## 구현 #1

|  |
| --- |
| package net.skhu.scanner.up;  import java.io.IOException;  import java.nio.file.Paths;  import java.util.ArrayList;  import java.util.Collections;  import java.util.Comparator;  import java.util.List;  import java.util.Map;  import java.util.Scanner;  import java.util.TreeMap;  class DataItem {  String s;  int count;  public DataItem(String s, int count) {  this.s = s;  this.count = count;  }  }  // ORDER BY count DESC, s  class DataItemComparator implements Comparator<DataItem> {  @Override  public int compare(DataItem o1, DataItem o2) {  if (o1 == null && o2 == null) return 0;  else if (o1 == null) return -1;  else if (o2 == null) return 1;  else {  int r = o1.count - o2.count;  if (r != 0) return r;  return o1.s.compareTo(o2.s);  }  }  }  public class Example3 {  public static void main(String[] args) throws IOException {  Map<String,Integer> map = new TreeMap<>();  String filePath = "e:/a.java";  Scanner scanner = new Scanner(Paths.get(filePath));  scanner.useDelimiter("[^a-zA-Z]+");  while (scanner.hasNext("[a-zA-Z]+")) {  String s = scanner.next();  Integer count = map.get(s);  if (count == null) count = 0;  count = count + 1;  map.put(s, count);  }  scanner.close();  List<DataItem> list = new ArrayList<>();  for (String s : map.keySet())  list.add(new DataItem(s, map.get(s)));  Collections.sort(list, new DataItemComparator());  for (DataItem d : list)  System.out.printf("%s %d\n", d.s, d.count);  }  } |

## 구현 #2 - lambda expression

|  |
| --- |
| package net.skhu.scanner.up2;  import java.io.IOException;  import java.nio.file.Paths;  import java.util.ArrayList;  import java.util.Collections;  import java.util.List;  import java.util.Map;  import java.util.Scanner;  import java.util.TreeMap;  class DataItem {  String s;  int count;  public DataItem(String s, int count) {  this.s = s;  this.count = count;  }  }  public class Example3 {  public static void main(String[] args) throws IOException {  Map<String,Integer> map = new TreeMap<>();  String filePath = "e:/a.java";  Scanner scanner = new Scanner(Paths.get(filePath));  scanner.useDelimiter("[^a-zA-Z]+");  while (scanner.hasNext("[a-zA-Z]+")) {  String s = scanner.next();  Integer count = map.get(s);  if (count == null) count = 0;  count = count + 1;  map.put(s, count);  }  scanner.close();  List<DataItem> list = new ArrayList<>();  for (String s : map.keySet())  list.add(new DataItem(s, map.get(s)));  // ORDER BY count DESC, s  Collections.sort(list, (o1, o2) -> {  if (o1 == null && o2 == null) return 0;  else if (o1 == null) return -1;  else if (o2 == null) return 1;  else {  int r = o2.count - o1.count;  if (r != 0) return r;  return o1.s.compareTo(o2.s);  }  });  for (DataItem d : list)  System.out.printf("%s %d\n", d.s, d.count);  }  } |

## 구현 #3 - stream API

|  |
| --- |
| package net.skhu.scanner.up3;  import java.io.IOException;  import java.nio.file.Paths;  import java.util.Collections;  import java.util.List;  import java.util.Map;  import java.util.Scanner;  import java.util.TreeMap;  import java.util.stream.Collectors;  class DataItem {  String s;  int count;  public DataItem(String s, int count) {  this.s = s;  this.count = count;  }  }  public class Example3 {  public static void main(String[] args) throws IOException {  Map<String,Integer> map = new TreeMap<>();  String filePath = "e:/a.java";  Scanner scanner = new Scanner(Paths.get(filePath));  scanner.useDelimiter("[^a-zA-Z]+");  while (scanner.hasNext()) {  String s = scanner.next();  Integer count = map.get(s);  if (count == null) count = 0;  count = count + 1;  map.put(s, count);  }  scanner.close();  List<DataItem> list = map.keySet().stream()  .map(s -> new DataItem(s, map.get(s)))  .collect(Collectors.toList());  // ORDER BY count DESC, s  Collections.sort(list, (o1, o2) -> {  if (o1 == null && o2 == null) return 0;  else if (o1 == null) return -1;  else if (o2 == null) return 1;  else {  int r = o2.count - o1.count;  if (r != 0) return r;  return o1.s.compareTo(o2.s);  }  });  list.stream().forEach(d -> System.out.printf("%s %d\n", d.s, d.count));  }  } |