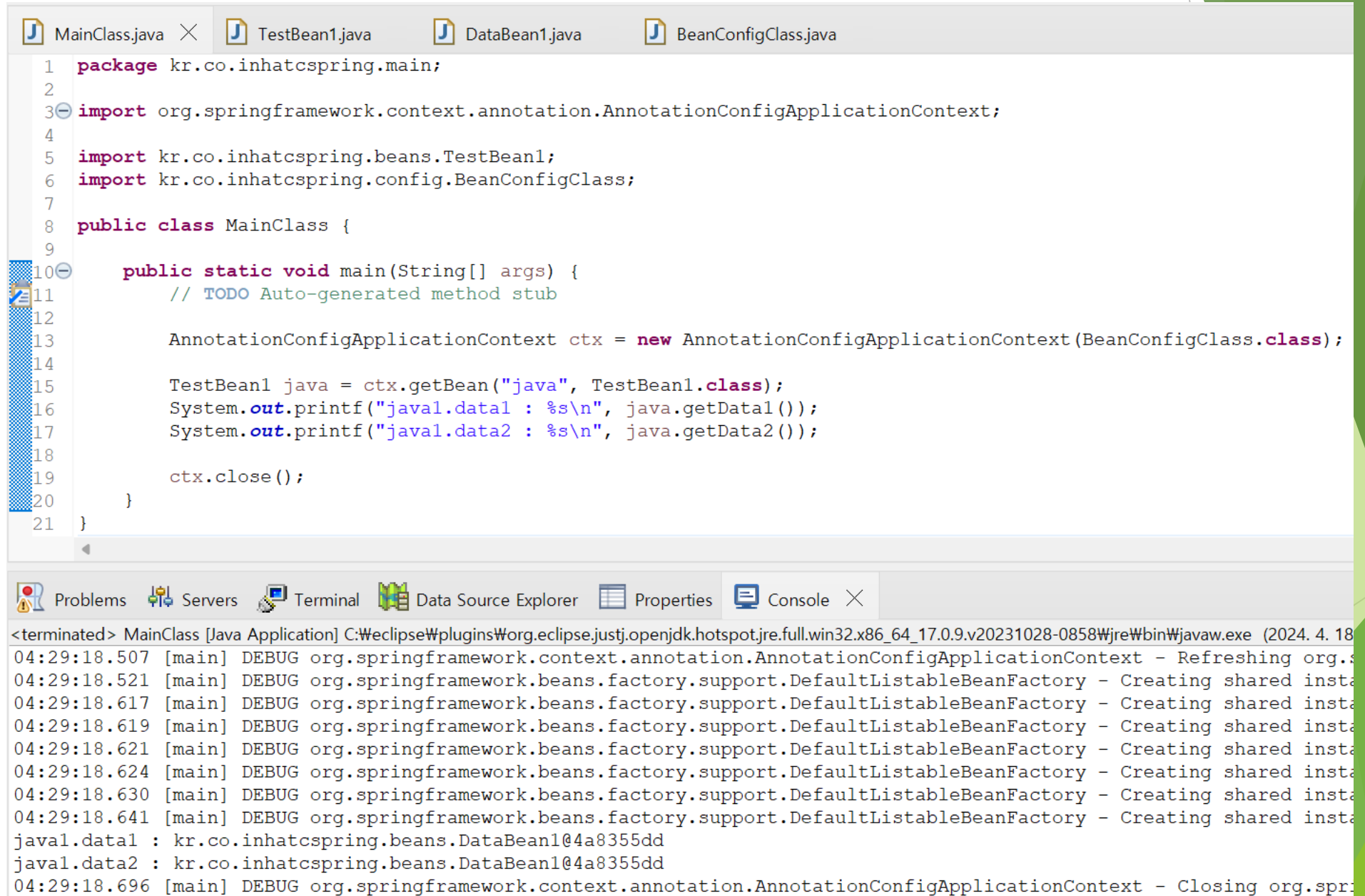


서버프로그래밍

담당교수: 송다혜

Type1: 로직 서술 – Q1



The screenshot displays an IDE window with four tabs: MainClass.java, TestBean1.java, DataBean1.java, and BeanConfigClass.java. The MainClass.java file is open, showing the following code:

```
1 package kr.co.inhatcspring.main;
2
3 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
4
5 import kr.co.inhatcspring.beans.TestBean1;
6 import kr.co.inhatcspring.config.BeanConfigClass;
7
8 public class MainClass {
9
10     public static void main(String[] args) {
11         // TODO Auto-generated method stub
12
13         AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext(BeanConfigClass.class);
14
15         TestBean1 java = ctx.getBean("java", TestBean1.class);
16         System.out.printf("javal.data1 : %s\n", java.getData1());
17         System.out.printf("javal.data2 : %s\n", java.getData2());
18
19         ctx.close();
20     }
21 }
```

The console output at the bottom shows the execution of the application, including debug messages from the Spring framework and the program's output:

```
<terminated> MainClass [Java Application] C:\WeclipseW\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (2024. 4. 18)
04:29:18.507 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Refreshing org.s
04:29:18.521 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
04:29:18.617 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
04:29:18.619 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
04:29:18.621 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
04:29:18.624 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
04:29:18.630 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
04:29:18.641 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
javal.data1 : kr.co.inhatcspring.beans.DataBean1@4a8355dd
javal.data2 : kr.co.inhatcspring.beans.DataBean1@4a8355dd
04:29:18.696 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Closing org.spr
```

Type1: 로직 서술

Q1: 다음의 코드에서 Autowire.BY_TYPE에 기반하여 자동주입이 되는 과정에 대해 서술하시오.

```
DataBean1.java × BeanConfigClass.java
1 package kr.co.inhatcspring.beans;
2
3 public class DataBean1 {
4
5 }
```

```
BeanConfigClass.java ×
1 package kr.co.inhatcspring.config;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.context.annotation.Bean;
5 import org.springframework.context.annotation.Configuration;
6
7 import kr.co.inhatcspring.beans.DataBean1;
8 import kr.co.inhatcspring.beans.TestBean1;
9
10 @Configuration
11 public class BeanConfigClass {
12
13     @Bean
14     public DataBean1 data1() {
15         return new DataBean1();
16     }
17
18     @Bean(autowire = Autowire.BY_TYPE)
19     public TestBean1 java() {
20         return new TestBean1();
21     }
22
23 }
```

```
MainClass.java × TestBean1.java × DataBean1.java
1 package kr.co.inhatcspring.beans;
2
3 public class TestBean1 {
4     private DataBean1 data1;
5     private DataBean1 data2;
6
7     public DataBean1 getData1() {
8         return data1;
9     }
10    public void setData1(DataBean1 data1) {
11        this.data1 = data1;
12    }
13    public DataBean1 getData2() {
14        return data2;
15    }
16    public void setData2(DataBean1 data2) {
17        this.data2 = data2;
18    }
19
20 }
```

A: BY_TYPE은 타입에 기반한 자동 주입이기 때문에, MainClass.java에서 'Testbean1' class를 가진 객체를 호출하면, 프레임워크가 Testbean1에 있는 property의 타입을 확인하고, BeanConfigClass.java에 등록된 객체 중 같은 타입의 객체를 자동으로 주입한다.

Type1: 로직 서술 – Q2

```
MainClass.java ×
1 package kr.co.inhatcspring.main;
2
3 import org.springframework.context.support.ClassPathXmlApplicationContext;
4
5 import kr.co.inhatcspring.beans.TestBean1;
6
7 public class MainClass {
8
9     public static void main(String[] args) {
10         // TODO Auto-generated method stub
11
12         ClassPathXmlApplicationContext ctx = new ClassPathXmlApplicationContext("kr/co/inhatcspring/config/beans.xml");
13
14         TestBean1 xml = ctx.getBean("xml", TestBean1.class);
15         System.out.printf("xml.data1 : %s\n", xml.getData1());
16         System.out.printf("xml.data2 : %s\n", xml.getData2());
17
18         ctx.close();
19     }
20 }
```

Problems Servers Terminal Data Source Explorer Properties Console ×

<terminated> MainClass (1) [Java Application] C:\WeclipseW\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (2024. 4. 18. 오전 4:44:08.935 [main] DEBUG org.springframework.context.support.ClassPathXmlApplicationContext - Refreshing org.springframework...

04:44:09.119 [main] DEBUG org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loaded 7 bean definitions from class

04:44:09.136 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of s

04:44:09.180 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of s

04:44:09.181 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of s

04:44:09.183 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of s

04:44:09.187 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of s

04:44:09.209 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of s

04:44:09.211 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of s

xml.data1 : kr.co.inhatcspring.beans.DataBean1@4b013c76

xml.data2 : kr.co.inhatcspring.beans.DataBean1@53fb3dab

04:44:09.251 [main] DEBUG org.springframework.context.support.ClassPathXmlApplicationContext - Closing org.springframework.con

Type1: 로직 서술

Q2: 다음의 코드에서 어노테이션에 기반하여 자동주입 되는 과정에 대해 서술하시오.

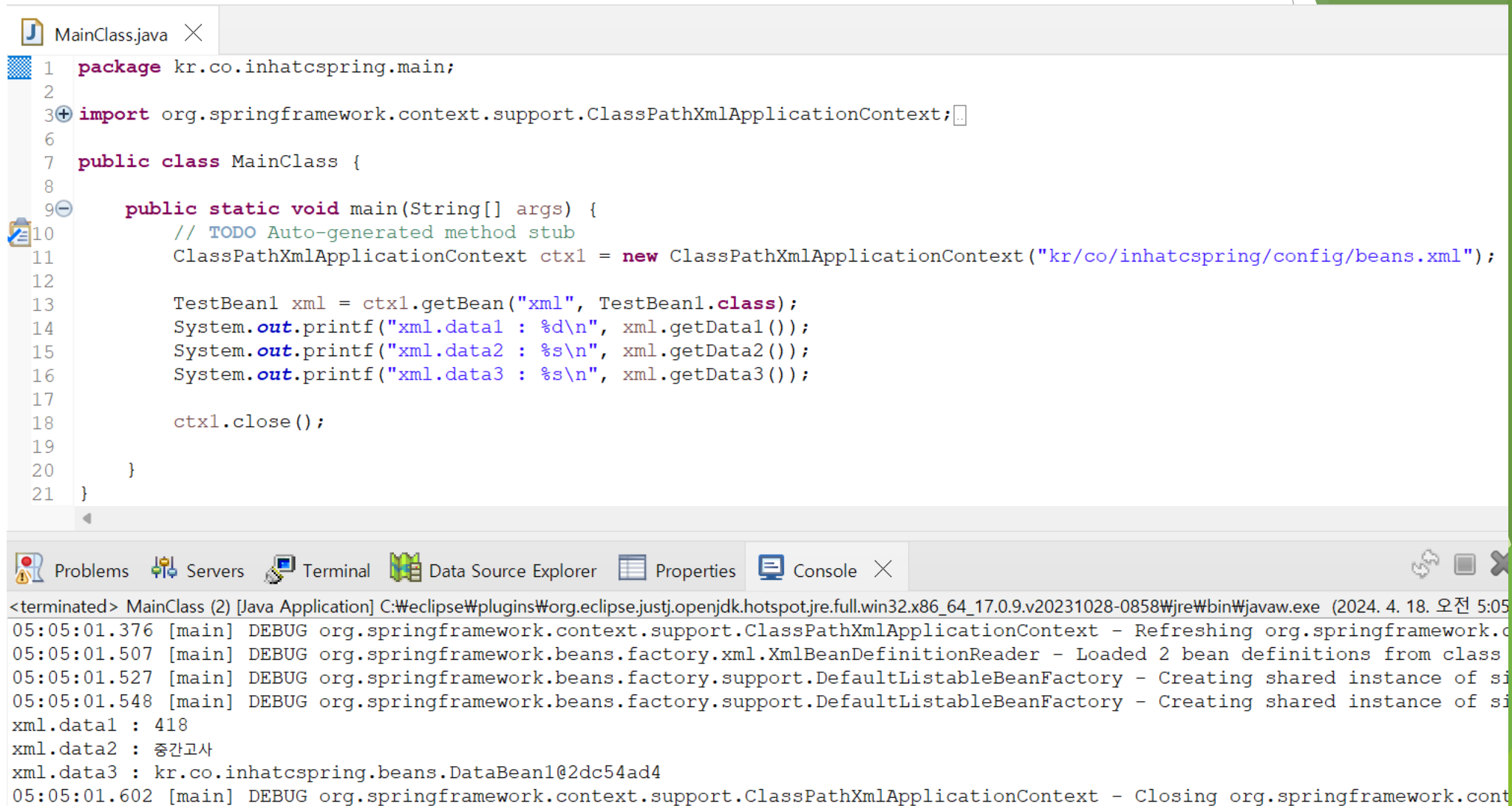
```
DataBean1.java X
1 package kr.co.inhatcspring.beans;
2
3 public class DataBean1 {
4
5 }
```

```
beans.xml X
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
6                           http://www.springframework.org/schema/beans/spring-beans.xsd
7                           http://www.springframework.org/schema/context
8                           http://www.springframework.org/schema/context/spring-context.xsd">
9
10
11 <context:annotation-config/>
12
13 <bean id='xml' class='kr.co.inhatcspring.beans.TestBean1'/>
14
15 <bean id='bean1' class='kr.co.inhatcspring.beans.DataBean1'/>
16 <bean id='bean2' class='kr.co.inhatcspring.beans.DataBean1'/>
17
18 </beans>
```

```
TestBean1.java X
1 package kr.co.inhatcspring.beans;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5
6
7 public class TestBean1 {
8     @Autowired
9     @Qualifier("bean1")
10     private DataBean1 data1;
11
12     @Autowired
13     @Qualifier("bean2")
14     private DataBean1 data2;
15
16     public DataBean1 getData1() {
17         return data1;
18     }
19
20     public DataBean1 getData2() {
21         return data2;
22     }
23
24 }
```

A: **Autowired** 어노테이션에 의해 TestBean1.java의 data1, data2 property에 객체를 주입하기 위해 프레임워크가 DataBean1 타입의 객체를 탐색한다. beans.xml에 등록된 DataBean1 타입의 객체가 두개이기 때문에, **Qualifier** 어노테이션을 통해 명시된 객체의 이름을 활용해 알맞은 객체를 찾아 주입한다.

Type2: 코드 작성 및 작성 근거 서술 - Q3



The screenshot displays the Eclipse IDE interface. The top editor window shows the code for `MainClass.java`. The code defines a package `kr.co.inhatcspring.main`, imports `org.springframework.context.support.ClassPathXmlApplicationContext`, and defines a `MainClass` with a `main` method. The `main` method creates a `ClassPathXmlApplicationContext` instance, retrieves a `TestBean1` bean, and prints its data fields. The bottom console window shows the execution output, including debug messages from Spring and the printed values of the bean's data fields.

```
1 package kr.co.inhatcspring.main;
2
3 import org.springframework.context.support.ClassPathXmlApplicationContext;
4
5
6
7 public class MainClass {
8
9     public static void main(String[] args) {
10         // TODO Auto-generated method stub
11         ClassPathXmlApplicationContext ctx1 = new ClassPathXmlApplicationContext("kr/co/inhatcspring/config/beans.xml");
12
13         TestBean1 xml = ctx1.getBean("xml", TestBean1.class);
14         System.out.printf("xml.data1 : %d\n", xml.getData1());
15         System.out.printf("xml.data2 : %s\n", xml.getData2());
16         System.out.printf("xml.data3 : %s\n", xml.getData3());
17
18         ctx1.close();
19     }
20 }
21 }
```

Problems Servers Terminal Data Source Explorer Properties Console

<terminated> MainClass (2) [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\java.exe (2024. 4. 18. 오전 5:05:05)

05:05:01.376 [main] DEBUG org.springframework.context.support.ClassPathXmlApplicationContext - Refreshing org.springframework.c

05:05:01.507 [main] DEBUG org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loaded 2 bean definitions from class

05:05:01.527 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of si

05:05:01.548 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of si

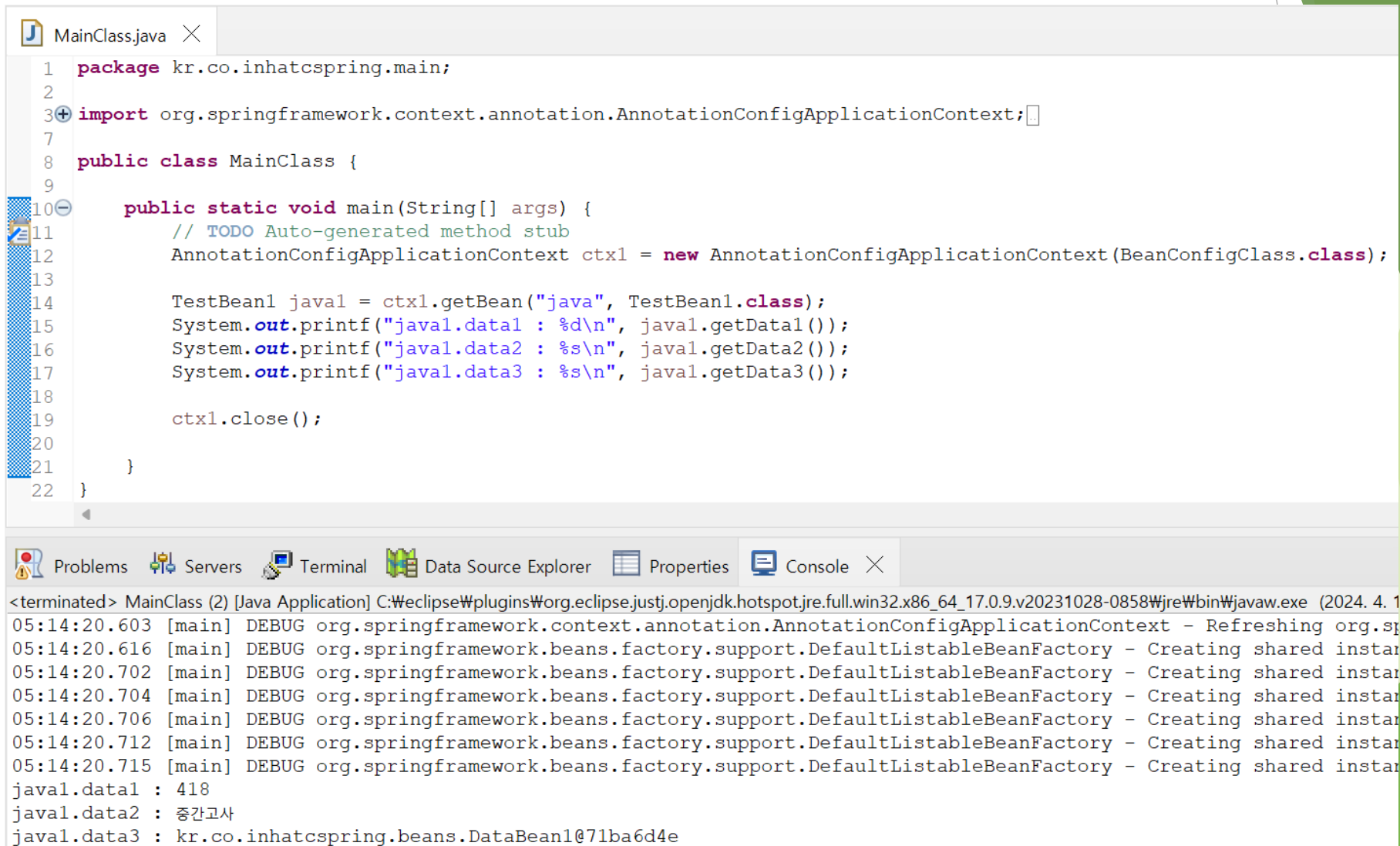
xml.data1 : 418

xml.data2 : 중간고사

xml.data3 : kr.co.inhatcspring.beans.DataBean1@2dc54ad4

05:05:01.602 [main] DEBUG org.springframework.context.support.ClassPathXmlApplicationContext - Closing org.springframework.cont

Type2: 코드 작성 및 작성 근거 서술 - Q3



The screenshot displays an IDE window with a Java file named `MainClass.java`. The code defines a package, imports `AnnotationConfigApplicationContext`, and implements a `main` method that creates an application context, retrieves a bean named `java1`, and prints its data. The bottom panel shows the console output of the program.

```
1 package kr.co.inhatcspring.main;
2
3 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
4
5
6
7
8 public class MainClass {
9
10     public static void main(String[] args) {
11         // TODO Auto-generated method stub
12         AnnotationConfigApplicationContext ctx1 = new AnnotationConfigApplicationContext(BeaConfigClass.class);
13
14         TestBean1 java1 = ctx1.getBean("java", TestBean1.class);
15         System.out.printf("java1.data1 : %d\n", java1.getData1());
16         System.out.printf("java1.data2 : %s\n", java1.getData2());
17         System.out.printf("java1.data3 : %s\n", java1.getData3());
18
19         ctx1.close();
20
21     }
22 }
```

Console Output:

```
<terminated> MainClass (2) [Java Application] C:\WeclipseW\pluginsW\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858W\jreW\binW\javaw.exe (2024. 4. 1
05:14:20.603 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Refreshing org.sp
05:14:20.616 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instan
05:14:20.702 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instan
05:14:20.704 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instan
05:14:20.706 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instan
05:14:20.712 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instan
05:14:20.715 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instan
java1.data1 : 418
java1.data2 : 중간고사
java1.data3 : kr.co.inhatcspring.beans.DataBean1@71ba6d4e
```

Type2: 코드 작성 및 작성 근거 서술

Q3: 다음의 xml 코드를 자바 코드상에서 setter 메서드를 활용한 자동 주입 코드로 작성하고, 작성한 코드의 근거에 대해 서술하시오.

```
DataBean1.java X
1 package kr.co.inhatcspring.beans;
2
3 public class DataBean1 {
4
5 }

TestBean1.java X
1 package kr.co.inhatcspring.beans;
2
3 public class TestBean1 {
4
5     private int data1;
6     private String data2;
7     private DataBean1 data3;
8
9     public int getData1() {
10         return data1;
11     }
12
13     public void setData1(int data1) {
14         this.data1 = data1;
15     }
16
17     public String getData2() {
18         return data2;
19     }
20
21     public void setData2(String data2) {
22         this.data2 = data2;
23     }
24
25     public DataBean1 getData3() {
26         return data3;
27     }
28
29     public void setData3(DataBean1 data3) {
30         this.data3 = data3;
31     }
32
33 }
```

```
BeanConfigClass.java X
1 package kr.co.inhatcspring.config;
2
3 import org.springframework.context.annotation.Bean;
4
5 @Configuration
6 public class BeanConfigClass {
7
8     @Bean
9     public TestBean1 java() {
10         TestBean1 t1 = new TestBean1();
11         t1.setData1(418);
12         t1.setData2("중간고사");
13         t1.setData3(new DataBean1());
14
15         return t1;
16     }
17
18 }
```

A: Configuration 어노테이션으로 BeanConfigClass.java 파일이 빈 등록을 위한 자바 파일임을 프레임워크에 명시하고, Bean 어노테이션으로 Bean 정보에 대해 등록했다. 그 후, setter 메서드를 통해 TestBean1에 값을 주입한다.

Type2: 코드 작성 및 작성 근거 서술 - Q4

```
MainClass.java ×
1 package kr.co.inhatcspring.main;
2
3 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
4
5 import kr.co.inhatcspring.beans.TestBean1;
6 import kr.co.inhatcspring.config.BeanConfigClass;
7
8 public class MainClass {
9
10     public static void main(String[] args) {
11         // TODO Auto-generated method stub
12
13         AnnotationConfigApplicationContext ctx2 = new AnnotationConfigApplicationContext(BeanConfigClass.class);
14
15         TestBean1 java1 = ctx2.getBean(TestBean1.class);
16         try {
17             java1.method1();
18         } catch (Exception e) {
19             e.printStackTrace();
20         }
21
22         ctx2.close();
23     }
24
25 }
```

Problems Servers Terminal Data Source Explorer Properties Console ×

<terminated> MainClass (3) [Java Application] C:\WeclipseW\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\java.exe (2024. 4. 10 05:28:26.454 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Refreshing org.s
05:28:26.466 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
05:28:26.508 [main] DEBUG org.springframework.context.annotation.ClassPathBeanDefinitionScanner - Identified candidate
05:28:26.511 [main] DEBUG org.springframework.context.annotation.ClassPathBeanDefinitionScanner - Identified candidate
05:28:26.573 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
05:28:26.574 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
05:28:26.576 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
05:28:26.577 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
05:28:26.579 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
05:28:26.639 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
05:28:26.656 [main] DEBUG org.springframework.aop.aspectj.annotation.ReflectiveAspectJAdvisorFactory - Found AspectJ m
05:28:26.753 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
05:28:26.782 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared insta
TestBean1의 method1 호출
afterMethod 호출

Type2: 코드 작성 및 작성 근거 서술

Q4-1: MainClass.java의 출력 결과에 대해 BeanConfigClass, TestBea1 각각에 올바른 어노테이션을 추가하시오.

Q4-2: Advisor.java에 '모든 패키지의 모든 클래스에 대해 method1이라는 이름을 가지고 있으며 매개변수를 0개 이상인 메서드가 호출될 때 weaving될 after advice'를 정의하는 코드를 작성하시오.

Q4-3: 4-1, 4-2에 작성한 코드의 작성 이유에 대해 서술하시오.

```
BeanConfigClass.java X
1 package kr.co.inhatcspring.config;
2
3 import org.springframework.context.annotation.ComponentScan;
4
5 @Configuration
6 @ComponentScan(basePackages = {"kr.co.inhatcspring.beans", "kr.co.inhatcspring.advisor"})
7 @EnableAspectJAutoProxy
8 public class BeanConfigClass {
9
10 }
11
12 }
```

```
TestBean1.java X
1 package kr.co.inhatcspring.beans;
2
3 import org.springframework.stereotype.Component;
4
5 @Component
6 public class TestBean1 {
7
8     public void method1() throws Exception{
9         System.out.println("TestBean1의 method1 호출");
10     }
11 }
12 }
```

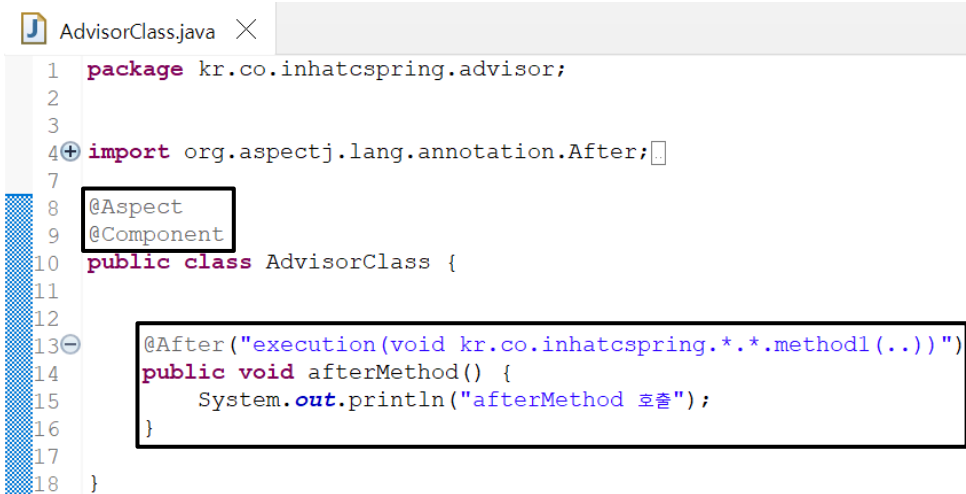
A: Configuration 어노테이션으로 BeanConfigClass.java가 객체 등록을 위한 파일임을 명시하고, ComponentScan 어노테이션으로 객체 등록을 위한 패키지의 경로를 지정한다. 자동으로 AOP 설정을 하기 위해 EnableAspectJAutoProxy 어노테이션을 명시하고, TestBean1.java에는 Component scan을 위해 Component 어노테이션을 작성한다.

Type2: 코드 작성 및 작성 근거 서술

Q4-1: MainClass.java의 출력 결과에 대해 BeanConfigClass, TestBea1 각각에 올바른 어노테이션을 추가하시오.

Q4-2: Advisor.java에 '모든 패키지의 모든 클래스에 대해 method1이라는 이름을 가지고 있으며 매개변수를 0개 이상인 메서드가 호출될 때 weaving될 after advice'를 정의하는 코드를 작성하시오.

Q4-3: 4-1, 4-2에 작성한 코드의 작성 이유에 대해 서술하시오.



```
AdvisorClass.java X
1 package kr.co.inhatspring.advisor;
2
3
4 import org.aspectj.lang.annotation.After;
5
6
7
8 @Aspect
9 @Component
10 public class AdvisorClass {
11
12
13     @After("execution(void kr.co.inhatspring.*.*.method1(..))")
14     public void afterMethod() {
15         System.out.println("afterMethod 호출");
16     }
17
18 }
```

A: AdvisorClass.java에는 AOP 설정을 위해 Aspect 어노테이션과 Component scan을 위한 Component 어노테이션을 작성한다.

After 어노테이션으로 아래에 작성될 메서드가 After advice임을 명시하고, asterisk(*)를 통해 모든 패키지의 모든 클래스 아래에 method1이라는 이름의 메서드에 대해 동작할 것을 정의한다. 매개변수의 자리에 ..을 작성해 매개변수가 0개 이상인 메서드임을 명시했다.

Type3: 단답형 답안 작성 및 근거 서술

Q5. 국제화 지원 등 문자열에 관련된 다양한 기능을 제공하며, 기업 실무용으로 사용되는 container를 beans.xml파일이 패키지 내부에 위치할 경우 어떤 메서드로 여는가?

A: **ClassPathXmlApplicationContext**

Type3: 단답형 답안 작성 및 근거 서술

Q6. 다음의 MainClass.java에서 객체가 생성되는 횟수와 생성 순서, 그 이유에 대해 서술하시오.

```
MainClass.java X
1 package kr.co.inhatcspring.main;
2
3 import org.springframework.context.support.ClassPathXmlApplicationContext;
4
5 import kr.co.inhatcspring.beans.TestBean;
6
7 public class MainClass {
8
9     public static void main(String[] args) {
10         // TODO Auto-generated method stub
11         ClassPathXmlApplicationContext ctx = new ClassPathXmlApplicationContext("kr/co/inhatcspring/config/beans.xml");
12
13         TestBean t1 = ctx.getBean("bean1", TestBean.class);
14         System.out.printf("t1: %s\n", t1);
15
16         TestBean t2 = ctx.getBean("bean2", TestBean.class);
17         System.out.printf("t2: %s\n", t2);
18
19         TestBean t3 = ctx.getBean("bean3", TestBean.class);
20         System.out.printf("t3: %s\n", t3);
21
22         TestBean t4 = ctx.getBean("bean1", TestBean.class);
23         System.out.printf("t4: %s\n", t4);
24
25         TestBean t5 = ctx.getBean("bean3", TestBean.class);
26         System.out.printf("t5: %s\n", t5);
27
28         ctx.close();
29     }
30 }
31 }
```

```
beans.xml X
http://www.springframework.org/schema/beans/spring-beans.xsd (xsi:schemaLocation)
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <beans xmlns="http://www.springframework.org/schema/beans"
4       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans
6                           http://www.springframework.org/schema/beans/spring-beans.xsd">
7
8     <bean id = 'bean1' class='kr.co.inhatcspring.beans.TestBean' lazy-init = "true"/>
9
10    <bean id = 'bean2' class='kr.co.inhatcspring.beans.TestBean'/>
11
12    <bean id = 'bean3' class='kr.co.inhatcspring.beans.TestBean' lazy-init = "true" scope = 'prototype'/>
13
14 </beans>
```

A: 4번, bean2->bean1->bean3->bean3

bean2는 lazy-init이 false(default)기 때문에, xml파일이 로딩될 때 객체가 가장 먼저 생성된다.

bean1은 lazy-init이 true이기 때문에 객체가 getbean으로 호출될 때 생성된다.

bean3는 lazy-init이 true고, scope이 prototype이기 때문에 객체가 getbean으로 호출될 때 생성되며, 그 후에 호출될 때 마다 새로 생성된다.