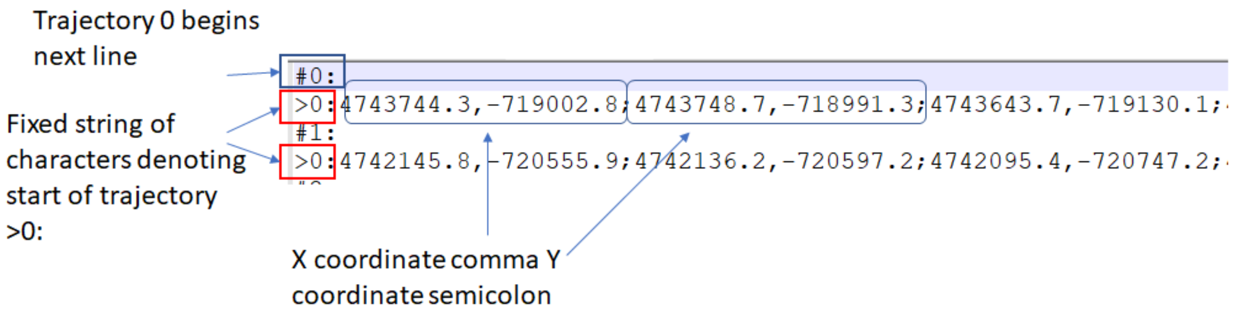
Setup + Build

Source code for AdaTrace is under AdaTrace/src. All required classes are included within the code package – there should be no need to download additional packages. Just import into a Java IDE (e.g., Eclipse), and the code should compile and build without problems.

There are 2 external jars, both of which are included within the code package: commons-math3-3.4.1.jar and kd.jar. Please add them to your project Java build path, otherwise AdaTrace won't build.

Input Data Files

The code package contains an input trajectory database: brinkhoff.dat. Here is a sample of the first few lines of an input database file and the meanings of the symbols:

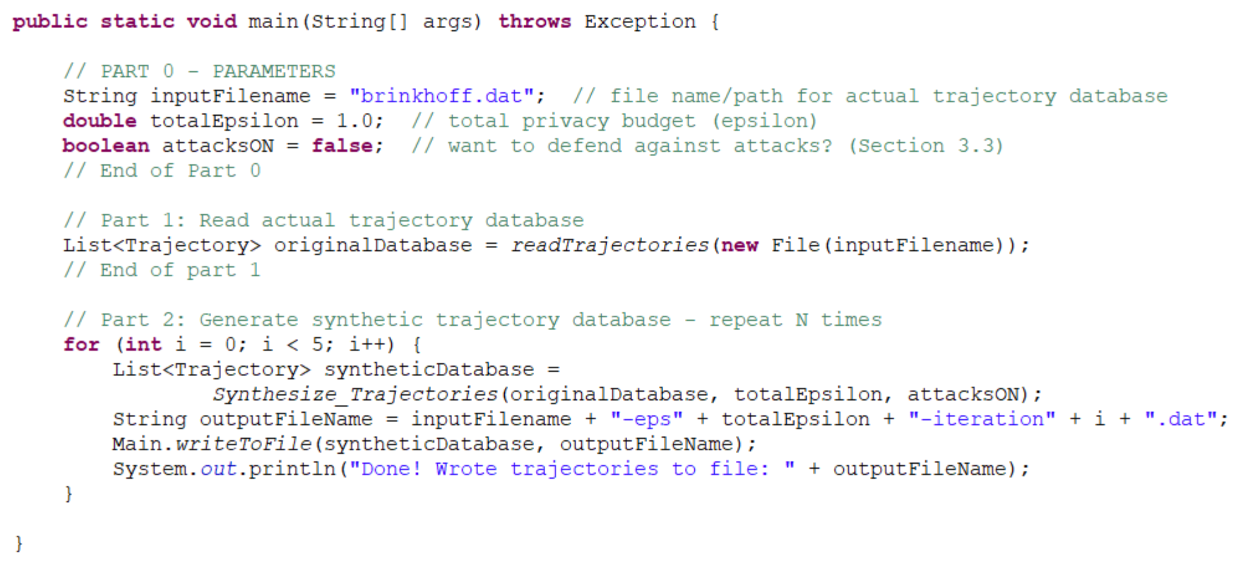

```
#0:
>0:4743744.3,-719002.8;4743748.7,-718991.3;4743643.7,-719130.1;
#1:
>0:4742145.8,-720555.9;4742136.2,-720597.2;4742095.4,-720747.2;
```
Trajectory 0 begins next line
Fixed string of characters denoting start of trajectory >0:
X coordinate comma Y coordinate semicolon

Trajectories are written line-by-line. On each line, trajectory's points are separated by semicolons. Each point is given in *x-coordinate, y-coordinate* format. Note that coordinates are assumed to be (x,y) in Cartesian space, not latitude,longitude.

AdaTrace assumes its input file is going to be in this format. If the file is formatted differently, AdaTrace will have unexpected/unknown behavior. Its output format (database of synthetic trajectories) follows the same file format.

Generating Trajectories

To generate synthetic trajectories using AdaTrace, go to Main.java, scroll all the way down to the main function. A screenshot showing the main function:



```
public static void main(String[] args) throws Exception {

    // PART 0 - PARAMETERS
    String inputFilename = "brinkhoff.dat";  // file name/path for actual trajectory database
    double totalEpsilon = 1.0;  // total privacy budget (epsilon)
    boolean attacksON = false;  // want to defend against attacks? (Section 3.3)
    // End of Part 0

    // Part 1: Read actual trajectory database
    List<Trajectory> originalDatabase = readTrajectories(new File(inputFilename));
    // End of part 1

    // Part 2: Generate synthetic trajectory database - repeat N times
    for (int i = 0; i < 5; i++) {
        List<Trajectory> syntheticDatabase =
                Synthesize_Trajectories(originalDatabase, totalEpsilon, attacksON);
        String outputFileName = inputFilename + "-eps" + totalEpsilon + "-iteration" + i + ".dat";
        Main.writeToFile(syntheticDatabase, outputFileName);
        System.out.println("Done! Wrote trajectories to file: " + outputFileName);
    }

}
```

The parameters you may want to change are located within this function:

- inputFilename: Should point to the actual (real) trajectory database, e.g., Brinkhoff
- totalEpsilon: The differential privacy budget epsilon
- attacksON: A boolean determining whether we want the trajectories to be attack-resilient or not. By default, it is set to false, thus AdaTrace will only enforce differential privacy. If set to true, AdaTrace will enforce resilience to Bayesian inference, partial sniffing, and outlier leakage attacks.

By default, each experiment is repeated 5 times due to the probabilistic nature of differential privacy and AdaTrace's trajectory generation algorithm. The output of each iteration is a synthetic trajectory database, which is written to a file on disk.

AdaTrace has several other parameters that you can play with, e.g.: budget distribution to sub-components, initial size for adaptive grid, … There are default values for them in place, most of which were found empirically. Note that results of AdaTrace will be slightly different in each run due to its probabilistic nature, the budget distribution, and the parameters within and outside the main function.

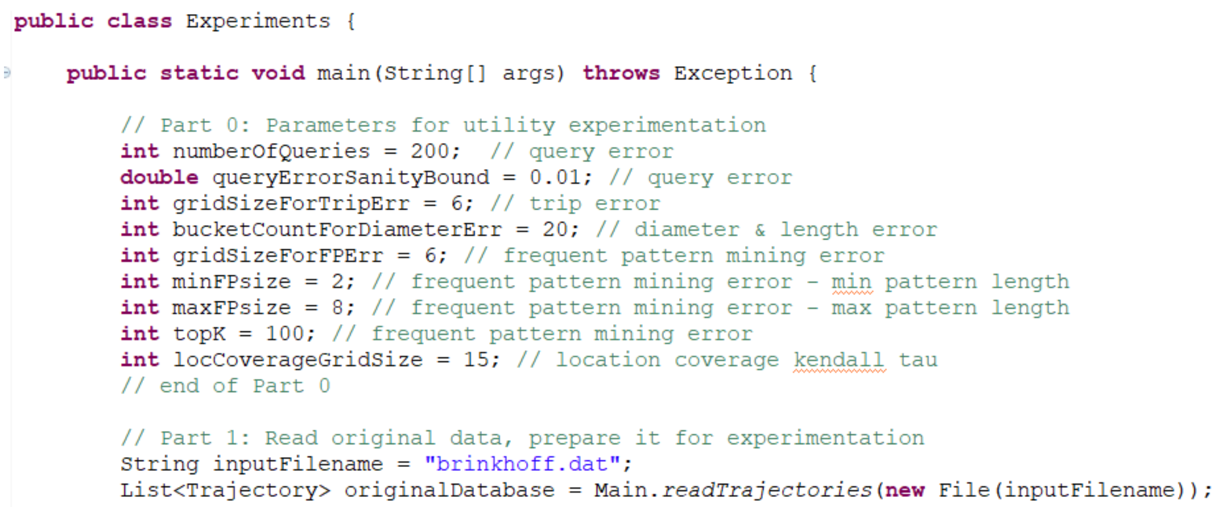
Evaluating Trajectory Quality (Experimentation)

Now that you generated trajectories using AdaTrace (or some other trajectory generation software), how do you empirically assess their quality?

First make sure that your synthetic trajectory databases follow the same file format described above. By default, AdaTrace's output files adhere to this format. Each database should be contained in one file.

Copy your output database files to the folder: AdaTrace/SYNTHETIC-DATASETS. You can have multiple files in this folder, each corresponding to a different synthetic trajectory database.

To run experiments, use Experiments.java class of AdaTrace. Here is the code from Experiments.java:



```java
public class Experiments {

    public static void main(String[] args) throws Exception {

        // Part 0: Parameters for utility experimentation
        int numberOfQueries = 200;  // query error
        double queryErrorSanityBound = 0.01; // query error
        int gridSizeForTripErr = 6; // trip error
        int bucketCountForDiameterErr = 20; // diameter & length error
        int gridSizeForFPErr = 6; // frequent pattern mining error
        int minFPsize = 2; // frequent pattern mining error - min pattern length
        int maxFPsize = 8; // frequent pattern mining error - max pattern length
        int topK = 100; // frequent pattern mining error
        int locCoverageGridSize = 15; // location coverage kendall tau
        // end of Part 0

        // Part 1: Read original data, prepare it for experimentation
        String inputFilename = "brinkhoff.dat";
        List<Trajectory> originalDatabase = Main.readTrajectories(new File(inputFilename));
```

Part 0 contains the parameters for experimentation, e.g., the number of queries, the sanity bound, grid sizes used in measuring errors, and so forth. Explanations of these parameters can be found in AdaTrace paper, Section 6.

VERY IMPORTANT -> Under Part 1, inputFilename should point to the database file containing actual (real) trajectories. All synthetic databases under AdaTrace/SYNTHETIC-DATASETS will be compared against the actual database pointed by inputFilename. In this example, this points to the Brinkhoff dataset.

When you run the code, results with respect to all error types will be printed to stdout.