Project 1: Logger

In many applications it is useful to save a sequence of short text messages along with the time each was written. Such a collection is called a log and they're used in a wide variety of situations: Ships keep track of where and when they are, computers keep track of user logins, etc.

Write a class called Logger. Objects of type Logger should have four public methods. A constructor that takes no parameters and initializes the object to be an empty log. One version of method `add` that takes a string and adds it to the log along with the current time and date. Another version of `add` takes a Date object as its first parameter and a string as its second parameter and adds the string to the log along with the date in the Date object. None of these three methods return or print anything, they simply update the log's state. If a string given to add is longer than 50 characters or contains an embedded newline character ("\n"), an `IllegalArgumentException` should be thrown and the object should not be changed.

The final method is `toString` which takes no parameters and returns the entire current log as a string. The returned string should be the concatenation of each log entry formatted as: the date and time as formatted by Date's toString method, followed by ": " (ie, colon and space), followed by the string provided for the log entry. The formatted log entries should be listed in increasing date and time order (entries with the same time and date may be ordered any way you like). A newline ("\n") should be inserted between each log entry. For example the following code snippet should print `true` .

```
Logger log = new Logger();
log.add(new Date("1 Jan 2017, 13:00:00 PST"), "Hello");
log.add(new Date("1 Jan 2017, 14:00:00 PST"), "World");
log.add(new Date("1 Jan 2017, 13:30:00 PST"), "Wonderful");
String expected = "Sun Jan 01 13:00:00 PST 2017: Hello\n" +
                  "Sun Jan 01 13:30:00 PST 2017: Wonderful\n" +
                  "Sun Jan 01 14:00:00 PST 2017: World";
System.out.println(log.toString().equals(expected));
```

You will need to compare two Date objects to know what order they should come in. You are no doubt familiar with `equals` as a method to compare two objects for equivalence. Date has that method, but also `compareTo`. The expression `d1.compareTo(d2)` will evaluate to an integer less than zero if `d1` comes before `d2` or 0 if they are equal or an integer greater than zero if `d1` should come after `d2`.

Arrays

Because of the requirement that toString produce a log in date and time order, and because add allows log entries to be added out of date and time order, you are required to keep your logs internally as an array of Date objects and an array of associated log strings, both in date and time order. So, the string at index *i* is associated with the Date at index *i*. When the user calls toString the string to return is relatively easy to construct from the data in the two arrays.

There are two complications that you must handle. Each time an add occurs, you must determine where in the arrays the new Date and string belong. If the position is not at the end of the entries, then some entries will have to be shifted to make room for the new entry. Also, since the log has no size bound, you must accommodate any number of entries. You should do this by having your arrays be an initially small number, like 32. Then, each time the size of your array is too small, make a new array double the size of the old array, copy the old contents into the new, and use the new array instead of the old.

If you need a refresher on how arrays work, reread as much of Chapter 7 of our textbook as needed.

Notes

- The Date class is the simplest way to get the current time formatted into a human-readable string. `new Date()` creates an object with the current time and date. The toString method of this object returns a string of exactly 28 characters. The no-argument constrictor and toString method of Date are not "deprecated", but the rest of the Date methods are. In general, if something is deprecated it should be avoided because the designers decided it was not a good idea and may be removed in the future. If you do things like `new Date("1 Jan 2017, 13:00:00 PST")` , you will receive warnings from the compiler of deprecation. This is okay for testing's sake, but shouldn't be used in actual code being sent to clients. For full details of the Date class, Google-search "Java 8 Date" and read the javadoc.
- The string representation of an empty log is just an empty string ("").
- It is conventional for toString to return a string that does not end with a newline. So, even though you are inserting a newline between each log entry, you should not add one to the end of the string being returned.

To Receive Credit

Follow the directions in Project Requirements and DBInbox Submission, and submit by 11:59pm, Sunday, February 12, 2017.

Questions?

If something is not clear, ask questions in class, on Piazza, or in office hours. Do not wait until the last minute to clear things up. Start early!

first published 2pm, Jan 17, 2017
modified 8am, Feb 1, 2017 to include information about compareTo