Timed Lab 3
Collections, Streams, and Lambdas

Evan Cahill

1 Problem Description

This is a two part timed lab. The first part will cover the collections API. The second part will test you on your knowledge of the stream API and lambda expressions. Each file will be graded separately from the other files, so be sure to attempt all the parts. Please see the instructions for each of the parts below:

Part 1: Collections (100 points)

This part of the timed lab will require that you modify two files:

`Assignment.java`

1. The `Assignment` class represents an assignment in CS 1331 and is composed of the `title`, `category`, and `maxGrade` of the assignment.
2. Instances of the `Assignment` class must behave properly as elements of any collection. For example, equivalent `Assignment` instances should be findable via the contains method of any collection class and duplicates should be removed in any Set class.
3. The significant fields of an `Assignment` are its `title` and `category`.
4. Lists of `Assignment` instances should be sortable using the `public static <T extends Comparable<? super T> sort(List<T> list)` method according to their natural order. Their natural ordering orders them first by their `category` and then by their `title`. *Hint: All enums have a natural ordering , which orders them according to the order that they were declared in the enum declaration.*

`Category.java`

1. The `Category` enum represents the category of an `Assignment`.
2. The enum should have the following values `HOMEWORK`, `EXAM`, and `LAB` declared in that order.

We have provided a main method in `Assignment.java` for you to test you solution. You are encouraged to use it. You will need to use the -ea switch when you run your program to enable the assertions. ex.

```
java -ea Assignment
```

Part 2: Streams & Lambdas (Up to 10 Extra Points)

`StreamEncoder.java`

You have been provided with a `StreamEncoder` class which contains several methods for working with encoded Strings. Your job is to complete the `decodeSentence(String)` method which takes in a sentence and returns the decoded sentence. You must use stream operations, such as map and reduce, and lambda / method references in your solution. To help you with this task you have been provided two helper methods:

1. `Stream<String> getWordStream(String)`
   Splits a sentence into words based on whitespace and returns a `Stream<String>` over the words in the sentence.
2. `String decodeWord(String)`
   Takes a single encoded word and returns the decoded word. Do not worry about how this method works, but if you are interested it uses ROT47 encoding.

Once again we have provided a main method that will print out a secret message when you have correctly implemented the `decodeSentence(String) method.`

2 Solution and Tips

We've provided all the source files you will need, but you will need to modify them. **Do not change any file names or things marked as do not modify.**

It is absolutely vital that your submission compiles. Non-compiling solutions will receive a zero: no exceptions!

Make sure to test your code.

3 Checkstyle

You must run checkstyle on your submission. The checkstyle cap for this assignment is **10** points.
Review the Style Guide and download the Checkstyle jar and associated XML file. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. You can easily count Checkstyle errors by piping the output of Checkstyle through `wc -l` and subtracting 2 for the two non-error lines printed above (which is how we will deduct points). For example:

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | wc -l
     2
```

Alternatively, if you are on Windows, you can use the following instead:

```
C:\> java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | findstr /v "Starting audit..." |
     findstr /v "Audit done"
0
```

Food for thought: is there a one-liner like above that shows you only the number of errors? Hint: `man grep`.

The Java source files we provide contain no Checkstyle errors. For this assignment, there will be a maximum of **10** points lost due to Checkstyle errors (1 point per error). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

4 Turn-in Procedure

Submit all of the Java source files you modified and resources your program requires to run to T-Square. Do not submit any compiled bytecode (`.class` files), the Checkstyle jar file, or the `cs1331-checkstyle.xml` file. When you're ready, double-check that you have submitted and not just saved a draft.

Please remember to run your code with Checkstyle!

Verify the Success of Your Submission to T-Square
Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
   (a) It helps insure that you turn in the correct files.
   (b) It helps you realize if you omit a file or files. [1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
   (c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight. Do not wait until the last minute!