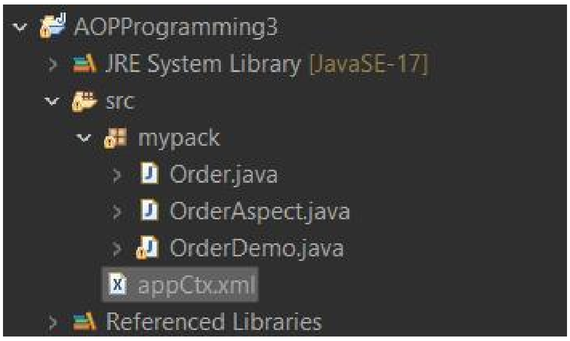
Q.3. Demonstrate Spring AOP After Throwing Advice with an example for Order class.



```
AOPProgramming3
  JRE System Library [JavaSE-17]
  src
    mypack
      Order.java
      OrderAspect.java
      OrderDemo.java
    appCtx.xml
  Referenced Libraries
```

The @AfterThrowing advice in Spring AOP is used to perform actions after a method throws an exception. Let's create an example using an Order class and XML-based configuration.

Order.java

```
package mypack;

public class Order {
private int orderId;

public Order(int orderId) {
this.orderId = orderId;
}

public void processOrder() {
// Simulate an exception during order processing
throw new RuntimeException("Error processing order " + orderId);
}
}
```

OrderAspect.java

```
package mypack;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.AfterThrowing;

public class OrderAspect {
@AfterThrowing(pointcut = "execution(* mypack.Order.processOrder())",throwing =
"exception")
public void afterThrowingAdvice(JoinPoint joinPoint, Exception exception) {
System.out.println("After Throwing Advice: Exception thrown in method " +
joinPoint.getSignature().getName() + ", Message: " + exception.getMessage());
}
}
```

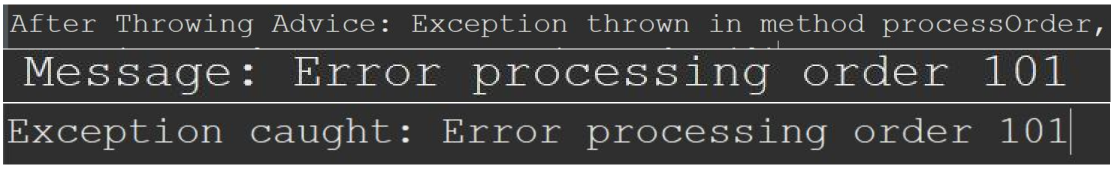
OrderDemo.java

```java
package mypack;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class OrderDemo {
public static void main(String[] args) {
// Load the Spring application context from XML
ApplicationContext ctx = new ClassPathXmlApplicationContext("appCtx.xml");
// Get the Order bean from the context
Order order = (Order) ctx.getBean("order");

// Call the method on Order (which will throw an exception)
try {
order.processOrder();
} catch (Exception e) {
// Handle the exception
System.out.println("Exception caught: " + e.getMessage());
}
}
}
```

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">

<!-- Define the Order bean -->
<bean id="order" class="mypack.Order">
<constructor-arg value="101"/>
</bean>

<!-- Define the OrderAspect bean -->
<bean id="orderAspect" class="mypack.OrderAspect"/>

<!-- Configure AOP aspect for the processOrder() method in the Order class -->
<aop:config>
<aop:aspect id="orderAspect" ref="orderAspect">
<aop:after-throwing method="afterThrowingAdvice"
pointcut="execution(* mypack.Order.processOrder())"
throwing="exception"/>
</aop:aspect>
</aop:config>

</beans>
```

Output:



```
After Throwing Advice: Exception thrown in method processOrder,
 Message: Error processing order 101
Exception caught: Error processing order 101
```