```
/* Ahmed Cheema / Havin Lim
Lab: 8 - Exceptions and File Handling
September 18th 2022
Sources : John David Stone's FileLister.java
Help obtained : None
We confirm that the above list of sources is complete AND that we have not
talked to anyone else (e.g., CSC207 students) about the solution to this
problem
*/

import java.io.*;

public class fileHandling {
      // Variables to count errors
      public static int emptyError = 0;
      public static int extraError = 0;

      // Function to read numbers into array and handle exceptions
      private static double[] readArray(BufferedReader source) throws
IOException {
            double[] arr = new double[12];
            try {
                  String line;
                  for (int i = 0; i < arr.length; i++) {
                        if ((line = source.readLine()) != null) {
                              arr[i] = Double.parseDouble(line);
                        }
                        else {
                              emptyError++;
                        }
                  }
            }
            catch (NumberFormatException e) {
                  System.err.println("Error: extra data on the line");
                  extraError++;
            }
            finally {
                  return arr;
            }
      }

      // Function to sum values in array
```

```
	public static double sum(double[] arr) {
		double sumNum = 0;
		for (int i = 0; i < arr.length; i++) {
			sumNum += arr[i];
		}
		return sumNum;
	}

	// Function to compute average of values in array
	public static double average(double[] arr) {
		double sumNum = sum(arr);
		return sumNum/arr.length;
	}

	// Function to read file, get the array of numbers,
	// print the values to console along with the sum
	// and average. Try/catch blocks from John David
	// Stone's FileLister.java file
	public static void openFile(String fileName) {
		BufferedReader src = null;
		try {
			src = new BufferedReader(new FileReader(fileName));
			double[] arr = readArray(src);
			if (emptyError > 0) {
				System.err.println("Error: The file does not contain
numbers.");
				throw new IOException();
			} else if (extraError > 0) {
				return;
			}
			for (int i = 0; i < arr.length; i++) {
				System.out.format("Number %d: %f%n",i+1,arr[i]);
			}
			System.out.format("%nThe sum of the array is: %f",
sum(arr));
			System.out.format("%nThe average of the array is: %f",
average(arr));
		} catch (FileNotFoundException e) {
			System.err.println("The file " + fileName +
			  " does not exist or " +
			  "cannot be opened for reading.");
		} catch (IOException e) {
```

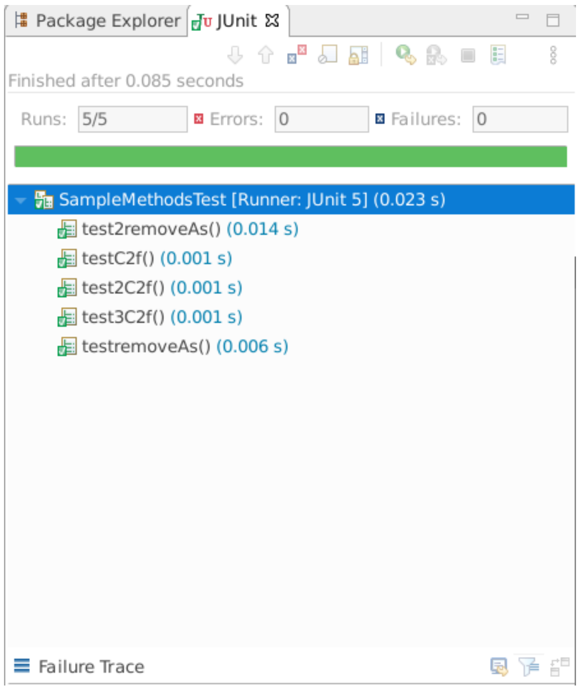
```
                  System.err.println("An error occurred during an attempt " +
                          "to read the file " + fileName + ".");
            } finally {
                  try {
                        if (src != null) {
                              src.close();
                        }
                  } catch (IOException e) {
                        System.err.println("An error occurred during an
attempt " +
                        "to close the file " + fileName + ".");
                  }
            }
      }

      // Client program
    public static void main(String[] args) {
        for (String fileName: args)
            openFile(fileName);
    }
}
```

6. If there is a text at any point of the file none of the numbers would be read into the program.
Since the maximum length of the array is fixed, numbers that are written after the maximum length would be ignored. Additionally, if the file has a shorter length compared to the maximum length, the rest of the arrays would be filled with 0(zero)s.

/* Ahmed Cheema / Havin Lim
Lab : 9 - Unit testing and Debugging with Eclipse
September 18 th 2022
Sources : None
Help obtained : None
We confirm that the above list of sources is complete AND that we have not talked to anyone else (e.g., CSC207 students) about the solution to this problem
***/**

SampelMethodsTest.java

```
package sampletesting;

import static org.junit.jupiter.api.Assertions.*;

class SampleMethodsTest extends SampleMethods {

      @Test
      void testC2f() {
            assertEquals(SampleMethods.c2f(0),32);
      }

      @Test
      void test2C2f() {
            assertEquals(SampleMethods.c2f(100),212);
      }
      @Test
      void test3C2f() {
            assertEquals(SampleMethods.c2f(50),122);
      }
      @Test
      void testremoveAs() {
            assertEquals(SampleMethods.removeAs("add"), "dd");
      }
      @Test
      void test2removeAs() {
            assertEquals(SampleMethods.removeAs("The cat leaped over the
hat!"), "The ct leped over the ht!");
      }

}
```



```
Finished after 0.085 seconds
Runs: 5/5   Errors: 0   Failures: 0

SampleMethodsTest [Runner: JUnit 5] (0.023 s)
    test2removeAs() (0.014 s)
    testC2f() (0.001 s)
    test2C2f() (0.001 s)
    test3C2f() (0.001 s)
    testremoveAs() (0.006 s)
```

```
/* Havin Lim
Lab : 10 - Inheritance
September 18th 2022
Sources : None
Help obtained : None
I confirm that the above list of sources is complete AND that I have not
talked to anyone else (e.g., CSC207 students) about the solution to this
problem
*/

// Counter.java
package tallying;

import java.nio.BufferUnderflowException;

public class Counter
{
    private int counter;
    private String name;

    Counter(String addName)
    {
        counter = 0;
        name = addName;
    }

    public int counterGet()
    {
        return counter;
    }

    public void counterIncrement()
    {
        counter += 1;
    }

    public void counterReset()
    {
        counter = 0;
    }

    public String toString()
    {
        return name+": "+counter;
    }



    class CounterWithMemory extends Counter {

            CounterWithMemory(String addName) {
                  super(addName);
            }

            public int tally = 0;

      public void storeTally() {
            System.out.format("%d%n", tally);
            tally++;
      }

      public int recoverTally() {
            if(tally != 0) {
            return tally;

            } else {
                  throw new BufferUnderflowException();
            }
      }


      public void reset() {
            tally = 0;
      }
    }
}
```