```java
public class SSSP {
  static Map<Integer, List<Integer>> graph;
  static Map<Integer, Double> distances;
  static final Integer N = ...;
  static final Double EPSILON = ...;
  public static void main(String[] args) {
    graph = makeGraph(N);
    distances = new TreeMap();
    Map<Integer, Double> newDistances = new TreeMap<>();
    newDistances.put(0, 0.0);
    for (int i = 1; i < N; i++) {
      newDistances.put(i, Double.MAX_VALUE);
    }
    MapReduce<Integer, Integer, Double, Double> mapReduce
            = new MapReduce<>();
    mapReduce.setMapperSupplier(SSSP.Mapper::new);
    mapReduce.setReducerSupplier(SSSP.Reducer::new);
    boolean done = false;
    while (!done) {
      distances.putAll(newDistances);
      mapReduce.setInput(
              listOfFiniteDistanceNodes(distances)
      );
      newDistances.putAll(mapReduce.call());
      done = withinEpsilon(distances, newDistances);
    }
    displayOutput(distances);
  }
}
```

FIGURE 17.18 The SSSP class used in Exercise 17.5.