Assessment

Task 1 (25 Marks)

Create a directory called `task01` inside your repository. In `task01` directory, create 2 sub directories called `src` and `classes`; they are for storing your Java source code and compiled classes respectively.

Create a `.gitignore` file to ignore the `classes` directory.

You can use any Java package name for this task.

In this task, you will write a command line calculator. The following shows how the calculator behaves. The > is the prompt

```
java -cp classes your.package.Main

Welcome.
> 3 + 4
7
> 10 - 2
8
> 50 / 5
10
> 3 * 3
9
> exit
Bye bye
```

The calculator accepts a line of input in the following format

```
number space operator space number
```

where a `number` can be a positive or negative integer or decimal number.

Table 1 shows the list of operations that the calculator supports

Operator	Explanation
+	Add
-	Subtract
/	Divide
*	Multiply

Table 1 Calculator operations

The numbers and operators are separated by a single space.

Every result evaluated by the calculator is saved in a variable called `$last`. This can be used in subsequent calculations. New calculations will overwrite the value in `$last`.

The following shows how `$last` behaves.

```
java -cp classes your.package.Main

Welcome.
> $last + 3
3
> 4 * 8
32
> 512 - $last
480
> 7 x 6
42
> $last + $last
84
> exit
Bye bye
```

Write a Java program to read a line and evaluate the mathematical expression according to the above described behaviour. Assume that all numbers and mathematical operators entered are correct.

You can use any Java package name for this task.

Task 2 (45 marks)

Create a directory called `task02` inside your repository. In `task02` directory, create 2 sub directories called `src` and `classes`; they are for storing your Java source code and compiled classes respectively.

Create a `.gitignore` file to ignore the `classes` directory.

You can use any Java package name for this task.

In this task, you will be writing a Java program to analyse how famous authors write their books by determining, for a given word that the author uses, what are the possible next words that the author will use.

You will do this by analysing a corpus to text by the author and determining the next word distribution. For example, the next word distribution for the following text, by Charles Dickens from A Tale of Two Cities, is shown in Table 2.

```
It is a far, far better thing that I do, than I have
ever done; it is a far, far better rest that I go to
than I have ever known.
```

Word	Next Word	Count
it	is	2
is	a	2
a	far	2
far	far	2
	better	2

better	rest	1
	thing	1
thing	that	1

Table 2 Next word distribution of the first six words

From Table 2, we see that the word 'is' follows 'it' 2 times in the text; the **words 'far' and 'better'** follows the word 'far' 2 times.

Before analysing each word, you should

1. **Stripped all** punctuations from the words under consideration:
   - **full stop** (.), comma (,), colon (:)
   - **exclamation** (!), dash (-)
   - **brackets** (() {})
   - quotes either single or double (' ")
   - any other punctuations
2. Words in different cases, e.g. `ALL`, `all`, `All`, should be treated as the same word.

Write a Java program to analyse all the files from a given **directory** and print out the next word distribution.

Your Java program should run with 1 parameter which is a directory containing all the files to be analysed. The program should run as follows

```
java -d classes your.package.Main <directory_name>
```

where `your.package.Main` is the fully qualified class name of your main Java class and `<directory_name>` is the directory containing the files.

After processing the files under a directory, print all the words, their corresponding next words and the probability of the next word occurring. The following example is the printout from Table 2

```
it
    is 1
is
    a 1
far
    far 0.5
    better 0.5
better
    rest 1
    thing 1
thing
    that 1
```

The next word probability is calculated by the frequency of the next word divided by the count of all the next words.

For example the next words for 'far', are 'far' and 'better' each with a frequency of 2; the total next words, for the word 'far' is 4. The next word probability are for each word following 'far' is calculated as follows

```
far = 2 / 4 = 0.5
better = 2 / 4 = 0.5
```

You will be provided with a ZIP file containing a list of files to be analysed. Unzip the file in `task02` directory. The unzipped file should create a directory called `texts` which contains 2 other directories: `frost` and `seuss`. The latter 2 directories contain text files which you will be using for this task.

Hint: look at `File.listFiles()` in the JDK 20 documentation.

Submission

You must submit your assessment by pushing it to your repository at GitHub.

Only commits on or before **Friday 1700 May 12 2023** will be accepted. Any commits after **Friday 1700 May 12 2023** will not be accepted. No other form of submission will be accepted (eg. ZIP file).

Remember to **make your repository public after Friday 1700 May 12 2023** so the instructors can review your submission.

After committing your work, post the following information to Slack channel `#01-sdf-submission`

1. Your name (as it appears in your NRIC)
2. Your email
3. Git repository URL. Remember to make your repository **PUBLIC after Friday 1700 May 12 2023**

It is your responsibility to ensure that all the above submission requirements are met. Your assessment submission will not be accepted if

1. Any of the 3 items mentioned above is missing from `#01-sdf-submission` channel, and/or
2. Your information did not comply with the submission requirements eg. not providing your full name, and/or
3. Your repository is not publicly accessible after **Friday 1700 May 12 2023**

You should post the submission information to the Slack channel `#01-sdf-submission` no later than **Friday 1715 May 12 2023**