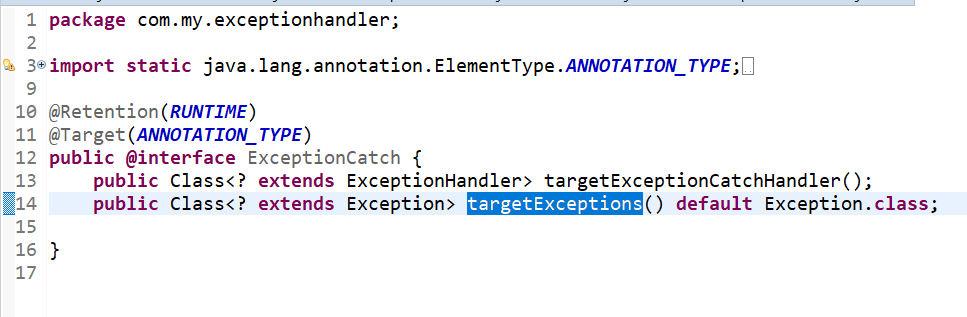
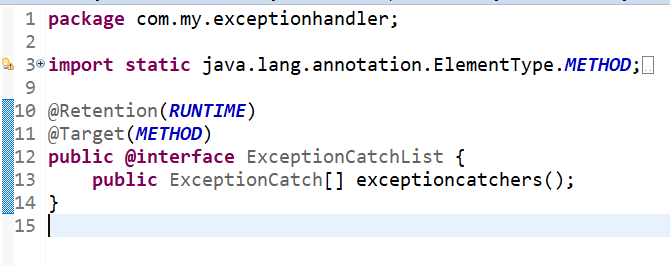
The Component “Y”

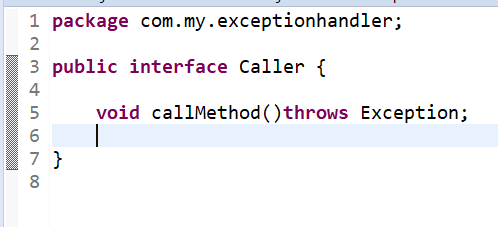
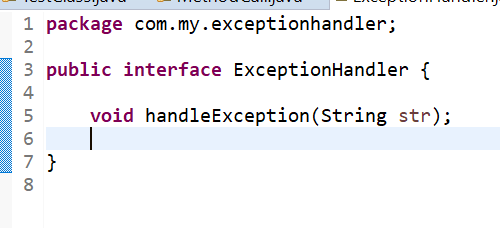
For some particular issues like exception handling where we need to save the logs we have previously seen that we need to make a new customized exception class and then we need to throw that customized exception class in each and every method.

But that’s where annotation comes for the rescue. We can make annotation and manipulate the object. Below is the custom utility example where when exception occurs it will save the exception details in the table.





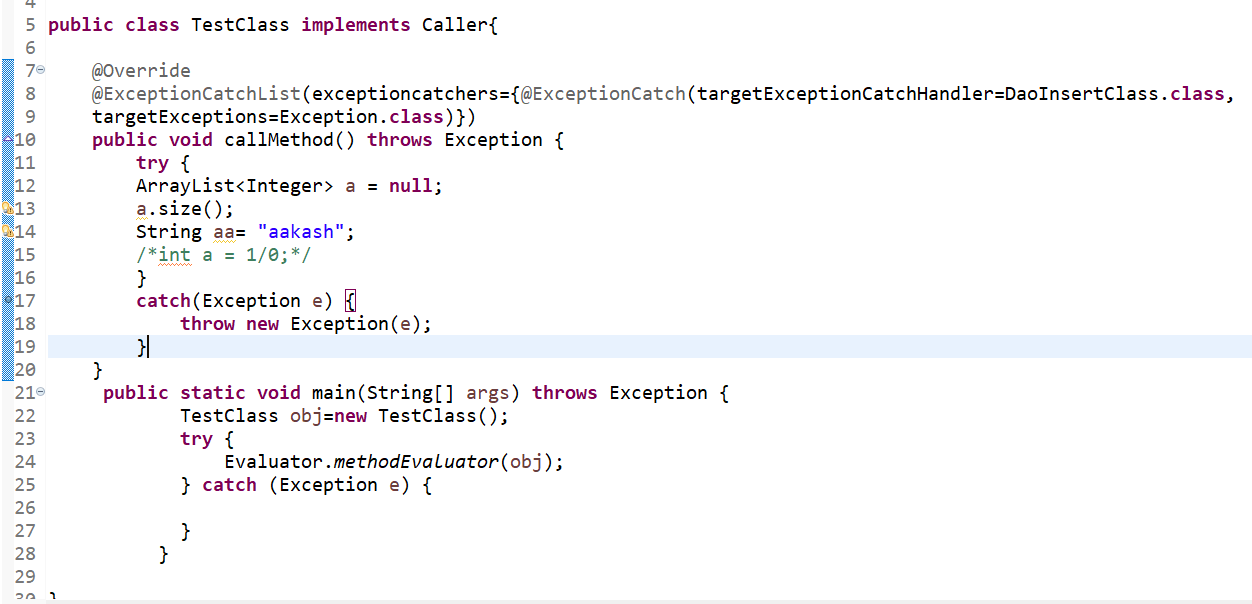
We are ready with the two annotations @ExceptionCatch which will take two parameters as targetExceptionCatchHandler variable(this must be any class which extends Exception.java) and targetExceptions variable(This can be any Exceptions which extends Exception.class, in our case it would be ACMException). ExceptionCatch annotation has target as annoatation\_type which means it can be declared around an annotation. Whereas ExceptionCatchList has target as method which means it can be declared around any method. Both have the retention value as RUNTIME which means it will be available to JVM through runtime of it.



We have 2 interfaces which have one method in it. Caller interface have callMethod() which will be used to call the methods inside it for the exception handling. ExceptionHandler interface has handleException method which will take String as an input. This method’s implementation will have the logic to save the logs for the exception.



The Evaluator.java has one static method which gets the method, annotations and parameters which are present above our Test class methods using JAVA reflections. This method first collects the data which are passed on the annotation on the method in form of how we defined the annotations. After fetching the data it will invoke the given method. If that methods throws the exceptions then it will catch and then the exception logs are passed to the handleExceptions class if the exceptions is of the type which we have passed in the annotation.

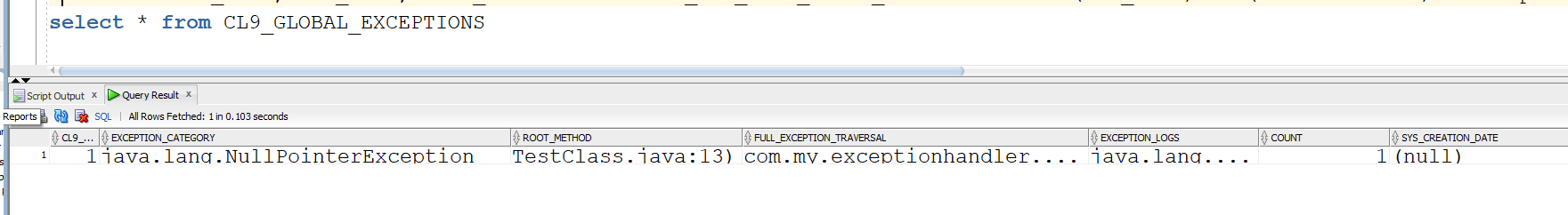


The TestClass implements the Caller interface and overrides the callMethod(). This method is annotated with @ExceptionCatchList which was our custom annotation which have @ExceptionCatch and have two parameters as targetExceptionHandler(It is the class name this exception has to be handled at the end) targetExceptions which will tell which all probable exceptions that has to be handled in the end.

In the main method we have called the static method (methodEvaluator) and passing the TestClass object which implements Caller interface as now it will be the instance of Caller.



The DaoInsertClass is the class which implements the ExceptionHandler and overriding the handleException which will take string as a parameter and then it will save the logs in the CL9\_GLOBAL\_EXCEPTIONS table by giving all the important information of the exception.



As seen we can identify the exception category as null pointer, the method name and the line number at which it came, the package address in full exception traversal, whole logs and sys\_creation\_Date when the error occurred.

**Advantages** :-

* We can get the time to analyze the logs and identify some exceptions before it gets logged into the system.
* Less dependcy on log files which are the most important part in finding the issue. The same thing will be servered by the table implemented. Moreover it will be much easy to get to the root cause of the problem.
* The exception could be recognised much faster.
* The pattern of the similar exception can be identified.
* The root cause of the exception can be identified without searching into the log files which would be easier than finding in the log files.

**Challenges** :-

* Tables will be created which will have large amount of data. For each exception we have to insert the data. Further it can be saved in optimized way like by if the same error is present then we would not save the data but we will save the number of time that happened so that we can get the idea out of number of files how many went for the particular same error.
* Tables need to be cleared or deleted on some configurable time. To implement that thing one script needs to be added which can run at EOD and deletes the table or clears the older table.
* The method which needs to be tested for exceptions should be implementing the Caller interface and also it must be overriding the method. This method should have the call for the method which needs to be tested. Thus it will have some code changes to be done.