Term 4 Assignment 1

Quicka Self-Driving

You are the Lead Systems Engineer for the Quicka ridesharing platform and are in charge of developing all the necessary software for drivers, riders, and self-driving cars.

For this assignment, you are asked to write the Control Operation System (aka COS-101) software for the Quicka Autonomous Vehicle. The computers onboard the QAV are outfitted with a JVM capable of running Java code.

When designing and writing your systems, you may use any IDE of your choice, but you must ensure that your code is able to run on the JVM.

Assignment 4.1: Driving Logs

The self-driving transports have a internal reporting and logging system that records events and issues during rides. These logfiles contain the last 1000 events/messages as well as their status codes.

The Logfile Format

By default, the logfiles are all called 'logfile.txt'. Each logfile is exactly 1000 lines long.

Each entry in the file consists of 4 fields:

1. a line number (starting at 0)
2. a POSIX timestamp
3. a message/event type
4. an event description

Each field (line number, timestamp etc.) is separated by a space, except for the description which ends with a full stop.

Log events come in three different types: `NOTIFY`, `WARNING`, and `ERROR`.

Task 1: Display

Because the logfiles are so big, we would only like to display a certain number of lines at a time. We prompt the user to choose start and end line numbers and print all the lines from start to end (inclusive).

The input constraints are as follows: $0 \leq start \leq end \leq 999$

You can safely assume that the inputs will fall within that range.

Instructions

1. Create a file called `Display.java`
2. Write a try-catch statement that attemps to create a Scanner object to read from the logfile ('logfile.txt').
3. If the file does not exist, catch the FileNotFoundException, print "Cannot find logfile!" then "Exiting..." and quit the program.
4. If the file exists, prompt the user to enter start and end indices
5. Print out all the lines between the start and end line numbers (inclusive).

Sample Input/Output Example 1:

```
Enter a start and end index separated by a space:
0 10
0: 1631077133.1621282 WARNING High CPU temp.
1: 1631077133.1621487 ERROR Carrier signal lost.
2: 1631077133.1621530 NOTIFY Waypoint set.
3: 1631077133.1621563 ERROR Motor failure.
4: 1631077133.1621597 NOTIFY Waypoint reached.
5: 1631077133.1621628 WARNING IMU sensor offset.
6: 1631077133.1621654 WARNING Speed limit exceeded.
7: 1631077133.1621683 NOTIFY Returning home.
8: 1631077133.1621709 WARNING High CPU temp.
9: 1631077133.1621737 WARNING Speed limit exceeded.
10: 1631077133.1621764 NOTIFY Rerouting.
```

Example 2:

```
Cannot find logfile!
Exiting...
```

Example 3:

```
Enter a start and end index separated by a space:
499 512
499: 1631077133.1634748 ERROR Battery depleted.
500: 1631077133.1634772 WARNING IMU sensor offset.
501: 1631077133.1634796 WARNING High CPU temp.
502: 1631077133.1634822 NOTIFY Rerouting.
503: 1631077133.1634843 ERROR Carrier signal lost.
```

```
504: 1631077133.1634867 NOTIFY Rerouting.
505: 1631077133.1634893 ERROR Motor failure.
506: 1631077133.1634915 ERROR Carrier signal lost.
507: 1631077133.1634941 ERROR Battery depleted.
508: 1631077133.1634967 WARNING High CPU temp.
509: 1631077133.1634991 ERROR GPS offline.
510: 1631077133.1635013 WARNING Signal degradation.
511: 1631077133.1635036 WARNING Signal degradation.
512: 1631077133.1635070 NOTIFY Rerouting.
```

Task 2: Fault Intolerant

When reviewing logfiles, we need to be aware of the first time an `ERROR` crops up in the log. This can help us see how and when things might have gone wrong.

For this task we will find the first event/message of type `ERROR` and print it out.

Instructions

1. Create a file called `FaultIntolerant.java`
2. As with Task 1, open the file for reading in a try-catch block.
3. Read through the file line by line
4. If the message/event type is `ERROR`, throw a new Exception with the line as the exception message.
5. Catch the thrown exception and print out the message to the user then exit the program immediately.
6. If no error message is found, exit the program without printing anything.

Sample Input/Output Example 1:

```
1: 1631077133.1621487 ERROR Carrier signal lost.
```

Example 2:

```
Cannot find logfile!
Exiting...
```

Task 3: Summary

We would like to be able to generate a summary for each logfile to give us a rough idea of its contents.

Instructions

1. Create a file called `Summary.java`.
2. As with Task 1, open the file for reading in a try-catch block.
3. Read through the entire file, counting how many of each message/event type occurs.
4. Print out a summary with the count of each event/message type.

As a sanity check: the total number of message/event type counts should equal 1000.

Hint: a useful method could be `String.split`.

Sample Input/Output Example 1:

```
Summary
--------------------
Notifications:  322
Warnings:       326
Errors:         352
```

Example 2:

```
Cannot find logfile!
Exiting...
```

Bonus:

Add the necessary code to allow any logfile file to be specified when running the program.

For example:

```
$ java Display newlog.txt
Enter a start and end index separated by a space:
>0 13
0: 1631077133.1634748 ERROR Battery depleted.
1: 1631077133.1634772 WARNING IMU sensor offset.
2: 1631077133.1634796 WARNING High CPU temp.
3: 1631077133.1634822 NOTIFY Rerouting.
4: 1631077133.1634843 ERROR Carrier signal lost.
5: 1631077133.1634867 NOTIFY Rerouting.
6: 1631077133.1634893 ERROR Motor failure.
7: 1631077133.1634915 ERROR Carrier signal lost.
8: 1631077133.1634941 ERROR Battery depleted.
9: 1631077133.1634967 WARNING High CPU temp.
10: 1631077133.1634991 ERROR GPS offline.
12: 1631077133.1635013 WARNING Signal degradation.
13: 1631077133.1635036 WARNING Signal degradation.
```

Note that you only need to do this for one of the tasks above in order to get the marks.

Submission Instructions

Submit only your source code files in a compressed `.zip` folder labeled with your student number an underscore and `assignment4`.

For example if your student number is `12345` submit a file `12345_assignment4.zip` that contains `Display.java`, `FaultIntolerant.java`, and `Summary.java`.

Late submissions incur a 10% penalty per day maximum of 5 days late.

Plagiarism Policy

All forms of plagiarism (programs, tutorials, practicals etc) are illegal and will be punished according to the University's rules. These rules are severe and can lead to rustication. The Computer Science Department checks all program submissions to ensure that the work submitted has not been copied. Work by other authors must be acknowledged as such under all circumstances. The department will assume that you have agreed to the plagiarism declaration for every of your hand-in.

Of course, we do encourage you to share and debate ideas and discuss your work, but we draw a careful line between that and plagiarism. Please pay close attention: we take it very seriously.

Marking

Section	Marks
Task 1: Display	16
Task 2: Fault Intolerant	10
Task 3: Summary	14
Total	**40**