TDD FileLoader v1.0.

In this solution, the production class actually loads a file from the disk

The Unit Test

```
package com.celestial.mockito.filetodb;

import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.Test;


/**
 *
 * @author selvy
 */
public class LiveFileLoaderTest
{
    // The inital design is described in this test
    /*
     The weakness should be obvious?  The file to be loaded and it's
location
     I use a shared network drive to run this code from different
machines, normally
     dev-ing from a PC.  When I ran the code on the laptop from a cafe
it
     immediately failed because the C: on the laptop was completely
different
     to the PC, so the original file C:/tmp/KeyboardHandler.txt did not
exist

     THIS IS A GREAT EXAMPLE OF WHY THE UNIT TEST AND CUT SHOULD NOT BE
STRONGLY
     LINKED TO ANY IO - NETWORK, DB, AND FILE SYSTEM
    */
     @Test
     public void load_all_of_file_using_inbuilt_Files_type()
     {
         // arrange
         String fileToLoad = "c:/tmp/KeyboardHandler.java.txt";
         FileLoader cut = new FileLoader(fileToLoad);
         int expectedBytesRead = 1383;

         // act
         int bytesRead = cut.loadFile(fileToLoad);

         // assert
         assertEquals(expectedBytesRead, bytesRead);
     }
}
```

The CUT FileLoader

```
package com.celestial.mockito.filetodb;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.*;


/**
 *
 * @author selvy
 */
public class FileLoader
{
   private  class IntWrapper
   {
      public   int value;
   }

   String fileToLoad;
   List<String> lines = Collections.emptyList();

   public FileLoader(String fileToLoad)
   {
       this.fileToLoad = fileToLoad;
   }

   int loadFile(String fname)
   {
       try
       {
           lines = Files.readAllLines(Paths.get(fname),
StandardCharsets.UTF_8);
       }
       catch (IOException e){}

       return calculateFileSize();
   }

   private int calculateFileSize()
   {
       IntWrapper result = new IntWrapper();

       lines.forEach(line -> {
               result.value += line.length();
       });
```

```
        return result.value;
    }
}
```