Continuous Position Calculator

Background:

This problem is essentially a simplified version of code implemented for one of our products. Under no circumstances will we use any of the code which you submit for evaluation in any of our products. As a demonstration of good faith, once you submit your answer, we will provide you with our version of the answer to this question which we have already completed.

Objective:

Create a Java class DemoGPS (as well as any support classes necessary) which reads in NMEA strings in realtime from a GPS and uses them to create a rolling average position which can be requested at any time.

Requirements:

1. Use the Java 1.8 API (assume all classes in the standard library are available).
2. Your class must be **final**.
3. Your constructor must take 2 arguments: a java.io.InputStream (it will be passed to your constructor already connected to a properly configured serial port connected to the GPS) from which you will read the NMEA strings, and an integer which specifies the number of GPS position readings to use in the average.
4. Your class must have a public **start** method which will start a new thread which will continually read the NMEA strings from the input.
5. Your class should have a public method called **getCurrentPosition** which returns the current position (i.e. the average of at most the last N GPS positions, where N is the number specified in the constructor, or whatever is available if fewer than N good positions have been read) as an object of a static inner class named **Position**. It should return null if no positions have been read.
6. **Position** must include the latitude and longitude as double precision numbers representing degrees, using the standard convention of positive for North and East, negative for South and West.
7. Your class must be thread-safe. **getCurrentPosition** may be called from more than one external thread simultaneously.
8. Your class should be commented as appropriate for production, so that someone unfamiliar with the code can take over maintenance of the class.

Environment:

1. DemoGPS will be a component in a larger system. It does not need a main method.
2. The GPS from which you are reading has been configured to output GGA and GSV data to you. You must therefore ignore the irrelevant NMEA lines.
3. The GPS outputs GGA and full GSV information (approximately) once per second.

4. Since the GPS data is transmitted to us over a serial line which can drop or scramble bytes, your code should deal with corrupted/missing lines gracefully.
5. The processing loop in your code should not have any termination conditions and should run indefinitely. (It may terminate if a java.io.EOFException is thrown.)
6. As many of our products run in resource-constrained environments, your code should avoid unnecessary indirection.

NMEA format documentation:

http://aprs.gids.nl/nmea/
https://web.archive.org/web/20190104055009/http://www.gpsinformation.org/dale/nmea.htm

Submission Guidelines:

Please submit a single java file (no zip or other archives) called DemoGPS.java to dheck@solartechnology.com. (If you need to create any additional classes, please make them inner classes and just note in a comment if you would have made it stand-alone.)