Some thoughts about the logical layout and flow of the program follow. These are just some ideas that will surely change as more accurate ideas form.

Math updated

==========

How to: Load raw data file

- Have a class called RawDataLine that has a data member for each point of data in a line of the file.
- Methods:
    - Read an instance of itself in from the file. Accepts an open stream to the file to be read from.
- Constructor initializes data members.

Questions:

- Who has an instance of this class?
    - Calculator has a list of instances of this class.
- What kind of stream should be used?
    - BufferedReader given a FileReader. A faster approach would be to have a BufferedInputStream given a FileInputStream, though less convenient.
- Does this class have value, or should it just have its work handled by the parent class using arrays to store the data?
    - ?

==========

How to: Write calculated data to file

- Have a class called OrderedPair that has two data members, xAxis and yAxis. This class is used to read and write these pairs in a uniform manner. For example, the xAxis would represent a time and the yAxis would represent a velocity in one case.
- Methods to:
    - Write itself to a file with an open stream passed to it when called.
    - Read itself from a file with an open stream passed to it when called.
- Constructor accepts integers and sets data members to given values.

Questions:

- Who has an instance of this class?
    - Calculator has a few instances of this class that it tells to write to the appropriate graph data file. To elaborate, one instance for time versus acceleration, one for time versus velocity, one for time versus displacement, and perhaps others as needed.
- What kind of streams should be used?

    - For now, trying ObjectInputStream and ObjectOutputStream each given a FileInputStream and FileOutputStream respectively.
- Does this class have value, or should it just not exist and allow parents to take on the task of reading and writing these pairs?
    - ?

==========

How to: Calculate the data

- Have a class called Calculator that tells several RawDataLine instances to read themselves in from the raw data file and then, using the appropriate data members, to calculate required results such as velocity and displacement at a given time in a given axis. These resulting data points are turned into instances of OrderedPair that will be read in and graphed later, but for now are told to write themselves to file.
- Methods to:
    - Do calculation on RawDataLines currently stored in a list to create instances of OrderedPair that then are told to write themselves to their respective graph data files..
- Constructor accepts:
    - Raw file path to open a stream on.
- This class opens the streams on every file being read and written to, one for reading the Raw data file and many for writing the pairs of graph data to files.

Questions:

- Who owns this class?
    - Likely when the drop down menu item to ‘Load Raw Flight File’ is clicked, or a file is drug onto the screen that is a Raw Flight File, this class is instantiated and passed the file path so it can open it and begin reading.
    - How will we know what files are Raw and which are graph data files? Signature at the beginning of the graph data files?
- How are the graph data files named and placed?
    - Should the user pick, or should their be a hard coded location for each flight?

==========

How to: Read a Graph data file

- Have a class owned by the new JTabbedPanel that opens a stream on every graph data file that needs to be graphed in this flights directory and starts having instances of OrderedPair read themselves in to convert them into the correct graphical format for the GUI to use.

- Methods to:
    - Open a stream on a graph data file and test for signature.
    - Create OrderedPair instances to tell them to read themselves in.
    - Convert OrderedPair into appropriate graphical format to be graphed.
- Constructor accepts
    - Flight directory path?

Questions:

- What kind of streams should be used?
    - The OrderedPair currently uses FileInputStream handed to an ObjectInputStream, but perhaps a faster way will be used later.

==========

Given acceleration and time

$A(t) = a$
$V(t) = \text{Integral } (A(t))\, dt = a * t + v_0$
$S(t) = \text{integral } (V(t))\, dt = (a * t \wedge 2) / 2 + v_0 * t + s_0$

v_0 is the previous velocity in this case
s_0 is the previous displacement in this case

Note: Changing to the following formulas for accuracy and simplicity from the Riemann Sums approach previously documented.

$V_i = ½ (a_{i-1} + a_i) * \Delta t + V_{i-1}$
$S_i = ½ (v_{i-1} + v_i) * \Delta t + S_{i-1}$
$\Delta t = (t_{i-1} - t_i)$

Note: $½ (a_{i-1} + a_i)$ is an average and for this project, we can instead use $1/10 (a_{i-10} + ... + a_i)$ for a greater smoothing of the data. This is true for $½ (v_{i-1} + v_i)$ as well. Another way to implement this larger averaging follows as only the endpoints of acceleration actually need to be divided by 2 when following the recurrence.

For 10 points of data:

$V_i = Vi\text{-}10 + ½ (a_{i-10} + a_i) * \Delta t + \sum_{n=1}^{i-1} An * \Delta t$

$S_i = Si\text{-}10 + ½ (v_{i-10} + v_i) * \Delta t + \sum_{n=1}^{i-1} Vn * \Delta t$

However, for ease of coding for the time being, we will use the following equations.

For 10 points of data:

$V_i = 1/10\ (a_{i-10} + ... + a_i) * \Delta t + V_{i-1}$
$S_i = 1/10\ (v_{i-10} + ... + v_i) * \Delta t + S_{i-1}$
$\Delta t = (t_{i-1} - t_i)$

Additionally, since there could be and most likely will be variations in Δt, we could take a roundabout way of averaging it.

$\Delta t = ((t_{i-3} + t_{i-1}) / 2 - (t_{i-2} + t_i) / 2)$

==========

For every reading, the axes of the accelerometer must be re-orthogonalized to get the true accelerations with respect to the original axes defined by the original position of the accelerometer when it started to take its readings. Without accounting for this, the integrated velocities and displacements would be completely off with respect to the launch coordinates.

A solution that needs further research is to use Rotational Matrices to re-orthogonalize the acceleration magnitudes before integrating them to get other values such as velocity and acceleration.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Ex: } \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

The above matrix would give the components of x, y, z denoted X', Y', Z' as a result of a rotation θ about the z axis.
Does this mean that we now need to correct for the spin about y and x by performing their respective rotational matrix on the X', Y', Z'?