LenaUbi Code Files:

- Program.cs (**LenaUbi** class and **main** method)
- Person.cs (**PersonInfo** struct and **Person** class)
- ClassroomDay.cs (**DateTimeComparer** and **ClassroomDay** class)

Summary

The purpose of the program is to process the raw data files containing Ubisense and LENA data and store it in an easily accessible data structure, in this case a Dictionary.

To construct this, data is reduced to a 0.1s resolution. So, at any time between the recorded period of that day to the tenth of a second, we extrapolate the x-y location of each subject in the classroom and determine whether they are talking from the values in the vocalization duration column of the CSV.

Format of the Data

The main Dictionary, *activities*, has keys referenced by DateTimes and values (another Dictionary) that store all pertinent information at that time; i.e. the x-y location of each individual and whether they were talking. Additional Dictionaries, such as pairStats and pairClose, count up and store the amount of time in which two subjects meet a certain requirement (such as being in proximity and talking). More information about data can be found within the code and comments.

Raw Data Specifications

With the current built-in functionality, the raw data files must have the following conditions:

- Raw data must be a CSV with columns containing the following information:
    - Date and time
    - Subject ID
    - Boolean for whether or not the time is during freeplay
    - X position
    - Y position
    - Vocalization duration
- The times must be sorted from earliest to latest. It is okay if there are repeat times (order not important) so long as they reference subjects of different IDs
- Rows of times should have one of the following:
    - Ubisense data **only**
    - LENA data **only**
    - Ubisense data and LENA data combined.

The files used in the main method of the code can serve as examples of proper files to use.

Process of Converting Data to a 0.1s resolution

First, the CSV is read in and all the raw data is stored for processing. The goal is to fill a dictionary of times for every tenth of a second between the first recorded time of that day and the last recorded time of that day. To extrapolate the x-y positions of a person at a time that does not exist in the raw data, a **linear interpolation** is used. For example, if we want to know the position of subject B1 at 9:00:23.2 (9:00am, 23.2 seconds), we take the two closest times, one before and one after, to interpolate the position at 23.2. A binary search is used to find these closest times; therefore, it is imperative that the input data be sorted by time beforehand. In the rare case that the raw data time lines up exactly with the target time, the x-y values are not interpolated and are just taken from the raw.

The code creates a time for every tenth of a second from start to end, but sometimes subjects may leave the room for extended periods of time, which could raise severe issues in the extrapolation. So, to ensure that the subject is actually in the room, the code checks that the two closest times preceding and succeeding the target time being extrapolated are within a minute of each other. Thus, if the subject leaves the room for an extended period of time, the values in between would not be extrapolated and the subject would not show up in the dictionary for those times.

Lines with only LENA data are stored in a separate dictionary and added in separately after the positional data are all interpolated. So, the code goes through constructing all the x-y positions for all times with a default value of **false** for talking. Then the code goes through and sets these talk values to **true** using the time windows provided by the extra LENA data.

For more details on the process, follow the comments in the code files. An example of the minimum number of method calls in the **main** method for a correct construction of 0.1s is given below:

```
LenaUbi obj = new LenaUbi(filename); //create a new LenaUbi object with filename
//read csv and set parameter to true if want to look at freeplay only
obj.readCSV(false);
//process data with the ClassroomDay class
ClassroomDay day = new ClassroomDay(obj.rawInfo, obj.lenaInfo);
//initialize object's activities Dict
obj.activities = day.getActivities();
//make pairs with all the IDs
List<String> a = obj.makePairs(obj.peopleId);
//count interactions but don’t write a file
obj.countInteractions(false);
```