Programming Assignment #5: BlackBoxOfChaos[1]

COP 3330, Fall 2019

Due: Tuesday, November 26, *before* 11:59 PM

Abstract

In this assignment, you will gain experience working with generics in Java. In particular, you will modify my *BlackBoxOfChaos* class so that it can hold any kind of object under the sun.

The *BlackBoxOfChaos* is an absurdist data structure. Sometimes it loses the values you place inside it. Sometimes it finds lost values again. Its behaviors are governed by a *ChaoticSpiral* class, and both the *BlackBoxOfChaos* and the *ChaoticSpiral* can throw a *ChaosException* under certain circumstances. It's important not to get too caught up in the details of how the *BlackBoxOfChaos* works, and to instead focus on carefully tracing the path followed by any value we insert into one of these boxes, modifying anything necessary as you go in order to make this data structure generic.

Deliverables

BlackBoxOfChaos.java

Note! The capitalization and spelling of your filename matter!

Note! Code must be tested on Eustis, but submitted via Webcourses.

1 Credit for the phrase "black box of chaos" goes to Professor Angell, who once commented that Java is a black box of chaos, given some of the complex things it does for us behind the scenes.

1. Problem Statement

Included with this assignment is a source file named *BlackBoxOfChaos.java*. You must modify that file as follows:

1. Modify the source code so that *BlackBoxOfChaos* is generic (i.e., so that the black boxes of chaos will be able to hold any type of data). Before setting out to accomplish that, you might want to review the notes on generics in Webcourses. In particular, consider reading the sections that provide a step-by-step walk-through of how to modify the vanilla *LinkedList* class we created this semester to make it generic.
2. You might have to modify one or more of the other classes in *BlackBoxOfChaos.java* in order to make the *BlackBoxOfChaos* class fully and properly generic.
3. Add your name and NID to lines 6 and 7 of *BlackBoxOfChaos.java*.
4. Remove *main()*. The *main()* method currently in *BlackBoxOfChaos.java* will not work once the class is generic.
5. Modify the return value of the *public static double difficultyRating()* method so that it returns a double on the range of 1.0 (ridiculously easy) through 5.0 (insanely difficult) indicating how difficult you found this assignment.
6. Modify the return value of the *public static double hoursSpent()* method so that it returns a realistic and reasonable estimate (greater than zero) of the number of hours you spent working on this assignment.

To make this class generic, you will have to change the return types of some of the methods I have implemented. However, please do not change static methods to non-static methods, or private methods to public methods, and vice-versa.

Here's a hint about how to approach this assignment. Highlight and/or copy and paste to reveal:

2. Special Requirement: No Compile-Time Warnings (*Super Important!*)

Here are some special restrictions regarding compile-time warnings:

- ★ Your code must not produce any warnings when compiled on Eustis. For this particular assignment, it's especially important not to have any *-Xlint:unchecked* warnings. Please note that some systems don't produce *-Xlint:unchecked* warnings! The safest way to check whether your code is producing such warnings is to compile and run your program on Eustis with the *test-all.sh* script.
- ★ You cannot use the *@SuppressWarnings* annotation (or any similar annotations) to suppress warnings in this assignment.

Please do not give your code to classmates and ask them to check whether their compilers generate compile-time warnings for your code. Remember, sharing code in this course is out of bounds for assignments.

3. Additional Requirements (*Super Important!*)

Please abide by the following additional requirements for this assignment. It's incredibly important that you follow these restrictions in order to receive full credit:

- ★ Do not add or remove lines of code before line 355, which is where the *difficultyRating()* method begins. You may modify lines 1 through 354 as necessary to make *BlackBoxOfChaos* generic, but you should not add, move, or delete lines. This is important because some of the test cases might examine the output produced by your program when certain exceptions are thrown, and those exceptions will include line numbers where those errors originated. You can, of course, add comments above the *difficultyRating()* and *hoursSpent()* methods, since those fall on lines 355 and below. You can – and must – also remove *main()*, which falls on lines 365 and below.
- ★ Do not modify *ChaoticSpiral.java* or *ChaosException.java*, and do not submit those files with this assignment. The only file you should modify is *BlackBoxOfChaos.java*, and that is the only file you should submit when you're finished with this assignment.
- ★ Other than making *BlackBoxOfChaos* generic (along with its support classes in *BlackBoxOfChaos.java*, as needed), do not alter the behaviors of these classes in any way. While the *BlackBoxOfChaos* might seem to be doing random, chaotic things, it's following careful patterns, and it must continue to do so in order to pass the test cases we'll use to grade this assignment.

4. Compiling and Running All Test Cases (and the *test-all.sh* Script!)

The test cases included with this assignment are designed to show you some ways in which we might test your code and to shed light on the expected functionality of your code. We've also included a script, *test-all.sh*, that will compile and run all test cases for you.

Recall that your code must compile, run, and produce precisely the correct output on Eustis in order to receive full credit. Here's how to make that happen:

1. At the command line, whether you're working on your own system or on Eustis, you need to use the *cd* command to move to the directory where you have all the files for this assignment. For example:

   ```
   cd Desktop/blackbox_assignment
   ```

   Warning: When working at the command line, any spaces in file names or directory names either need to be escaped in the commands you type, or the entire name needs to be wrapped in double quotes. For example:

   ```
   cd blackbox\ assignment
   ```

   ```
   cd "blackbox assignment"
   ```

   It's probably easiest to just avoid file and folder names with spaces.

2. To compile your program with one of my test cases:

```
javac BlackBoxOfChaos.java TestCase01.java
```

3. To run this test case and redirect the program's output to a text file:

```
java TestCase01 > myoutput.txt
```

4. To compare your program's output against the sample output file I've provided for this test case:

```
diff myoutput.txt sample_output/TestCase01-output.txt
```

   If the contents of *myoutput.txt* and *TestCase01-output.txt* are exactly the same, *diff* won't print anything to the screen. It will just look like this:

```
seansz@eustis:~$ diff myoutput.txt sample_output/TestCase01-output.txt
seansz@eustis:~$ _
```

   Otherwise, if the files differ, *diff* will spit out some information about the lines that aren't the same.

5. I've also included a script, *test-all.sh*, that will compile and run all test cases for you. You can run it on Eustis by placing it in a directory with *BlackBoxOfChaos.java* and all the test case files and typing:

```
bash test-all.sh
```

Super Important: Using the *test-all.sh* script to test your code on Eustis is the safest, most sure-fire way to make sure your code is working properly before submitting. Note that this script might have limited functionality on Mac OS systems or Windows systems that aren't using the Linux-style bash shell.

5. Transferring Files to Eustis

When you're ready to test your project on Eustis, using MobaXTerm to transfer your files to Eustis isn't too hard, but if you want to transfer them using a Linux or Mac command line, here's how you do it:

1. At your command line on your own system, use *cd* to go to the folder that contains all your files for this project (*BlackBoxOfChaos.java*, *test-all.sh*, the test case files, and the *sample_output* folder).
2. From that directory, type the following command (replacing `YOUR_NID` with your actual NID) to transfer that whole folder to Eustis:

```
scp -r $(pwd) YOUR_NID@eustis.eecs.ucf.edu:~
```

   Warning: Note that the `$(pwd)` in the command above refers to your current directory when you're at the command line in Linux or Mac OS. The command above transfers the *entire contents* of your current directory to Eustis. That will include all subdirectories, so for the love of all that is good, please don't run that command from your desktop folder if you have a ton of files on your desktop!

6. Deliverables (Submitted via Webcourses, Not Eustis)

Submit a single source file, named *BlackBoxOfChaos.java*, via Webcourses. The source file should meet all restrictions and requirements set forth in this document. Be sure to include your name and NID on lines 6 and 7.

7. Grading Criteria and Miscellaneous Requirements

> ***Important Note:*** When grading your programs, we will use different test cases from the ones we've released with this assignment, to ensure that no one can game the system and earn credit by simply hard-coding the expected output for the test cases we've released to you. You should create additional test cases of your own in order to thoroughly test your code. In creating your own test cases, you should always ask yourself, "What kinds of inputs could be passed to this program that don't violate any of the input specifications, but which haven't already been covered in the test cases included with the assignment?"

The *tentative* scoring breakdown (not set in stone) for this programming assignment is:

100%	Passes test cases with 100% correct output formatting and abides by all restrictions listed above. This portion of the grade includes tests of the *difficultyRating()* and *hoursSpent()* methods. See note about additional deductions below.

(*Important!*) Additional point deductions may be imposed for violating any of the restrictions listed above or failing to meet any of the requirements set forth in this document. You should also still abide by the style restrictions and other general restrictions from the previous assignments. While points will not be *awarded* for following all restrictions and requirements listed in this PDF, points may be *deducted* for failure to do so.

(*Important!*) Your program must be submitted via Webcourses, and it must compile and run on Eustis in order to be eligible for credit. Programs that do not compile and run on Eustis will automatically receive zero credit.

8. Final Thoughts (*Abbreviated*)

- ★ Please be sure to submit a *.java* file (not a *.class* file, and certainly not a *.doc* or *.pdf* file).
- ★ After submitting to Webcourses, please consider downloading your source code, re-compiling it, and re-testing it in order to ensure that you uploaded the correct version.
- ★ Please remove *main()* from *BlackBoxOfChaos.java*, and then double checking that your program still compiles before submitting.
- ★ Please do not articulate a *package* in your *BlackBoxOfChaos.java* file.
- ★ Please be aware that the test cases included with this assignment are not comprehensive. Please be sure to create your own test cases and thoroughly test your code. Sharing test cases with other students is allowed (and encouraged!), as long as those test cases don't include any solution code for the assignment.

Start early! Work hard! Ask questions! Good luck!