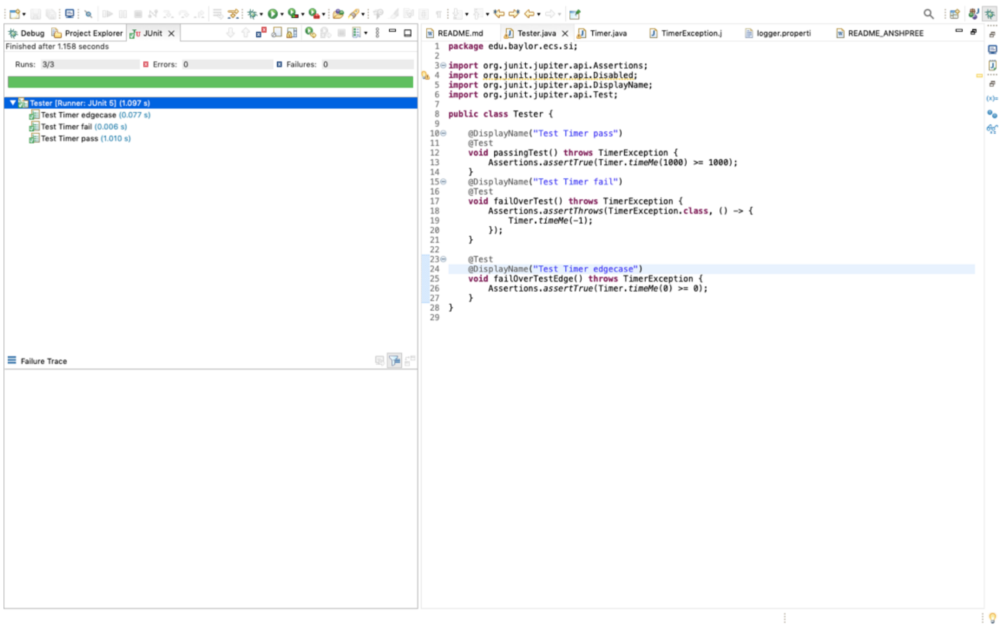
1. See logger.log, why is it different from the log to console?
Answer: Logger.log is different from log to console because if we see in logger.properties (in folder src/main/resources) we see that the level of file logging is defined as "All" while the level of logging on sonsole is defined as only the info level.(ie it is only confined to info level)
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']
Answer: This line is present in the logger.log file which is of the level "Finer".
3. What does Assertions.assertThrows do?
Answer: assertThrows assert makes sure that an exception of type TimerException.class is thrown.
4. See TimerException and there are 3 questions
   1. What is serialVersionUID and why do we need it? (please read on Internet)
   Answer: serialVersionUID is used during deserialisation to verify that the sender and receiver of a serialised object have loaded classes for that object that are compatible with respect to serialisation. If the receiver has loaded a class for the object that has a different serialVersionUID than that of the corresponding sender's class, then deserialisation will result in an InvalidClassException.
   2. Why do we need to override constructors?
   Answer: We need to override the constructors because we have to call the super(parameter_list) inside the constructors of the TimerException class which is basically the constructors of the Exception class and that way we can pass the parameters to the Exception class constructor.
   3. Why we did not override other Exception methods?
   Answer:We did not override other Exception constructors because in the Timer class we only make use of two constructors new TimerException("Sleep exception", e) and newTimerException("Cannot be less than zero”). We did not override other Exception methods because they can be simply called using the object of TimeException class, while parent class constructors cannot be called using an object we need to invoke using the keyword super in the child class.

5. The Timer.java has a static block static {}, what does it do? (determine when called by debugger)
Answer: It is in static block because it is called once , when the class itself is initialised, no matter how many objects of that type you create. This is because it it is an option for initialising or setting up the class at run-time
6. What is README.md file format how is it related to bitbucket? (https://confluence.atlassian.com/bitbucketserver/markdown-syntax-guide-776639995.html)
Answer: If your repository contains a README.md file at the root level, Bitbucket displays its contents on the repository’s Source page if the file has the .md extension.
7. Why is the test failing? what do we need to change in Timer? (fix that all tests pass and describe the issue)
Answer: The test is failing because it excepted TimeException however it excepted NullPointerException because the variable timeNow was null so to make the test case pass we initialised the variable timeNow.

8. What is the actual issue here, what is the sequence of Exceptions and handlers (debug)
Answer: In the try block when it receives negative timeToWait it throws an exception and the rest of the sentences in the try block gets ignored and doesn't get executed and then it moves to the catch block where the exception thrown is not caught hence it finally moves to the finally block where it uses the timeNow variable which is null (because we did not initialise it) and hence it throws NullPointerException and thus due to exception chaining when we assert the thrown exception we receive the NullPointerException and not the TimeException.

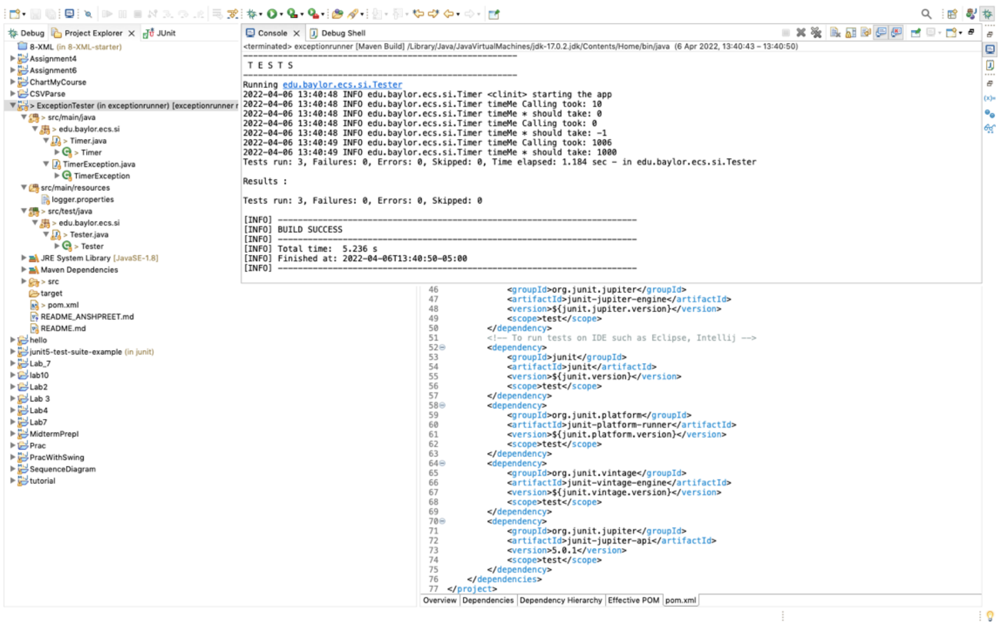
9. Make a printScreen of your eclipse JUnit5 plugin run (JUnit window at the bottom panel)



```
Finished after 1.158 seconds
Runs: 3/3    Errors: 0    Failures: 0

Tester [Runner: JUnit 5] (1.097 s)
    Test Timer edgecase (0.077 s)
    Test Timer fail (0.006 s)
    Test Timer pass (1.010 s)
```

```java
package edu.baylor.ecs.si;

import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

public class Tester {

    @DisplayName("Test Timer pass")
    @Test
    void passingTest() throws TimerException {
        Assertions.assertTrue(Timer.timeMe(1000) >= 1000);
    }
    @DisplayName("Test Timer fail")
    @Test
    void failOverTest() throws TimerException {
        Assertions.assertThrows(TimerException.class, () -> {
            Timer.timeMe(-1);
        });
    }

    @Test
    @DisplayName("Test Timer edgecase")
    void failOverTestEdge() throws TimerException {
        Assertions.assertTrue(Timer.timeMe(0) >= 0);
    }
}

```

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



```
<terminated> exceptionrunner [Maven Build] /Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java  (6 Apr 2022, 13:40:43 – 13:40:50)
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running edu.baylor.ecs.si.Tester
2022-04-06 13:40:48 INFO edu.baylor.ecs.si.Timer <clinit> starting the app
2022-04-06 13:40:48 INFO edu.baylor.ecs.si.Timer timeMe Calling took: 10
2022-04-06 13:40:48 INFO edu.baylor.ecs.si.Timer timeMe * should take: 0
2022-04-06 13:40:48 INFO edu.baylor.ecs.si.Timer timeMe Calling took: 0
2022-04-06 13:40:48 INFO edu.baylor.ecs.si.Timer timeMe * should take: -1
2022-04-06 13:40:49 INFO edu.baylor.ecs.si.Timer timeMe Calling took: 1006
2022-04-06 13:40:49 INFO edu.baylor.ecs.si.Timer timeMe * should take: 1000
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.184 sec - in edu.baylor.ecs.si.Tester

Results :

Tests run: 3, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.236 s
[INFO] Finished at: 2022-04-06T13:40:50-05:00
[INFO] ------------------------------------------------------------------------
```

```xml
                <groupId>org.junit.jupiter</groupId>
                <artifactId>junit-jupiter-engine</artifactId>
                <version>${junit.jupiter.version}</version>
                <scope>test</scope>
            </dependency>
            <!-- To run tests on IDE such as Eclipse, Intellij -->
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>${junit.version}</version>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>org.junit.platform</groupId>
                <artifactId>junit-platform-runner</artifactId>
                <version>${junit.platform.version}</version>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>org.junit.vintage</groupId>
                <artifactId>junit-vintage-engine</artifactId>
                <version>${junit.vintage.version}</version>
                <scope>test</scope>
            </dependency>
            <dependency>
                <groupId>org.junit.jupiter</groupId>
                <artifactId>junit-jupiter-api</artifactId>
                <version>5.0.1</version>
                <scope>test</scope>
            </dependency>
        </dependencies>
</project>
```

11. What category of Exceptions is TimerException and what is NullPointerException
Answer: TimerException extends Exception class, while the NullPointerException extends RuntimeException.

12. Push the updated/fixed source code to your own repository.

https://github.com/anshpreetkaur1/Assignment9.git