TDD FileLoader v1.2.

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

The Unit Test

```
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.ArrayList;
import java.util.List;
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.Test;
import org.mockito.MockedStatic;
import org.mockito.Mockito;

/**
 *
 * @author selvy
 */
public class FileLoaderTest
{
   // To use a different type of file system loader, pass a lambda to
loadFile()
   // as shown here
   /*
       int bytesRead = cut.loadFile((fname) ->
       {
          List<String> result = null;
          try
          {
              result = Files.readAllLines(Paths.get(fname),
StandardCharsets.UTF_8);
          }
          catch (IOException e){}
          return result;
       });
   */

    // Redesign the FileLoader so that the machenism to load files up
can be
    // passed in as a lambda - still titghtly coupled the file system
    @Test
    public void load_all_of_file_using_inbuilt_Files_type_as_lambda()
    {
        // arrange
```

```
        String fileToLoad = "c:/tmp/KeyboardHandler.txt";
        FileLoader cut = new FileLoader(fileToLoad);
        int expectedBytesRead = 10;    //1371;
        List<String> pretendFileContent = new ArrayList<>();
        pretendFileContent.add("Hello");
        pretendFileContent.add("world");
        MockedStatic<Files> ff = Mockito.mockStatic(Files.class);
        ff.when(() -> Files.readAllLines(Paths.get(fileToLoad),
StandardCharsets.UTF_8)).thenReturn(pretendFileContent);

        // act
        int bytesRead = cut.loadFile((fname) ->
        {
           List<String> result = null;
           try
           {
               result = Files.readAllLines(Paths.get(fname),
StandardCharsets.UTF_8);
           }
           catch (IOException e){}
           return result;
        });

        // 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
{
    class   IntWrapper
    {
```

```
        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();
    }

    public List<String> getLines() {
        return lines;
    }

    int loadFile(ILoader func)
    {
        lines = func.loadFile(fileToLoad);
        return calculateFileSize();
    }

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

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

        return result.value;
    }
}
```