Packages, Wrapper Classes, Streams, and File Handling in Java

Objective

Learn how to use packages, wrapper classes, file I/O, and streams in Java by building a program that reads, processes, and writes data.

Requirements

1. Create a package called `utilities` containing utility classes for file handling and string processing.
2. Write a program to read a text file, count the number of words and lines, and write the result to another file.
3. Use wrapper classes to handle conversions (e.g., `Integer.parseInt()`) when processing data from files.
4. Implement error handling for file operations (e.g., file not found, read/write errors).
5. Add functionality to compute word frequency in the file and write the results to a file.
6. Use Java streams to filter and display lines containing a specific keyword.
7. Create a test program in a different package to demonstrate the functionalities of the `utilities` package.

Exercises

Packages

1. Create an additional package named `mathutilities` containing a class that provides basic arithmetic operations. Demonstrate usage from another package.

Listing 1: mathutilities/MathHelper.java

```
package mathutilities;

public class MathHelper {
    public static int add(int a, int b) {
        // TODO: Implement addition
    }
    public static int subtract(int a, int b) {
        // TODO: Implement subtraction
    }
}
```

Wrapper Classes

1. Implement a method that reads a list of numbers from a file, converts them using wrapper classes, and calculates their sum.

Listing 2: utilities/NumberHelper.java

```
package utilities;

import java.io.*;
import java.util.*;

public class NumberHelper {
    public static int sumNumbersFromFile(String filePath) throws IOException {
        // TODO: Implement sum calculation using wrapper classes
    }
}
```

File Handling

1. Modify the `FileHelper` class to check if a file exists before reading or writing.
2. Implement a method that appends text to an existing file without overwriting its contents.

Listing 3: utilities/FileHelper.java

```
package utilities;

import java.io.*;
import java.util.*;

public class FileHelper {
    public static boolean fileExists(String filePath) {
        // TODO: Implement file existence check
    }
    public static void appendToFile(String filePath, String content) throws IOExcep
        // TODO: Implement file appending
    }
}
```

Streams

1. Use Java Streams to sort the lines of a file in alphabetical order before writing them to a new file.
2. Implement a stream-based solution to find the longest word in a given file.

Listing 4: utilities/StreamHelper.java

```
package utilities;

import java.io.*;
```

```
import java.util.*;
import java.util.stream.*;

public class StreamHelper {
    public static List<String> sortFileLines(String filePath) throws IOException {
        // TODO: Implement sorting lines with Streams
    }
    public static String findLongestWord(String filePath) throws IOException {
        // TODO: Implement finding longest word with Streams
    }
}
```

Hint:

Listing 5: utilities/FileHelper.java

```
package utilities;

import java.io.*;
import java.util.*;
import java.util.stream.*;

public class FileHelper {
    public static List<String> readFile(String filePath)
                    throws IOException {
        // TODO: Implement reading a file line by line
    }

    public static void writeFile(String filePath,
                    List<String> lines)
                    throws IOException {
        // TODO: Implement writing lines to a file
    }

    public static Map<String, Integer> countWordFrequency(String filePath)
                    throws IOException {
        // TODO: Implement word frequency counting
    }

    public static List<String> filterLines(String filePath, String keyword)
                    throws IOException {
        // TODO: Implement filtering lines containing a keyword
    }
}
```

Listing 6: utilities/StringHelper.java

```
package utilities;
```

```java
public class StringHelper {
    public static int countWords(String line) {
        // TODO: Count and return the number of words in a line
    }
}
```

Listing 7: MainApp.java

```java
import utilities.*;
import java.io.*;
import java.util.*;

public class MainApp {
    public static void main(String[] args) {
        // TODO: Read a file, count words and lines, and write results
    }
}
```

Listing 8: test/TestApp.java

```java
package test;

import utilities.*;
import java.io.*;
import java.util.*;

public class TestApp {
    public static void main(String[] args) {
        // TODO: Demonstrate functionalities of utilities package
    }
}
```

Submission

Submit the code files and a report explaining your implementation.