Sorry I didn’t get back to you all on Friday. Here is some more info on the close approach analysis project.

The basics of what we will do, summarized with just two spacecraft for simplicity:

1) Load the satellite orbits from the space-track data

2) Propagate both orbits over the same time span (something like 7 days, this would be a configurable input), output from this is a time-history of position and velocity vectors of the satellites at some regular time-step like 1 minute. We will use the Orekit library for generating this data.

3) Calculate the separation of the satellites at each time step in each of the x-y-z components and the total separation (there are also orbit relative frames that we would use, i.e. radial/in-track/cross-track)

4) Look for close approaches within a certain configurable threshold and report those—both what the closest approach distance is and how long the separation is below the threshold

5) Output results with text reports and plots

Part of the complexity of the big problem is of course that there are thousands of objects/satellites to analyze, each vs. each other one. Once we get the basic problem working for two objects (along with unit tests), we will expand to analyze the whole catalog, look at doing the processing in parallel, and if we have any extra time we could add calculations for “probability of collision”.

f you have any questions as you think about which project to work on please email me.

Here are the links that I pulled up today:

Satellite catalog (two-line element sets) for use in the close approach analysis project:

<http://celestrak.com/>

<https://www.space-track.org/> (free account creation required)

Cesium.js library (would be used for 3D orbit visualization project):

<https://cesium.com/platform/cesiumjs/>

Orbit visualization using Cesium.js is also on the CelesTrak site:

<http://celestrak.com/cesium/orbit-viz.php?tle=/pub/TLE/catalog.txt&satcat=/pub/satcat.txt&referenceFrame=1>

Introduction material to orbits and spacecraft flight dynamics:

<https://en.wikipedia.org/wiki/Orbit>

- Read up until "Newtonian analysis of orbital motion"

<https://en.wikipedia.org/wiki/Orbital_elements>

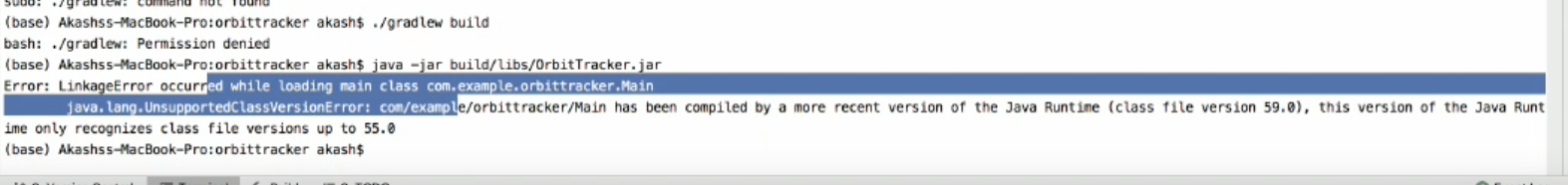
- Learn about classical elements

Two other good pages to read:

<https://en.wikipedia.org/wiki/Orbital_mechanics>

<https://en.wikipedia.org/wiki/Flight_dynamics_(spacecraft)>

-Galen



Break down task into an outline of the processing steps, inputs, outputs, data that will be generated, etc., before starting deatiled coding (either in skeleton code and comments or in plain text).

* Load in all the data - few thousand satellites
  + <https://www.celestrak.com/NORAD/elements/active.txt>
  + Determine 7 day period
* Set up orbits
  + Use Kepler Orbits?
  + Find a way to import a text file in Java (pretty sure I know a package for this)
  + Use orekit library for data
* Determine which satellites are close to each other
  + Input a threshold value for when satellites are considered close
  + Determine what satellites and the times for when distance is below threshold
  + Output satellites name and time
  + Needs to be an efficient way cause we have a lot of data
  + Maybe split orbits into leo meo and geo to decrease the number of comparisons by threefold
  + Calculate the position of each satellite every minute
    - Position can be calculate by using the orbit of the satellite
  + Compare positions and see if value is below threshold
  + Output results to a text file

Inputs

* Data from celestrak and possibly spacetrack
* 7 day period to investigate
* Threshold for what is considered close

Outputs

* Satellites that are in close contact and for how long
* Plots of close contact satellites so it easier to visualize
  + Plot can be time versus distance to closest satellite?

More detailed outline for close approach

* Use the given values (from 2 line data sets) to determine if orbit is leo meo or geo
  + Compare leo first, then meo, then geo for the process below
* Find the appropiate orbit based on the 2 line data set
* Iterate through a 7 day period
  + For each minute, find the updated position for the satellite
  + Check satellites to see if any are below threshold range
  + Make note of satellites that are close to each other and the time
* Complete process for meo orbits
* Complete process for geo orbits