finish

```java
package com.example;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertSame;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations="classpath:applicationContext.xml")
public class HelloBeanJunitSpringTest {
    @Autowired
    private ApplicationContext context;

    @Test
    public void test1() {
        Student student1 = this.context.getBean("student1", Student.class);
        assertEquals(25, student1.getAge());
        System.out.println(student1);
    }

    @Test
    public void test2() {
        Student student3 = (Student)this.context.getBean("student3");
        Student student4 = this.context.getBean("student3", Student.class);
        assertSame(student3, student4);
        System.out.println(student4);
    }
}
```

- -right-click > Run As > JUnit Test
- -결과 -> JUnit View에 초록색 bar

5)만일 해당 객체를 찾을 수 없다는 오류가 계속 발생하면
- -해당 Project > right-click > Build Path > Libraries tab
- -spring-test-5.3.10.jar 선택 후 [Remove] 로 삭제
- -Classpath 선택
- -[Add External JARs...] Click
- -Local M2 Repository(e.g

        C:\Users\instructor\.m2\repository\org\springframework\spring-test\5.3.10)에서 직접 jar(spring-test-5.3.10.jar)를 선택할 것
        -[Order and Export] tab에서 spring-test-5.3.10.jar 선택 후 [Up] button을 클릭
        -해당 DIDemo/src 바로 아래까지 올리고 [Apply and Close] Click

Task 13. Java Annotation과 XML 을 이용한 DI 설정 방법 : Java file에 XML file을 포함시켜 사용하는 방법

1. In Package Explorer > right-click > New > Java Projectn
    1)Project Name : SpringDemo5
    2)JRE
        -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

2. src > right-click > New > Package
    1)Package name : com.example
    2)Finish

3. com.example.Student.java

```
    package com.example;

    import java.util.List;

    public class Student {
        private String name;
        private int age;
        private List<String> hobbys;
        private double height;
        private double weight;
    }
```

4. Java Project를 Spring Project로 변환
    1)SpringDemo5 Project > right-click > Configure > Convert to Maven Project
        -Project : /SpringDemo5
        -Group Id : SpringDemo5
        -Artifact Id : SpringDemo5
        -version : 0.0.1-SNAPSHOT
        -Packaging : jar
        -Finish

    2)SpringDemo5 Project > right-click > Spring > Add Spring Project Nature

    3)pom.xml file에 Spring Context Dependency 추가하기

```
      <version>0.0.1-SNAPSHOT</version>
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.10</version>
            </dependency>
        </dependencies>
```

    4)pom.xml > right-click > Run As > Maven install
        [INFO] BUILD SUCCESS 확인

5. Lombok library 추가
    1)https://mvnrepository.com/에서 'lombok'으로 검색
    2)'Project Lombok' click
    3)1.18.20 click
    4)depency copy해서 pom.xml에 붙여넣기

```
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.10</version>
            </dependency>
            <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
            <dependency>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <version>1.18.20</version>
                <scope>provided</scope>
            </dependency>
        </dependencies>
```

5)pom.xml > right-click > Run As > Maven install
    [INFO] BUILD SUCCESS 확인

6. Student.java lombok Annotation 붙이기
    1)Student.java

```
        package com.example;

        import java.util.List;

        import lombok.AllArgsConstructor;
        import lombok.Data;
        import lombok.NonNull;
        import lombok.RequiredArgsConstructor;

        @Data
        @RequiredArgsConstructor
        @AllArgsConstructor
        public class Student {
            private @NonNull String name;
            private @NonNull int age;
            private @NonNull List<String> hobbys;
            private double height;
            private double weight;
        }
```

7. SpringDemo5/resources folder 생성
    1)SpringDemo5 project > right-click > Build Path > Configure Build Path
    2)Source Tab > Add Folder
    3)SpringDemo5 선택 확인
    4)Create New Folder > Folder name : resources > Finish > OK
    5)SpringDemo5/resources(new) 확인
    6)Apply and Close

8. Bean Configuration XML 작성
    1)SpringDemo5/resources > right-click > New > Spring Bean Configuration File
    2)File name : applicationContext.xml > Finish

9. /resources/applicationContext.xml

```
    <?xml version="1.0" encoding="UTF-8"?>
    <beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

        <bean id="student3" class="com.example.Student">
            <constructor-arg value="지리산" />
            <constructor-arg value="30" />
            <constructor-arg>
                <list>
                    <value>등산</value>
                    <value>게임</value>
                    <value>독서</value>
                </list>
            </constructor-arg>
            <property name="height" value="165" />
            <property name="weight">
                <value>49</value>
            </property>
        </bean>
    </beans>
```

10. com.example.ApplicationConfig.java

```
package com.example;

import java.util.Arrays;
import java.util.List;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.ImportResource;

@Configuration
@ImportResource("classpath:ApplicationContext.xml")
public class ApplicationConfig {
    @Bean
    public Student student1(){
        List<String> hobbys = Arrays.asList("독서", "영화감상", "요리");

        Student student = new Student("백두산", 25, hobbys);
        student.setHeight(165);
        student.setWeight(45);

        return student;
    }
}
```

11. com.example.MainClass.java

```
package com.example;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainClass {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context = new
        AnnotationConfigApplicationContext(ApplicationConfig.class);
        Student student1 = context.getBean("student1", Student.class);
        System.out.println(student1);

        Student student3 = context.getBean("student3", Student.class);
        System.out.println(student3);

        context.close();
    }
}
```

12. Java Application 실행

```
Student(name=백두산, age=25, hobbys=[독서, 영화감상, 요리], height=165.0, weight=45.0)
Student(name=지리산, age=30, hobbys=[등산, 게임, 독서], height=165.0, weight=49.0)
```

13. JUnit 5를 사용한 DI test class 작성하기
    1)com.example > right-click > New > JUnit Test Case
    2)Select [New JUnit Jupiter test]
    3)Name : HelloBeanJUnitTest
    4)Finish
    5)[New JUnit Test Case] 창에서 Select [Perform the follwing action:] > Add JUnit 5 library to the build path
    6)OK

14. pom.xml에 dependency 추가
    1)JUnit 5 설치
        -http://mvnrepository.com에서 'junit'로 검색
        -검색 결과 목록에서 'JUnit Jupiter API' Click
        -version 목록에서 5.8.1 click

    2)dependency 복사해서 pom.xml에 붙여넣기

```xml
<!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.8.1</version>
    <scope>test</scope>
</dependency>
```

    3)pom.xml > right-click > Maven Install
        [INFO] BUILD SUCCESS
        -만일 ERROR 발생하면 다음과 같이 조치한다.
        -SpringDemo5 > right-click > Maven > Update Project
        -SpringDemo5가 check되어 있음을 확인하고 OK
        -다시 pom.xml > right-click > Maven Install
            [INFO] BUILD SUCCESS

15. JUnit 5를 사용한 Test
    1)com.example.HelloBeanJUnitTest.java

```java
package com.example;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertSame;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

class HelloBeanJUnitTest {
    private ApplicationContext context;

    @BeforeEach
    public void init() {
        this.context = new AnnotationConfigApplicationContext(ApplicationConfig.class);
    }

    @Test
    public void test1(){
        Student student1 = (Student)context.getBean("student1");
        assertEquals("백두산", student1.getName());
        System.out.println(student1);
    }

    @Test
    public void test2() {
        Student student3 = context.getBean("student3", Student.class);
        Student student4 = (Student)context.getBean("student3");
        assertSame(student3, student4);
```

```
            System.out.println(student3);
        }
    }
```

2)HelloBeanJUnitTest.java > right-click > Run As > JUnit Test

-JUnit 창에 Green Bar

```
        Student(name=백두산, age=25, hobbys=[독서, 영화감상, 요리], height=165.0, weight=45.0)
        Student(name=홍지민, age=30, hobbys=[등산, 게임, 독서], height=165.0, weight=49.0)
```

Task 14. Lab

1. In Package Explorer > right-click > New > Java Project
    1)Project name : DIDemo3
    2)JRE
        -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

2. src > right-click > New > Package
    1)Package name : com.example
    2)Finish

3. POJO class 작성
    1)com.example > right-click > New > Class
    2)Class Name : Hello

```
    package com.example;

    public class Hello{
        private String name;
        private Printer printer;

        public String sayHello(){
            return "Hello " + name;
        }

        public void print(){
            this.printer.print(sayHello());
        }
    }
```

    3)com.example > right-click > New > Interface
    4)interface name : Printer

```
    package com.example;

    public interface Printer{
        void print(String message);
    }
```

    5)com.example > right-click > New > Class
    6)Class Name : StringPrinter
    7)Interfaces : com.example.Printer

```
    package com.example;

    public class StringPrinter implements Printer{
        private StringBuffer buffer = new StringBuffer();

        @Override
        public void print(String message){
            this.buffer.append(message);
        }

        public String toString(){
            return this.buffer.toString();
        }
    }
```

8)com.example > right-click > New > Class
9)Class Name : ConsolePrinter
10)Interfaces : com.example.Printer

```
    package com.example;

    public class ConsolePrinter implements Printer{

        @Override
        public void print(String message){
            System.out.println(message);
        }
    }
```

4. Java Project를 Spring Project로 변환

1)DIDemo3 Project > right-click > Configure > Convert to Maven Project
    -Project : /DIDemo3
    -Group Id : DIDemo3
    -Artifact Id : DIDemo3
    -version : 0.0.1-SNAPSHOT
    -Packaging : jar
    -Finish

2)DIDemo3 Project > right-click > Spring > Add Spring Project Nature

3)pom.xml file에 Spring Context Dependency 추가하기

```
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.10</version>
        </dependency>
    </dependencies>
```

4)pom.xml > right-click > Run As > Maven install
    [INFO] BUILD SUCCESS 확인

5. Lombok library 추가

1)https://mvnrepository.com/에서 'lombok'으로 검색
2)'Project Lombok' click
3)1.18.20 click
4)depency copy해서 pom.xml에 붙여넣기

```
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.10</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.20</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
```

5)pom.xml > right-click > Run As > Maven install
    [INFO] BUILD SUCCESS 확인

6. Hello.java에 lombok Annotation으로 수정하기

```java
package com.example;

import lombok.NoArgsConstructor;
import lombok.Setter;

@Setter
@NoArgsConstructor
public class Hello {
    private String name;
    private Printer printer;

    public String sayHello(){
        return "Hello " + name;
    }

    public void print(){
        this.printer.print(sayHello());
    }
}
```

7. src/config folder 생성
    1)/src > right-click > New > Folder
    2)Folder name : config

8. Bean Configuration XML 작성
    1)/src/config > right-click > New > Other > Spring > Spring Bean Configuration File
    2)File name : beans.xml > Finish

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="hello" class="com.example.Hello">
        <property name="name" value="Spring" />
        <property name="printer" ref="printer" />
    </bean>
    <bean id="printer" class="com.example.StringPrinter" />
    <bean id="consolePrinter" class="com.example.ConsolePrinter" />
</beans>
```

9. DI Test class 작성
    1)/src > right-click > New > Package
    2)Name : com.example.test
    3)Finish
    4)/src/com.example.test > right-click > New > Class
    5)Name : HelloBeanTest

```java
package com.example.test;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

import com.example.Hello;
import com.example.Printer;
```

```
    public class HelloBeanTest {
        public static void main(String [] args){
            ApplicationContext context = new GenericXmlApplicationContext("config/beans.xml");

            Hello hello = (Hello)context.getBean("hello");
            System.out.println(hello.sayHello());
            hello.print();

            Printer printer = (Printer)context.getBean("printer");
            System.out.println(printer.toString());

            Hello hello2 = context.getBean("hello", Hello.class);
            hello2.print();

            System.out.println(hello == hello2);  //Singleton Pattern
        }
    }
```

6)Java Application 실행

```
    Hello Spring
    Hello Spring
    true
```

10. JUnit 5 Library 설치

1)JUnit 5 설치

-http://mvnrepository.com에서 'junit'로 검색
-검색 결과 목록에서 'JUnit Jupiter API' Click
-version 목록에서 5.8.1 click

2)dependency 복사해서 pom.xml에 붙여넣기

```
    <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.8.1</version>
        <scope>test</scope>
    </dependency>
```

3)pom.xml > right-click > Maven Install

```
    [INFO] BUILD SUCCESS
```

-만일 ERROR 발생하면 다음과 같이 조치한다.
-SpringDemo5 > right-click > Maven > Update Project
-SpringDemo5가 check되어 있음을 확인하고 OK
-다시 pom.xml > right-click > Maven Install

```
        [INFO] BUILD SUCCESS
```

11. JUnit 5를 사용한 Test

1)com.example.test > right-click > New > Class
2)Name : HelloBeanJUnitTest

```
    package com.example.test;

    import static org.junit.jupiter.api.Assertions.assertEquals;
    import static org.junit.jupiter.api.Assertions.assertSame;

    import org.junit.jupiter.api.BeforeEach;
    import org.junit.jupiter.api.Test;
    import org.springframework.context.ApplicationContext;
    import org.springframework.context.support.GenericXmlApplicationContext;

    import com.example.Hello;

    public class HelloBeanJUnitTest {
        private ApplicationContext context;

        @BeforeEach
        public void init() {
            this.context = new GenericXmlApplicationContext("config/beans.xml");
        }

        @Test
        public void test1(){
            Hello hello = (Hello)context.getBean("hello");
            assertEquals("Hello Spring", hello.sayHello());
            hello.print();
        }

        @Test
        public void test2(){
            Hello hello = (Hello)context.getBean("hello");
            Hello hello2 = context.getBean("hello", Hello.class);
            assertSame(hello, hello2);
        }
    }
```

3)만일 해당 객체를 찾을 수 없다는 오류가 계속 발생하면
- -해당 Project > right-click > Build Path > Configure Build Path > Libraries tab
- -Classpath 선택
- -[Add External JARs...] Click
- -Local M2 Repository(e.g C:\Users\사용자아이디\.m2\repository\org\junit\jupiter\junit-jupiter-api\5.8.1)에서 직접 jar(junit-jupiter-api-5.8.1.jar)를 선택할 것
- -[Order and Export] tab에서 junit-jupiter-api-5.8.1.jar 선택 후 [Up] button을 클릭
- -해당 Project/src 바로 아래까지 올리고 [Apply and Close] Click

4)HelloBeanJUnitTest.java > right-click > Run As > JUnit Test
- -JUnit 창에 Green Bar

12. Spring TestContext Framework

1)Spring-Test library 설치

2)pom.xml 수정

```
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>5.3.10</version>
        <scope>test</scope>
    </dependency>
```

3)pom.xml > right-click > Maven Install
- -만일 Error 발생시 DIDemo3 > right-click > Maven > Update Project... > Ok
- -다시 Maven Install 실행

4)Spring-Test를 사용할 DI test class-HelloBeanJUnitSpringTest.java 작성하기
- -/src/com.example.test > New > Class
- -Name : HelloBeanJUnitSpringTest
- -Finish

```
        package com.example.test;

        import static org.junit.jupiter.api.Assertions.assertEquals;
        import static org.junit.jupiter.api.Assertions.assertSame;

        import org.junit.jupiter.api.Test;
        import org.junit.jupiter.api.extension.ExtendWith;
        import org.springframework.beans.factory.annotation.Autowired;
        import org.springframework.context.ApplicationContext;
        import org.springframework.test.context.ContextConfiguration;
        import org.springframework.test.context.junit.jupiter.SpringExtension;
```

```java
        import com.example.Hello;

        @ExtendWith(SpringExtension.class)
        //JUnit 5.x에서 사용
        @ContextConfiguration(locations="classpath:config/beans.xml")
        public class HelloBeanJUnitSpringTest {
            @Autowired
            ApplicationContext context;

            @Test
            public void test1(){
                Hello hello = (Hello)context.getBean("hello");
                assertEquals("Hello Spring", hello.sayHello());
                hello.print();
            }

            @Test
            public void test2(){
                Hello hello = (Hello)context.getBean("hello");
                Hello hello2 = context.getBean("hello", Hello.class);
                assertSame(hello, hello2);
            }
        }
```

5)right-click > Run As > Junit Test
6)결과 -> Junit View에 초록색 bar
7)만일 해당 객체를 찾을 수 없다는 오류가 계속 발생하면
  -해당 Project > right-click > Build Path > Configure Build Path > Libraries tab
  -spring-test-5.3.10.jar 선택 후 [Remove] 로 삭제
  -Classpath 선택
  -[Add External JARs...] Click
  -Local M2 Repository(e.g C:\Users\사용자아이디\.m2\repository\org\springframework\spring-test\5.3.10)에서 직접 jar(spring-test-5.3.10.jar)를 선택할 것
  -[Order and Export] tab에서 spring-test-5.3.10.jar 선택 후 [Up] button을 클릭
  -해당 Project/src 바로 아래까지 올리고 [Apply and Close] Click

13. src/com.example/StringPrinter.java 수정

```java
    package com.example;

    import org.springframework.stereotype.Component;

    @Component("stringPrinter")
    public class StringPrinter implements Printer{
        private StringBuffer buffer = new StringBuffer();
    ...
```

14. src/com.example/ConsolePrinter.java 수정

```java
    package com.example;

    import org.springframework.stereotype.Component;

    @Component("consolePrinter")
    public class ConsolePrinter implements Printer{
    ...
```

15. /src/com.example/Hello.java 수정

```java
    package com.example;

    //import org.springframework.beans.factory.annotation.Autowired;
    //import org.springframework.beans.factory.annotation.Qualifier;
    import org.springframework.beans.factory.annotation.Value;
    import org.springframework.stereotype.Component;
```

```java
//import javax.annotation.Resource;

import javax.inject.Inject;
import javax.inject.Named;

import lombok.NoArgsConstructor;
import lombok.Setter;

@Setter
@NoArgsConstructor
@Component
public class Hello {
    @Value("Spring")
    private String name;

    //@Autowired(required = true)
    //@Qualifier("stringPrinter")
    //@Resource(name = "stringPrinter")
    /*
      @Resource annotation 사용하려면 mvnrepository.com에서 Javax Annotation API 검색해서
      <dependency>
        <groupId>javax.annotation</groupId>
        <artifactId>javax.annotation-api</artifactId>
        <version>1.3.2</version>
      </dependency>

      @Inject annotation 사용하려면 mvnrepository.com에서 Javax Inject API 검색해서
      <dependency>
        <groupId>javax.inject</groupId>
        <artifactId>javax.Inject</artifactId>
        <version>1</version>
      </dependency>
    */
    @Inject
    @Named("stringPrinter")
    private Printer printer;

    public String sayHello(){
        return "Hello " + name;
    }

    public void print(){
        this.printer.print(sayHello());
    }
}
```

16. 기존의 설정file과 충돌이 발생하기 때문에 /src/config/beans.xml 삭제

17. 새로운 설정 file 생성
    1)src/config > right-click > New > Other > Spring > Spring Bean Configuration File
    2)File name : annos.xml > Finish
    3)Namespace tab > context Check

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-4.3.xsd">

    <context:component-scan base-package="com.example" />
</beans>
```

18. /src/com.example.test/HelloBeanJUnitSpringTest.java 수정하기
```
	package com.example.test;

	import static org.junit.jupiter.api.Assertions.assertEquals;
	import static org.junit.jupiter.api.Assertions.assertSame;

	import org.junit.jupiter.api.Test;
	import org.junit.jupiter.api.extension.ExtendWith;
	import org.springframework.beans.factory.annotation.Autowired;
	import org.springframework.context.ApplicationContext;
	import org.springframework.test.context.ContextConfiguration;
	import org.springframework.test.context.junit.jupiter.SpringExtension;

	import com.example.Hello;

	@ExtendWith(SpringExtension.class)
	@ContextConfiguration(locations="classpath:config/annos.xml")
	public class HelloBeanJUnitSpringTest {
		@Autowired
		ApplicationContext context;

		@Test
		public void test1(){
			Hello hello = (Hello)context.getBean("hello");
			assertEquals("Hello Spring", hello.sayHello());
			hello.print();
		}

		@Test
		public void test2(){
			Hello hello = (Hello)context.getBean("hello");
			Hello hello2 = context.getBean("hello", Hello.class);
			assertSame(hello, hello2);
		}
	}
```

1)right-click > Run As > Junit Test
2)결과 -> Junit View에 초록색 bar

Task 15. Lab with JUnit 5 Jupiter
1. In Package Explorer > right-click > New > Java Project
    1)Project Name : DIDemo4
    2)JRE
        -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

2. src > right-click > New > Package
    1)Package name : com.example
    2)Finish

3. com.example.Student.java, com.example.StudentInfo.java
    1)Student.java
```
	package com.example;

	import java.util.List;

	public class Student {
		private String name;
		private int age;
```

```
        private List<String> hobbys;
        private double height;
        private double weight;
    }
```

2)StudentInfo.java

```
    package com.example;

    public class StudentInfo {
        private Student student;
    }
```

4. Java Project를 Spring Project로 변환

1)DIDemo4 Project > right-click > Configure > Convert to Maven Project
- -Project : /DIDemo4
- -Group Id : DIDemo4
- -Artifact Id : DIDemo4
- -version : 0.0.1-SNAPSHOT
- -Packaging : jar
- -Finish

2)DIDemo4 Project > right-click > Spring > Add Spring Project Nature

3)pom.xml file에 Spring Context Dependency 추가하기

```
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.10</version>
        </dependency>
    </dependencies>
```

4)pom.xml > right-click > Run As > Maven install

[INFO] BUILD SUCCESS 확인

5. Lombok library 추가

1)https://mvnrepository.com/에서 'lombok'으로 검색

2)'Project Lombok' click

3)1.18.20 click

4)depency copy해서 pom.xml에 붙여넣기

```
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.10</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.20</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
```

5)pom.xml > right-click > Run As > Maven install

[INFO] BUILD SUCCESS 확인

6. Student.java, StudentInfo.java lombok Annotation 붙이기

1)Student.java

```
    package com.example;
    import java.util.List;

    import lombok.AllArgsConstructor;
    import lombok.Data;
    import lombok.NonNull;
    import lombok.RequiredArgsConstructor;

    @Data
    @RequiredArgsConstructor
    @AllArgsConstructor
    public class Student {
        private @NonNull String name;
        private @NonNull int age;
        private @NonNull List<String> hobbys;
        private double height;
        private double weight;
    }
```

2)StudentInfo.java

```
    package com.example;

    import lombok.AllArgsConstructor;
    import lombok.Setter;

    @Setter
    @AllArgsConstructor
    public class StudentInfo {
        private Student student;

        public void printInfo(){
            if(this.student != null){
                System.out.println("Name : " + this.student.getName());
                System.out.println("Age : " + this.student.getAge());
                System.out.println("Hobbies");
                this.student.getHobbys().forEach(hobby -> System.out.println(hobby));
                System.out.println("Height : " + this.student.getHeight());
                System.out.println("Weight : " + this.student.getWeight());
            }
        }
    }
```

7. com.example.config package 생성
    1)com.example > right-click > New > Package
    2)Name : com.example.config
    3)Finish

8. com.example.config.ApplicationConfig.java 생성
    1)com.example.config > right-click > New > Click
    2)Name : ApplicationConfig
    3)Finish

```
    package com.example.config;

    import java.util.Arrays;
    import java.util.List;

    import org.springframework.context.annotation.Bean;
    import org.springframework.context.annotation.Configuration;

    import com.example.Student;
    import com.example.StudentInfo;

    @Configuration
    public class ApplicationConfig {
        @Bean
        public Student student1() {
            List<String> list = Arrays.asList("독서", "영화감상", "요리");
            Student student1 = new Student("백두산", 25, list);
            student1.setHeight(165);
            student1.setWeight(45);
            return student1;
        }

        @Bean
        public StudentInfo studentInfo() {
            return new StudentInfo(this.student1());
        }
    }
```

9. com.example.MainClass.java

```
package com.example;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import com.example.config.ApplicationConfig;

public class MainClass {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context = new
        AnnotationConfigApplicationContext(ApplicationConfig.class);
        Student student1 = context.getBean("student1", Student.class);
        System.out.println(student1);

        StudentInfo studentInfo = context.getBean("studentInfo", StudentInfo.class);
        studentInfo.setStudent(student1);
        studentInfo.printInfo();

        context.close();
    }
}
```

10. Java Application 실행

```
Student(name=백두산, age=25, hobbys=[독서, 영화감상, 요리], height=165.0, weight=45.0)
Name : 백두산
Age : 25
Hobbies
독서
영화감상
요리
Height : 165.0
Weight : 45.0
```

11. Student.java 수정

```
package com.example;

import java.util.List;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import lombok.Getter;
import lombok.Setter;

@Component
@Setter
```

```java
@Getter
public class Student {
    @Value("백두산")
    private String name;
    @Value("25")
    private int age;
    @Value("등산, 게임, 독서")
    private List<String> hobbys;
    @Value("162.5")
    private double height;
    @Value("49.2")
    private double weight;
}
```

12. StudentInfo.java 수정

```java
package com.example;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import lombok.NoArgsConstructor;
import lombok.Setter;

@NoArgsConstructor
@Component
public class StudentInfo {
    @Setter(onMethod_ = @Autowired)
    private Student student;

    public void printInfo(){
        if(this.student != null){
            System.out.println("Name : " + this.student.getName());
            System.out.println("Age : " + this.student.getAge());
            System.out.println("Hobbies");
            this.student.getHobbys().forEach(hobby -> System.out.println(hobby));
            System.out.println("Height : " + this.student.getHeight());
            System.out.println("Weight : " + this.student.getWeight());
        }else {
            System.out.println("Null");
        }
    }
}
```

13. ApplicationConfig.java 수정

```java
package com.example.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

import com.example.StudentInfo;

@Configuration
@ComponentScan(basePackages = {"com.example"})
public class ApplicationConfig {
    @Bean
    public StudentInfo studentInfo() {
        return new StudentInfo();
    }
}
```

14. MainClass.java 수정

```
package com.example;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import com.example.config.ApplicationConfig;

public class MainClass {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context = new
        AnnotationConfigApplicationContext(ApplicationConfig.class);
        StudentInfo info = context.getBean("studentInfo", StudentInfo.class);
        info.printInfo();
        context.close();
    }
}
```

15. MainClass 실행

```
Name : 백두산
Age : 25
Hobbies
등산, 게임, 독서
Height : 162.5
Weight : 49.2
```