Monotonic Set

1 Introduction

This assignment is meant to get you familiar with using Collections to write a specific type of set: a monotonic set.

2 Problem Description

A set is a collection which cannot contain duplicates. Your job is to write an implementation of set which can only be added to. To do this, you will be writing an implementing class of `MonotonicSet` called `MySimpleSet`.

3 Solution Description

Provided files (Do not modify these):

- `MonotonicSet`: An interface which defines the set.
- `MySimpleSetTester`: A testing program to make sure your set functions correctly.

You are required to write the following class:

- `MySimpleSet`: An implementation of `MonotonicSet` using an `ArrayList`.

Finally, you are required to Javadoc the class and methods of `MySimpleSet`. **In the class header, include whether the set is A) Always-increasing B) Non-increasing or C) Non-decreasing, along with a brief description of why.**
Running the program should look something like this:

```
$ java MySimpleSetTester
Testing toString:
[Hey!, Listen!, Sorry, I couldn't resist., (you know you would if you could)]
Testing size:
4
Testing iterator:
2.0
4.0
4.2
```

4 Checkstyle

Review the CS 1331 Code Conventions and download the Checkstyle jar file and the configuration file to the directory that contains your Java source files. Run Checkstyle on your code like this (in the directory containing all your Java source files):

```
$ 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 the errors by piping the output of Checkstyle through `grep`.

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | grep -cEv "(Starting
    audit...|Audit done)"
0
$
```

The `-c` option tells `grep` to count matching lines instead of printing them, `E` means use `egrep` (extended `grep`) syntax, and `v` means invert the match. Here we use an inverted match to discard the two non-error lines of Checkstyle's output. If you have `egrep` you could leave off the `-E` and just use `egrep`.

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | egrep -cv "(Starting
    audit...|Audit done)"
0
$
```

Alternatively, if you are Windows, you can use this command to count the number of style errors by piping output through the `findstr` program.

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

The Java source files we provide contain no Checkstyle errors. You are responsible for any Checkstyle errors you introduce when modifying these files. For this assignment, there will be a maximum of 50 points lost due to Checkstyle errors. In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

5 Javadoc

Add Javadoc comments to all classes and methods except `main`. The tags we are looking for are: **@author, @version, @param, @return**.

6 Tips

- Take a look at the API for any questions you might have. This assignment should not take more than 30 minutes to an hour to complete.

7 Turn-in Procedure

Submit all of the Java source files you create to T-Square. Do not submit the classes provided to you, 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. Theses files include, but are not limited to:

- `MySimpleSet.java`

8 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 ensure 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 Wednesday. Do not wait until the last minute!