```
public class WordCount {
  static List<String> text;
  static int numThreads = ...;
  ...
  public static void main(String[] args) {
    text = readFile("document.tex");
    List<List<String>> inputs = splitInputs(text, numThreads);
    MapReduce<List<String>, String, Long, Long> mapReduce = new MapReduce<>();
    mapReduce.setMapperSupplier(WordCount.Mapper::new);
    mapReduce.setReducerSupplier(WordCount.Reducer::new);
    mapReduce.setInput(inputs);
    Map<String, Long> map = mapReduce.call();
    displayOutput(map);
  }
  ...
  static class Mapper extends Mapper<List<String>, String, Long> {
    public Map<String, Long> compute() {
      Map<String, Long> map = new HashMap<>();
      for (String word : input) {
        map.merge(word, 1L, (x, y) -> x + y);
      }
      return map;
    }
  }
  static class Reducer extends Reducer<String, Long, Long> {
    public Long compute() {
      long count = 0;
      for (long c : valueList) {
        count += c;
      }
      return count;
    }
  }
}
```

FIGURE 17.6 A MapReduce-based WordCount application.