CS 350 Project SRS Version 0.1

Part 1: Utility Application

The first part of the project serves two basic but valuable purposes. First, it is a utility for users who want to interact later with the larger project to come. It is very common for different stakeholders and resources to use different definitions for their data; e.g., the axes and units in coordinate systems. This leads to endless confusion and chances to make conversion mistakes. Second, it allows users to plan simple flight paths for spacecraft. Specifically, it calculates the distances traveled on different axes from a list of coordinates. The implementation as usual involves nothing more than basic arithmetic, and there are only three functional methods (two virtually identical). However, managing the complexity to perform the right actions on the right data is nontrivial. There are many combinations to test, so a good design and test plan are critical.

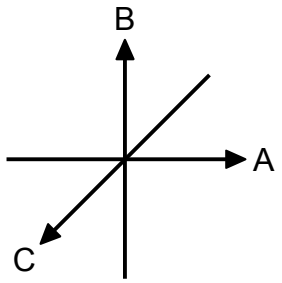
Specifications

See the Javadoc for full details. There is only one source file, but it contains an inner class and three enumerations. As always, implement the public contract as specified. Not all details are explicitly stated. Be sure to ask for clarification if you are unsure.

Operational Details

The client instantiates a `WaypointPlanner` by defining how to interpret the axes of the input stream. Each line consists of a triple containing three comma-delimited real values that specify coordinates in three-dimensional space according to the user's coordinate system. A line ends with a newline. Whitespace is ignored.

The columns in the input stream can be any of six permutations of *x*, *y*, and *z*, namely *xyz*, *xzy*, *yxz*, *yzx*, *zxy*, and *zyx*. This native external format maps onto the canonical internal format defined as axes *A*, *B*, and *C*, where the origin is (0,0,0) and the arrows indicate increasing values:



Each native axis can increase in the same direction as the canonical axis it is mapped to or in the opposite direction. For example, native *z* might increase downward on *B*. There are six permutations of the axes and eight of the directions for a total of 48 possible input definitions. Similarly, the input unit may be defined as meters, kilometers, feet, or miles. The canonical unit is always meters. This utility converts them all to the canonical format for consistency with other (nonexistent) tools in the larger project. It can return the coordinates in either format and in any unit.

The second feature is to be able to compute the distances traveled from the first to the last coordinates. There are two variants: the total distance as a single number, and a list of intermediate distances between each of the coordinates (which sums to the total). Flight planners may need to know how much distance is covered on different combinations of axes; e.g., one-dimensional *x* distance on a line, two-dimensional *xy* distance on a plane, or three-dimensional *xyz* distance within a volume. The unit of distance can be selected.

See the resources on the task link for examples.

Deliverables

Implement all the Javadoc functionality unless indicated otherwise. Be sure to check for all errors. Submit only the Java source file. There is only one chance to get this solution correct. Although this is a small programming task, work on it as a team. You will evaluate each other at the end of the project.