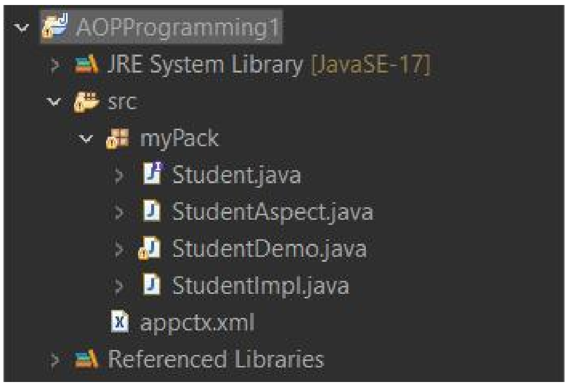
Programming Assignment No. 01

Spring and AOP

Q.1 Demonstrate Spring AOP JoinPoint and Advice Arguments with an example for Student class and interface related with the Student class.



```
AOPProgramming1
  JRE System Library [JavaSE-17]
  src
    myPack
      Student.java
      StudentAspect.java
      StudentDemo.java
      StudentImpl.java
    appctx.xml
  Referenced Libraries
```

Let's consider a simple example with a Student interface and its implementation class. We'll create an aspect to log information before and after the execution of methods in the Student class using Spring AOP.

Create the Student interface and its implementation.

Student.java

```
package myPack;

public interface Student {
void study();
String getName();
void setName(String name);
}
```

StudentAspect.java

```
package myPack;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Before;

public class StudentAspect {
@Before("execution(* myPack.Student.*(..))")
public void beforeMethodExecution(JoinPoint jp) {
System.out.println("Before executing method: " + jp.getSignature().getName());
Object[] args = jp.getArgs();
for (Object arg : args) {
System.out.println("Argument: " + arg);
}
}
@After("execution(* myPack.Student.*(..))")
public void afterMethodExecution(JoinPoint jp) {
System.out.println("After executing method: " + jp.getSignature().getName());
}
}
```

StudentDemo.java

```
package myPack;

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

public class StudentDemo{

public static void main(String[] args) {
ApplicationContext ctx = new ClassPathXmlApplicationContext("appctx.xml");
Student student = (Student) ctx.getBean("student");
student.study();
student.setName("Raju");
System.out.println("Student name: " + student.getName());
}
}
```

StudentImpl.java

```
package myPack;

public class StudentImpl implements Student {
private String name;

@Override
public void study() {
System.out.println(getName() + " is Learning Codes.");
}

@Override
public String getName() {
return name;
}

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

In this example, the beforeMethodExecution method will be executed before any method in the Student class, and it will print the method name and arguments. The afterMethodExecution method will be executed after any method in the Student class and will print the method name.

Appctx.xml

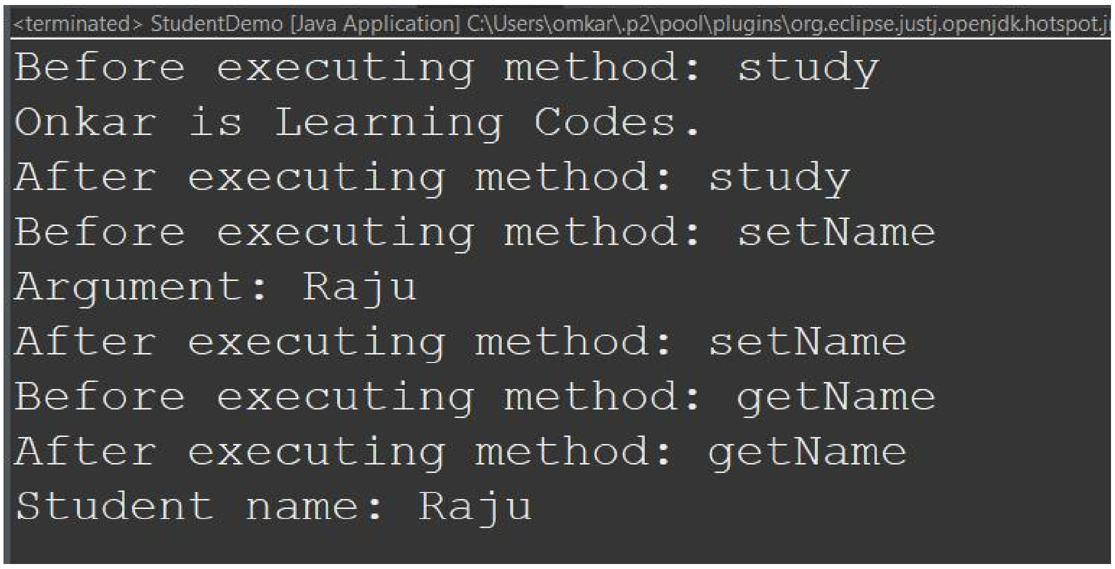
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:aop="http://www.springframework.org/schema/aop"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">

<aop:aspectj-autoproxy proxy-target-class="true"/>
<bean id="studyBean" class="myPack.StudentAspect"></bean>
<bean id="student" class="myPack.StudentImpl">
<property name="name" value="Onkar"/>
</bean>

<bean id="studentAspect" class="myPack.StudentAspect"/>
<aop:config>
<aop:aspect id="studentAspect" ref="studentAspect">
<aop:before method="beforeMethodExecution" pointcut="execution(*
myPack.Student.*(..))"/>
<aop:after method="afterMethodExecution" pointcut="execution(*
myPack.Student.*(..))"/>
</aop:aspect>
</aop:config>

</beans>
```

Output:



```
<terminated> StudentDemo [Java Application] C:\Users\omkar\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.j
Before executing method: study
Onkar is Learning Codes.
After executing method: study
Before executing method: setName
Argument: Raju
After executing method: setName
Before executing method: getName
After executing method: getName
Student name: Raju
```