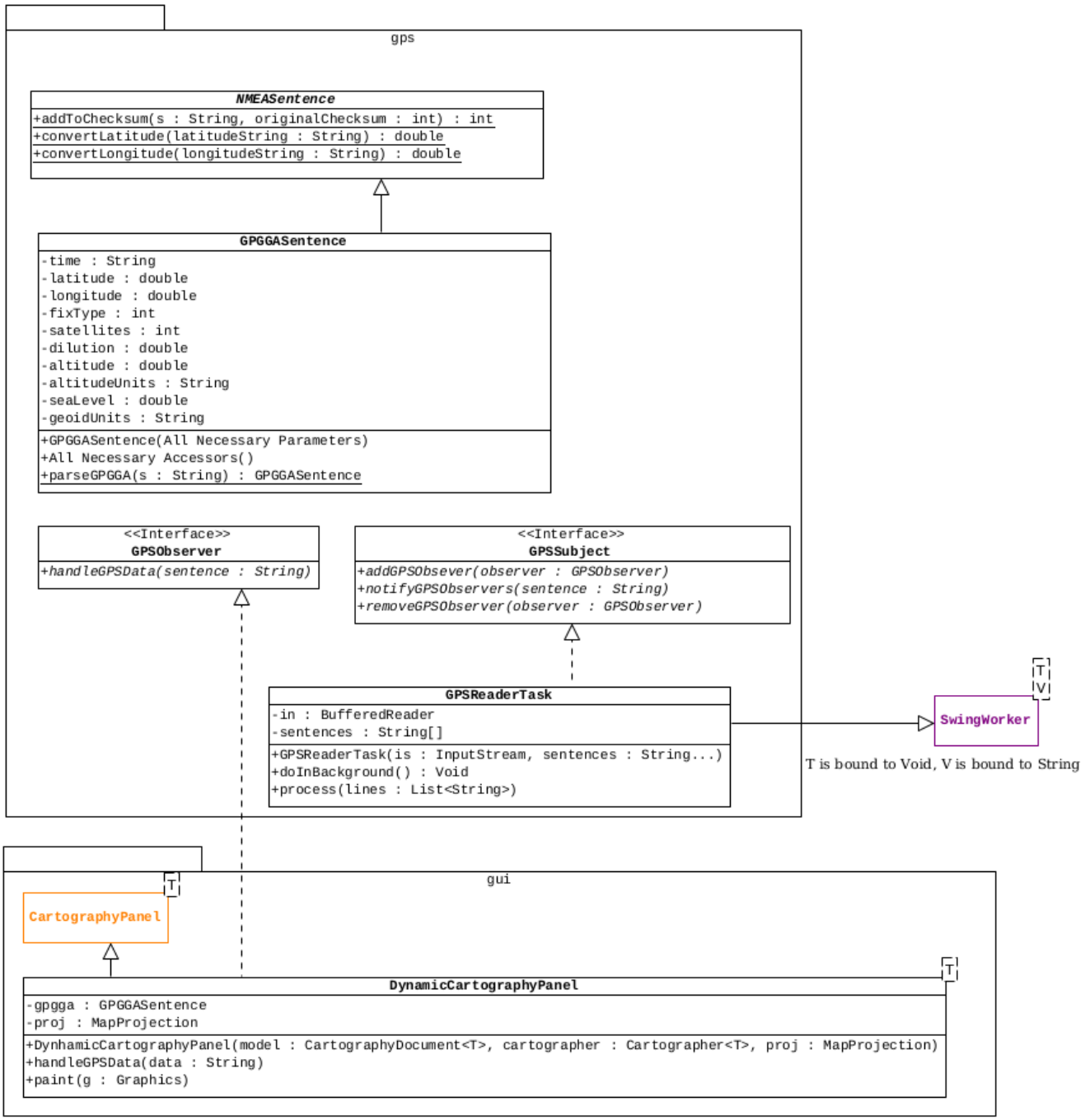
Navigation System v4

Purpose

Version 4 of the navigation system will provide the user with position/location information from a GPS receiver.

Design

The design of the system is summarized in the following UML class diagrams. Note that the components in jade green are part of the Java API and the components in orange are "old" (i.e., were developed for a previous version of the Navigation System).


gps
NMEASentence
+addToChecksum(s : String, originalChecksum : int) : int
+convertLatitude(latitudeString : String) : double
+convertLongitude(longitudeString : String) : double
GPGGASentence
-time : String
-latitude : double
-longitude : double
-fixType : int
-satellites : int
-dilution : double
-altitude : double
-altitudeUnits : String
-seaLevel : double
-geoidUnits : String
+GPGGASentence(All Necessary Parameters)
+All Necessary Accessors()
+parseGPGGA(s : String) : GPGGASentence
<<Interface>>
GPSObserver
+handleGPSData(sentence : String)
<<Interface>>
GPSSubject
+addGPSObsever(observer : GPSObserver)
+notifyGPSObservers(sentence : String)
+removeGPSObserver(observer : GPSObserver)
GPSReaderTask
-in : BufferedReader
-sentences : String[]
+GPSReaderTask(is : InputStream, sentences : String...)
+doInBackground() : Void
+process(lines : List<String>)
T
V
SwingWorker
T is bound to Void, V is bound to String
gui
T
CartographyPanel
T
DynamicCartographyPanel
-gpgga : GPGGASentence
-proj : MapProjection
+DynhamicCartographyPanel(model : CartographyDocument<T>, cartographer : Cartographer<T>, proj : MapProjection)
+handleGPSData(data : String)
+paint(g : Graphics)

Specifications

This section contains design specifications for some of the components above. For the others, the UML diagrams should provide all of the information that you need.

The `DynamicCartographyPanel` Class

A `DynamicCartographyPane` object is a `CartographyPanel` object that can handle dynamic position/location updates.

The `handleGPSData()` must parse the NMEA sentence it is passed and store it as an attribute.

The `paint()` method must:

1. Add a `Rectangle2D.Double` object to the first element of the `zoomStack` that is centered on the current location/position and is 2km wide and 2km high.
2. Call the parent `paint()` method to invoke the street network.
3. Project and transform the current position/location.
4. Render a filled circle (in red) that is centered on the current location and is 8 pixels wide and 8 pixels high.

The `GPSReaderTask` Class

The constructor is passed the `InputStream` to read from (which is typically obtained from the GPS device) and a variable number of NMEA sentences that must be processed.

The `doInBackground()` method must continuously (until the task is cancelled) read lines from the `BufferedReader` (that decorates the `InputStream` passed to the constructor), and invoke `publish()` for each line.

The `process()` method must notify all `GPSObserver` objects of each line that it is supposed to process (as defined by `sentences[]`). (Note: For performance reasons, multiple lines might be coalesced into a "chunk", which is why `process()` is passed a `List<String>` and not an individual `String`.

Examples

This section contains examples of what we hope the maps will look like.