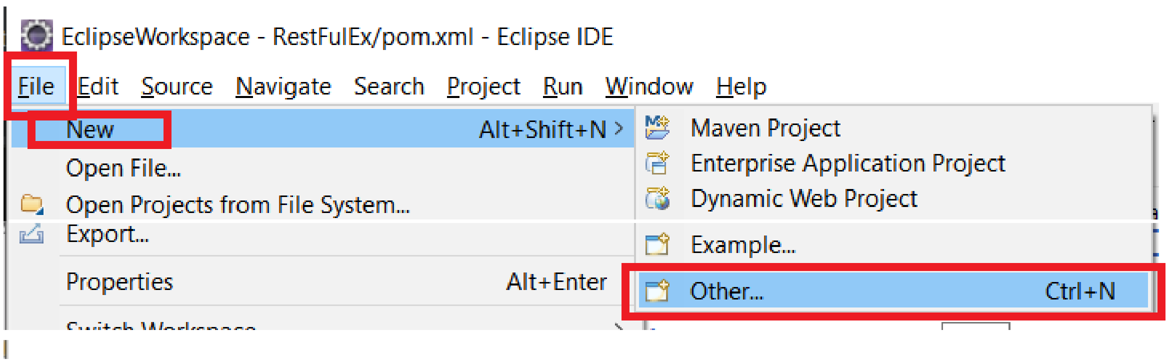
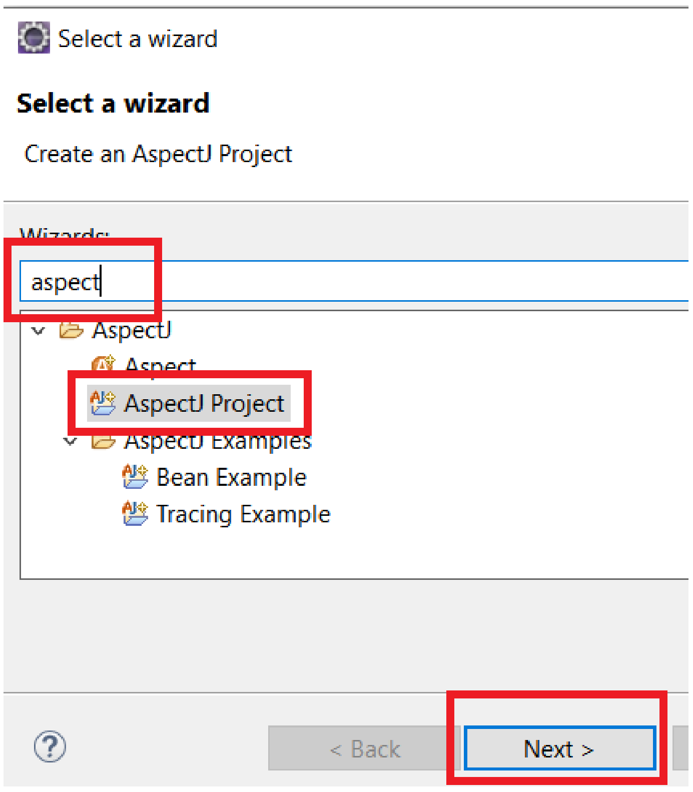
Steps to Create an AOP Project

Step 1 : Creating AspectJ Project.

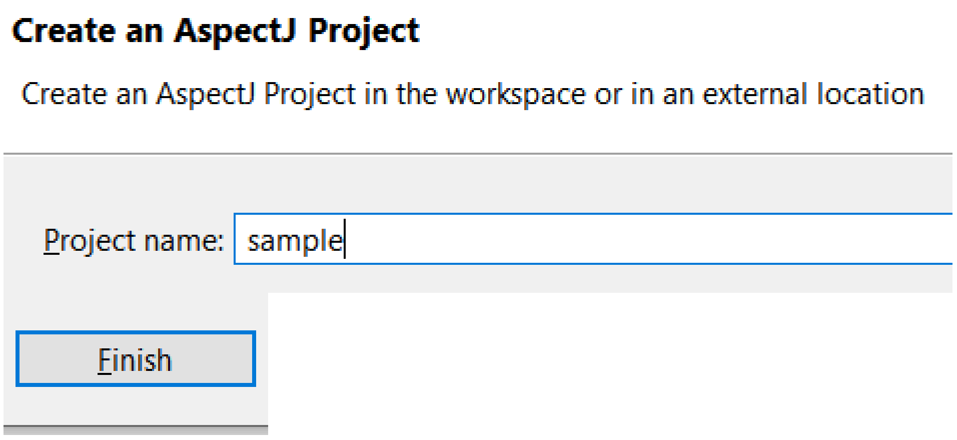
1.1 : Open Eclipse. Go To File > New > Other.


EclipseWorkspace - RestFulEx/pom.xml - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help
New Alt+Shift+N >
Open File...
Open Projects from File System...
Export...
Properties Alt+Enter
Maven Project
Enterprise Application Project
Dynamic Web Project
Example...
Other... Ctrl+N

1.2 : Search for 'aspect' and Select 'AspectJ Project'. Then Click on Next.



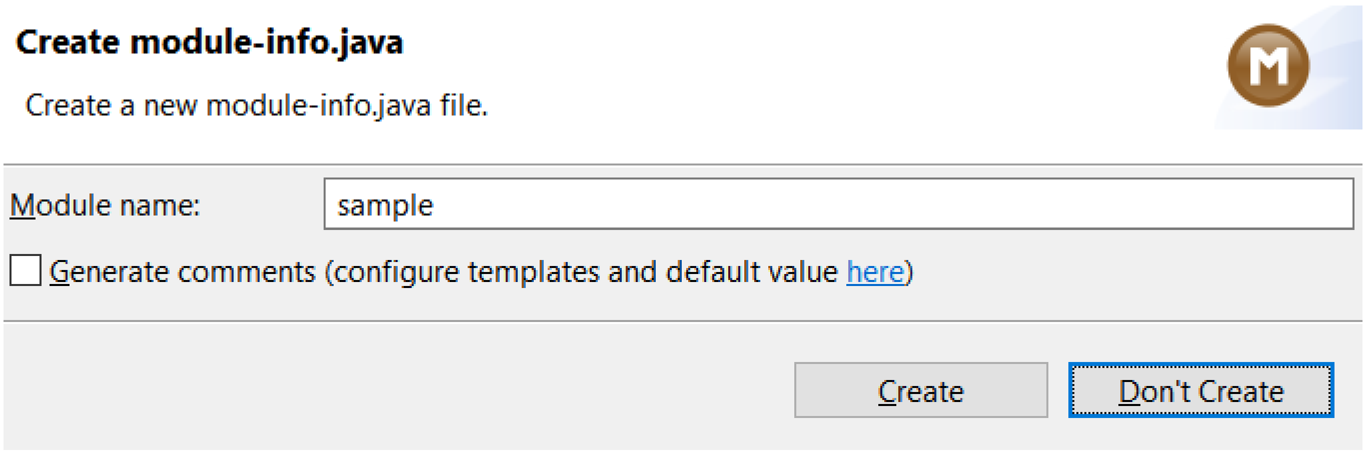
1.3 : Enter Project Name of your wish, and click on Finish.


Create an AspectJ Project

Create an AspectJ Project in the workspace or in an external location

Project name: sample

1.4 : If asked to create module-info.java file, select 'Don't Create'.

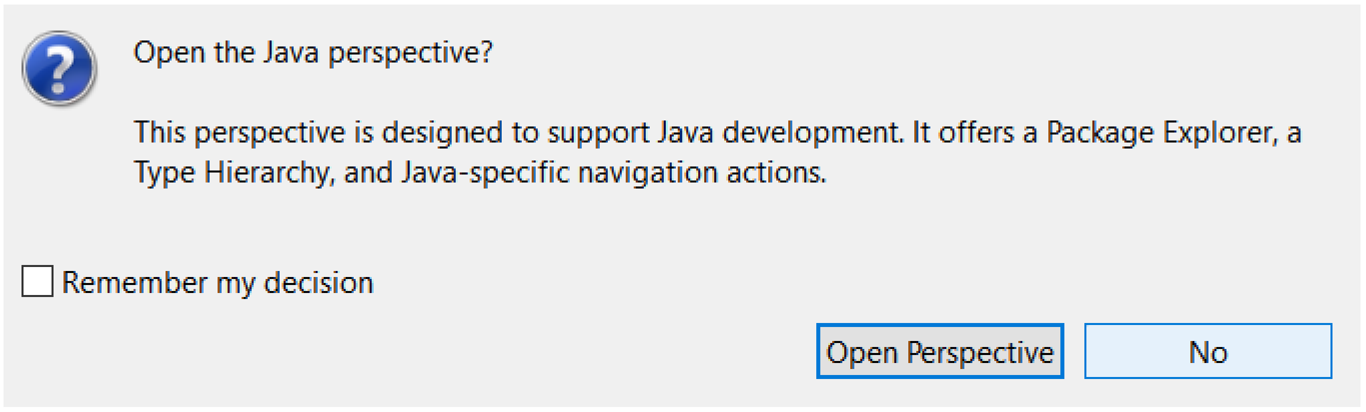

Create module-info.java

Create a new module-info.java file.

Module name: sample

☐ Generate comments (configure templates and default value here)

1.5 : Finally if you are asked to Open Java Perspective, just choose **NO.**


Open the Java perspective?

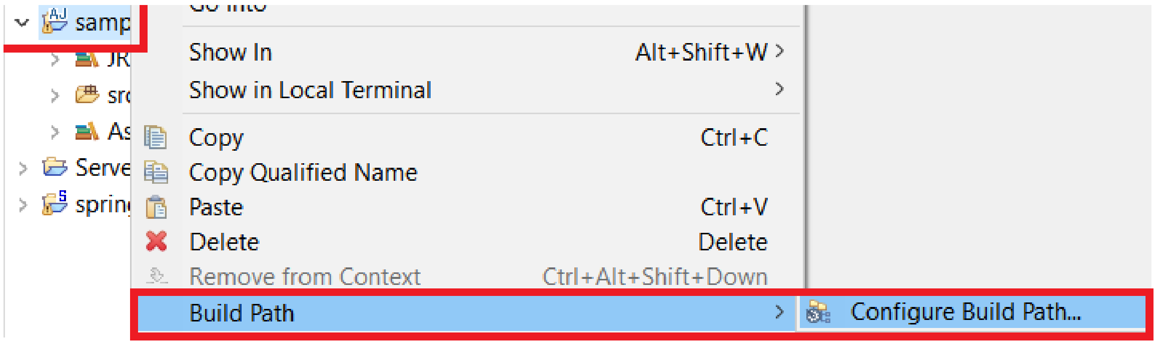
This perspective is designed to support Java development. It offers a Package Explorer, a Type Hierarchy, and Java-specific navigation actions.

☐ Remember my decision

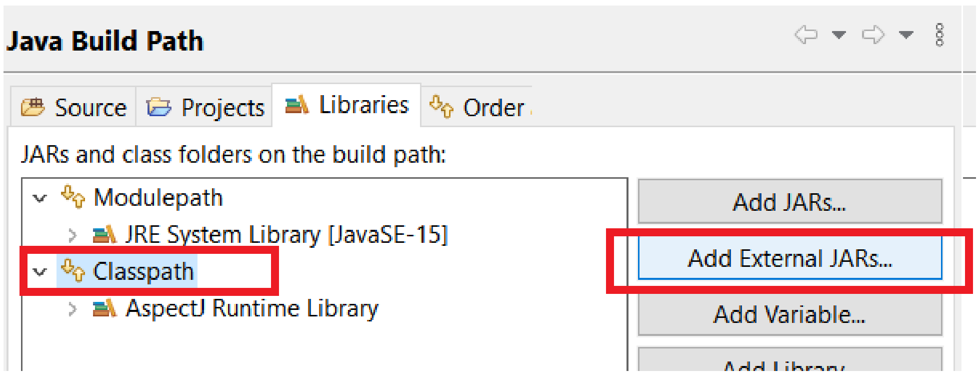
This creates your AspectJ project.

Step 2 : Adding the Spring Libraries.

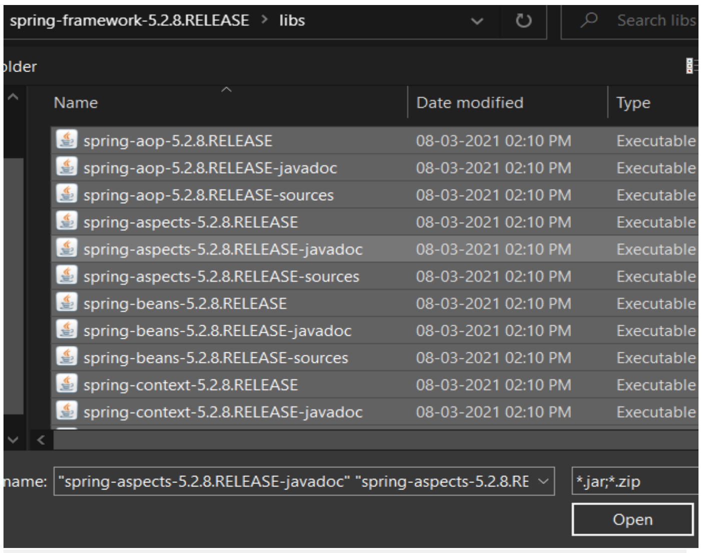
2.1 : Right click on your Newly created AspectJ project, Choose Build Path > Configure Build Path.



2.2 On Java Build Path wizard, Choose **Classpath**and then select **Add External JARs.**



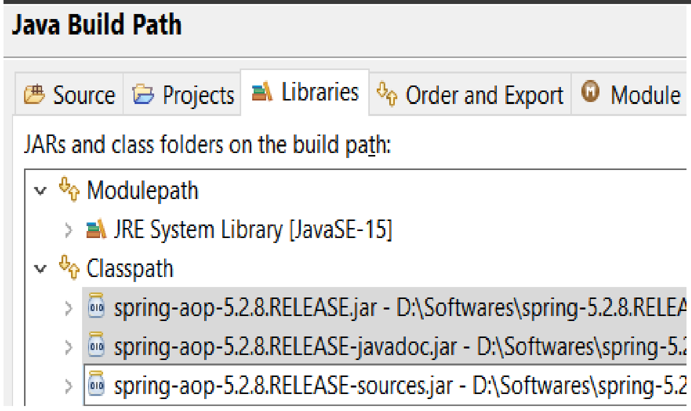
2.3 : Choose all the Spring Libraries you've downloaded, and click on OPEN. This will add all libraries to Classpath.


spring-framework-5.2.8.RELEASE > libs

Name	Date modified	Type
spring-aop-5.2.8.RELEASE	08-03-2021 02:10 PM	Executable
spring-aop-5.2.8.RELEASE-javadoc	08-03-2021 02:10 PM	Executable
spring-aop-5.2.8.RELEASE-sources	08-03-2021 02:10 PM	Executable
spring-aspects-5.2.8.RELEASE	08-03-2021 02:10 PM	Executable
spring-aspects-5.2.8.RELEASE-javadoc	08-03-2021 02:10 PM	Executable
spring-aspects-5.2.8.RELEASE-sources	08-03-2021 02:10 PM	Executable
spring-beans-5.2.8.RELEASE	08-03-2021 02:10 PM	Executable
spring-beans-5.2.8.RELEASE-javadoc	08-03-2021 02:10 PM	Executable
spring-beans-5.2.8.RELEASE-sources	08-03-2021 02:10 PM	Executable
spring-context-5.2.8.RELEASE	08-03-2021 02:10 PM	Executable
spring-context-5.2.8.RELEASE-javadoc	08-03-2021 02:10 PM	Executable

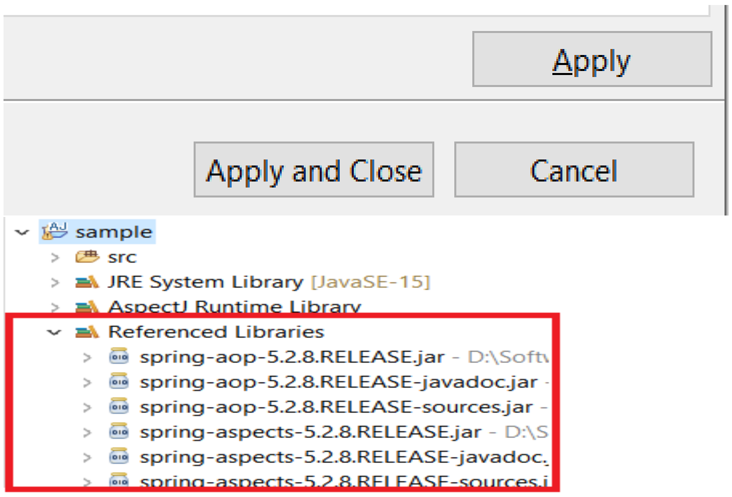
name: "spring-aspects-5.2.8.RELEASE-javadoc" "spring-aspects-5.2.8.RE

.jar;.zip



spring-aop-5.2.8.RELEASE.jar - D:\Softwares\spring-5.2.8.RELEA
spring-aop-5.2.8.RELEASE-javadoc.jar - D:\Softwares\spring-5.2
spring-aop-5.2.8.RELEASE-sources.jar - D:\Softwares\spring-5.2

2.4 Finally click on Apply & Close, now you are ready to work with Aspects in Spring.



```
sample
  src
  JRE System Library [JavaSE-15]
  AspectJ Runtime Library
  Referenced Libraries
    spring-aop-5.2.8.RELEASE.jar - D:\Softw
    spring-aop-5.2.8.RELEASE-javadoc.jar -
    spring-aop-5.2.8.RELEASE-sources.jar -
    spring-aspects-5.2.8.RELEASE.jar - D:\S
    spring-aspects-5.2.8.RELEASE-javadoc.
    spring-aspects-5.2.8.RELEASE-sources.j
```

Problem Statement 1 : Write a program to demonstrate Spring AOP – before advice.

Solution :

beforeaop.java

```
package bvimit.edu;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class beforeaop {

        @Pointcut("execution(int beforeoperation.*(..))")
        public void p(){}

        @Before("p()")
        public void myadvice(JoinPoint jp)
        {
                System.out.println("before advice");
        }
}
```

beforeoperation.java

```
package bvimit.edu;

publicclass beforeoperation {
publicvoid msg() {System.out.println("method 1");}
publicint m(){System.out.println("method 2 with return");return 2;}
publicint k(){System.out.println("method 3 with return");return 3;}
    }
```

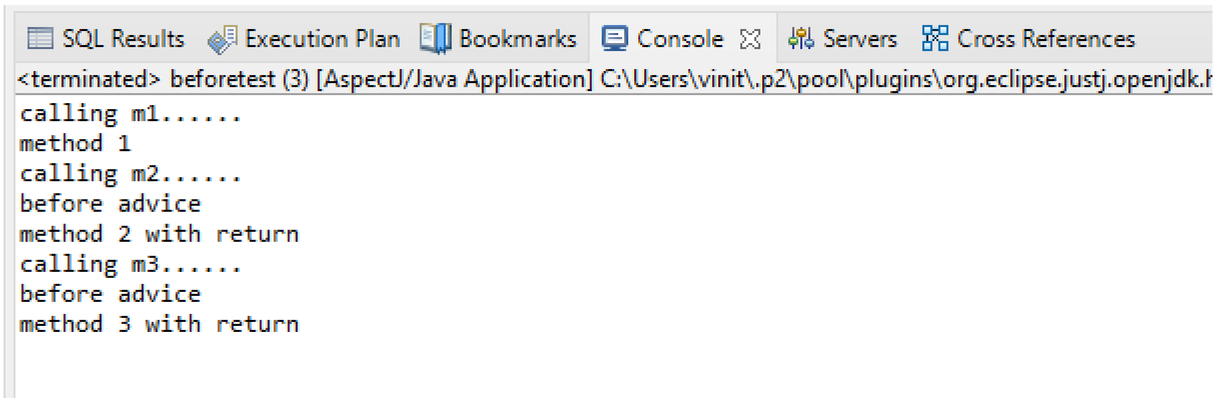
aopctx1.xml

```
<?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="opBean" class="bvimit.edu.beforeoperation"></bean>

<bean id="trackMyBean" class="bvimit.edu.beforeaop"></bean>



<bean
class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCrea
tor"></bean>

</beans>
```

beforetest.java

```
package bvimit.edu;

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

public class beforetest {

	public static void main(String[] args) {
		ApplicationContext context = new
ClassPathXmlApplicationContext("aopctx1.xml");
		beforeoperation e = (beforeoperation) context.getBean("opBean");
		System.out.println("calling m1......");
		e.msg();
		System.out.println("calling m2......");
		e.m();
		System.out.println("calling m3......");
		e.k();


	}

}
```

Output :



```
<terminated> beforetest (3) [AspectJ/Java Application] C:\Users\vinit\.p2\pool\plugins\org.eclipse.justj.openjdk.
calling m1......
method 1
calling m2......
before advice
method 2 with return
calling m3......
before advice
method 3 with return
```

Problem Statement 2 : Write a program to demonstrate Spring AOP – after advice.

Solution :

Afteraopdata.java

```
package bvimit.edu;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class afteraopdata {

        @Pointcut("execution(int afteroperation.*(..))")
        public void p(){}

        @After("p()")
        public void myadvice(JoinPoint jp)
        {
                System.out.println("after advice");
        }
}
```

afteroperation.java

```
package bvimit.edu;

publicclass afteroperation {
publicvoid msg() {System.out.println("method 1");}
publicint m(){System.out.println("method 2 with return");return 2;}
publicint k(){System.out.println("method 3 with return");return 3;}
    }
```

aopctx.xml

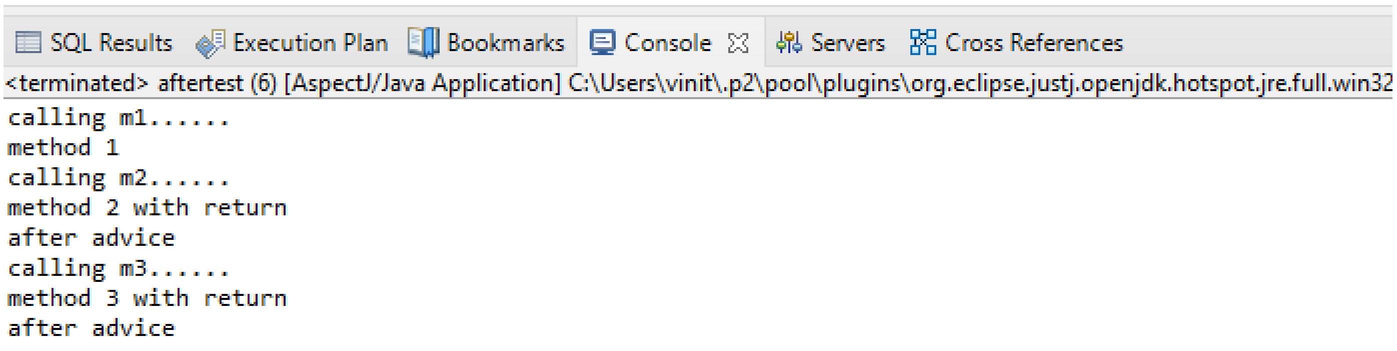
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="opBean" class="bvimit.edu.afteroperation"></bean>

<bean id="trackMyBean" class="bvimit.edu.afteraopdata"></bean>



<bean
class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCrea
tor"></bean>
</beans>
```

aftertest.java

```
package bvimit.edu;

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

public class aftertest {

	public static void main(String[] args) {
		ApplicationContext context = new
ClassPathXmlApplicationContext("aopctx.xml");
		afteroperation e = (afteroperation) context.getBean("opBean");
		System.out.println("calling m1......");
		e.msg();
		System.out.println("calling m2......");
		e.m();
		System.out.println("calling m3......");
		e.k();


	}

}
```

Output :



```
<terminated> aftertest (6) [AspectJ/Java Application] C:\Users\vinit\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
calling m1......
method 1
calling m2......
method 2 with return
after advice
calling m3......
method 3 with return
after advice
```

Problem Statement 3 : Write a program to demonstrate Spring AOP – around advice.

Solution :

Bankaopdata.java

```
package bvimit.edu;

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class Bankaopdata {

	@Pointcut("execution(* Bank.*(..))")
	public void a() {}

	@Around("a()")
	public Object myadvice(ProceedingJoinPoint p)throws Throwable
	{
		System.out.println("Around concern Before calling actual method");
		Object obj=p.proceed();
		System.out.println("Around Concern After calling actual method");
		return obj;
	}
}
```

Bank.java

```
package bvimit.edu;

publicclass Bank {
	publicvoid welcome() {System.out.println("welcome to bank");}
	publicint icici() {System.out.println("icici bank interest rate");return 7;}
	publicint pnb() {System.out.println("pnb bank interest rate");return 6;}
}
```

Bankaopdata.xml

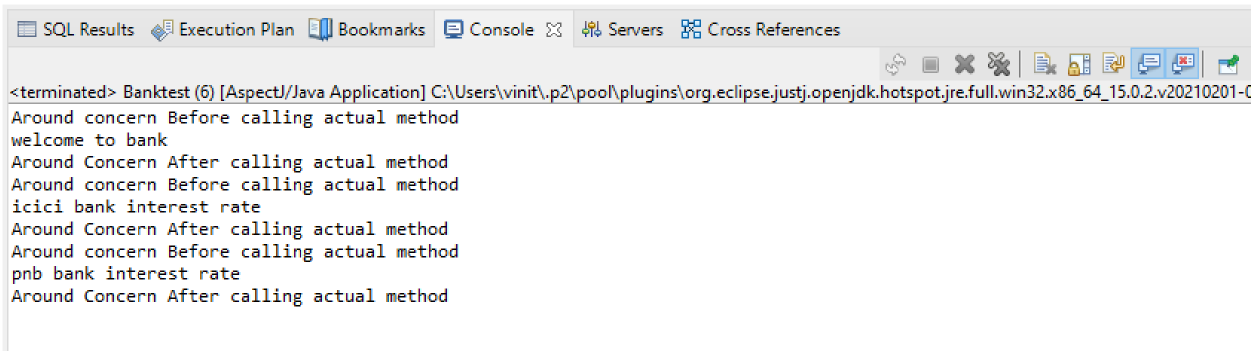
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="opBean" class="bvimit.edu.Bank"></bean>
<bean id="trackMyBean" class="bvimit.edu.Bankaopdata"></bean>

<bean
class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCrea
tor"></bean>
</beans>
```

Banktest.java

```
package bvimit.edu;

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

publicclass Banktest {

	privatestatic ApplicationContext context;

	publicstaticvoid main(String[] args) {
		context = new ClassPathXmlApplicationContext("Bankaopdata.xml");

		Bank e =(Bank) context.getBean("opBean");
		System.out.println("Calling welcome method...");
		e.welcome();
		System.out.println("Calling icici method...");
		e.icici();
		System.out.println("Calling pnb method...");
		e.pnb();
	}
```

}

Output :



```
<terminated> Banktest (6) [AspectJ/Java Application] C:\Users\vinit\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0
Around concern Before calling actual method
welcome to bank
Around Concern After calling actual method
Around concern Before calling actual method
icici bank interest rate
Around Concern After calling actual method
Around concern Before calling actual method
pnb bank interest rate
Around Concern After calling actual method
```

Problem Statement 4 : Write a program to demonstrate Spring AOP – after returning advice.

Solution :

Bankaopdata.java

```
package bvimit.edu;

import org.aspectj.lang.JoinPoint;
importorg.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.AfterReturning;
importorg.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
importorg.aspectj.lang.annotation.Pointcut;

@Aspect
publicclass Bankaopdata {

        @AfterReturning(
                        pointcut ="execution(* Bank.*(..))",
                        returning="result")
publicvoid myadvice(JoinPoint jp,Object result)
{
        System.out.println("AfterReturning concern");
        System.out.println("Result in advice" +result);
        }
}
```

Bank.java

```
package bvimit.edu;

publicclass Bank {
        publicvoid welcome() {System.out.println("welcome to bank");;}
        publicint icici() {System.out.println("icici bank interest rate");return 7;}
        publicint pnb() {System.out.println("pnb bank interest rate");return 6;}
}
```

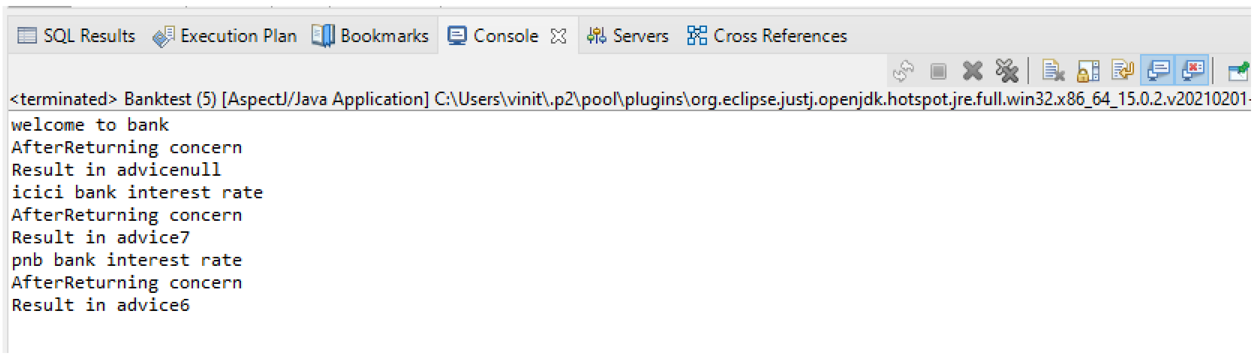
Bankaopdata.xml

```
<?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="opBean" class="bvimit.edu.Bank"></bean>
<bean id="trackMyBean" class="bvimit.edu.Bankaopdata"></bean>

<bean
class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCrea
tor"></bean>
</beans>
```

Banktest.java

```
package bvimit.edu;

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

publicclass Banktest {

        privatestatic ApplicationContext context;

        publicstaticvoid main(String[] args) {
                context = new ClassPathXmlApplicationContext("Bankaopdata.xml");

                Bank e =(Bank) context.getBean("opBean");
                //System.out.println("Calling welcome method...");
                e.welcome();
                //System.out.println("Calling icici method...");
                e.icici();
                //System.out.println("Calling pnb method...");
                e.pnb();
        }

}
```

Output :



```
<terminated> Banktest (5) [AspectJ/Java Application] C:\Users\vinit\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201
welcome to bank
AfterReturning concern
Result in advicenull
icici bank interest rate
AfterReturning concern
Result in advice7
pnb bank interest rate
AfterReturning concern
Result in advice6
```

Problem Statement 5 : Write a program to demonstrate Spring AOP – after throwing advice.

Solution :

Operationaop_at.java

```
package bvimit.edu;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Aspect;

@Aspect
publicclass Operationaop_at {
@AfterThrowing(
                pointcut = "execution(* Operation_at.*(..))", throwing = "error")
        publicvoid myadvice(JoinPoint jp, Throwable error)
        {
                System.out.println("AfterThrowing concern");
                System.out.println("Exception is: "+error);
                System.out.println("end of after throwing advice....");
        }
        }
```

Operation_at.java

```
package bvimit.edu;
publicclass Operation_at {

        publicvoid validate(int att)throws Exception{
                if(att<75) {
                        thrownew ArithmeticException("Not eligible for exam");
                }
                else {
                        System.out.println("Eligible for exam");
                }
        }
}
```

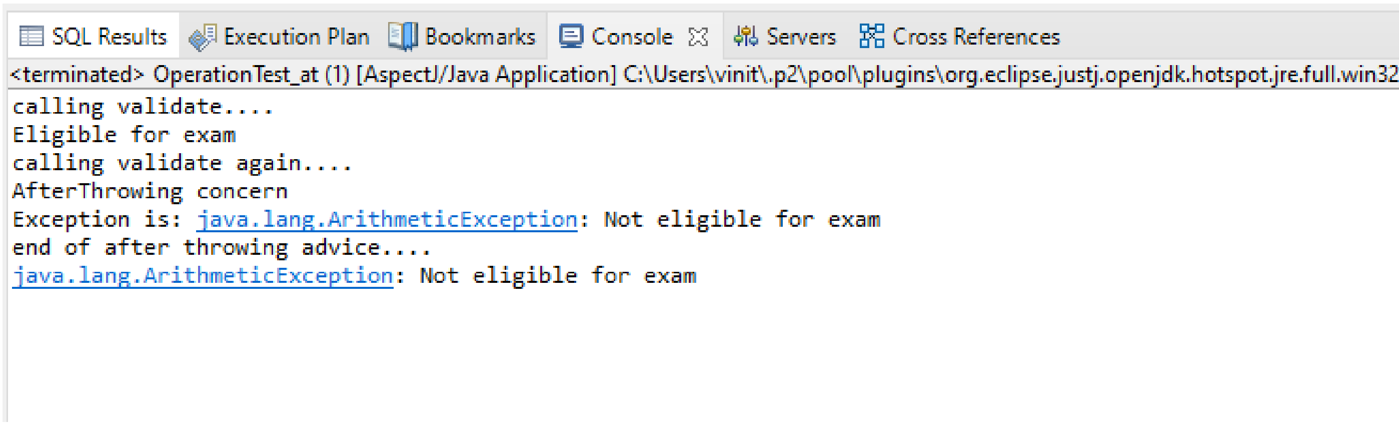
validctx.xml

```
<?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="opBean" class="bvimit.edu.Operation_at"></bean>

<bean id="trackMyBean" class="bvimit.edu.Operationaop_at"></bean>

<bean
class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCrea
tor"></bean></beans>
```

TestValidation.java

```
package bvimit.edu;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class OperationTest_at {
private static ApplicationContext context;
                public static void main(String[] args) {
ApplicationContext context = new ClassPathXmlApplicationContext("validctx.xml");
                         Operation_at op = (Operation_at) context.getBean("opBean");
                        System.out.println("calling validate....");
                         try {
                                 op.validate(85);
                         }catch(Exception e){System.out.println(e);}

                         System.out.println("calling validate again....");

                         try {
                                 op.validate(25);
                         }catch(Exception e){System.out.println(e);}
                         }
                        }
```

Output :



```
<terminated> OperationTest_at (1) [AspectJ/Java Application] C:\Users\vinit\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
calling validate....
Eligible for exam
calling validate again....
AfterThrowing concern
Exception is: java.lang.ArithmeticException: Not eligible for exam
end of after throwing advice....
java.lang.ArithmeticException: Not eligible for exam
```

Problem Statements 6: Write a program to demonstrate Spring AOP –pointcuts.

<u>Solution:</u>

Operation_pc.java

```
package bvimit.edu;
publicclass Operation_pc {
		publicvoid msg() {System.out.println("method 1");}
		publicint m() {System.out.println("method 2 with return");return 2;}
		publicint k() {System.out.println("method 3 with return");return 3;}
		}
```

Aopdata_pc.java

```
package bvimit.edu;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Pointcut;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
@Aspect
public class Aopdata_pc {

	@Pointcut("execution(int Operation.*(..))")
	public void p(){}

	@After("p()")
	public void myadvice(JoinPoint jp)
	{
		System.out.println("After advice");
	}
	@Pointcut("execution(* Operation.*(..))")
	public void i(){}

	@Before("i()")
	public void myadvice1(JoinPoint jp)
	{
		System.out.println("Before advice");
}
}
```

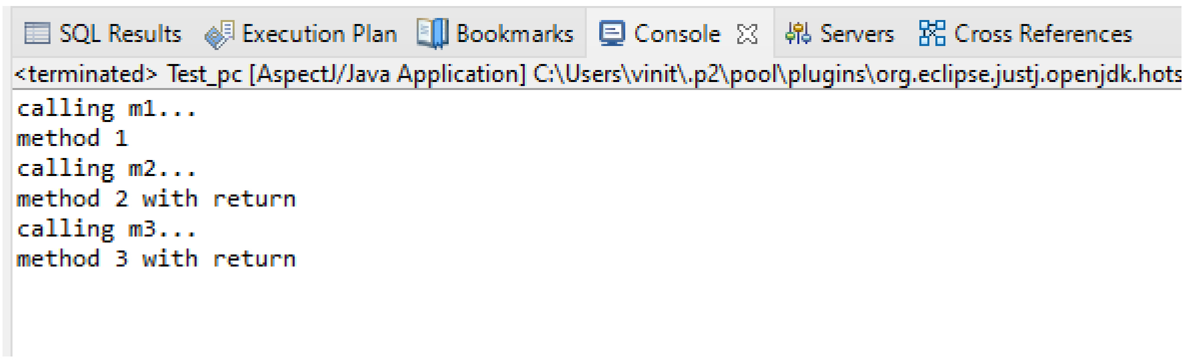
Test_pc.java

```
package bvimit.edu;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test_pc {

public static void main(String[] args) {


ApplicationContext context = new ClassPathXmlApplicationContext("aopctx_pc.xml");

                Operation_pc e=(Operation_pc)context.getBean("opBean");

                System.out.println("calling m1...");

                e.msg();

                System.out.println("calling m2...");

                e.m();

                System.out.println("calling m3...");

                e.k();

                }

}
```

aopctx_pc.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="opBean" class="bvimit.edu.Operation_pc"></bean>

<bean id="trackMyBean" class="bvimit.edu.Aopdata_pc"></bean>

<bean
class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCrea
tor"></bean>

</beans>
```

Output:



```
<terminated> Test_pc [AspectJ/Java Application] C:\Users\vinit\.p2\pool\plugins\org.eclipse.justj.openjdk.hots
calling m1...
method 1
calling m2...
method 2 with return
calling m3...
method 3 with return
```