CS5001 Object-Oriented Modelling, Design and Programming
Practical 1 – Text Alignment

School of Computer Science
University of St Andrews

Due Friday week 4, weighting 20%
MMS is the definitive source for deadlines and weightings.

Now that you are familiar with your programming environment from having completed the introductory exercises, you are going to write a small program in your first practical that reads in a number of paragraphs of text from a file and aligns or justifies the text to a specified margin with the line wrapping at a specified line length. The program should output the aligned text to the terminal.

For this practical, you may develop your code in any IDE or editor of your choice. However, you must ensure that your source code is in a folder named `CS5001-p1/src` and your main method is in a file called `TextAlignment.java`.

Requirements

Your program should have the following features.

- Accept three arguments from the command line.
  - The first argument is the name (and path) of the file containing the text
  - The second argument is the text alignment type. This can be either left, right, centre or justify.
  - The third argument is the line length which should be greater than zero.
  - If any of the argument is missing or invalid, you should output the following message:

```
usage: java TextAlignment <filename> <alignmentType> <lineLength>
```

  - If the specified file does not exist in the given location, you should output:

```
File not found:
```

    followed by a space and the name of the file. See the sample runs below for exact formatting.
- Read the file specified in the first argument.

  - In order to read in a file "test.txt" in your Java program to an array of Strings, where each String in the array represents a paragraph of text in the file, you can use the following line in your program.

```
String[] paragraphs = FileUtil.readFile("test.txt");
```

  - Or you can see the section below on using Scanner.

- Go through the String array or other data structure you have and align or justify each paragraph of text in the specified alignment type with the specified line length.
  - The alignment type can be left, right, centre or justify.
  - The line length is the maximum number of characters in a line. This can be exceeded if a word is longer than the line length for the left, right and centre alignment types. However, if the alignment type is justify, then the line length should not be exceeded.
  - If the alignment type is justify, then your program should also make sure that words are split and hyphenated at the end of the line. The length of the line, including the hyphen, should not exceed the line length.
- Print the results to the terminal.
  - The formatted text should be printed verbatim to standard output with no additional characters.
- The automated checker will run your program with a number of different files, alignment types, and line lengths. You should make sure that your program works correctly with all of options.
  - If there are any ambiguities in the specification, the automated checker results are the definitive source of what is required.

Hints

So far, we have not covered reading text from a file, so we have provided you with a FileUtil.java class to deal with reading from a file which is located on StudRes. Alternatively, you can look up other ways of reading files yourself. You can see that you might find the `Scanner` object very useful when reading through the text. You can create a `Scanner` that reads through a text file using the following code:

```
File textFile = new File(filename);
Scanner textScanner = new Scanner(textFile);
```

Then you can use the `Scanner` object's methods to get the strings you need from the file. You can find information on this class at its page in the Java API, and you might pay particular attention to the `hasNext`, `next`, and `useDelimiter` methods.

You may find it useful to look up other features in the Java API, including Strings, different subclasses of Exception, and regular expressions.

We have provided you with four pieces of example text which you can use to test your program. The files are available at:

https://studres.cs.st-andrews.ac.uk/CS5001/Coursework/p1/Tests/

The files java.txt and java_short.txt are generated by ChatGPT.

When running your program on files in different directories remember to use the absolute or relative path to the file. For instance, if you are running your program from CS5001-p1/src but your text files are in CS5001-p1, you could use the following command, specifying the relative path to `some_text.txt`, to align the text in `some_text.txt`:

```
java TextAlignment "../some_text.txt" left 80
```

Note: you must include the quote marks around the file name if there are spaces in the file name/path.

The expected output of a few examples is shown in Appendix A – Sample Runs. You can find further examples of output by examining the tests provided (see Automated Checking below).

For this practical you may find the documentation of arrays and strings particularly useful in showing you how to find out the length of an array and of a string, how to extract substrings from a string, how to find the occurrence of certain characters in a string, split a string, remove leading and trailing white space, etc.

You should start breaking down the problem in an object-oriented style. You should have classes that perform the alignment and file reading separately and also separate from the main method. You could also consider using different classes for the different alignment types. Try to only use static methods and fields when it is necessary. Practise encapsulation.

Automated Checking

You are provided with some basic unit tests to check your code provides the required functionality. The tests can be found at `/cs/studres/CS5001/Coursework/p1/Tests`.

In order to run the automated checker on your program before saving it to an archive, log into one of the School computers running Linux, then **change directory** to your `CS5001-p1` directory and execute the following command:

```
stacscheck /cs/studres/CS5001/Coursework/p1/Tests
```

If the automated checker doesn't run, or the build fails, or all tests fail, you may have mis-typed a command or not have followed the instructions above.

You should open the provided test files and make sure you understand what the tests are doing and try to come up with some interesting tests of your own. You can run your own via the command line, or via the automated checker. You can even create new tests in a local sub-directory in your assignment directory and run stacscheck with your own directory as the argument, for example:

```
stacscheck /cs/home/<yourusername>/CS5001-p1/MyTests
```

When you are ready to submit, make sure that you also run the checker on the archive you are preparing to submit, by calling, for example:

```
stacscheck --archive CS5001-p1.zip /cs/studres/CS5001/Coursework/p1/Tests
```

You can see the documentation for the automated checker at:

https://studres.cs.st-andrews.ac.uk/Library/stacscheck/

Deliverables – Software (and README)

If you have attempted any enhancements beyond those mentioned in the handout, you should include a short `README` file in your assignment folder (`CS5001-p1`). You should describe what you have done, including any instructions for compiling, running and using your program. Hand in a `.zip` archive of your entire assignment folder, which includes your src directory, the `README` file (if applicable), and any local test sub-directories, via MMS.

Marking

Grades will be awarded according to the General Mark Descriptors in the Feedback section of the School Student Handbook. The following table shows some example descriptions for projects that would fall into each band.

Grade	Examples
1–3	Little or no working code, and little or no understanding shown.
4–6	Acceptable code for part of the problem, but with serious issues such as not compiling or running. Some understanding of the issues shown.
7	A running program that reads text from a file and attempts alignment, but with serious problems such as incorrect text processing, incorrectly formatted output, and crashes. Poor code quality.
8–10	Reads from a file and aligns text with one of the alignment types, achieving the correct line length and answer most of the time. Possibly includes bugs or misformatted output. Code quality is legible but with major room for improvement, perhaps all being done in a single method.
11–13	Two or more alignment types implemented correctly, with correct line length for each type, but with problems such as poor error-handling or invalid output formatting. Code quality is reasonable, with correct indentation and some method decomposition.
14–16	Almost all required functionality implemented correctly, with only minor bugs or missing features. Code is clear and well-structured, with good method and class decomposition, Java code conventions followed, and some use of Javadoc.
17–18	All required functionality implemented correctly, with appropriate error-handling and robustness. High-quality code that is easy to understand, decomposed into meaningful clases and short methods with very little code duplication, with high-quality comments and Javadoc.
19–20	Outstanding implementations of all features, and possibly some extra features, with exceptional clarity of design, no code duplication, concise but full documentation, and an appropriate object-oriented structure.

Submissions will be marked according to the School's General Mark Descriptors, which can be found in the student handbook at

http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

Lateness

The standard penalty for late submission applies (Scheme A: 1 mark per 24-hour period, or part thereof):

http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Good Academic Practice

The University policy on Good Academic Practice applies:

https://www.st-andrews.ac.uk/students/rules/academicpractice/

We will run automated plagiarism checks on all submissions, so please don't be tempted to share your code with another student or use code you find online.

Appendix A – Sample Runs

Four sample files are provided on StudRes.... You can use these files to test your work. Some sample desired outputs are shown below, and you can see more by looking at the test files.

The examples below assume that you are working in the `src` directory, and that the text files are located in the parent directory. The `$` symbol represents the prompt in the terminal, but your terminal prompt might have some more information in it.

```
$ javac *.java
$ java TextAlignment /cs/studres/CS5001/Coursework/p1/Tests/java.txt
usage: java TextAlignment <filename> <alignmentType> <lineLength>
$ java TextAlignment /cs/studres/CS5001/Coursework/p1/Tests/nonexistent.txt left 80
File not found: /cs/studres/CS5001/Coursework/p1/Tests/nonexistent.txt (No such file or dire
$ java TextAlignment /cs/studres/CS5001/Coursework/p1/Tests/java.txt left 80
At St. Andrews University, nestled amidst the historic charm of Scotland, one
student's academic journey has taken an exciting turn as they delve into the
world of programming. With a thirst for knowledge and a passion for
problem-solving, this diligent student has embarked on a compelling quest to
master the intricacies of Java, a versatile and widely-used programming
language. The picturesque campus, with its centuries-old architecture, serves as
an inspiring backdrop to their coding endeavors. Through lectures, hands-on
labs, and collaborative projects, this student is not only honing their coding
skills but also learning to think critically, analyze algorithms, and create
software solutions that have real-world applications. So..... With each line of
code they write, they inch closer to becoming a proficient Java developer,
equipped to tackle the challenges of the digital age and contribute to the
ever-evolving tech landscape. Wow!!!! St. Andrews provides the perfect
environment for nurturing their intellectual curiosity, and their journey into
the world of Java is just the beginning of a promising academic and professional
```

```
adventure.
$ java TextAlignment /cs/studres/CS5001/Coursework/p1/Tests/java.txt justify 80
At St. Andrews University, nestled amidst the historic charm of Scotland, one s-
tudent's academic journey has taken an exciting turn as they delve into the wor-
ld of programming. With a thirst for knowledge and a passion for problem-solvin-
g, this diligent student has embarked on a compelling quest to master the intri-
cacies of Java, a versatile and widely-used programming language. The picturesq-
ue campus, with its centuries-old architecture, serves as an inspiring backdrop
to their coding endeavors. Through lectures, hands-on labs, and collaborative p-
rojects, this student is not only honing their coding skills but also learning
to think critically, analyze algorithms, and create software solutions that have
real-world applications. So..... With each line of code they write, they inch c-
loser to becoming a proficient Java developer, equipped to tackle the challenges
of the digital age and contribute to the ever-evolving tech landscape. Wow!!!!
St. Andrews provides the perfect environment for nurturing their intellectual c-
uriosity, and their journey into the world of Java is just the beginning of a p-
romising academic and professional adventure.
```