This assignment is a collaboration between:

- Megan Fantes (U56999246)
- Efstathios Karatsiolis (U65214507)

PA 3
Design Document

Describe any design decisions you made. If you used something other than a nested-loops join, describe the tradeoffs of the algorithm you chose. Discuss and justify any changes you made to the API. Describe any missing or incomplete elements of your code.

- We chose to implement Join.java with a nested-loop join. It is the most straight-forward algorithm to implement, and allows us to understand the data-flow of the simpleDB before trying to write code for more difficult join algorithms.

- The Predicate, JoinPredicate, and Filter .java files contain helper methods that are called during a Join. Implementing them was straightforward, so we did not make any dramatic design decisions, because most of the methods were getters and setters.

- In HashEquiJoin, we added a method to streamline the join process:
  - processList()
    - This method is a simple helper method that loops through the field of two input tuples and combines them. If the list iterator over a tuples of tuples is not null, then this method is called to combine the two tuples in the HashEquiJoin.

Describe how long you spent on the lab, and whether there was anything you found particularly difficult or confusing.

- This assignment took us about 20 hours of coding between 2 people.
- Implementing HashEquiJoin was slightly confusing, it seemed the skeleton of the file was missing a few methods for the logic of the join algorithm.