Kevin Kulda

Dr. Cerny

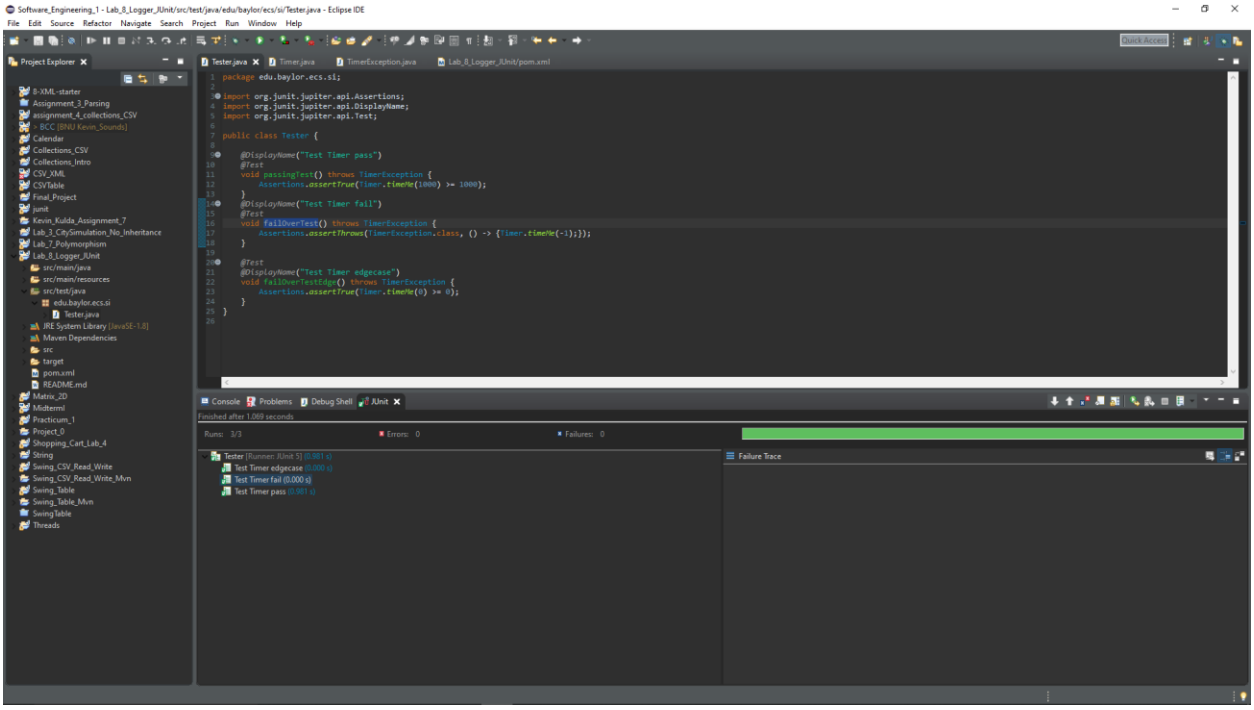
30-October-2019

Tasks to answer in your own README.md that you submit on Canvas:

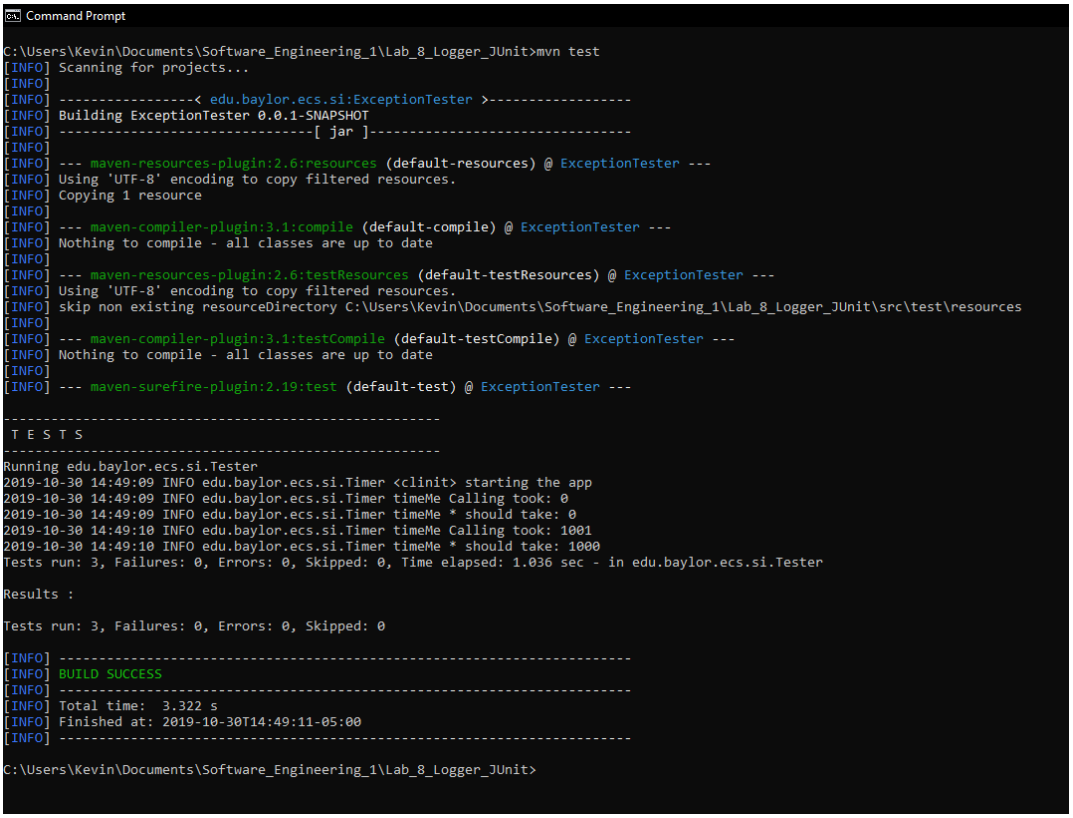
1. See logger.log, why is it different from the log to console?
    1. Logger.log is a file and will save the logging information after the program exits.
    2. Logger .log is set to receive ALL logging levels of repors.
    3. Log to the console is set to only to receive INFO level logs.
2. Where does this line come from? FINER org.junit.jupiter.engine.execution.ConditionEvaluator logResult Evaluation of condition [org.junit.jupiter.engine.extension.DisabledCondition] resulted in: ConditionEvaluationResult [enabled = true, reason = '@Disabled is not present']
    1. This line comes from the logger.log file after running the Test as a Junit Test.
    2. This statement is generated when a test is run on a method that is not explicitly enabled to be tested. By default it is enabled, and because the method in this case does not have the @Disabled annotation it is considered enabled and is run.
3. What does Assertions.assertThrows do?
    1. It asserts that execution of the supplied executable throws an exception of the expected type and returns the exception. If no exception is thrown, or if an exception of a different type is thrown, this method will fail.
4. See TimerException and there are 3 questions
    1. What is serialVersionUID and why do we need it? (please read on Internet)
        - Simply put, the serialVersionUID is a unique identifier for Serializable classes. This is used during the deserialization of an object, to ensure that a loaded class is compatible with the serialized object. If no matching class is found, an InvalidClassException is thrown.
    2. Why do we need to override constructors?
        - We must override constructors because they are not inherited in subclasses. Note, constructors have the same name as the class they are in so if they were inherited the class would have a constructor with the name of its parent class.
    3. Why we did not override other Exception methods?
        - If a method declares to throw a given exception, the overriding method in a subclass can only declare to throw that exception or its subclass. Therefore, there is no point to override an Exception method because you cannot and should not try to change the behavior of the superclass methods with regards to throwing appropriate types of Exceptions.
5. The Timer.java has a static block static {}, what does it do? (determine when called by debugger)
    1. A static block in Java is a block of code that is executed at the time of loading a class for use in a Java application. It is used for initializing static Class members in general, and is also known as a ‘Static Initializer’. The most powerful use of a

static block can be realized while performing operations that are required to be executed only once for a Class in an application lifecycle.
    2. In Timer.java the static block creates and initializes the logger config file, loads the config file data, and then closes the file so the logger is all set up for the program.
    3. It is called by debugger at the beginning of running the program.
6. What is README.md file format how is it related to bitbucket? (https://confluence.atlassian.com/bitbucketserver/markdown-syntax-guide-776639995.html)
    1. The README.md file format is “markdown”. It is related to bitbucket because the bitbucket server uses markdown for formatting text. Markdown generally uses a set of inline notation to format the text.
7. Why is the test failing? what do we need to change in Timer? (fix that all tests pass and describe the issue)
    1. The test is failing because the TimerException is never being thrown from the timeMe method. In Timer we need to put the if statement that throws the TimerException outside of the try catch block so the exception is thrown from the method and not caught in the try catch block.
8. What is the actual issue here, what is the sequence of Exceptions and handlers (debug)
    1. The actual issue is that the test expects a TimerException to be thrown and it is not. In the timeMe method the if statement that correctly should throw the TimerException is placed within a try-catch block. In the try-catch block the TimerException is contained even though it is not caught and rethrown. Therefore, the TimerException is never thrown from the method or seen by the test method. Instead the exception is contained within the try-catch block and the flow of execution improperly finishes in the method. Because this exception never is thrown from the method, the assert statement fails.

9. Make a printScreen of your eclipse JUnit5 plugin run (JUnit window at the bottom panel)



10. Make a printScreen of your eclipse Maven test run, with console



11. What category of Exceptions is TimerException and what is NullPointerException
    1. TimerException is an example of a User-Defined Run-Time Exception.
    2. A NullPointerException is a RuntimeException. In java, this exception is thrown when an application attempts to use an object reference that has the null value.
12. Push the updated/fixed source code to your own repository.
    1. Done 😊