

<참고사항>

제공된 교안 pdf 중 순서
변경이 필요한 부분 정보

Page 73 ~ 101 내용이
Page 34 뒤로 옮겨져야함

02. 스프링의 이해

1. 스프링 프레임워크란?
2. 스프링 개발 환경 구축
3. 스프링 DI와 IoC

Section 01

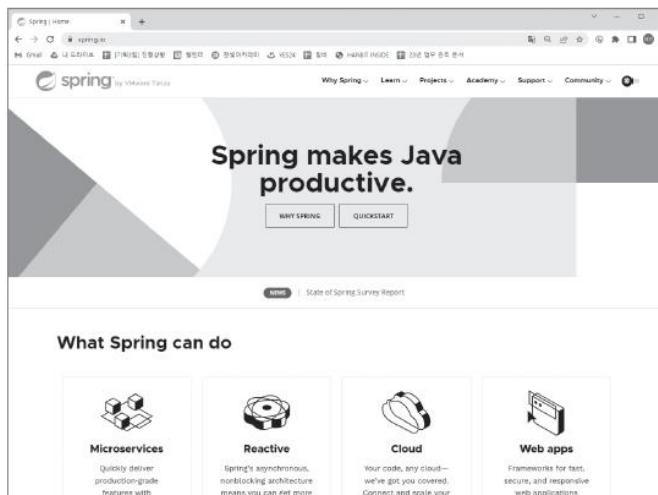
스프링 프레임워크란?

■ 스프링 프레임워크

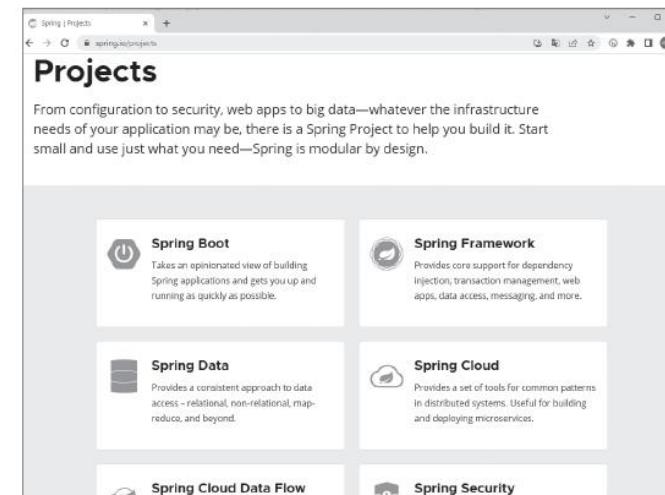
- 자바 기반의 애플리케이션을 개발하기 위한 오픈소스 프레임워크로 일반적으로 줄여서 스프링이라고 함
- 웹 애플리케이션 제작에 뛰어난 강점이 있어 여러 산업 분야에서 널리 사용됨

■ 스프링 MVC

- 스프링을 기반으로 하는 하위 프레임워크
- 웹 애플리케이션 개발에 최적화된 프레임워크



(a) 스프링 공식 홈페이지

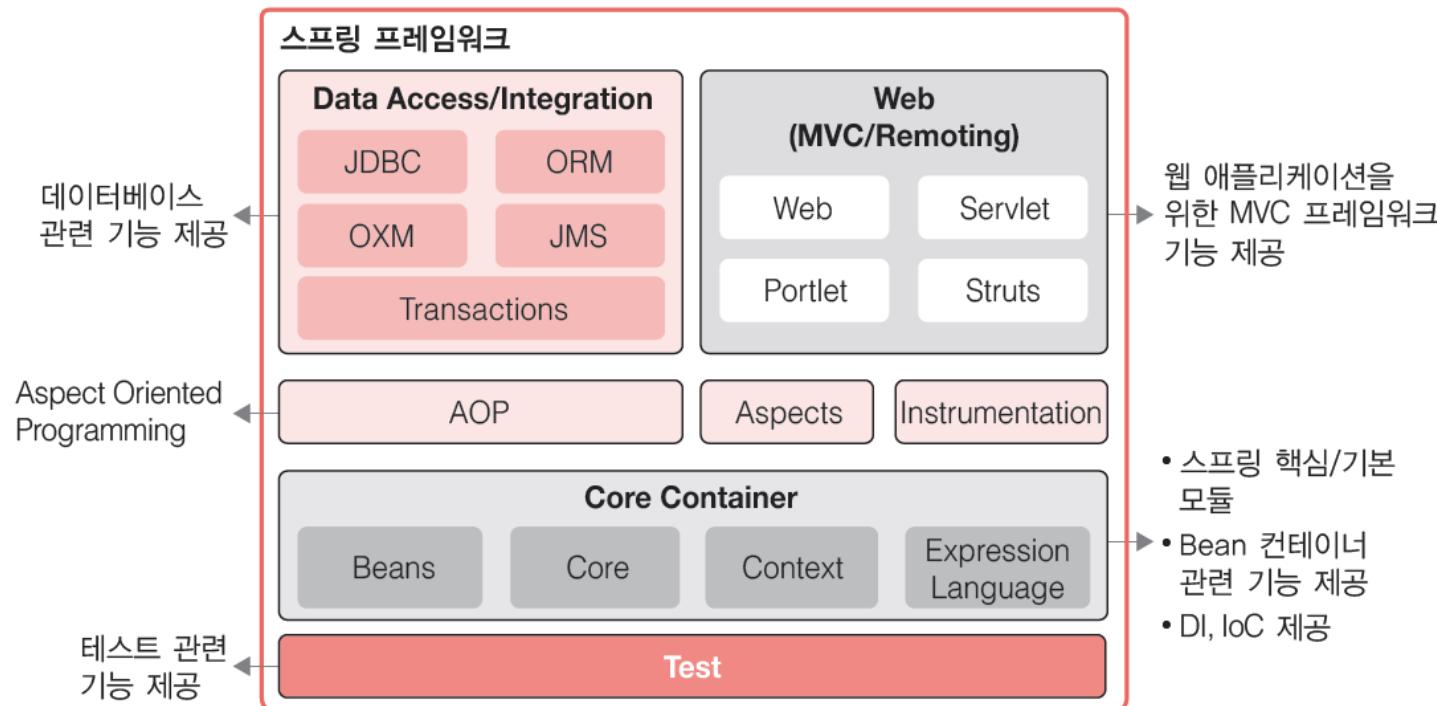


(b) 다양한 스프링 프로젝트

[그림 1] 스프링에서 진행하는 다양한 프로젝트

■ 스프링 프레임워크의 모듈

- 스프링은 다양한 모듈로 이루어져 있음
- 모듈의 종류는 방대하기 때문에 필요할 때 사용할 모듈의 라이브러리를 개발 프로젝트에 가져와 사용하면 됨



[그림 2] 필요에 따라 특정 모듈만 이용할 수 있는 경량 컨테이너인 스프링

■ 스프링에서 제공하는 대표적인 모듈

- 이러한 모듈을 사용하려면 프로젝트에 모듈에 대한 '의존 설정'을 해야 함

표 2-1 스프링 모듈

스프링 모듈명	기능
spring-core	스프링의 핵심인 DI(Dependency Injection)와 IoC(Inversion of Control)를 제공
spring-webmvc	스프링에서 제공하는 컨트롤러(Controller)와 뷰(View)를 이용한 스프링 MVC 구현 기능 제공
spring-jdbc	데이터베이스를 쉽게(적은 양의 코드) 다룰 수 있는 기능 제공
spring-tx	스프링에서 제공하는 트랜잭션 관련 기능 제공
spring-security	애플리케이션 보안을 담당하는 기능 제공

Section 02

스프링 개발 환경 구축

■ 실습환경

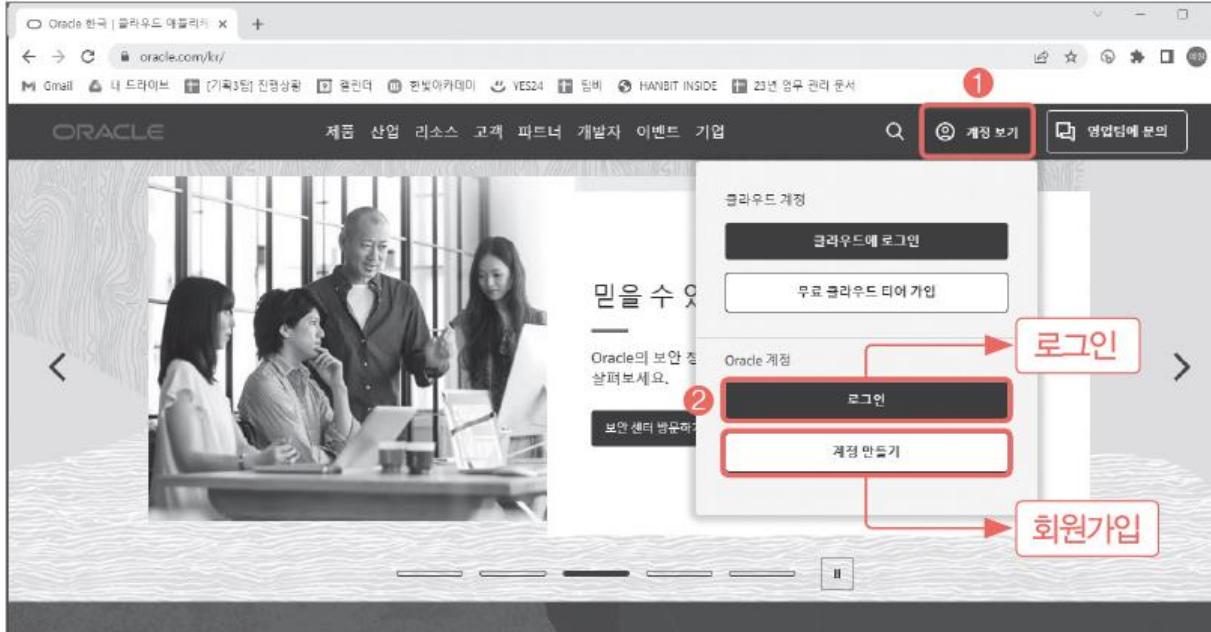
항목	프로그램	이 책의 버전
자바 개발도구	JDK	JDK 11
통합개발환경	이클립스(2~8장)	이클립스 2022-12-R
	STS(9~14장)	STS3
서블릿 컨테이너	아파치 톰캣	Apache Tomcat 9
데이터베이스	MariaDB	MariaDB Server 10

- JDK11을 사용하는 이유

- ✓ 만약 컴퓨터에 JDK11이 아닌 다른 버전이 설치되어 있다면 '프로그램 추가/제거'를 이용해서 제거 후 JDK11을 설치해야 함
- ✓ 9장부터 사용하는 STS3 IDE에서 JDK11을 기반으로 실습이 진행하는데, 만약 JDK11이 아닌 다른 버전을 사용하면 실습에 문제가 발생할 수 있기 때문임

■ 1. 웹 브라우저에서 오라클 홈페이지(<https://www.oracle.com/kr/index.html>)에 접속하기

- 설치 파일을 내려받기 위해서는 오라클 계정이 있어야 함
- 오른쪽 상단의 '계정 보기'를 클릭하여 계정이 있다면 <로그인>을, 계정이 없다면 <계정 만들기>를 클릭



- 페이지 하단의 Java 11을 선택하고 Windows를 선택한 뒤 jdk-11.0.14_windows-x64_bin.exe를 클릭
- 라이선스에 동의하고 <Download jdk-11.0.14_windows-x64_bin.exe> 버튼을 클릭하여 다운로드

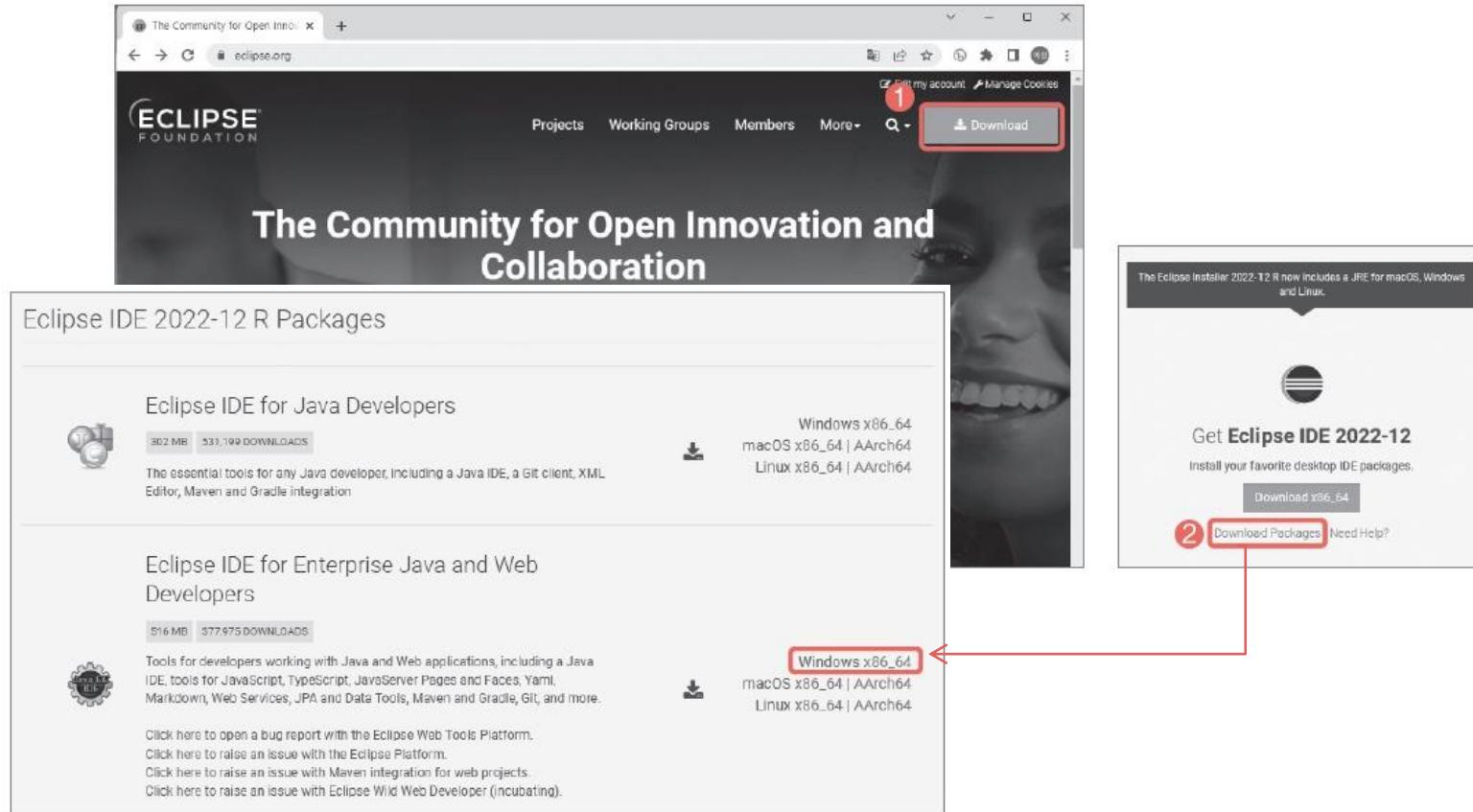
The screenshot shows the Java SE Development Kit 11.0.14 download page. At the top, there is a 'Java 11' tab with a red box and a circled '1'. On the left, there is a 'Windows' tab with a red box and a circled '2'. In the center, there is a checkbox labeled 'I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE' with a red box and a circled '4'. At the bottom right, there is a 'Download jdk-11.0.14_windows-x64_bin.exe' button with a red box and a circled '5'. A red box also highlights the download link for the x64 Compressed Archive at the bottom.

Product/file description	File size	Download
x64 Installer	140.24 MB	jdk-11.0.14_windows-x64_bin.exe (3)
x64 Compressed Archive	157.79 MB	jdk-11.0.14_windows-x64_bin.zip

2. 이클립스 설치하기

KITRI 2024 교안 - Song Young Ohk

- 1. 웹 브라우저에서 이클립스(<https://www.eclipse.org/>) 홈페이지에 접속하여 우측 상단의 <Download> 버튼을 클릭하기
 - 이어지는 화면에서 <Download Package>를 클릭하고 설치 진행
- 2. 2022-12 R Packages의 'Eclipse IDE for Enterprise Java and Web Developers'를 선택하고, 'Windows x86_64'를 클릭하여 다운로드



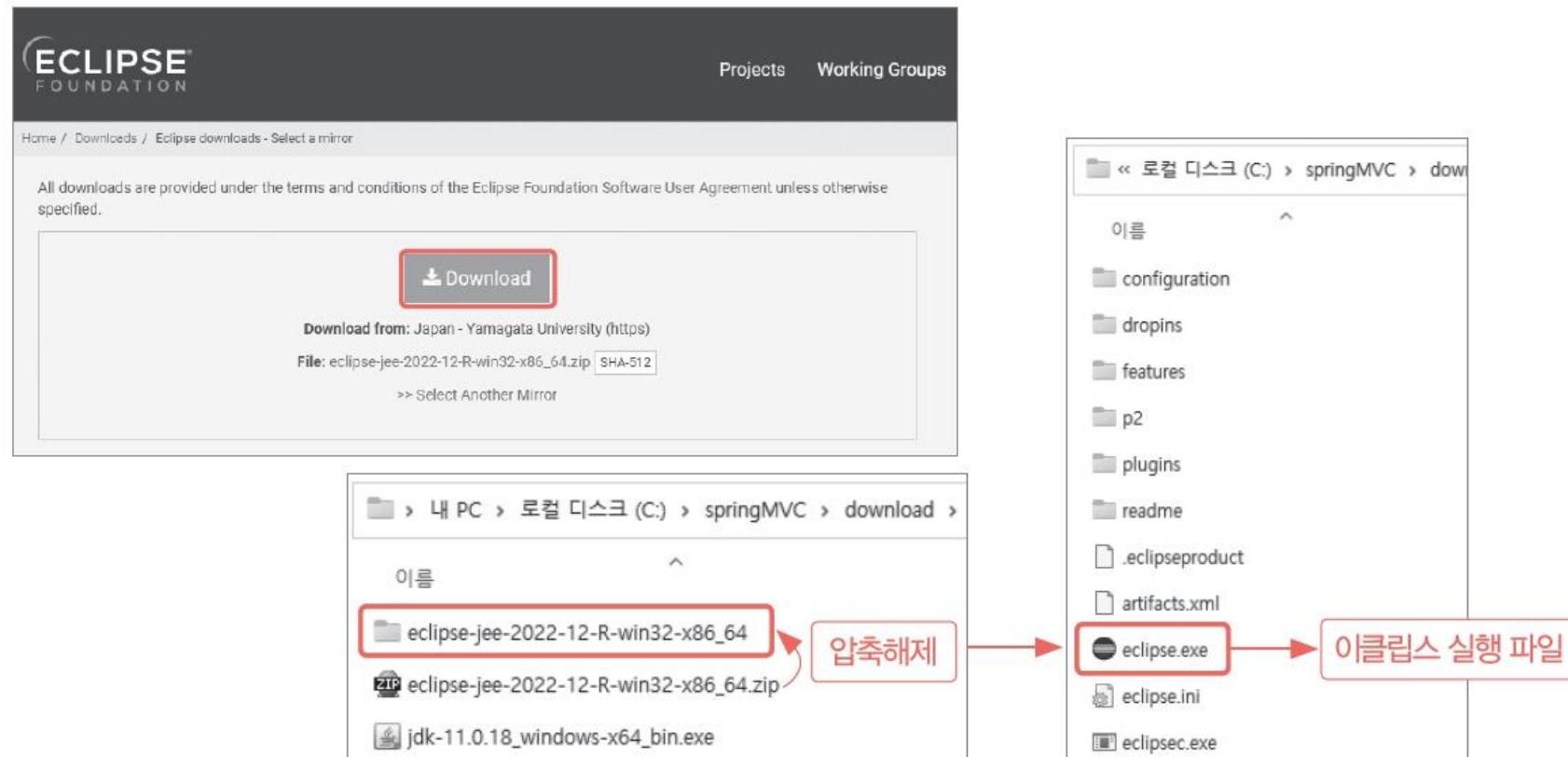
2. 이클립스 설치하기

KITRI 2024 교안 - Song Young Ohk

■ 3. <Download> 버튼을 클릭하여 eclipse-jee-2022-12-R-win32-x86_64.zip 파일 다운로드하기

- 다운로드 위치: C:\springMVC\download

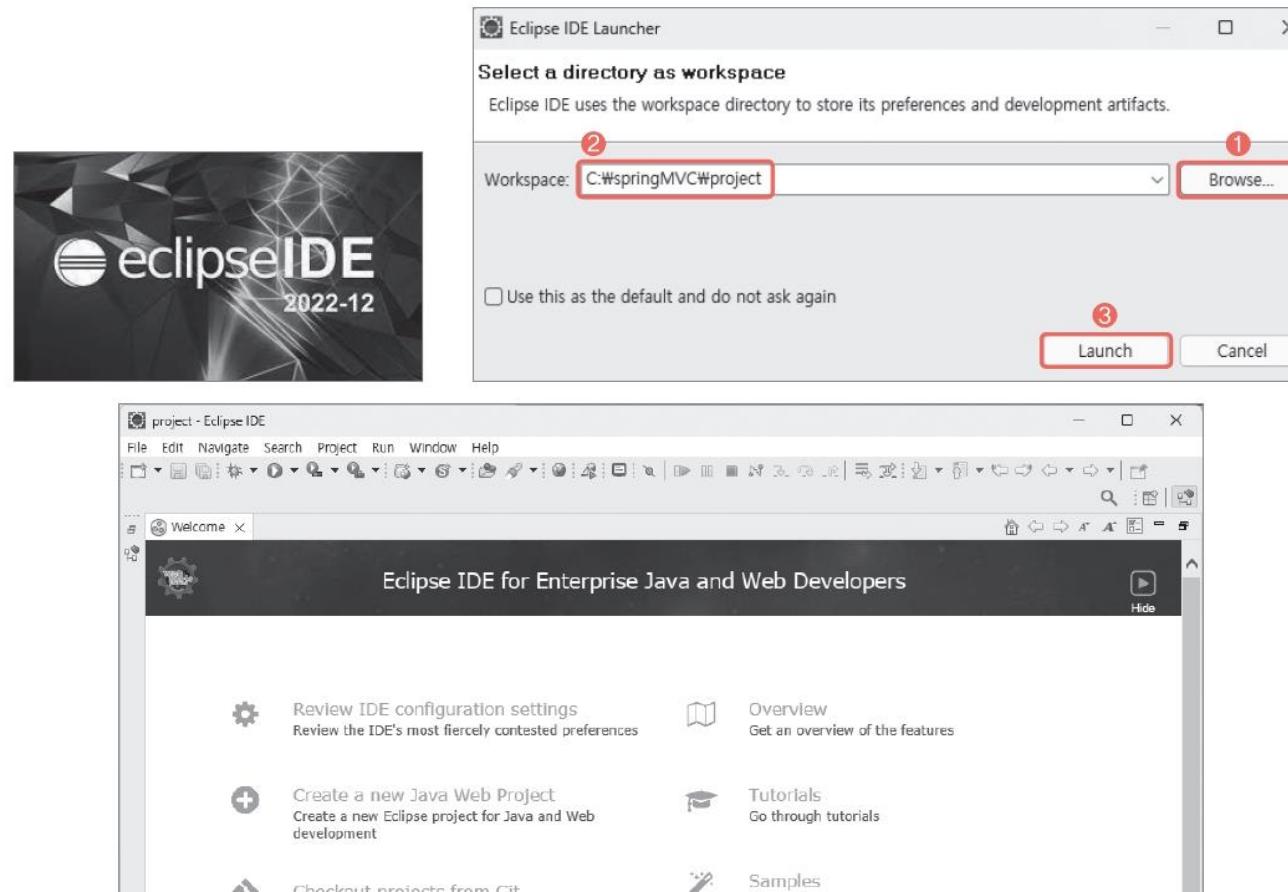
■ 4. 다운로드한 이클립스는 압축파일로, 압축 해제하여 이클립스 실행 파일(eclipse.exe)을 더블클릭해서 이클립스를 실행하기



2. 이클립스 설치하기

KITRI 2024 교안 - Song Young Ohk

- 5. 이클립스 로고 창이 잠시 보이고, Workspace를 설정하는 창이 뜨면 <Browse...>를 클릭하여 Workspace 경로를 변경하고 <Launch> 클릭하기
 - Workspace 경로: ‘C:\springMVC\project’
- 6. Welcome 페이지가 뜨면 이클립스가 정상적으로 실행된 것임



■ 이클립스의 자바 버전(JDK) 설정하기

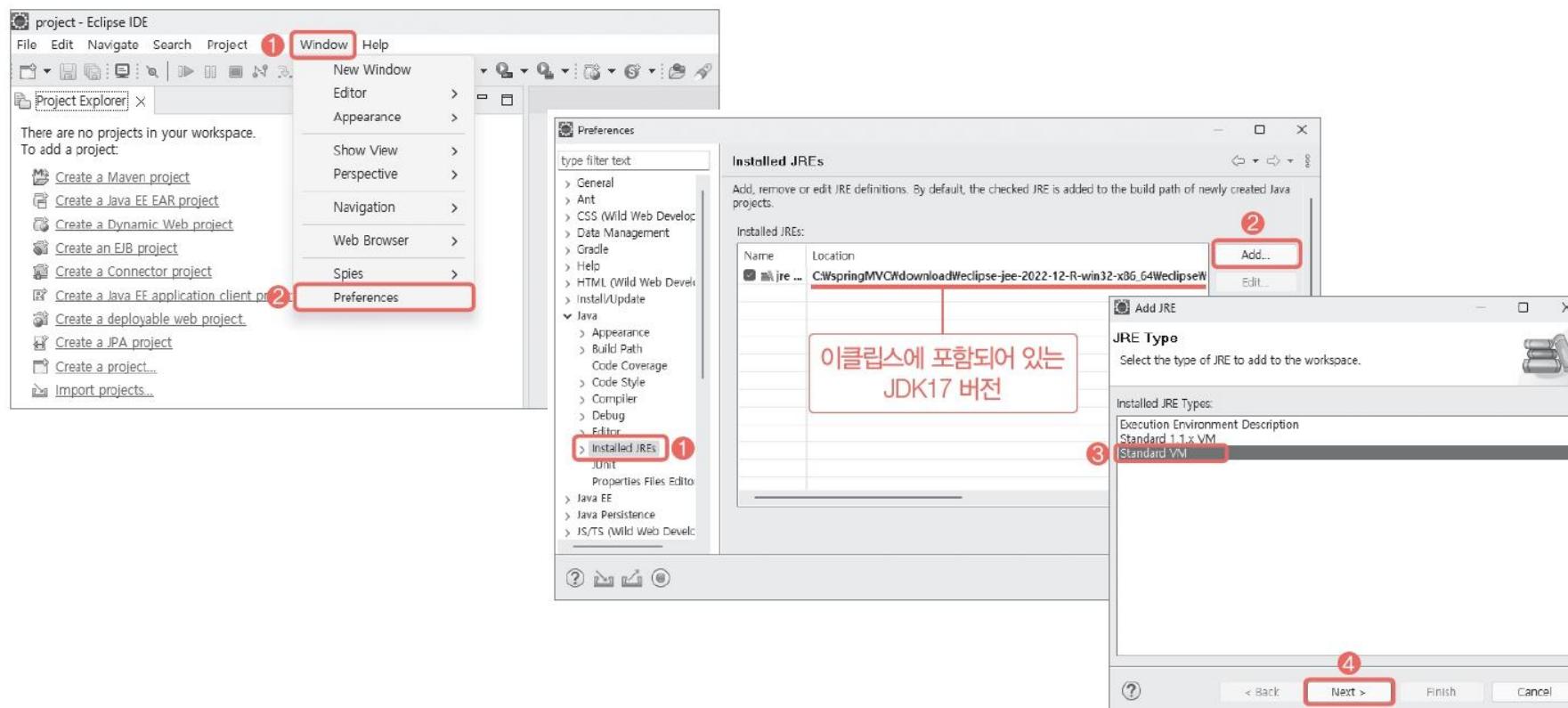
- 자바 JDK11과 이클립스 2022-12-R을 설치했음
- 그런데 이클립스 2022-12-R에는 자바의 최신 버전인 17이 내장되어 있기 때문에 이클립스의 자바 버전을 JDK17이 아닌 JDK11로 변경해야 함

2. 이클립스 설치하기

KITRI 2024 교안 - Song Young Ohk

■ 이클립스의 자바 버전(JDK) 설정하기

- 1. 이클립스 상단 메뉴에서 [Window]-[Preferences]를 선택
- 2. [Preferences] 창 왼쪽 메뉴에서 [Java]-[Installed JREs]를 클릭해 이클립스에 기본적으로 포함되어 있는 JDK17의 경로를 확인하기
 - ✓ <Add...>를 클릭하고, [Add JRE] 창에서 Standard VM을 선택한 후 <Next> 클릭

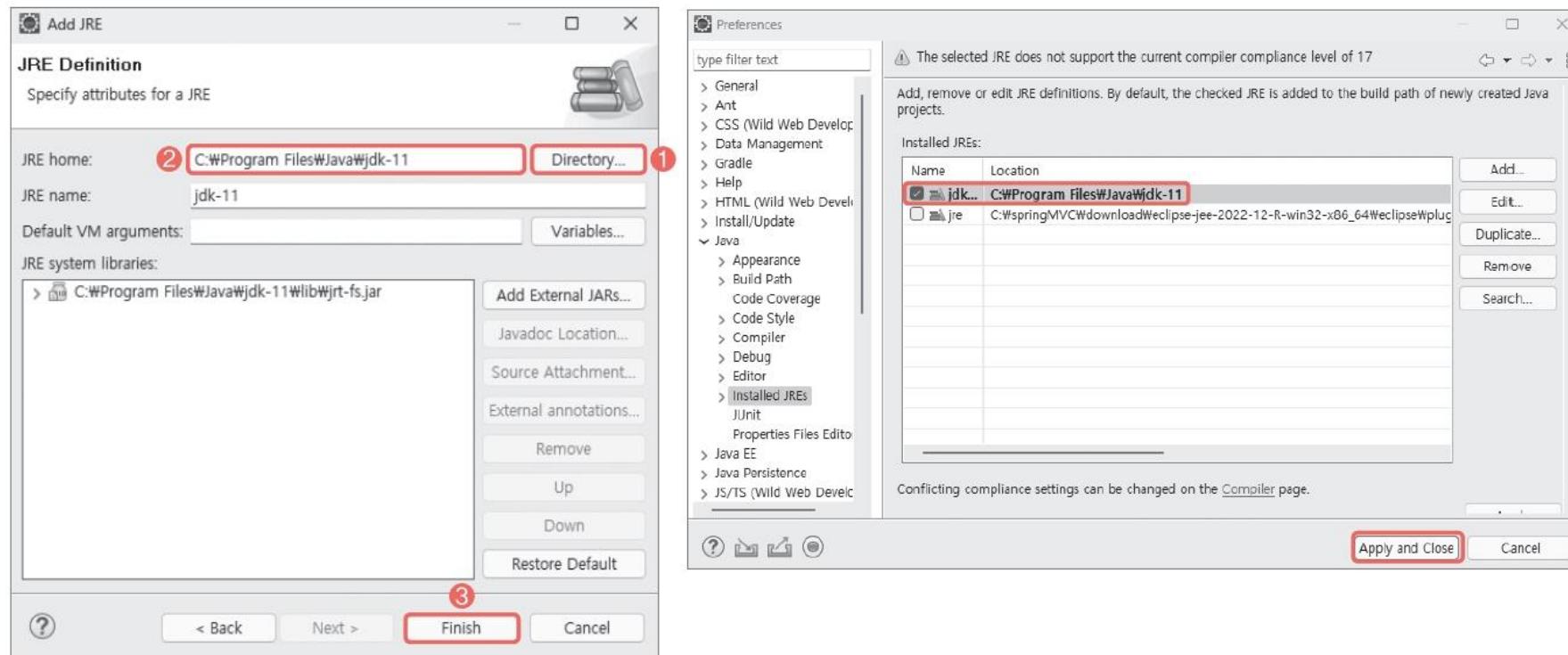


2. 이클립스 설치하기

KITRI 2024 교안 - Song Young Ohk

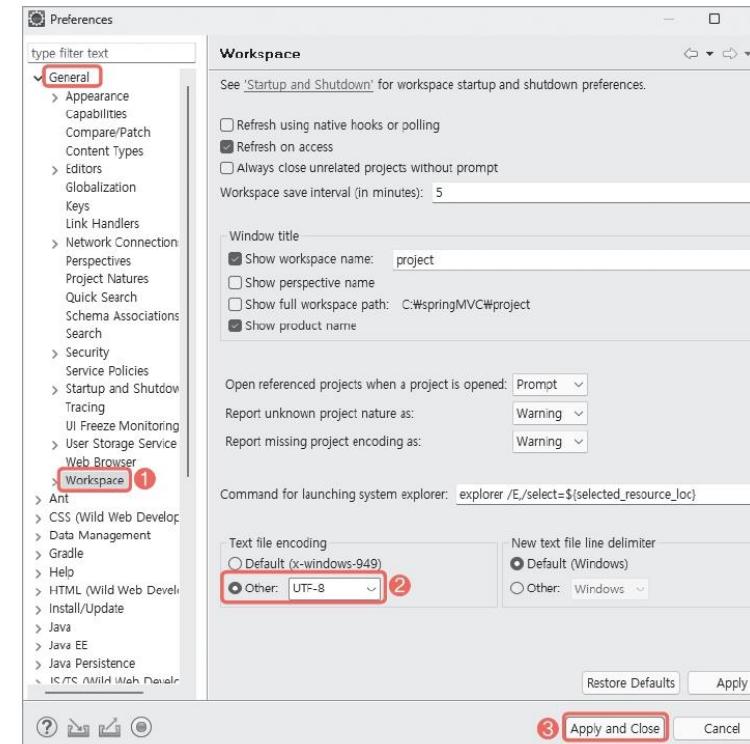
■ 이클립스의 자바 버전(JDK) 설정하기

- 3. [JRE Definition] 창에서 <Directory>를 클릭하여 JRE home을 앞에서 설치한 JDK11 경로 설정하고 <Finish> 클릭
 - ✓ JDK11 경로: C:\Program Files\Java\jdk-11.0.14
- 4. 추가된 jdk-11의 체크박스에 체크하여 default로 설정하고, <Apply and Close>를 클릭해서 설정 종료



■ 인코딩 설정하기(UTF-8)

- 이클립스는 기본적으로 인코딩이 MS-949로 되어 있음. 이것을 UTF-8로 변경하기
- 1. 이클립스의 [Window]-[Preferences]를 클릭
- 2. [Preferences] 창의 왼쪽 메뉴에서 [General]-[Workspace] 선택
- 3. 화면 하단의 Text file encoding에서 MS-949를 UTF-8로 변경하고, <Apply and Close>를 클릭해서 인코딩 설정을 종료하기



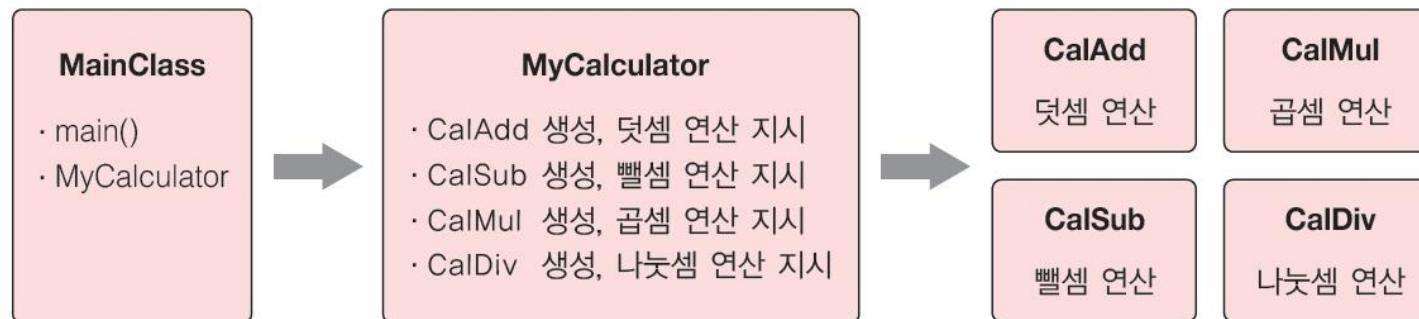
Section 03

스프링 DI와 IoC

■ 스프링 DI와 IoC 학습을 위한 간단한 계산기 자바 프로젝트

- 계산기 자바 프로젝트의 구조

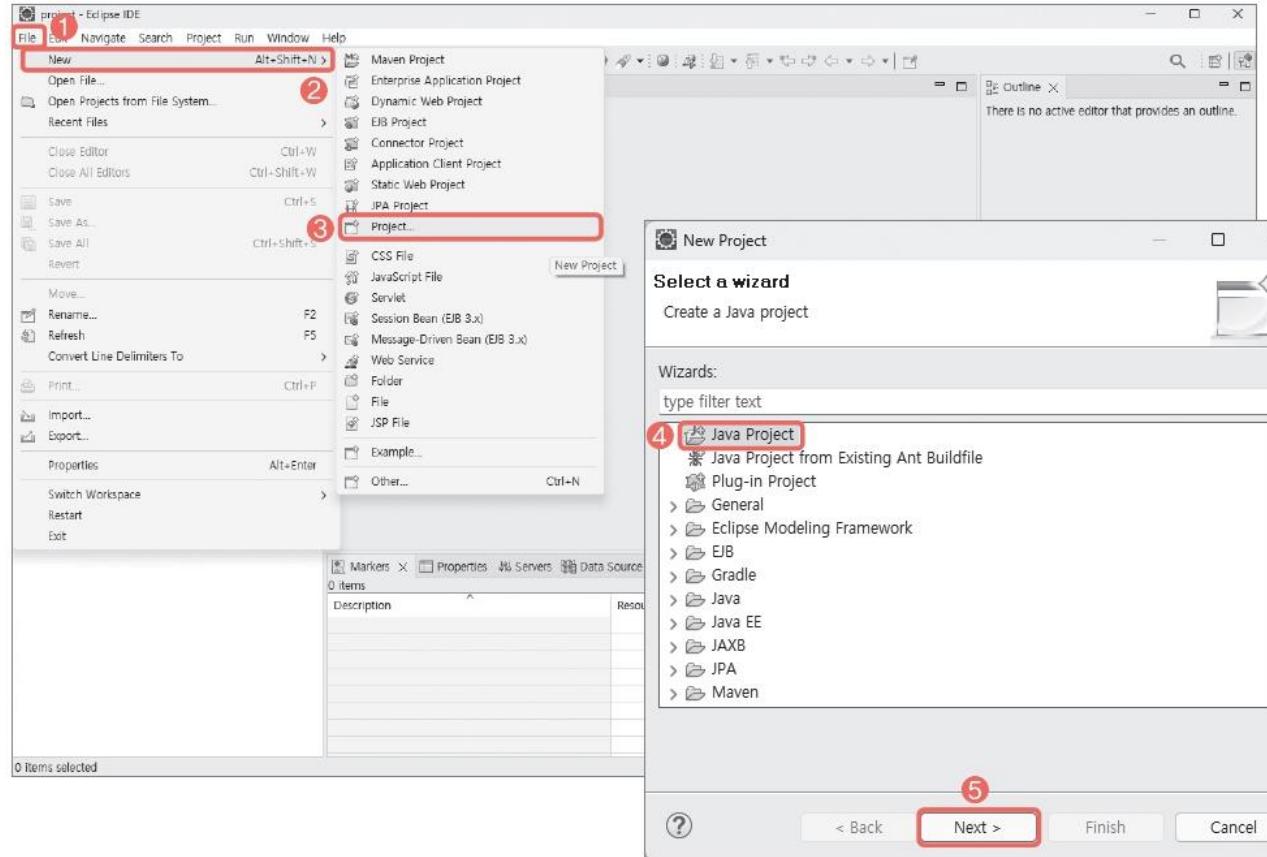
- ✓ MainClass의 main()에서 프로그램이 시작되면 MyCalculator를 생성
- ✓ MyCalculator는 덧셈, 뺄셈, 곱셈, 나눗셈을 위한 각각의 객체(CalAdd, CalSub, CalMul, CalDiv)를 생성
- ✓ 생성된 객체는 내부에서 사칙연산을 실행



[그림 3] 계산기 프로젝트(MyCalculator)의 구성

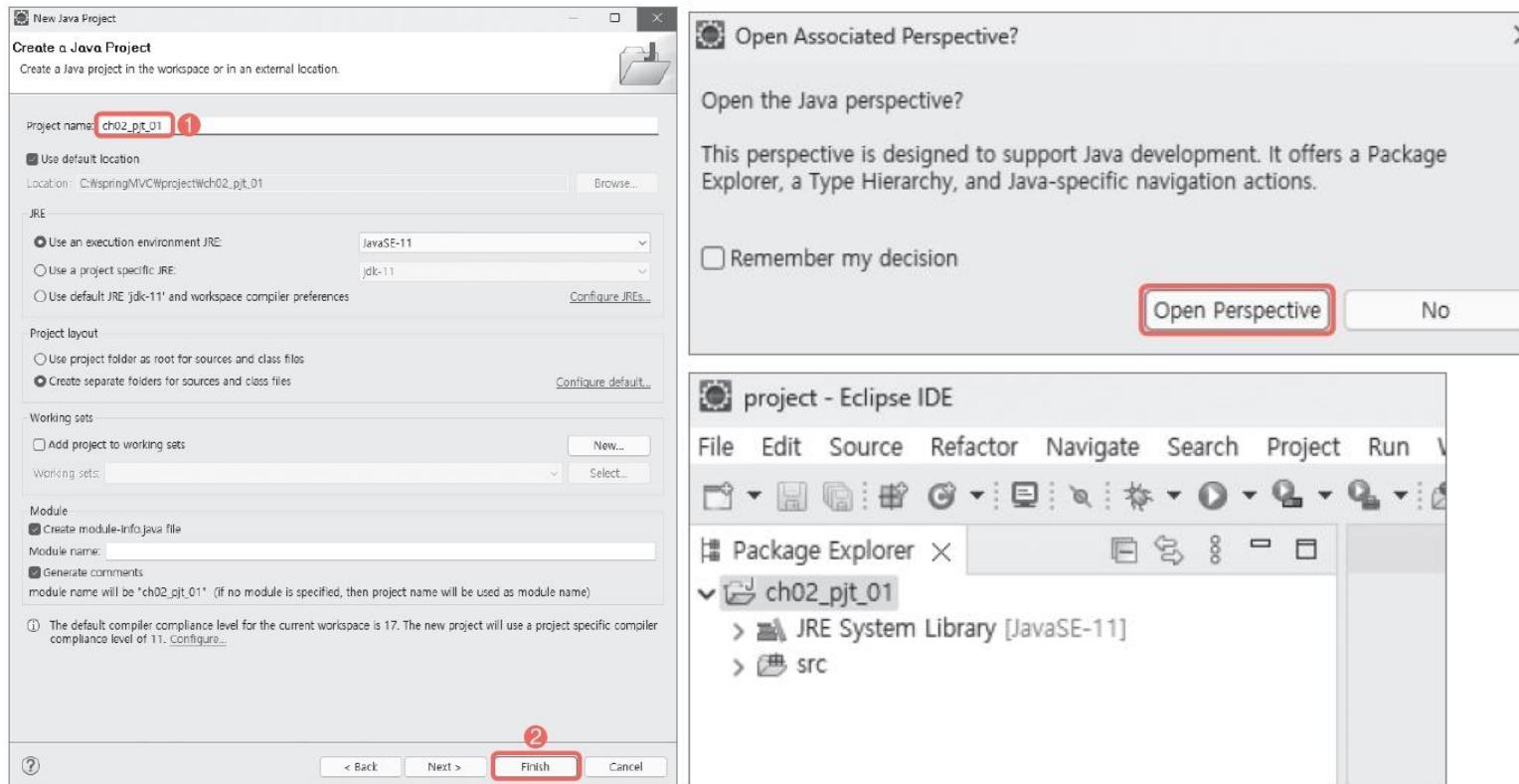
■ ch02_pjt_01 프로젝트 생성하기

- 1. [File]-[New]-[Project]를 클릭하고, [New Project] 창에서 'Java Project'를 선택하고 <Next> 클릭



■ ch02_pjt_01 프로젝트 생성하기

- 2. [New Java Project] 창에서 Project name을 ch02_pjt_01로 하고 <Finish> 클릭
- 3. [New module-info.java] 창이 나오면 <Don't Create>를 클릭
 - ✓ 만약 [Open Associated Perspective?] 창이 나오면 <Open Perspective>를 클릭
- 4. 프로젝트가 정상적으로 생성된 것을 확인하기

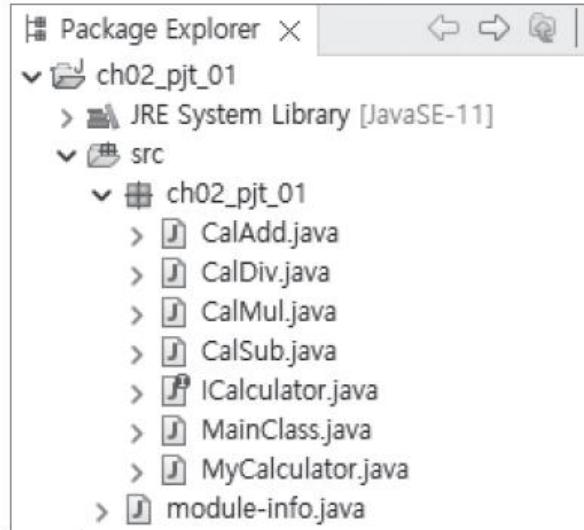


1. 계산기 프로젝트 만들기

KITRI 2024 교안 - Song Young Ohk

■ 클래스와 인터페이스 만들기

- 생성된 프로젝트에 다음과 같이 클래스와 인터페이스 생성하기



(a) 생성 완료된 전체 프로젝트 구조

클래스	인터페이스
MainClass.java	ICalculator.java
MyCalculator.java	
CalAdd.java	
CalSub.java	
CalMul.java	
CalDiv.java	

(b) 생성할 클래스와 인터페이스

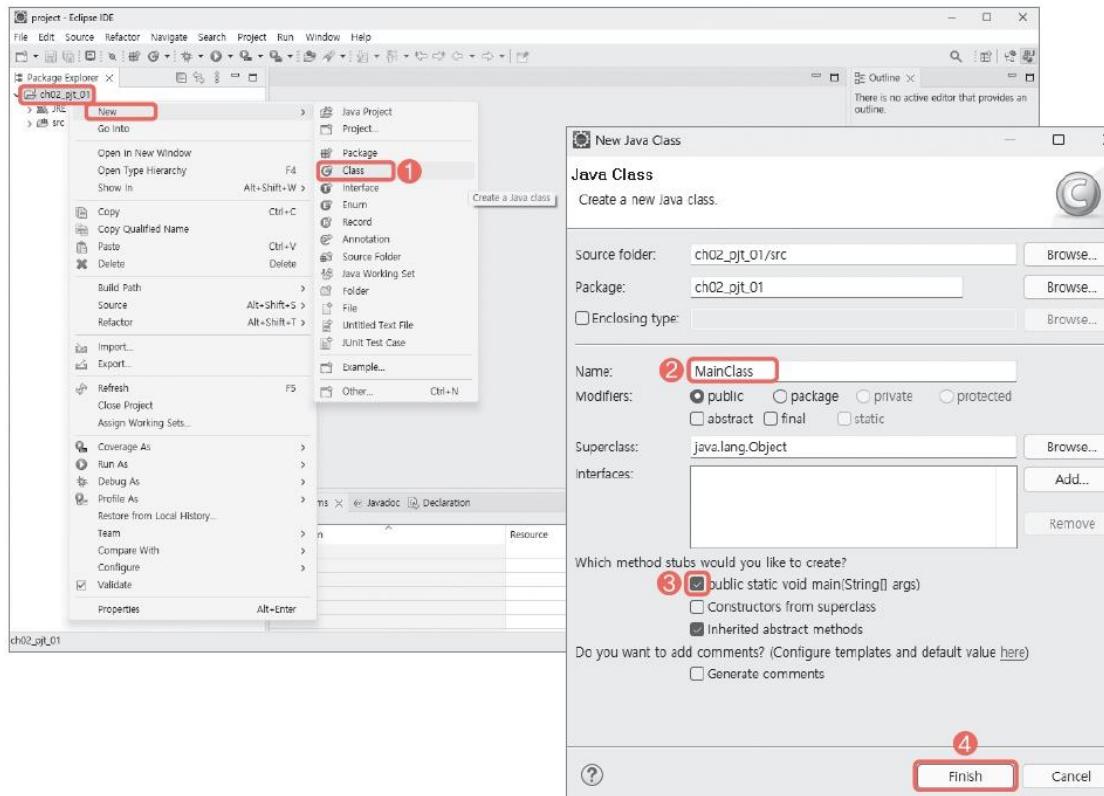
[그림 4] 계산기 프로젝트에 필요한 클래스와 인터페이스

1. 계산기 프로젝트 만들기

KITRI 2024 교안 - Song Young Ohk

■ 클래스 생성

- ch02_pjt_01에서 마우스 오른쪽 버튼을 클릭하고 [New]-[Class]를 선택하기. Name에 생성할 클래스 이름 입력
 - ✓ MainClass를 생성할 때 public static void main (String[] args)에 체크하면 main() 메서드를 자동으로 생성해줌
 - ✓ 그 외의 클래스를 생성할 때는 체크하지 않음

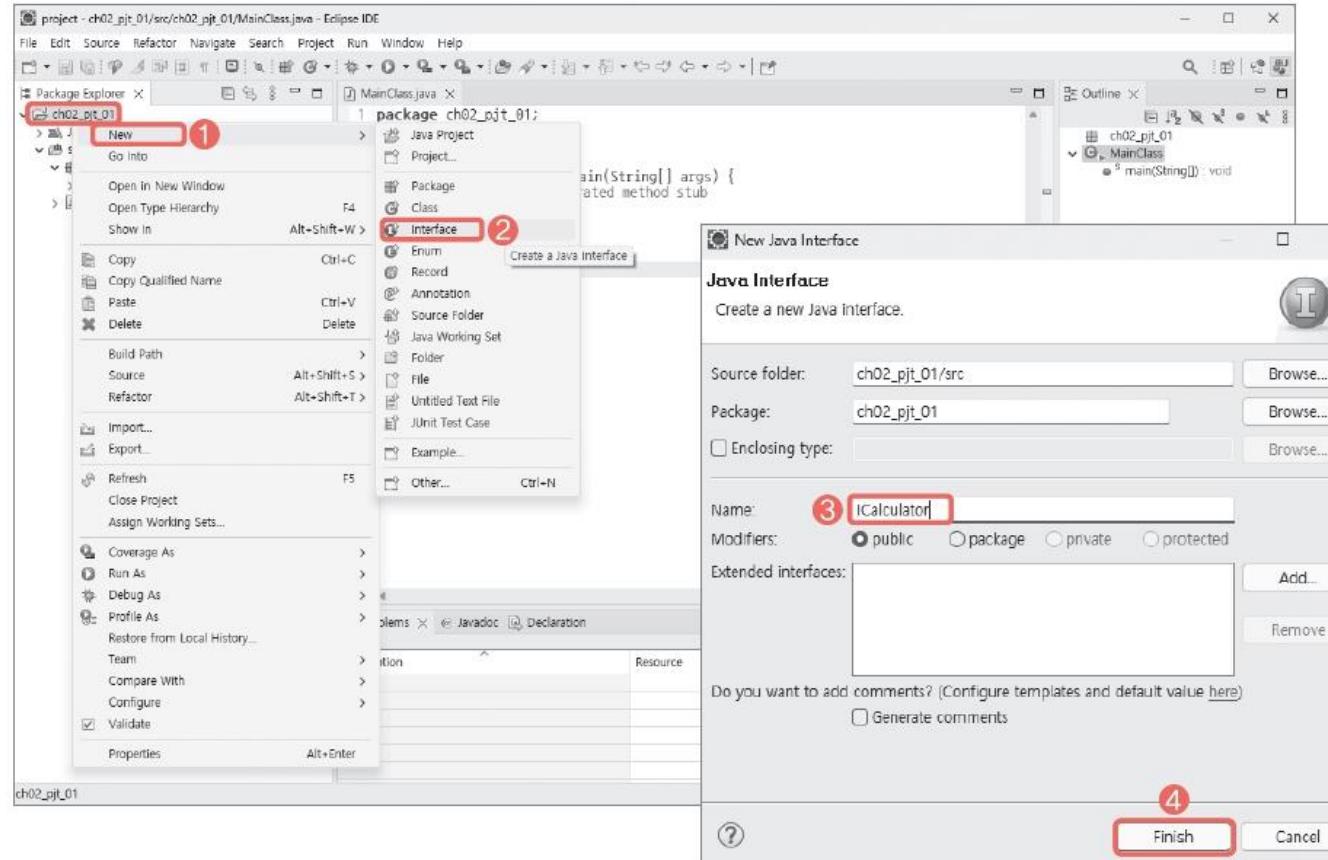


1. 계산기 프로젝트 만들기

KITRI 2024 교안 - Song Young Ohk

■ 인터페이스 생성

- ch02_pjt_01에서 마우스 오른쪽 버튼을 클릭하고 [New]-[Interface]를 선택하고, Name에 ICalculator 입력



1. 계산기 프로젝트 만들기

KITRI 2024 교안 - Song Young Ohk

MainClass 클래스

코드 2-1

```
01 package ch02_pjt_01;
02 public class MainClass {
03     public static void main(String[] args) {
04
05         MyCalculator calculator = new MyCalculator();
06         calculator.calAdd(10, 5);
07         calculator.calSub(10, 5);
08         calculator.calMul(10, 5);
09         calculator.calDiv(10, 5);
10
11     }
12 }
```

ch02_pjt_01\src\ch02_pjt_01\MainClass.java

코드 2-2

```
01 package ch02_pjt_01;
02 public class MyCalculator {
03
04     public void calAdd(int fNum, int sNum) {
05         ICalculator calculator = new CalAdd();          // CalAdd 객체 생성
06         int value = calculator.doOperation(fNum, sNum); // 덧셈 실행
07         System.out.println("result : " + value);
08     }
09
10    public void calSub(int fNum, int sNum) {
11        ICalculator calculator = new CalSub();          // CalSub 객체 생성
12        int value = calculator.doOperation(fNum, sNum); // 뺄셈 실행
13        System.out.println("result : " + value);
14    }
15
16    public void calMul(int fNum, int sNum) {
17        ICalculator calculator = new CalMul();          // CalMul 객체 생성
18        int value = calculator.doOperation(fNum, sNum); // 곱셈 실행
19        System.out.println("result : " + value);
20    }
21
22    public void calDiv(int fNum, int sNum) {
23        ICalculator calculator = new CalDiv();          // CalDiv 객체 생성
24        int value = calculator.doOperation(fNum, sNum); // 나눗셈 실행
25        System.out.println("result : " + value);
26    }
27 }
```

MyCalculator 클래스

ch02_pjt_01\src\ch02_pjt_01\MyCalculator.java

1. 계산기 프로젝트 만들기

KITRI 2024 교안 - Song Young Ohk

■ ICalculator 인터페이스

코드 2-3

```
01 package ch02_pjt_01;
02 public interface ICalculator {
03     public int doOperation(int firstNum, int secondNum);
04 }
```

ch02_pjt_01\src\ch02_pjt_01\ICalculator.java

■ CalAdd 클래스

코드 2-4

```
01 package ch02_pjt_01;
02 public class CalAdd implements ICalculator {
03     @Override
04     public int doOperation(int firstNum, int secondNum) {
05         return firstNum + secondNum;
06     }
07 }
```

ch02_pjt_01\src\ch02_pjt_01\CalAdd.java

■ CalSub 클래스

코드 2-5

```
01 package ch02_pjt_01;
02 public class CalSub implements ICalculator {
03     @Override
04     public int doOperation(int firstNum, int secondNum) {
05         return firstNum - secondNum;
06     }
07 }
```

ch02_pjt_01\src\ch02_pjt_01\CalSub.java

■ CalMul 클래스

코드 2-6

```
01 package ch02_pjt_01;
02 public class CalMul implements ICalculator {
03     @Override
04     public int doOperation(int firstNum, int secondNum) {
05         return firstNum * secondNum;
06     }
07 }
```

ch02_pjt_01\src\ch02_pjt_01\CalMul.java

■ CalDiv 클래스

코드 2-7

ch02_pjt_01\src\ch02_pjt_01\CalDiv.java

```
01 package ch02_pjt_01;
02 public class CalDiv implements ICalculator {
03     @Override
04     public int doOperation(int firstNum, int secondNum) {
05         return secondNum != 0 ? (firstNum / secondNum) : 0;
06     }
07 }
```

실행 결과

```
result : 15
result : 5
result : 50
result : 2
```

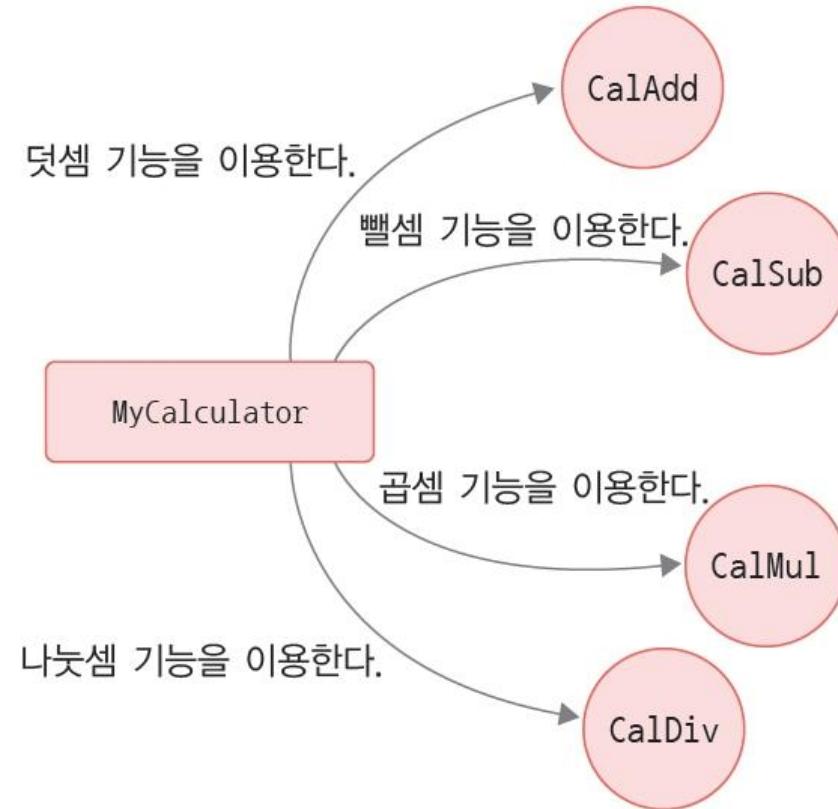
■ DI(Dependency Injection)

- 의존성 주입이라고 함
- 의존성 주입이란 필요한(의존하는) 객체를 직접 생성하지 않고 외부에서 주입하는 방식을 의미함

■ 의존의 개념

- 계산기 프로그램을 실행하면 main()([코드 2-2])에서 MyCalculator를 생성하고 calAdd(), calSub(), calMul(), calDiv()를 호출함
- 그리고 이 메서드들은 동일하게 연산에 필요한 객체를 직접 생성함
- MyCalculator에서 연산에 필요한 객체를 생성하는 것을 'MyCalculator는 CalAdd, CalSub, CalMul, CalDiv 객체를 이용한다'라고 함
- 즉 MyCalculator는 자신이 직접 연산하지 않고, 각각의 연산 객체들(CalAdd, CalSub, CalMul, CalDiv)에게 연산 업무를 전달함

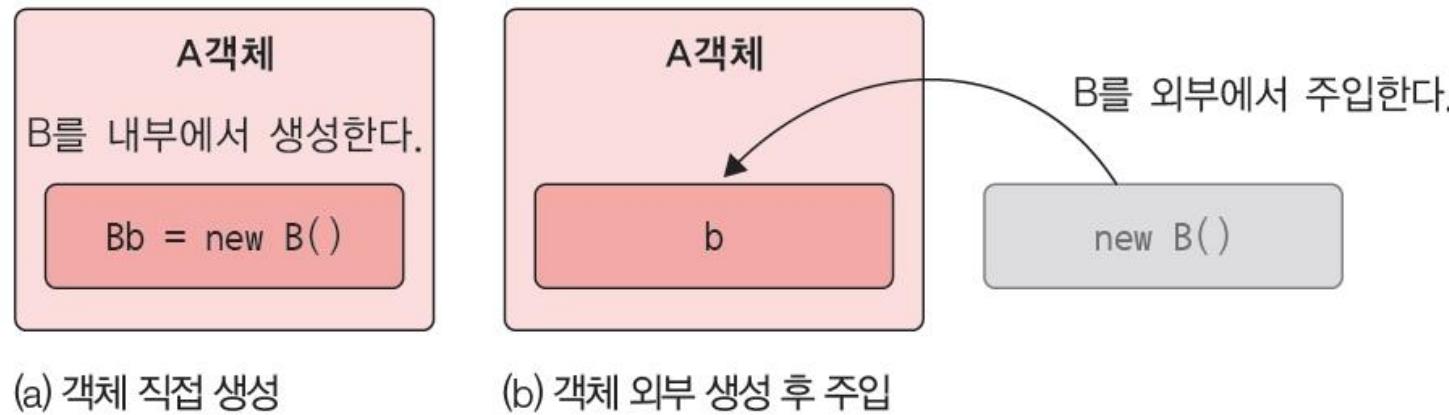
■ 의존의 개념



[그림 5] MyCalculator에서 연산 업무를 전달하는 형태

■ 의존의 개념

- MyCalculator가 다른 객체들을 이용한다'는 것을 다르게 표현해보기
- CalAdd 입장에서는 main() 함수가 MyCalculator에 지시한 덧셈 연산 업무를 MyCalculator를 대신해서 덧셈 연산을 처리하고 있음
- 즉, 'MyCalculator는 CalAdd에 의존한다'라고 할 수 있음



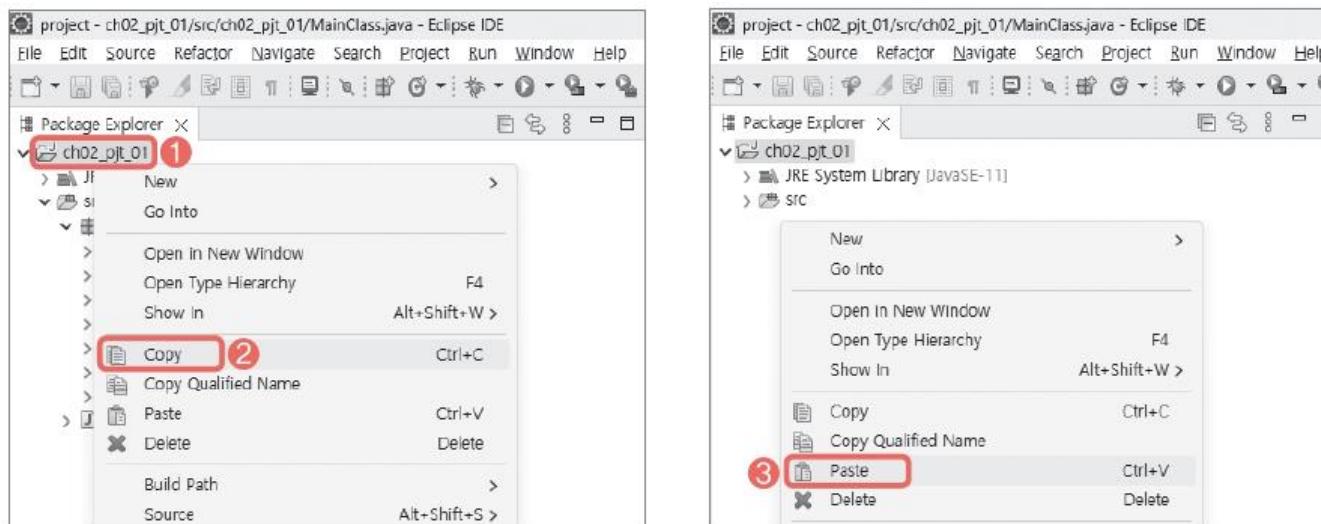
[그림 6] 필요한 객체를 내부 또는 외부에서 생성하는 주입함

■ ch02_pjt_01 프로젝트를 외부에서 주입하는 방식으로 변경하기

- ch02_pjt_01을 ch02_pjt_02로 복사해서 CalAdd, CalSub, CalMul, CalDiv를 MyCalculator가 직접 생성하지 않고 외부에서 주입하는 방식으로 변경하기

■ ch02_pjt_02 생성하기

- 1. ch02_pjt_01에서 마우스 오른쪽 버튼을 클릭하고 [Copy]를 선택하고, Package Explorer의 흰색 공간에서 마우스 오른쪽 버튼을 클릭하고 [Paste]를 선택

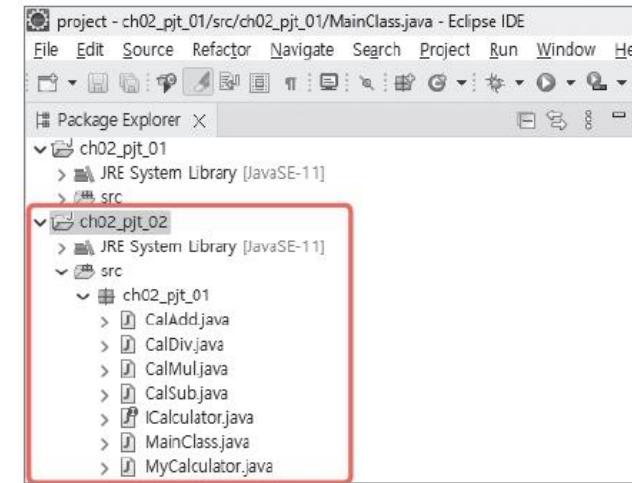
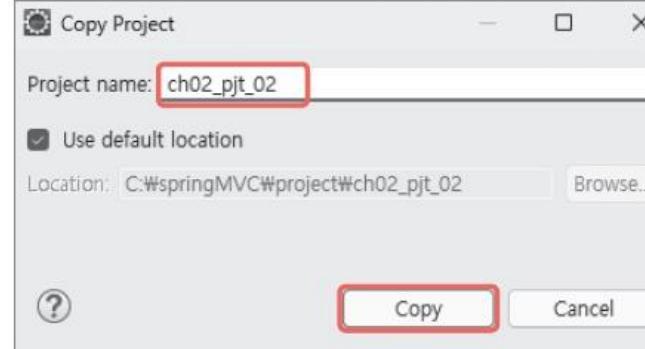


3. 계산기 프로그램을 DI 방식으로 변경하기

KITRI 2024 교안 - Song Young Ohk

■ ch02_pjt_02 생성하기

- 2. [Copy Project] 창의 Project name에 ch02_pjt_02를 입력하고 <Copy> 클릭
- 3. ch02_pjt_02가 생성된 것을 확인하기



3. 계산기 프로그램을 DI 방식으로 변경하기

KITRI 2024 교안 - Song Young Ohk

■ MyCalculator 클래스 수정하기

- MyCalculator가 의존하는 CalAdd, CalSub, CalMul, CalDiv를 MyCalculator가 직접 생성하지 않고 외부에서 주입하는 방식으로 변경하기 위해 수정함
- 이를 매개변수를 이용해서 연산에 필요한 객체를 외부에서 받으면 됨
- **calAdd(), calSub(), calMul(), calDiv() 코드 수정하기**
 - ✓ MyCalculator 클래스가 CalAdd 클래스를 외부에서 주입받을 수 있도록 [코드 2-2]의 calAdd(), calSub(), calMul(), calDiv() 부분을 수정

코드 2-8

ch02_pjt_01\src\ch02_pjt_02\MyCalculator.java

```
01 package ch02_pjt_01;
02 public class MyCalculator {
03
04     public void calAdd(int fNum, int sNum, CalAdd calAdd) { // CalAdd 객체 주입
05         int value = calAdd.doOperation(fNum, sNum);           // 덧셈 실행
06         System.out.println("result : " + value);
07     }
08
09     public void calSub(int fNum, int sNum, CalSub calSub) { // CalSub 객체 주입
10        int value = calSub.doOperation(fNum, sNum);          // 뺄셈 실행
11        System.out.println("result : " + value);
12    }
13
14     public void calMul(int fNum, int sNum, CalMul calMul) { // CalMul 객체 주입
15        int value = calMul.doOperation(fNum, sNum);          // 곱셈 실행
16        System.out.println("result : " + value);
17    }
18
19     public void calDiv(int fNum, int sNum, CalDiv calDiv) { // CalDiv 객체 주입
20        int value = calDiv.doOperation(fNum, sNum);          // 나눗셈 실행
21        System.out.println("result : " + value);
22    }
23 }
```

3. 계산기 프로그램을 DI 방식으로 변경하기

KITRI 2024 교안 - Song Young Ohk

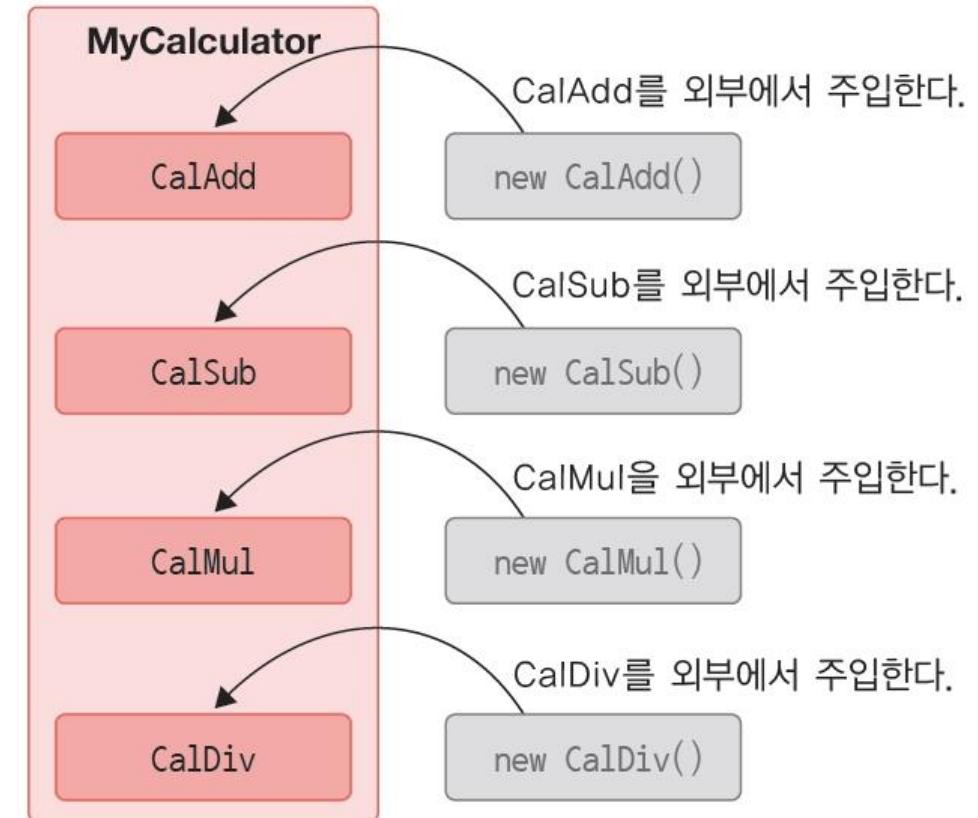
■ MainClass 수정하기

- MyCalculator 수정이 끝났다면 MyCalculator를 사용하는 MainClass도 수정
- main()에서 MyCalculator를 사용할 때 예전과 달리 연산에 필요한 객체들(CalAdd, CalSub, CalMul, CalDiv)을 주입해야 함

코드 2-9

```
ch02_pjt_01\src\ch02_pjt_02\MainClass.java  
01 package ch02_pjt_01;  
02 public class MainClass {  
03     public static void main(String[] args) {  
04         MyCalculator calculator = new MyCalculator();  
05  
06         calculator.calAdd(10, 5, new CalAdd()); // 객체 주입  
07         calculator.calSub(10, 5, new CalSub()); // 객체 주입  
08         calculator.calMul(10, 5, new CalMul()); // 객체 주입  
09         calculator.calDiv(10, 5, new CalDiv()); // 객체 주입  
10     }  
11 }  
12 }
```

■ MainClass 수정하기



[그림 7] 연산에 필요한 객체를 외부에서 주입받는 형태의 MyCalculator 구조

3. 계산기 프로그램을 DI 방식으로 변경하기

KITRI 2024 교안 - Song Young Ohk

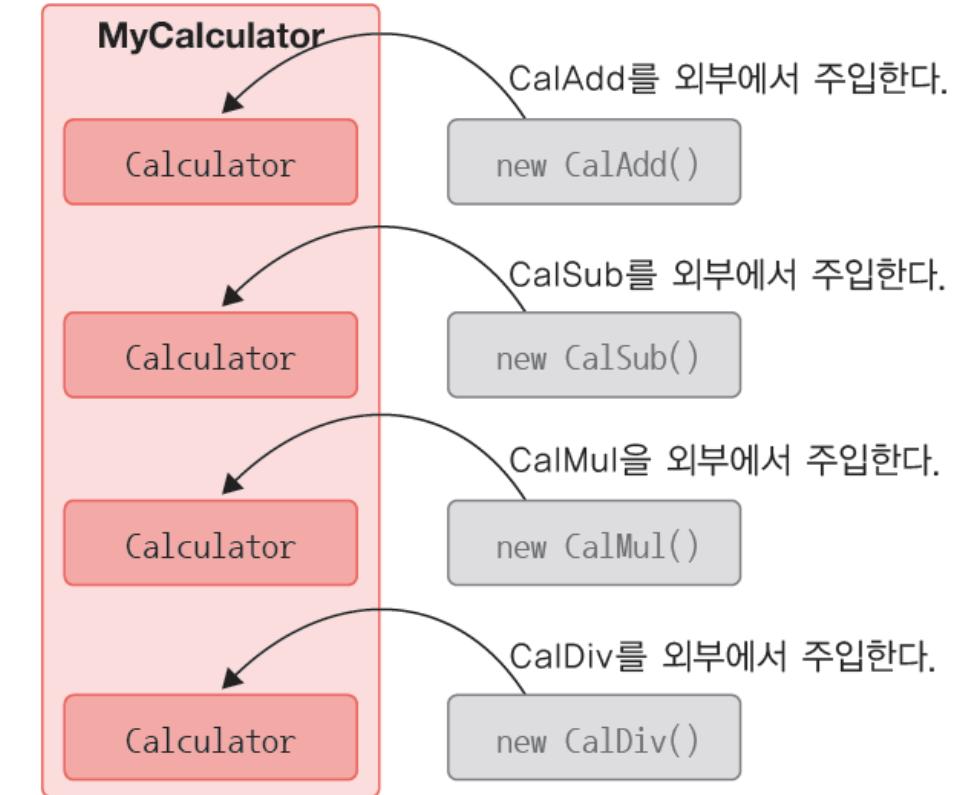
■ 인터페이스를 활용하도록 수정하기

- MyCalculator가 필요한 객체를 외부에서 받는 코드는 변형하는데 인터페이스를 이용하지 않으니 동일한 코드를 계속 반복 작성해야 했음
- 인터페이스를 활용하여 MyCalculator가 CalAdd, CalSub, CalMul, CalDiv를 외부에서 받을 때 ICalculator를 이용하도록 MyCalculator와 MainClass를 최종 수정하기

코드 2-10

```
01 package ch02_pjt_01;
02 public class MyCalculator {
03
04     public void calculate(int fNum, int sNum, ICalculator calculator) {
05         int value = calculator.doOperation(fNum, sNum);    // 연산 실행
06         System.out.println("result : " + value);
07     }
08 }
```

ch02_pjt_01\src\ch02_pjt_02\MyCalculator.java



[그림 8] Icalculator를 이용하여 유연해진 프로그램

3. 계산기 프로그램을 DI 방식으로 변경하기

KITRI 2024 교안 - Song Young Ohk

■ 인터페이스를 활용하도록 수정하기

- MyCalculator를 사용하도록 MainClass의 main() 메서드도 수정하기

코드 2-11

ch02_pjt_01\src\ch02_pjt_02\MyCalculator.java

```
01 package ch02_pjt_01;
02 public class MainClass {
03     public static void main(String[] args) {
04
05         MyCalculator calculator = new MyCalculator();
06         calculator.calculate(10, 5, new CalAdd());
07         calculator.calculate(10, 5, new CalSub());
08         calculator.calculate(10, 5, new CalMul());
09         calculator.calculate(10, 5, new CalDiv());
10     }
11 }
```

■ IoC(inversion of control)

- 제어의 역전이라고 함
- 프로그램의 제어권을 개발자가 컨트롤하는 것이 아니라 외부에서 컨트롤하는 방식을 의미함
- 말 그대로 제어의 주체가 개발자에서 스프링으로 바뀐 것
- IoC를 이해하기 위해 ch02_pjt_02를 복사해서 ch02_pjt_03 생성하기

■ CalAssembler 클래스

- ch02_pjt_03에서 수정할 부분은 main()임
- main()의 역할
 - ✓ 프로그램의 시작을 나타냄(JVM이 가장 먼저 찾음)
 - ✓ 따라서 지금처럼 MyCalculator, CalAdd, CalSub, CalMul, CalDiv 등과 같은 프로그램에 필요한 모든 객체를 main()에서 생성한다는 것은 main()에 과다한 업무를 부여하는 것과 같음
- 프로그램 실행에 필요한 객체는 별도의 클래스에서 생성하도록 수정함
 - ✓ CalAssembler 클래스의 역할 : 프로그램에서 사용하는 객체들을 생성하고 주입함

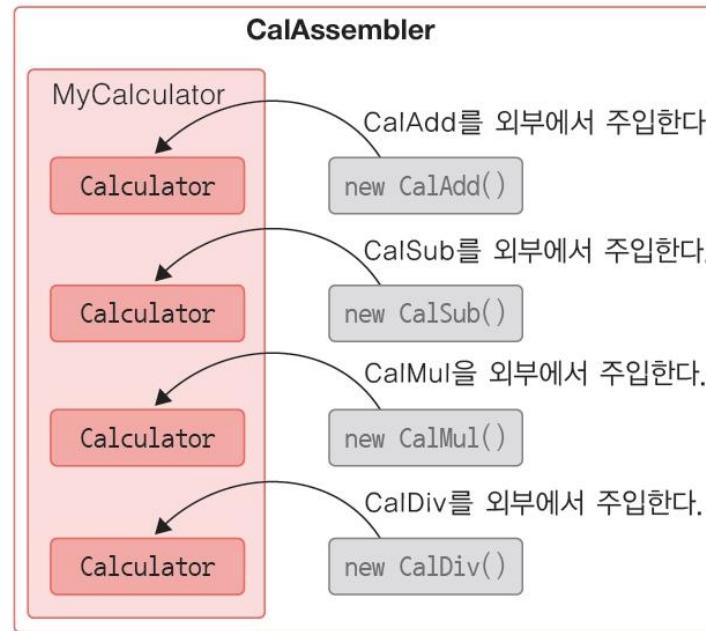
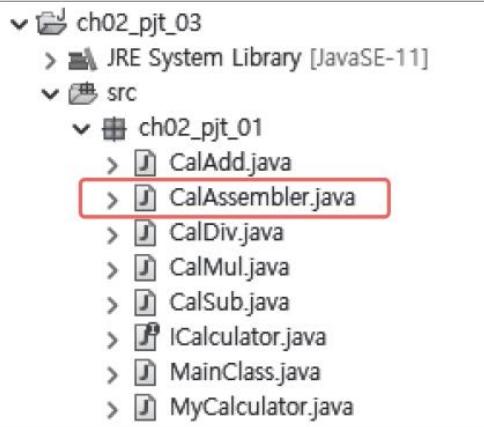
5. 계산기 프로젝트를 IoC 방식으로 변경하기

KITRI 2024 교안 - Song Young Ohk

CalAssembler 클래스

- 1. CalAssembler 클래스를 만들기

- 2. 프로그램에서 사용하는 객체들을 생성하고 주입하는 CalAssembler 코딩하기



(a) 프로젝트에 추가된 CalAssembler 클래스

(b) 객체를 생성하고 주입하는 CalAssembler 클래스

[그림 9] CalAssembler 클래스의 역할

코드 2-12

ch02_pjt_01\src\ch02_pjt_03\CalAssembler.java

```
01 package ch02_pjt_01;
02 public class CalAssembler {
03     MyCalculator calculator;
04     CalAdd calAdd;
05     CalSub calSub;
06     CalMul calMul;
07     CalDiv calDiv;
08
09     public CalAssembler() {
10         calculator = new MyCalculator();
11         calAdd = new CalAdd();
12         calSub = new CalSub();
13         calMul = new CalMul();
14         calDiv = new CalDiv();
15
16         assemble();
17     }
18
19     public void assemble() {
20         calculator.calculate(10, 5, calAdd);
21         calculator.calculate(10, 5, calSub);
22         calculator.calculate(10, 5, calMul);
23         calculator.calculate(10, 5, calDiv);
24     }
}
```

■ CalAssembler 클래스

- 3. 이제 main() 메서드에서 하던 일을 CalAssembler에서 하게 됨

- ✓ CalAssembler는 생성자에서 프로그램에 필요한 객체들(MyCalculator, CalAdd, CalSub, CalMul, CalDiv)을 모두 생성하고 연산을 자동으로 실행(assemble())함
- ✓ 이에 따라 MainClass의 main() 메서드를 다음과 같이 수정하기

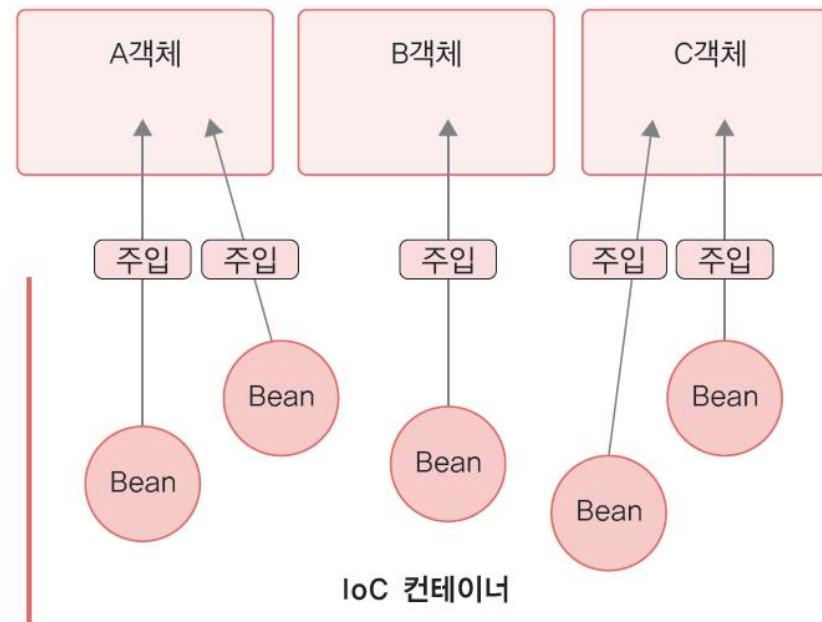
코드 2-13

ch02_pjt_01\src\ch02_pjt_03\MainClass.java

```
01 package ch02_pjt_01;
02 public class MainClass {
03     public static void main(String[] args) {
04         new CalAssembler();
05     }
06 }
```

■ CalAssembler 클래스

- [코드 2-13] 4행에서 볼 수 있듯이 main() 메서드에서는 CalAssembler 객체만 생성하도록 변경됨
- IoC 컨테이너: CalAssembler와 같이 객체를 생성하고 조립하는 특별한 공간
- 빈(Bean): IoC 컨테이너의 객체
- 다시 말해 스프링의 IoC 컨테이너는 빈을 생성하고 필요한 곳에 주입(DI)하는 특별한 공간임



[그림 10] 빈을 생성하고 필요한 곳에 주입하는 IoC 컨테이너

03. 스프링 맛보기

1. 메이븐 프로젝트
2. 처음 만들어보는 스프링 프로젝트
3. 디렉터리와 pom.xml 파일의 이해
4. 스프링을 이용한 계산기 프로그램

Section 01

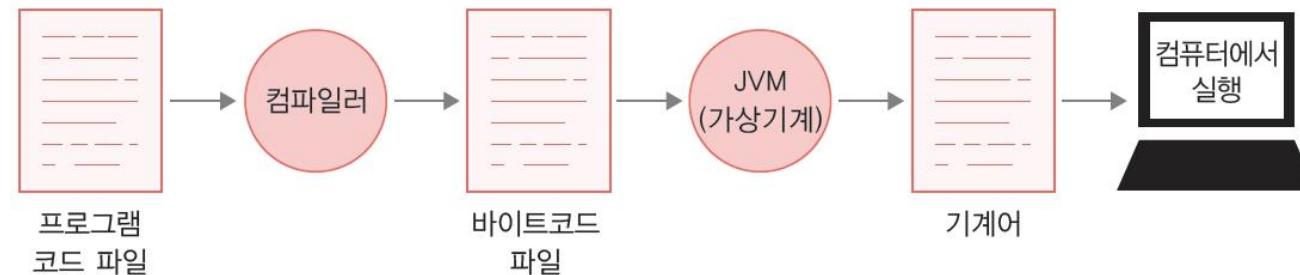
메이븐 프로젝트

■ 컴파일(compile)

- 코딩한 코드 파일을 컴파일러(compiler)가 바이트코드(bytecode) 파일로 변환하는 과정을 뜻함
- 바이트코드 파일은 JVM에 의해 기계어로 바뀌어 컴퓨터에서 실행됨

■ 빌드(build)

- 컴파일보다 넓은 의미로 라이브러리 다운로드 및 연결, 컴파일, 링크, 패키징 등 애플리케이션 제작에 필요한 전반적인 과정을 뜻함
 - ✓ 링크: 서로 다른 파일을 연결 해서 메서드 호출 등의 업무가 가능하게 만드는 것
 - ✓ 패키징: 구현된 각 각의 기능을 하나로 합쳐서 실행 파일을 만드는 것
 - ✓ 빌드는 많은 과정을 담당하는데 이러한 과정은 개발자가 직접 하는 것이 아니고, 빌드툴에 의해 모두 자동화되어 있음



[그림 1] 프로그램 코드 파일은 컴파일러를 통해 바이트코드 파일로 변환된다.

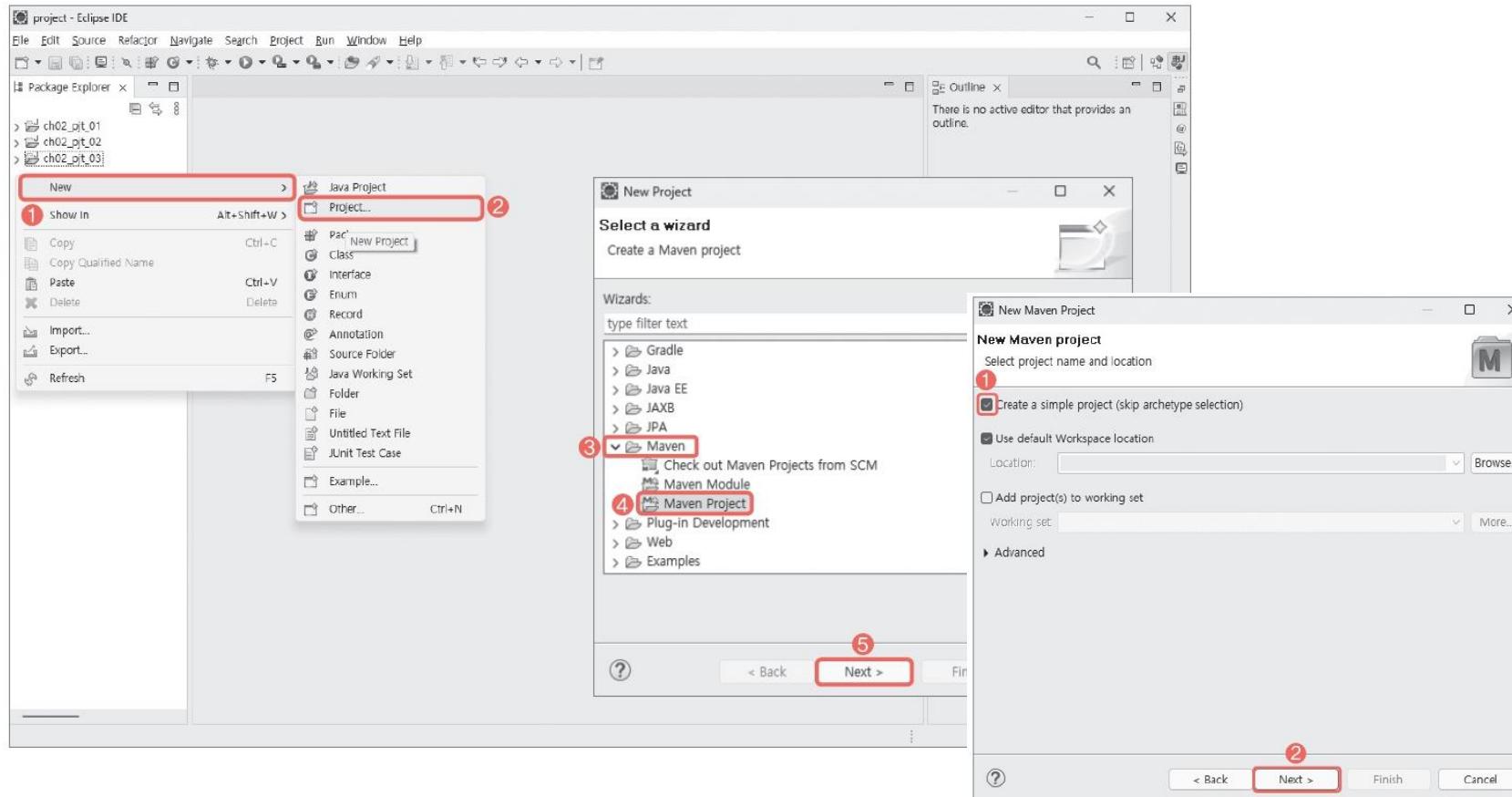
■ 빌드툴의 종류

- Ant: 과거에 많이 사용했지만, 내부 스크립트가 다소 복잡해서 점차 사용 빈도가 줄고 있음
- Maven: 오래 전부터 스프링 애플리케이션 개발에 사용된 툴로 많은 산업 현장에서 여전히 사용되고 있음
 - ✓ 앞으로 진행할 스프링 프로젝트는 메이븐(Maven)을 사용함
 - ✓ 메이븐은 이클립스에 기본적으로 설치가 되어 있어 별도의 설치 과정이 필요 없음
- Gradle: 최근에 인기를 얻고 있음
 - ✓ 메이븐보다 약 두 배가량 빠름

2. 메이븐 프로젝트 생성하기

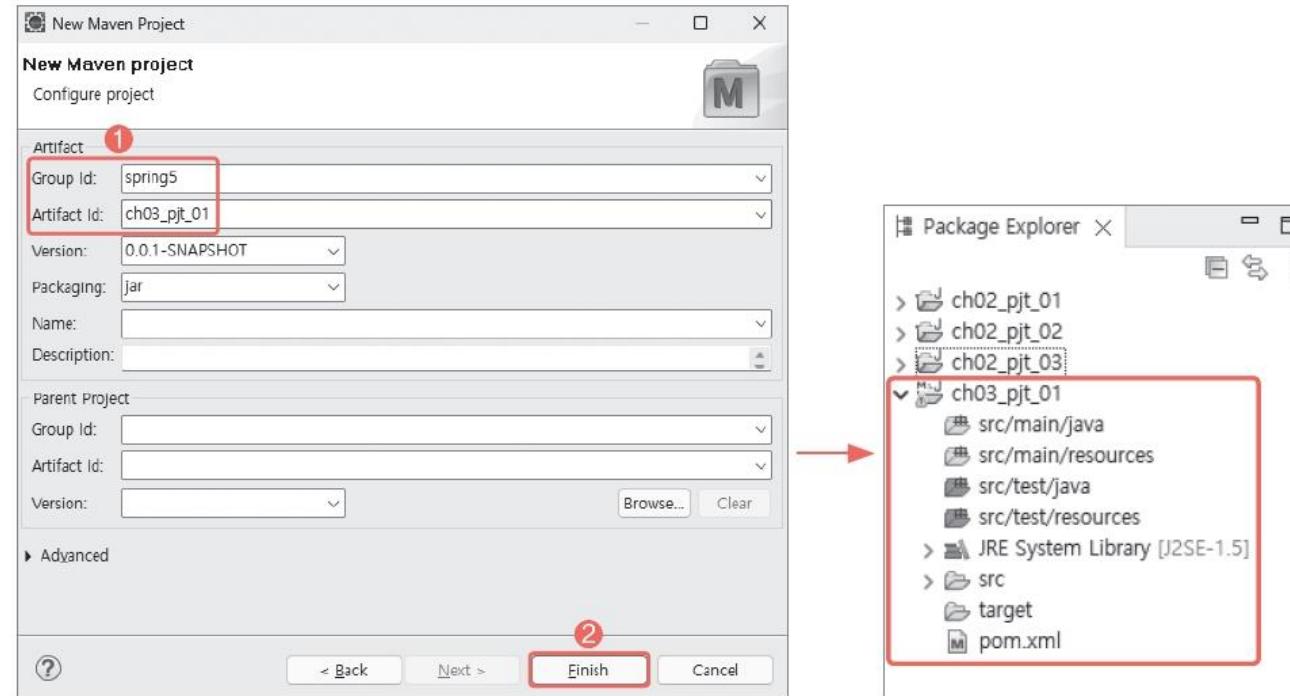
KITRI 2024 교안 - Song Young Ohk

- 1. Package Explorer에서 마우스 오른쪽 버튼을 클릭하여 [New]-[Project...]를 선택하기
 - [New Project] 창에서 Maven 폴더의 Maven Project를 선택한 후 <Next> 클릭
- 2. [New Maven Project] 창이 나오면 'Create a simple project(skip archetype selection)'에 체크하고 <Next> 클릭



■ 3. Artifact 정보를 입력하는 창에 Group Id, Artifact Id를 입력하고 <Finish> 버튼을 클릭

- Group Id: spring5
- Artifact Id: ch03_pjt_01

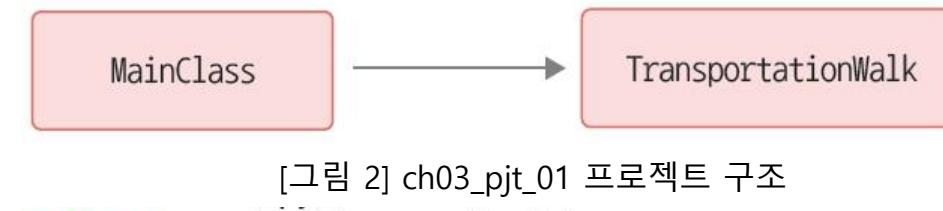


■ 프로젝트의 시나리오

- ① MainClass에서 이동수단(TransportationWalk) 객체를 생성한다.
- ② 생성된 이동수단(TransportationWalk) 객체의 move() 메서드를 호출한다.
- ③ move() 메서드 호출 시 콘솔 창에 해당하는 이동수단(도보)을 출력한다.

■ 프로젝트의 구조

- MainClass 클래스는 TransportationWalk 클래스를 사용함



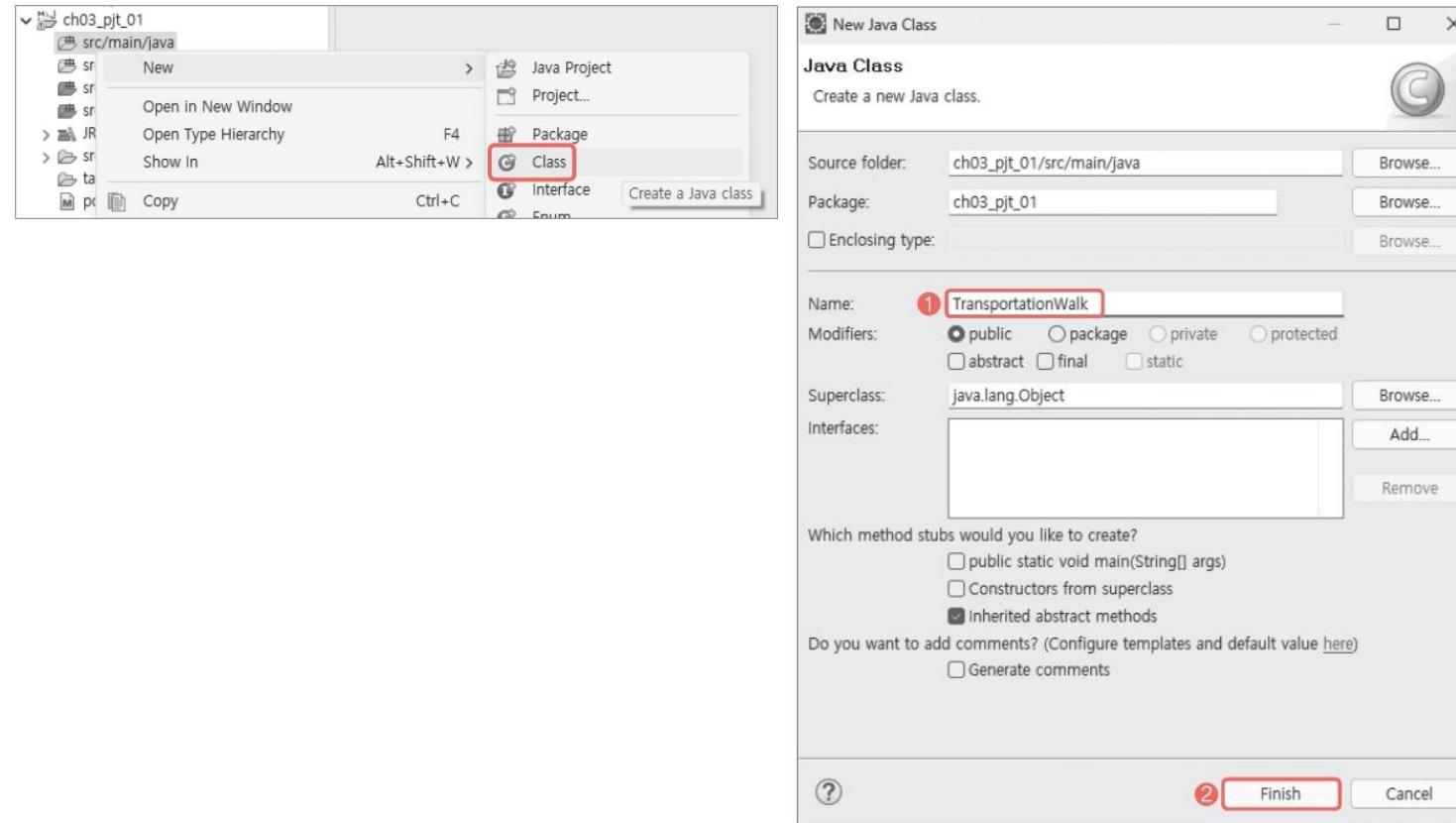
[그림 2] ch03_pjt_01 프로젝트 구조

3. 메이븐 프로젝트 실행하기

KITRI 2024 교안 - Song Young Ohk

■ 클래스 생성과 코딩

- 1. ch03_pjt_01의 [src/main/java]에서 마우스 오른쪽 버튼을 클릭하여 새로운 Class 파일을 생성하기
- 2. 클래스 이름을 TransportationWalk로 하고 <Finish> 클릭



■ 클래스 생성과 코딩

- 3. TransportationWalk.java 파일을 다음과 같이 작성함

코드 3-1

ch03_pjt_01\src\main\java\ch03_pjt_01\TransportationWalk.java

```
01 package ch03_pjt_01;
02 public class TransportationWalk {
03     public void move() {
04         System.out.println("도보로 이동합니다!");
05     }
06 }
```

- 4. MainClass 클래스를

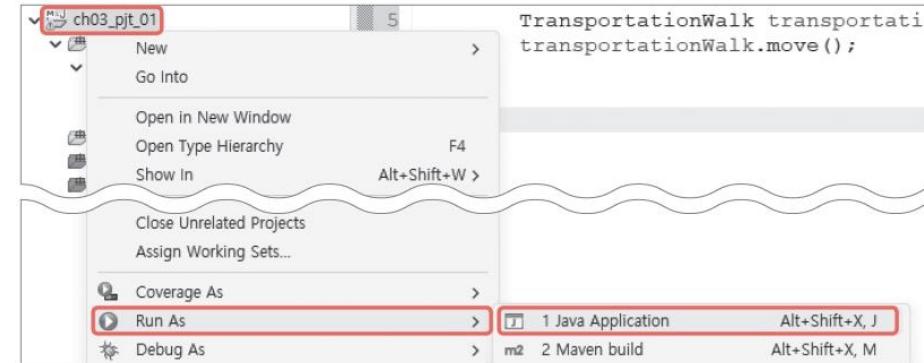
코드 3-2

ch03_pjt_01\src\main\java\ch03_pjt_01>MainClass.java

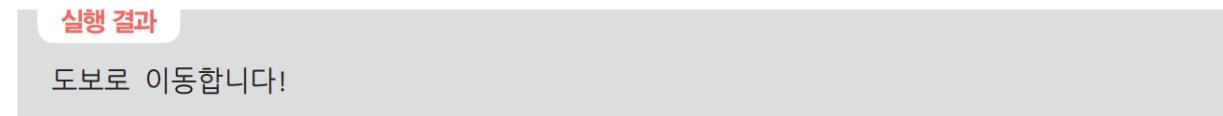
```
01 package ch03_pjt_01;
02 public class MainClass {
03     public static void main(String[] args) {
04         TransportationWalk transportationWalk = new TransportationWalk();
05         transportationWalk.move();
06     }
07 }
```

■ 클래스 생성과 코딩

- 5. 작성한 코드를 모두 완료했으면 ch03_pjt_01에서 마우스 오른쪽 버튼을 클릭하고 [Run As]-[Java Application]을 선택하여 프로그램을 실행하기



- 6. main()에서 TransportationWalk의 move()를 호출한 결과 확인하기



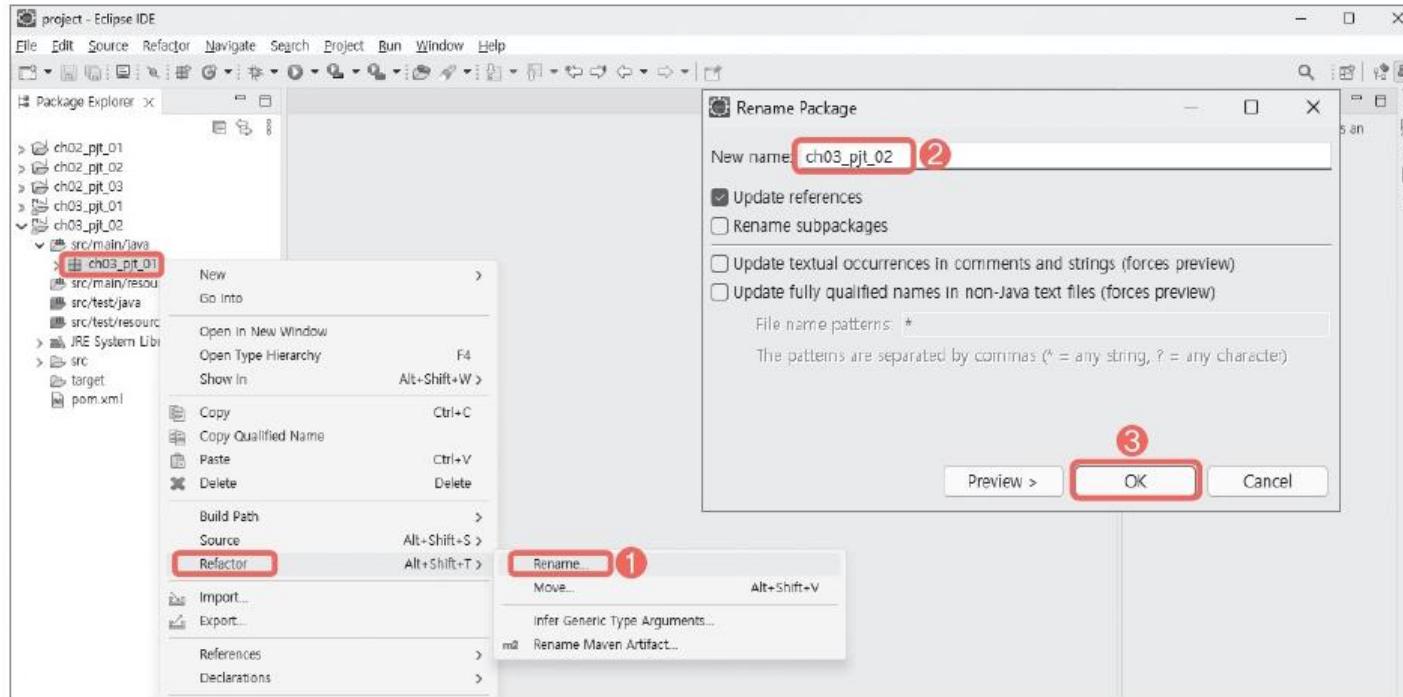
Section 02

처음 만들어보는
스프링 프로젝트

1. 프로젝트 만들기

KITRI 2024 교안 - Song Young Ohk

- 1. ch03_pjt_01을 복사해서 붙인 후, [Copy Project] 창이 열리면 Project name에 ch03_pjt_02를 작성한 후 <Copy>를 클릭하기
- 2. ch03_pjt_02의 [src/main/java]에서 ch03_pjt_01 패키지명에 마우스 오른쪽 버튼을 클릭하여 [Refactor]-[Rename]을 선택하면 [Rename Package] 창이 열림.
 - New name에 ch03_pjt_02를 입력한 후 <OK>를 클릭



■ Ch03_pjt_02가 ch03_pjt_01과 다른 점

- GenericXmlApplicationContext를 이용해서 main()에 TransportationWalk를 직접 생성하지 않고 스프링 IoC 컨테이너에서 Bean으로 생성하여 사용함
- 이를 위해 메이븐을 이용해서 spring-context-5.2.9.RELEASE.jar를 가져와야 함

■ pom.xml 코딩

코드 3-3

ch03_pjt_02\pom.xml

```
01 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
02   <modelVersion>4.0.0</modelVersion>
03   <groupId>spring5</groupId>
04   <artifactId>ch03_pjt_02</artifactId>
05   <version>0.0.1-SNAPSHOT</version>
06
07   <!-- spring-context 모듈 -->
08   <dependencies>
09     <dependency>
10       <groupId>org.springframework</groupId>
11       <artifactId>spring-context</artifactId>
12       <version>5.2.9.RELEASE</version>
13     </dependency>
14   </dependencies>
15
```

2. pom.xml 작업하기

KITRI 2024 교안 - Song Young Ohk

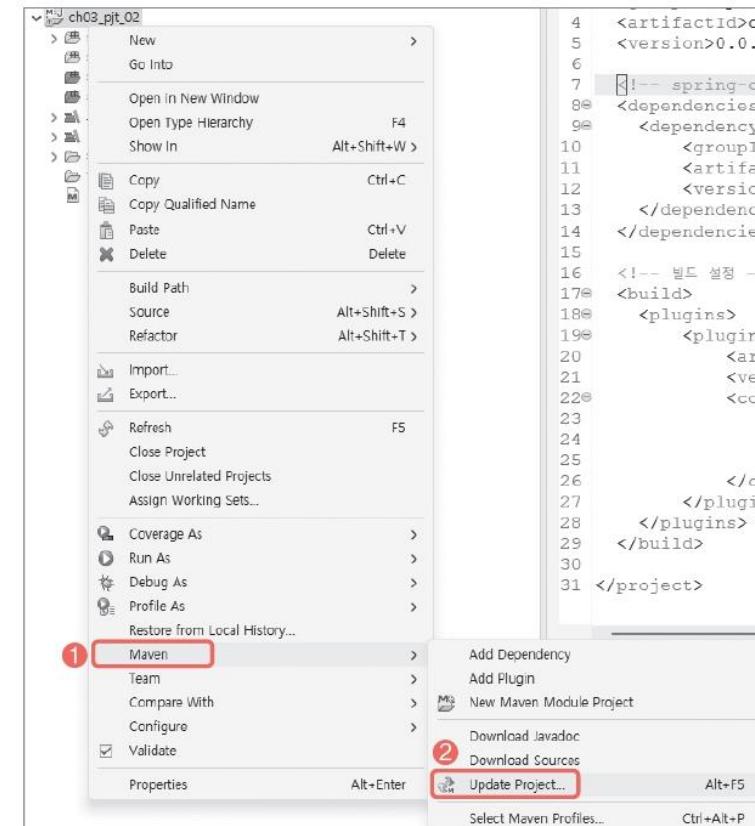
```
16    <!-- 빌드 설정 -->
17    <build>
18        <plugins>
19            <plugin>
20                <artifactId>maven-compiler-plugin</artifactId>
21                <version>3.1</version>
22                <configuration>
23                    <source>11</source>
24                    <target>11</target>
25                    <encoding>utf-8</encoding>
26                </configuration>
27            </plugin>
28        </plugins>
29    </build>
30
31 </project>
```

3. 프로젝트 업데이트하기

KITRI 2024 교안 - Song Young Ohk

JRE 라이브러리 버전을 11로 변경하기

- ch03_pjt_02 프로젝트에서 사용되는 JRE 라이브러리 버전이 1.5로 설정되어 있음([J2SE-1.5])
- 프로젝트 이름 위에서 마우스 오른쪽 버튼을 클릭하여 [Maven]-[Update Project...(<Alt>+<F5>)]를 선택



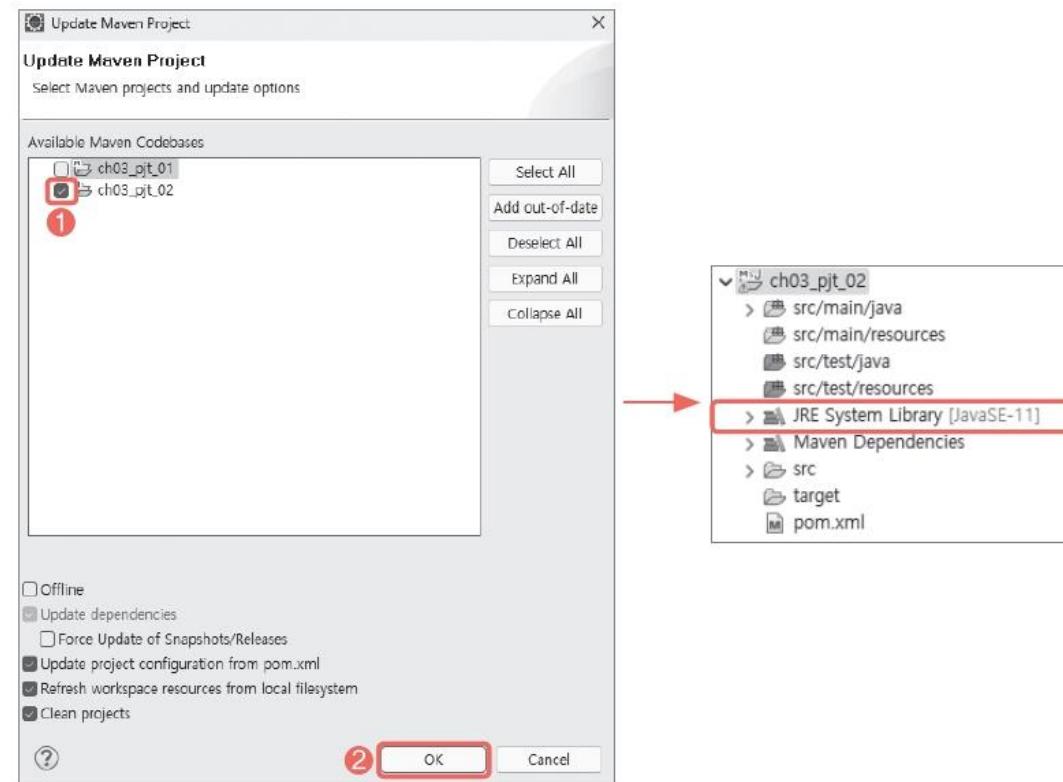
[그림 3] 메이븐 프로젝트 업데이트

3. 프로젝트 업데이트하기

KITRI 2024 교안 - Song Young Ohk

JRE 라이브러리 버전을 11로 변경하기

- [Update Maven Project] 창에서 프로젝트를 선택하고 <OK>를 클릭해서 JRE11 버전 ([JavaSE-11])으로 변경

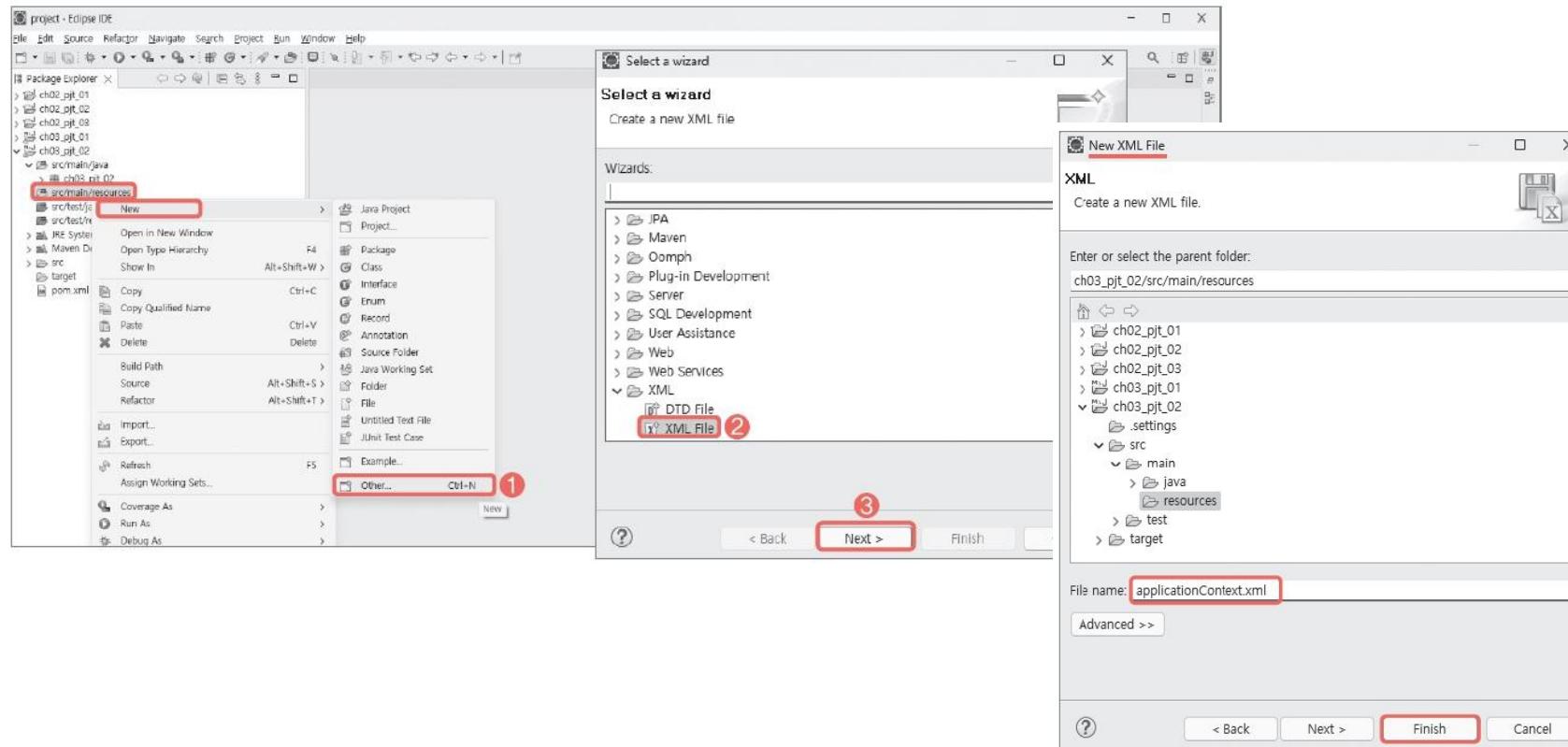


4. applicationContext.xml 생성하기

KITRI 2024 교안 - Song Young Ohk

■ IoC 컨테이너 역할을 하는 스프링 설정 파일: applicationContext.xml

- 1. [src/main/resources]에서 마우스 오른쪽 버튼을 클릭하여 [New]-[Other...]를 선택(<Ctrl>+<N>)하기. 이어서 [Select a wizard] 창에서 스크롤을 아래쪽으로 내려 [XML]-[XML File]을 선택한 후 <Next> 클릭
- 2. [New XML File] 창에서 File name을 applicationContext.xml로 하고 <Finish> 클릭



4. applicationContext.xml 생성하기

KITRI 2024 교안 - Song Young Ohk

■ IoC 컨테이너 역할을 하는 스프링 설정 파일: applicationContext.xml

- 3. applicationContext.xml을 코딩하기

코드 3-4

ch03_pjt_02\src\main\resources\applicationContext.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <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">
04
05   <bean id="tWalk" class="ch03_pjt_02.TransportationWalk" />
06
07 </beans>
```

4. applicationContext.xml 생성하기

KITRI 2024 교안 - Song Young Ohk

■ IoC 컨테이너 역할을 하는 스프링 설정 파일: applicationContext.xml

- 4. [src/main/java]의 MainClass.java를 수정하기

코드 3-5

ch03_pjt_02\src\main\java\MainClass.java

```
01 package ch03_pjt_02;  
02  
03 import org.springframework.context.support.GenericXmlApplicationContext;  
04  
05 public class MainClass {  
06     public static void main(String[] args) {  
07  
08 //     TransportationWalk transportationWalk = new TransportationWalk();  
09 //     transportationWalk.move();  
10  
11         GenericXmlApplicationContext ctx =  
12             new GenericXmlApplicationContext("classpath:applicationContext.xml");  
13  
14         TransportationWalk transportationWalk =  
15             ctx.getBean("tWalk", TransportationWalk.class);  
16         transportationWalk.move();  
17  
18     }  
19 }
```

실행 결과

도보로 이동합니다!

Section 03

디렉터리와
pom.xml 파일의 이해

■ 디렉터리 구성

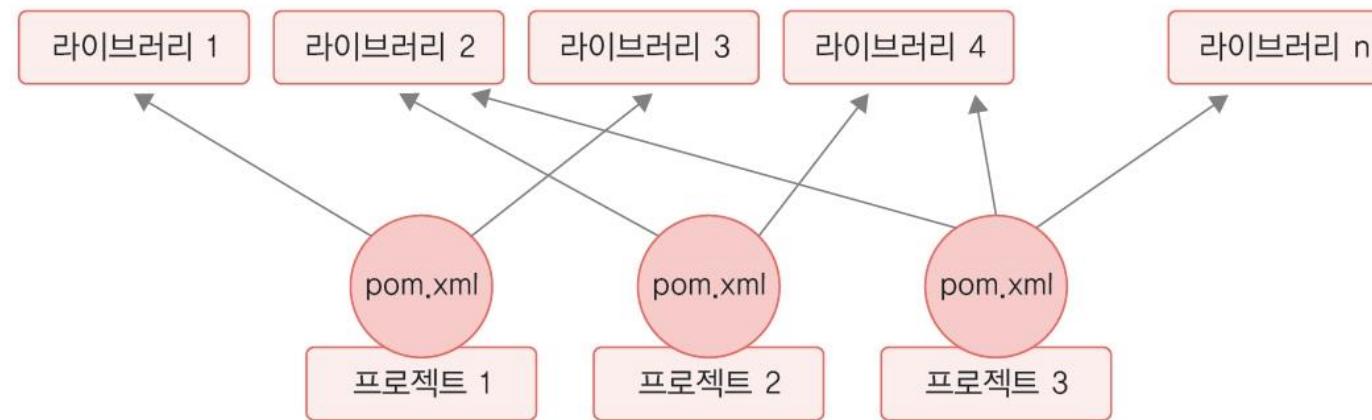
- java(ch03_pjt_02/src/main/java) 디렉터리
 - ✓ 앞으로 만들어지는 자바 파일들이 관리되는 곳임
 - ✓ 즉 프로젝트에 필요한 자바 파일을 이곳에 만들면 됨
- resources(ch03_pjt_02/src/main/resources) 디렉터리
 - ✓ 자원을 관리하는 폴더로 스프링 설정 파일(XML) 또는 프로퍼티 파일 등이 관리됨

[표 1] 디렉토리 구성과 역할

디렉터리	역할
ch03_pjt_02	프로젝트 root
ch03_pjt_02/src/main/java	.java 파일 관리
ch03_pjt_02/src/main/resources	자원 관리

■ 메이븐 설정 파일: pom.xml

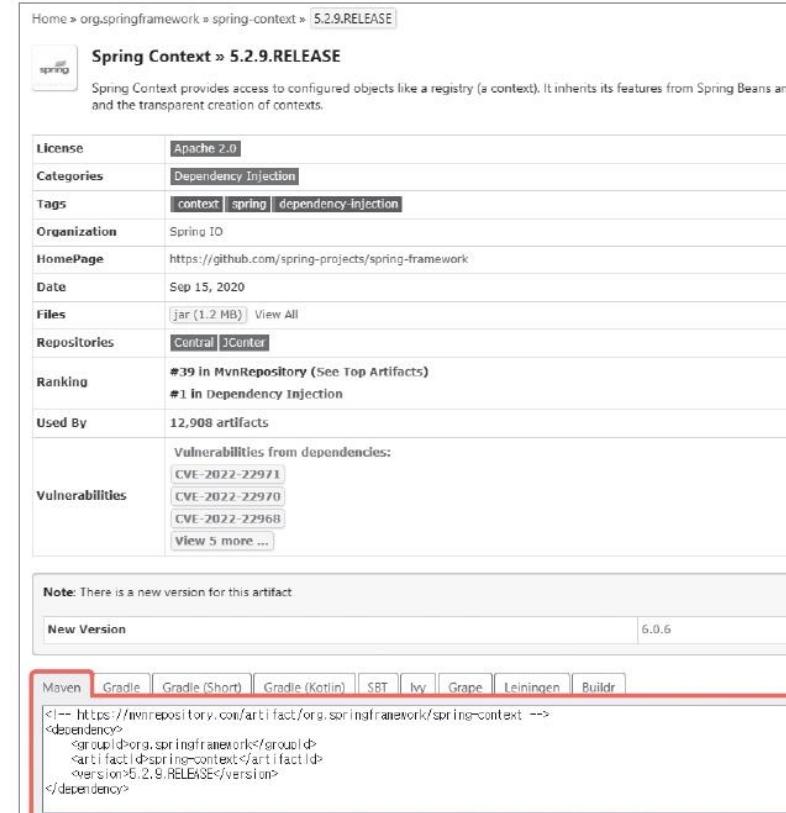
- 필요한 라이브러리를 연결해주고 빌드 설정을 담당함
 - ✓ 과거에는 개발자가 스프링에 필요한 라이브러리를 프로젝트에 직접 연결해서 사용하기도 했음
- 빌드에 필요한 정보도 가지고 있음
- 다시 말해 pom.xml은 스프링 프로젝트에 필요한 라이브러리와 빌드에 필요한 정보가 적혀 있는 명세서



[그림 5] pom.xml에 다운로드가 필요한 모듈을 설정

■ 메인 리포지터리

- 아파치 재단에서 관리하는 메일 리포지터리에서 모듈을 다운로드함
 - ✓ 메일 리포지터리 주소: <https://mvnrepository.com/>



[그림 6] 메인 리포지터리의 spring-context 모듈

■ 로컬 리포지터리

- 다운로드한 라이브러리(jar)가 저장된 개발자 컴퓨터
- spring-context 모듈 설정
 - ✓ 모듈 하나를 artifact(아티팩트)라는 단위로 관리함
 - ✓ org.springframework 그룹에 있는 spring-context라는 모듈의 5.2.9.RELEASE라는 버전을 프로젝트에서 사용(의존)하는 코드

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.2.9.RELEASE</version>
    </dependency>
</dependencies>
```

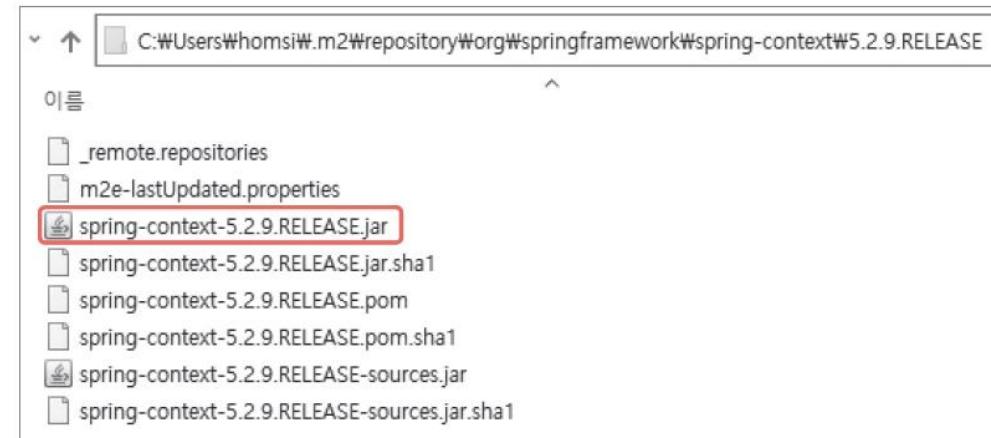
■ 로컬 리포지터리

- 로컬 리포지터리 경로

C:\Users\사용자\.m2\repository\

- spring-context-5.2.9.RELEASE.jar 경로

C:\Users\homsi\.m2\repository\org\springframework\spring-context\5.2.9.RELEASE\spring-context-5.2.9.RELEASE.jar



[그림 9] 로컬 리포지터리에서 확인할 수 있는 spring-context 모듈

■ `org.springframework` 그룹의 또 다른 모듈

- 개발자는 `spring-context-5.2.9.RELEASE.jar`가 필요(의존해야 함)해서 다운로드함
- 나머지 .jar 파일들은 `spring-context-5.2.9.RELEASE.jar`가 의존하고 있기 때문에 같이 다운로드됨
 - ✓ `\spring-aop\5.2.9.RELEASE\spring-aop-5.2.9.RELEASE.jar`
 - ✓ `\spring-beans\5.2.9.RELEASE\spring-beans-5.2.9.RELEASE.jar`
 - ✓ `\spring-core\5.2.9.RELEASE\spring-core-5.2.9.RELEASE.jar`
 - ✓ `\spring-expression\5.2.9.RELEASE\spring-expression-5.2.9.RELEASE.jar`



[그림 10] 다운로드한 모듈과 프로젝트 구조

■ pom.xml 파일에서 프로젝트 빌드를 설정하는 부분의 코드

✓ ([코드 3-3] 17~29행)

- java 파일을 컴파일할 때 JDK11 버전으로 컴파일한다는 것과 자바 소스를 UTF-8로 인코딩한다는 것을 명시한 내용
- 프로젝트 빌드 설정

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.1</version>
      <configuration>
        <source>11</source>
        <target>11</target>
        <encoding>utf-8</encoding>
      </configuration>
    </plugin>
  </plugins>
</build>
```

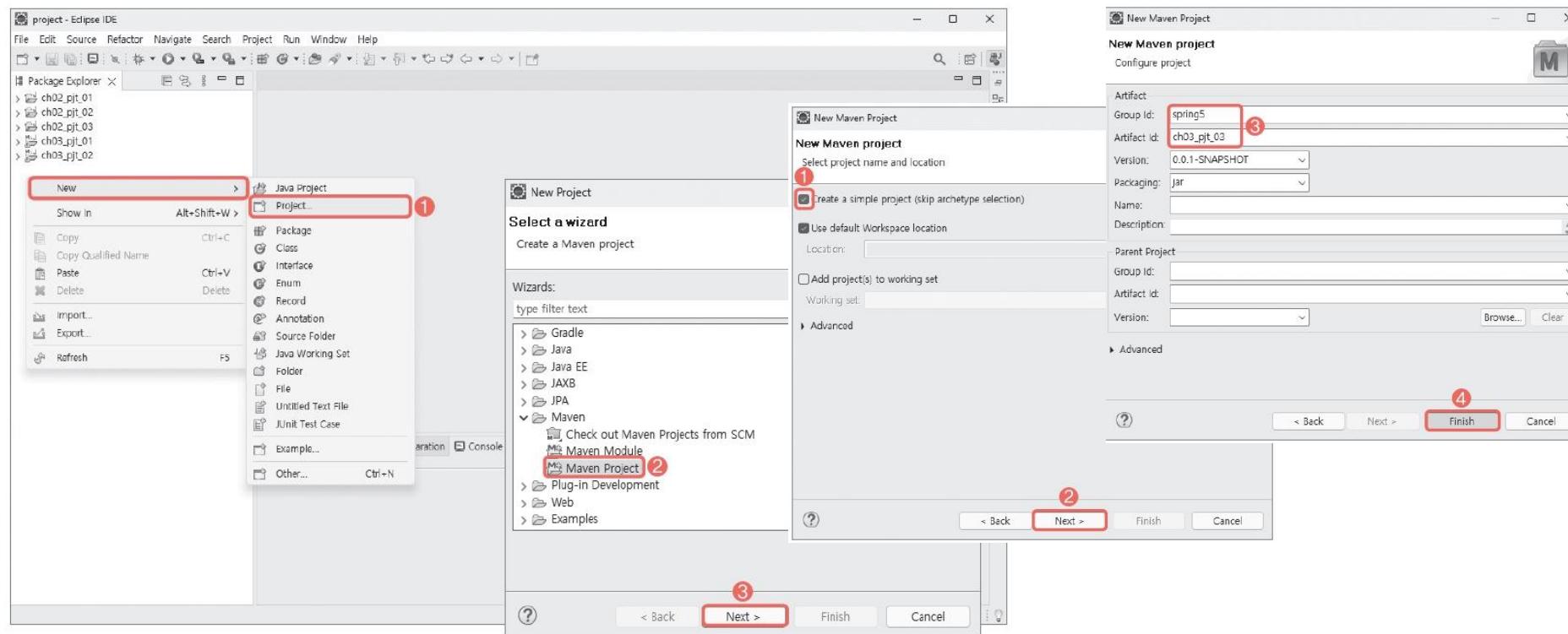
Section 04

**스프링을 이용한
계산기 프로그램**

1. 메이븐을 이용한 스프링 계산기 프로그램

KITRI 2024 교안 - Song Young Ohk

- 1. Package Explorer 탭의 흰 바탕에서 마우스 오른쪽 버튼을 클릭하여 [New]-[Project...]를 선택하기. [New Project] 창에서 'Maven'의 'Maven Project'를 선택한 후 <Next> 클릭
- 2. [New Maven Project] 창에서 'Create a simple project(skip archetype selection)'에 체크한 후 <Next> 클릭하고 Artifact 입력하고 <Finish> 클릭
 - Group Id: spring5
 - Artifact Id: ch03_pjt_03



■ 3. 생성된 ch03_pjt_03 프로젝트의 pom.xml에 스프링과 빌드를 설정하기

코드 3-6

ch03_pjt_03\pom.xml

```
01 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
02   <modelVersion>4.0.0</modelVersion>
03   <groupId>spring5</groupId>
04   <artifactId>ch03_pjt_02</artifactId>
05   <version>0.0.1-SNAPSHOT</version>
06
07   <!-- spring-context 모듈 -->
08   <dependencies>
09     <dependency>
10       <groupId>org.springframework</groupId>
11       <artifactId>spring-context</artifactId>
12       <version>5.2.9.RELEASE</version>
13     </dependency>
14   </dependencies>
15
16   <!-- 빌드 설정 -->
17   <build>
18     <plugins>
19       <plugin>
20         <artifactId>maven-compiler-plugin</artifactId>
21         <version>3.1</version>
```

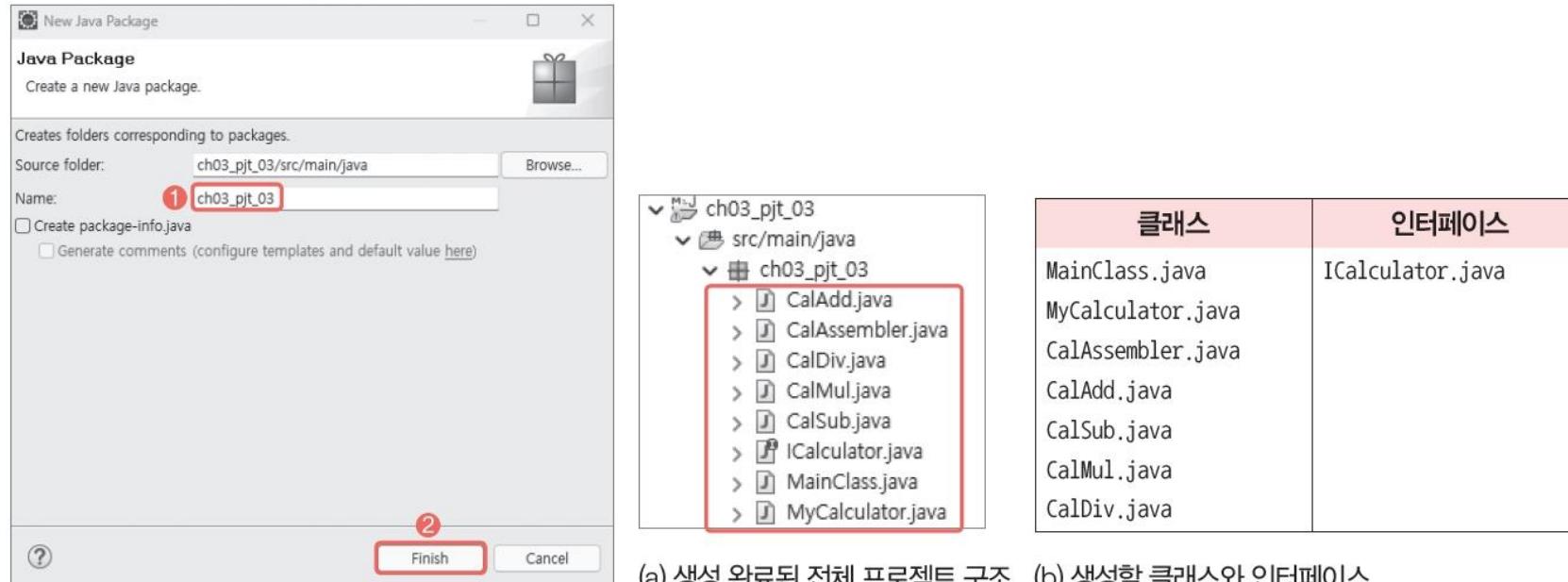
■ 3. 생성된 ch03_pjt_03 프로젝트의 pom.xml에 스프링과 빌드를 설정하기

```
22         <configuration>
23             <source>11</source>
24             <target>11</target>
25             <encoding>utf-8</encoding>
26         </configuration>
27     </plugin>
28 </plugins>
29 </build>
30
31 </project>
```

1. 메이븐을 이용한 스프링 계산기 프로그램

KITRI 2024 교안 - Song Young Ohk

- 4. ch03_pjt_03 프로젝트에서 사용되는 JRE 라이브러리 버전이 1.5로 설정([J2SE-1.5])되어 있다면 메이븐 프로젝트를 업데이트하기
- 5. [src/main/java]에 ch03_pjt_03 패키지를 생성하기
- 6. ch03_pjt_03 패키지에 다음과 같이 클래스와 인터페이스 만들고 코딩하기



[그림 11] 스프링을 활용한 계산기 프로젝트에 필요한 클래스와 인터페이스

■ 클래스와 인터페이스 코딩하기

코드 3-7

ch03_pjt_03\src\main\java\ch03_pjt_03>MainCalss.java

```
01 package ch03_pjt_03;
02 import org.springframework.context.support.GenericXmlApplicationContext;
03
04 public class MainClass {
05     public static void main(String[] args) {
06
07         GenericXmlApplicationContext ctx =
08             new GenericXmlApplicationContext("classpath:applicationContext.xml");
09
10         CalAssembler calAssembler =
11             ctx.getBean("calAssembler", CalAssembler.class);
12         calAssembler.assemble();
13
14     }
15 }
```

1. 메이븐을 이용한 스프링 계산기 프로그램

KITRI 2024 교안 - Song Young Ohk

코드 3-8

ch03_pjt_03\src\main\java\ch03_pjt_03\ICalculator.java

```
01 package ch03_pjt_03;
02 public interface ICalculator {
03     public int doOperation(int firstNum, int secondNum);
04 }
```

코드 3-9

ch03_pjt_03\src\main\java\ch03_pjt_03\MyCalculator.java

```
01 package ch03_pjt_03;
02 public class MyCalculator {
03     public void calculate(int fNum, int sNum, ICalculator calculator) {
04         // ICalculator 객체 주입
05         int value = calculator.doOperation(fNum, sNum); // 연산 실행
06         System.out.println("result : " + value);
07     }
08 }
```

1. 메이븐을 이용한 스프링 계산기 프로그램

KITRI 2024 교안 - Song Young Ohk

코드 3-10

ch03_pjt_03\src\main\java\ch03_pjt_03\CalAssembler.java

```
01 package ch03_pjt_03;
02 public class CalAssembler {
03
04     MyCalculator calculator;
05     CalAdd calAdd;
06     CalSub calSub;
07     CalMul calMul;
08     CalDiv calDiv;
09
10     public CalAssembler(MyCalculator calculator, CalAdd calAdd, CalSub calSub,
11                         CalMul calMul, CalDiv calDiv) {
12
13         this.calculator = calculator;
14         this.calAdd = calAdd;
15         this.calSub = calSub;
16         this.calMul = calMul;
17         this.calDiv = calDiv;
18     }
19
20     public void assemble() {
21         calculator.calculate(10, 5, calAdd);
22         calculator.calculate(10, 5, calSub);
23         calculator.calculate(10, 5, calMul);
24         calculator.calculate(10, 5, calDiv);
25     }
}
```

1. 메이븐을 이용한 스프링 계산기 프로그램

KITRI 2024 교안 - Song Young Ohk

코드 3-11

ch03_pjt_03\src\main\java\ch03_pjt_03\CalAdd.java

```
01 package ch03_pjt_03;
02 public class CalAdd implements ICalculator {
03     public int doOperation(int firstNum, int secondNum) {
04         return firstNum + secondNum;
05     }
06 }
```

코드 3-12

ch03_pjt_03\src\main\java\ch03_pjt_03\CalSub.java

```
01 package ch03_pjt_03;
02 public class CalSub implements ICalculator {
03     public int doOperation(int firstNum, int secondNum) {
04         return firstNum - secondNum;
05     }
06 }
```

코드 3-13

ch03_pjt_03\src\main\java\ch03_pjt_03\CalMul.java

```
01 package ch03_pjt_03;
02 public class CalMul implements ICalculator {
03     public int doOperation(int firstNum, int secondNum) {
04         return firstNum * secondNum;
05     }
06 }
```

코드 3-14

ch03_pjt_03\src\main\java\ch03_pjt_03\CalDiv.java

```
01 package ch03_pjt_03;
02 public class CalDiv implements ICalculator {
03     public int doOperation(int firstNum, int secondNum) {
04         return secondNum != 0 ? (firstNum / secondNum) : 0;
05     }
06 }
```

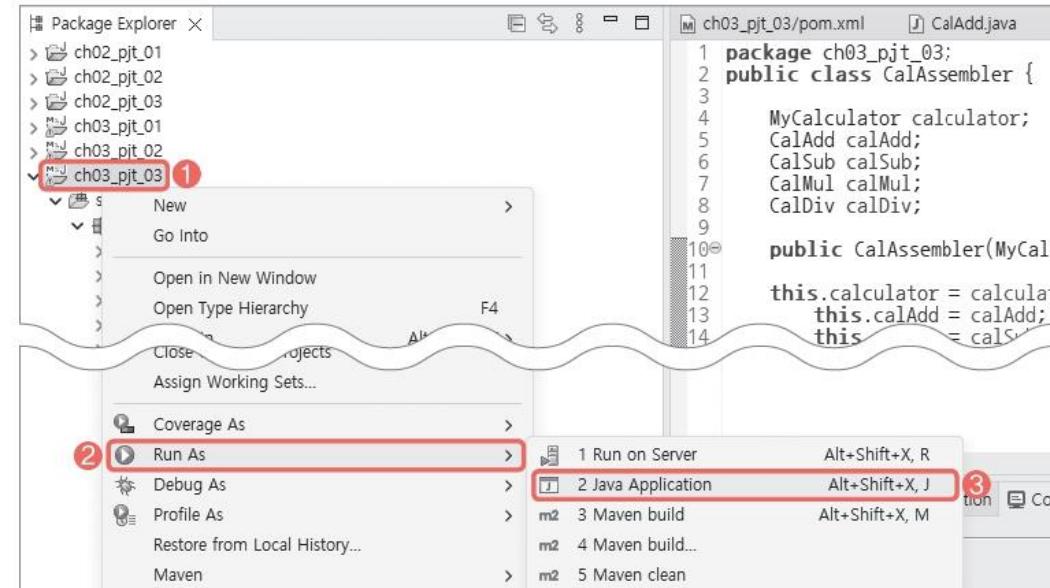
■ 7. [src/main/resources]에 applicationContext.xml 생성하기

코드 3-15

ch03_pjt_03\src\main\resources\applicationContext.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <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">
04
05     <bean id="cAdd" class="ch03_pjt_03.CalAdd" />
06     <bean id="cSub" class="ch03_pjt_03.CalSub" />
07     <bean id="cMulw" class="ch03_pjt_03.CalMul" />
08     <bean id="cDiv" class="ch03_pjt_03.CalDiv" />
09
10     <bean id="myCalculator" class="ch03_pjt_03.MyCalculator" />
11
12     <bean id="calAssembler" class="ch03_pjt_03.CalAssembler" >
13         <constructor-arg ref="myCalculator" />
14         <constructor-arg ref="cAdd" />
15         <constructor-arg ref="cSub" />
16         <constructor-arg ref="cMul" />
17         <constructor-arg ref="cDiv" />
18     </bean>
19
20 </beans>
```

■ 8. ch03_pjt_03 프로그램을 실행하기([Run As]-[Java Application])



실행 결과

```
result : 15
result : 5
result : 50
result : 2
```