CSCI 230 Summer 2019 Due, Monday, July 15, 11:59 pm
HW 3, **50 points**
Java Programming – Implementing a dynamic linked-based list

Collaboration policy: Individual Assignment. You can look up and ask others how to write Java code, but you cannot look up how to solve this problem or get anyone other than Nate or Dr. McCauley to help you with it. You can ask for clarification or Java help on Slack.

In the Dropbox, three Java files are provided (`List` and `Node` are identical from those provided with HW2):

- `List.java` (a List interface) and
- `Node.java` definition of the node objects stored in the list
- `LinkedList.java` (a class definition for a dynamic linked-based list that implements the `List` interface. One new method stub (see below) and new comments in some others).

You **may only** modify the `LinkedList` class. In particular, in the `LinkedList` class you **may only** modify/add the methods listed in **Part 1**, and under no circumstances are you allowed to remove, add, or modify any other line of code in this class (this include instance variables, class variables, constants, etc.).

Lastly, you **may not** add a package structure!

Part 1

In the `LinkedList` class please add the following method (note, remove and setNode and getNode do not have to be implemented):

- `public void addInOrder(AnyType t) throws NullPointerException` this method assumes that the list is currently sorted in ascending order and inserts t into the list in its proper sorted position in the list. (Note the LinkedList stores `Comparables`.)

In the `addInOrder` method above, there is a TODO comment - this is where you add your code. In some of the previous methods implemented in HW2, there is a YOU CAN IGNORE THIS METHOD.

Part 2

The provided `LinkedList` class has a main method with some sample calls in it. In the main please add additional test cases that demonstrate you have fully evaluated the operational correctness of the methods implemented in **Part 1**. To receive full credit, these test cases **must** be included.

Submission

Create a ZIP file that only contains a single folder named lastname (mine would be McCauley), copy the `LinkedList.java` file into the folder (do not include any sub-folders or any other project files in the folder). Zip the folder, it should now be automatically named Lastname – you should not have to change the name of the zip file generated.

<*lastname*>.zip e.g., McCauley.zip

***** If the assignment is not submitted in the correct format – your grade will suffer.

Submit the ZIP file via OAKS in the Dropbox that corresponds to the assignment. Resubmit as many times as you like, the newest submission will be the graded submission.

Per the syllabus, late assignments will not be accepted – no exceptions. Please do not email the TA or me your assignment after the due date, we will not accept it. *It is of higher points value to submit a compiling and running partial solution on time, than a complete solution late.*

Grading Rubric

In particular, the assignment will be graded as follow, if the submitted solution

If code does not compile, grade of 0	
Compiles	5
Runs	5
Thoroughness of test cases	5
Passes all grader test cases	35

Note: the documentation is done for you, but please add your name to the file!