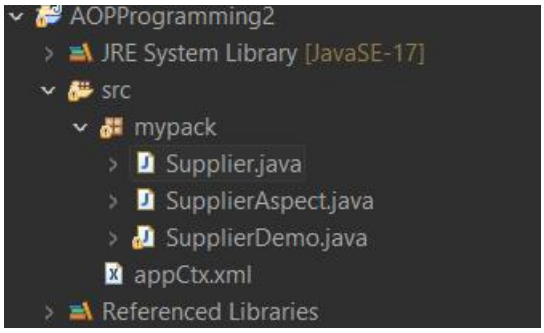
Q.2. Demonstrate Spring AOP After Returning Advice with an example for Supplier class.



In Spring AOP, the @AfterReturning advice is used to perform actions after the method execution, only if the method completes normally (i.e., without any exception). Let's create a simple example using a Supplier class to demonstrate @AfterReturning advice.

Supplier.java

```
package mypack;

public class Supplier {
private String name;

public Supplier(String name) {
this.name = name;
}

public String getName() {
return name;
}

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

1. The Supplier class has a getName() method that returns the name of the supplier.

2. The SupplierAspect class is an aspect that contains the @AfterReturning advice. This advice is triggered after the getName() method of the Supplier class is executed successfully.

3. The SupplierDemo class is the entry point of the application. It creates an instance of the Supplier class and calls its methods. The Spring AOP aspect is applied automatically, and you should see the message from the @AfterReturning advice in the console after calling the getName() method.

SupplierAspect.java

```
package mypack;

import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;
```

```
@Aspect
@Component
public class SupplierAspect {

@AfterReturning(pointcut = "execution(* mypack.Supplier.getName())",returning =
"result")
public void afterReturningAdvice(String result) {
System.out.println("After Returning Advice: The returned value is " + result);
}
}
```

SupplierDemo.java

```
package mypack;

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

public class SupplierDemo {

public static void main(String[] args) {
// Load the Spring application context from XML
ApplicationContext ctx = new ClassPathXmlApplicationContext("appctx.xml");

// Get the Supplier bean from the context
Supplier supplier = (Supplier) ctx.getBean("supplier");

// Calling the method on Supplier
String supplierName = supplier.getName();
System.out.println("Supplier Name: " + supplierName);

// Changing the Supplier name
supplier.setName("Onkar Supplier");

// Calling the method again on Supplier
String updatedSupplierName = supplier.getName();
System.out.println("Updated Supplier Name: " + updatedSupplierName);
}
}
```

appCtx.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">

<!-- Define the Supplier bean -->
<bean id="supplier" class="mypack.Supplier">
<constructor-arg value="Yash Supplier"/>
</bean>

<!-- Define the SupplierAspect bean -->
<bean id="supplierAspect" class="mypack.SupplierAspect"/>

<!-- Configure AOP aspect for the getName() method in the Supplier class -->
<aop:config>
<aop:aspect id="supplierAspect" ref="supplierAspect">
<aop:after-returning method="afterReturningAdvice"
pointcut="execution(* mypack.Supplier.getName())"
returning="result"/>
</aop:aspect>
</aop:config>

</beans>
```

Output: