```
public class KMeans {
  static final int numClusters = ...;
  static final double EPSILON = 0.01;
  static List<Point> points;
  static Map<Integer, Point> centers;

  public static void main(String[] args) {
    points = readFile("cluster.dat");
    centers = Point.randomDistinctCenters(points);
    MapReduce<List<Point>, Integer, List<Point>, Point> mapReduce
            = new MapReduce<>();
    mapReduce.setMapperSupplier(KMeans.Mapper::new);
    mapReduce.setReducerSupplier(KMeans.Reducer::new);
    mapReduce.setInput(splitInput(points, numWorkerThreads));
    double convergence = 1.0;
    while (convergence > EPSILON) {
      Map<Integer, Point> newCenters = mapReduce.call();
      convergence = distance(centers, newCenters);
      centers = newCenters;
    }
    displayOutput(centers);
  }
  static class Mapper extends Mapper<List<Point>, Integer, List<Point>> {
    public Map<Integer, List<Point>> compute() {
      Map<Integer, List<Point>> map = new HashMap<>();
      for (Point point : input) {
        int myCenter = closestCenter(centers, point);
        map.putIfAbsent(myCenter, new LinkedList<>());
        map.get(myCenter).add(point);
      }
      return map;
    }
  }
  static class Reducer extends Reducer<Integer, List<Point>, Point> {
    public Point compute() {
      List<Point> cluster = new LinkedList<>();
      for (List<Point> list : valueList) {
        cluster.addAll(list);
      }
      return Point.barycenter(cluster);
    }
  }
}
```

FIGURE 17.7 A MapReduce-based KMeans application.