```
1  public class MapReduce<IN, K, V, OUT> implements Callable<Map<K, OUT>> {
2    private List<IN> inputList;
3    private Supplier<Mapper<IN, K, V>> mapperSupplier;
4    private Supplier<Reducer<K, V, OUT>> reducerSupplier;
5    private static ForkJoinPool pool;
6    public MapReduce() {
7      pool = new ForkJoinPool();
8      mapperSupplier = () -> {throw new UnsupportedOperationException("No mapper supplier");}
9      reducerSupplier = () -> {throw new UnsupportedOperationException('No reducer supplier");}
10   }
11   public Map<K, OUT> call() {
12     Set<Mapper<IN, K, V>> mappers = new HashSet<>();
13     for (IN input : inputList) {
14       Mapper<IN, K, V> mapper = mapperSupplier.get();
15       mapper.setInput(input);
16       pool.execute(mapper);
17       mappers.add(mapper);
18     }
19     Map<K, List<V>> mapResults = new HashMap<>();
20     for (Mapper<IN, K, V> mapper : mappers) {
21       Map<K, V> map = mapper.join();
22       for (K key : map.keySet()) {
23         mapResults.putIfAbsent(key, new LinkedList<>());
24         mapResults.get(key).add(map.get(key));
25       }
26     }
27     Map<K, Reducer<K, V, OUT>> reducers = new HashMap<>();
28     mapResults.forEach(
29             (k, v) -> {
30               Reducer< K, V, OUT> reducer = reducerSupplier.get();
31               reducer.setInput(k, v);
32               pool.execute(reducer);
33               reducers.put(k, reducer);
34             }
35     );
36     Map<K, OUT> result = new HashMap<>();;
37     reducers.forEach(
38             (key, reducer) -> {
39               result.put(key, reducer.join());
40             }
41     );
42     return result;
43   }
44   ...
45 }
```

FIGURE 17.8 The MapReduce implementation.