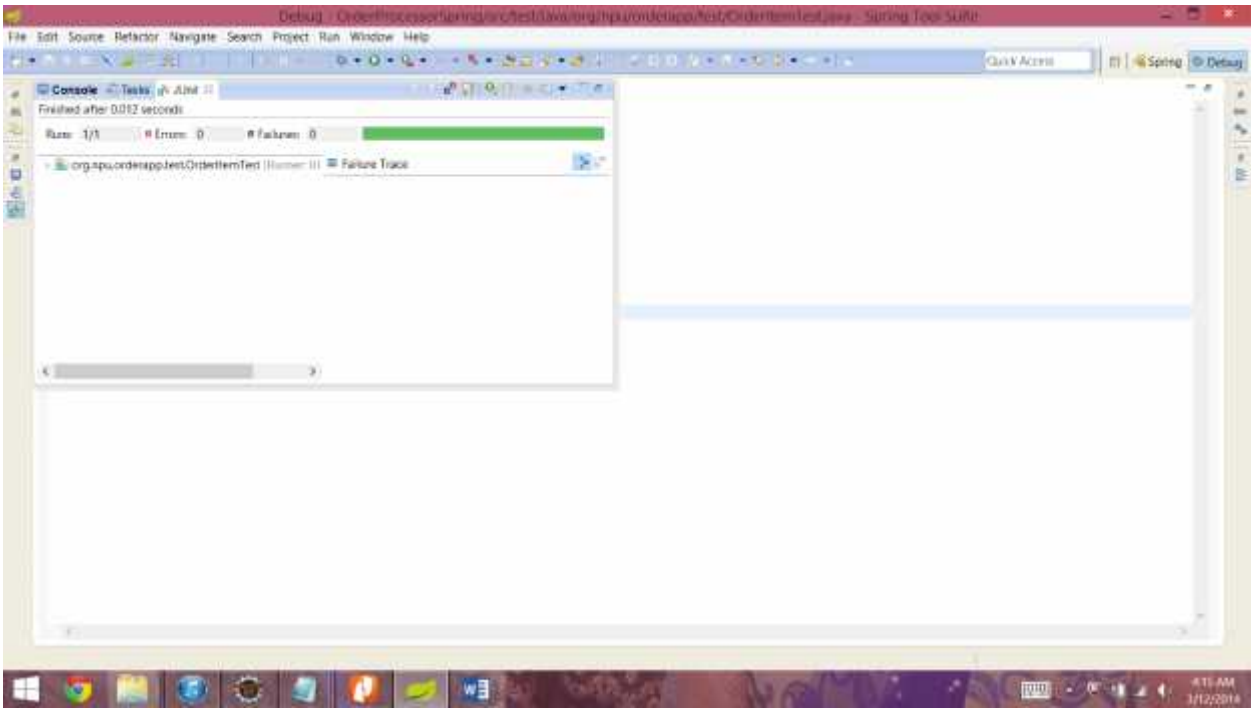
1. The exceptions you have used – the class name for the exception and under what circumstances it would be triggered.

All the exceptions are written in "org.npu.orderapp.exceptions" package.

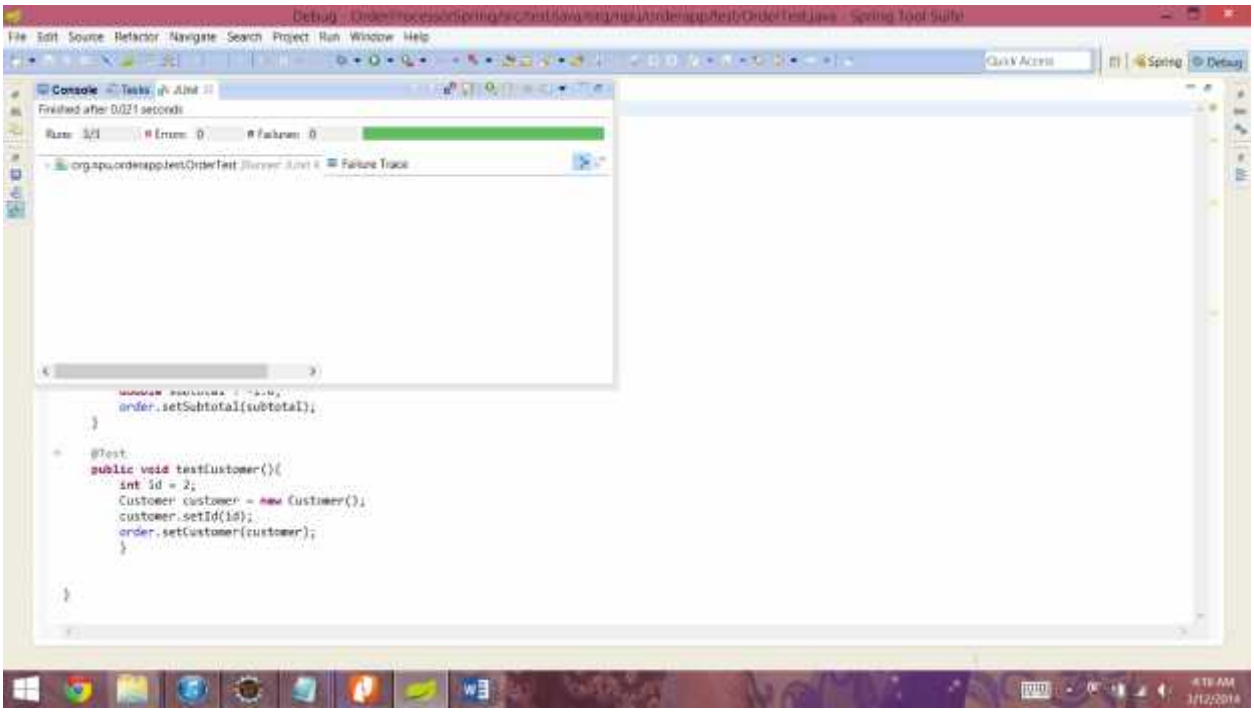
The Class name for the Exceptions are as:

1. OrderItemException
2. OrderException
3. TaxServiceException

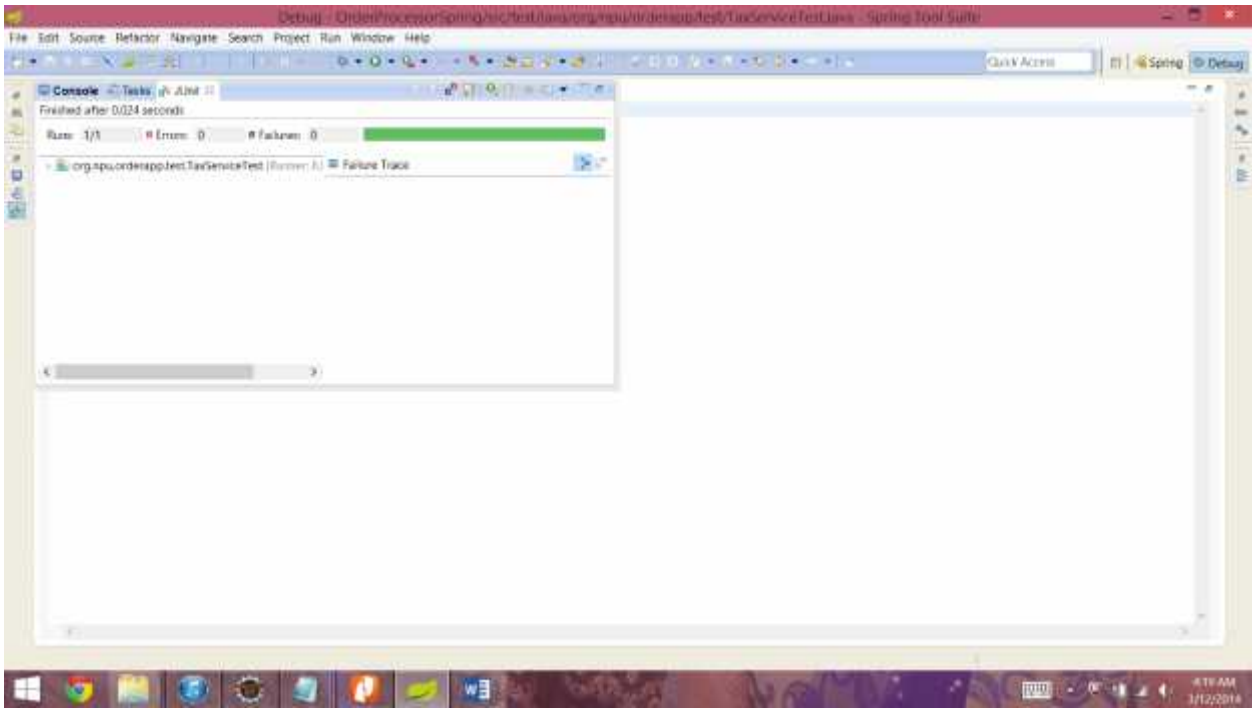
1.**OrderItemException**: This exception is triggered when the quantity of the order increases by a certain amount of value which I gave i.e 10, then this exception is triggered and this exception is called in setQuantity() method of the OrderItem class. The Output of the running JUnit test of OrderItemTest.java is as follow:



2. OrderException: This exception is triggered when the SubTotal or say the calculation of the total amount is in negative number then this exception is triggered and is called in Order.java class and the method is testSubTotal().This Exception is triggered when the subtotal entered is less than 0.0 i.e. it is a negative number. This exception is tested in OrderTest class using setSubTotal(double subtotal).The Output of the running JUnit test of OrderItemTest.java is as follow:



3. TaxServiceException: This exception is triggered when the property tax map already contains the same value.This exception is tested in TaxServiceTest class, in the method setTax(Map<String, Double>tax). If the Value of the map is already there then this will handle the system to crash.



The AOP Aspects you have created along with the different types of advice you used. Explicitly state in which class/method the different types of advice are found. Explain the point cuts that you have chosen – explain when they would be triggered.

- **@Before** advice has been created in LoggingAspects.java class and used in OrderProcessorServiceImpl class for newOrder() method. The pointcut is newOrder() method where the advice logBeforeMethods() will be applied before execution of pointcut newOrder().Advice logBeforeMethods() will print details regarding pointcut such as signature and method arguments.

- **@After** advice has been created in LoggingAspects.java class and used in OrderProcessorServiceImpl.java class for newOrder() method. The pointcut is newOrder() method where the advice logAfterMethods will be applied before execution of pointcut newOrder(). Advice logAfterMethods will print details regarding pointcut such as signature and method arguments.

- **@Around** advice has been created in LoggingAspects.java class used in AccountingServiceImpl.java class for computeTax() method. This pointcut would be triggered before to print the heading and after computeTax() is computed, the pointcut would be triggered to show that the tax is calculated successfully and will print the calucated tax that will be returned by computeTax() method.

- **@AfterReturning** advice has been created in AspectExample.java class and used in AccountingServiceIntlRules.java class for computetax () method. The advice would be triggered only if the tax has been calculated successfully without exception. Advice logAfterReturning will display the signature of the joinpoint and also will display the value returned by the joinpoint i.e the tax computed by pointcut computetax () method

- **@AfterThrowing** advice has been created in AspectExample.java class and used in OrderProcessor.java class for OrderitemException() method. The advice would be triggered only if the pointcut OrderitemException() throws ArithmeticException to show that the exception has been thrown and would display the exception.

TheScreenShots are as follows: