实验内容

- 计划课时：1.5*45
- List 与 Iterarot 接口
- Map 与 HashMap
- Set、HashSet 与 TreeSet
- 集合框架与 Collections
- 必须完成的题目：1、2、3、4(不包含里面的进阶或者选做内容)
- 加分考核点说明：基本加分项(画正字的一划)与特别加分项(星号)，如无标识特别加分项均为基本加分项。

题目 1：List

1. 编写 NameDao 接口：

方法:

```
//将数组中的字符串放入列表，使用forEach循环(PTA)
public List<String> getNameFromArray(String[] names);

//在列表中搜索到与name相同的项目，就返回该name所在的序号，找不到返回-1。
public int searchName(List<String> nameList, String name);

//根据指定i，从列表中移除第i个项目，如果i指定错误(负数或者超出列表最大值)抛出IllegaArgumentException异常(暂略)，移除成功返回true(未学异常，暂时不用抛出异常)
public boolean removeFromList(List<String> nameList, int i);

//将nameList中所有值为name进行删除，并返回删除的个数。使用for(int i = 0….这种方式实现。加分点。(PTA)
public int deleteByName(List<String> nameList, String name);
/*测试说明：测试的nameList中包含10个”a”与10个”b”，观察返回结果是否符合预期。
改进：使用Iterator方式进行改进*/

//将传入的nameList转化为String类型数组
public String[] nameListtoStringArr(List<String> nameList);

//使用List的sort方法对nameList进行排序，然后使用List的forEach方法将其输出
public  void sortNameList(List<String> nameList);
```

编写 NameDaoImpl 实现 NameDao 接口。

2. 在 main 方法中编写相关代码进行测试
3. 选做加分-**单元测试**：使用 JUnit 进行测试
   - 主要讲解添加 JUnit library，@Test , @BeforeAll, @BeforeEach，

- 参考代码：JuintTest.java
- 考资料：[单元测试之 JUnit5 入门](https://blog.csdn.net/swordcenter/article/details/79279094)

4. 选做：使用 Queue 接口，ArrayDeque 或 LinkedList 模拟栈和队列的操作。

题目 2：Map 与 HashMap

1. 编写 Student 类，只有 String name 与 int age 两个属性和他们的 setter/getter 方法，覆盖其 hashCode 与 equals 方法。思考：编写 hashCode 方法与 equals 方法需要考虑什么原则？
2. 覆盖 equals 方法，当学生的 name 相同的时候 equals 为 true。相应的，覆盖 hashCode 方法。
3. 在 Util 类中编写方法

   public static List<Student> makeStudentList(int n)

   随机生成 n 个学生的 List。

方法编写指导：

a.随机生成姓名:调用 Util.getRandomName (int min, int max)。

b.随机生成年龄:在 Util 类中编写方法：

public static int getRandomAge(int min, int max)

(如果 min 或者 max 为负数或者 min>max，抛出 IllegaArgument 异常，可以复用 util.getRandomNumber (int min,int max)函数, (未学异常，暂时不用抛出异常))

c.创建学生对象，放入 List，然后返回这个 List。

4. 在 Util 类中编写方法

   public static Map<String, Student> makeStudentMap(List<Student> stuList)

   将 List 中每个 Student 的 name 作为 key, Student 对象本身作为 value 放入 Map 中，并返回该 List。

5. 在 Util 类中编写方法

   public static Student getStudentFromList(List< Student> stuList, String name)

   根据指定的 name 返回对应的学生，如果没有找到返回 null

6. 在 Util 类中编写方法

   public static Student getStudentFromMap(Map<String,Student> stuMap, String name)

   根据指定的 name 返回对应的学生，如果没有找到返回 null。

7. 加分：对比测试：

   在初始化阶段调用 makeStudentList 方法生成 stuList 列表，里边包含 1kw 个 Student。

   在初始化阶段调用 makeStudentMap 将生成的 stuList 转化为 stuMap 映射表。

   对 stuList 调用 getStudentFromList,统计执行时间。

   对 stuMap 调用 getStudentFromMap，统计执行时间。

思考：步骤 6 中，哪个方法搜索速度快？为什么？

8. 遍历输出 StudentMap 前 10 个的 key，前 10 个 value，前 10 个 entry。

题目 3：Set、HashSet 与 TreeSet(基本加分)

1.题目 2 步骤 7 生成的 stuList 与 stuMap 的 size 各是多少？试分析为什么 size 不一样。
2.用一个集合存储所有不重名的字符串，如何实现？
3.改写 Student 类，让其实现 Comparable 接口，实现对 name 的比较。生成几个 Student 对象，将其放入 TreeSet。然后打印出 TreeSet 中的内容。
4.为 Student 类编写 NameAgeComparator，先对 name 比较再对 age 进行比较。生成几个 Student 对象，将其放入一 TreeSet，并为该 TreeSet 指定 NameAgeComparator 比较器。然后打印出 TreeSet 中的内容。与 3 比较，看看有什么不同？如何使用 Lambda 表达式进行改写？
5.修改 Student 类的 hashCode 方法，让其直接返回-1。生成几个 Student 对象，然后放入 HashMap 或者 HashSet，再查看集合中的元素，看起 size 是多少。回答：为什么？

题目 4：Collections(基本加分)

1. 改写 Student 类，增加 int grade 属性。
2. 随机生成 100 个 Student 对象放入，students 数组。
3. 将 students 数组转换成 studentList(List 类型)
4. 使用 Collections.sort 对 students 的成绩进行排序。
5. 使用 Collections 的 min 和 main 方法找到成绩最高的和成绩最低学生。
6. 使用 Collections 的 binarySearch 查找成绩等于 60 的学生。
7. 编写方法从 studentList 找出 grade 小于 60 的学生，放入 otherList。
8. 使用 List 的 removeAll 方法，从 studentList 中移除 otherList。
9. 使用 Collections 的 shuffle 方法，打乱 studentList 中学生排列的顺序。
10. 将 studentList 转换为数组。

题目 5：倒排索引(特别加分)(PTA)

倒排索引（Inverted index），也常被称为反向索引，是一种索引方法，用来存储某个单词存在于哪些文档之中。是信息检索系统中最常
用的数据结构。通过倒排索引，可以根据单词快速获取包含这个单词的文档列表。

本作业主要完成以下四个功能：

(1). 建立索引 createIndex：首先输入 100 行字符串，用于构建倒排索引，每行字符串由若干不含标点符号的、全部小写字母组成的单词构成，每个单词之间以空格分隔。依次读入每个单词，并组成一个由<单词, 每个单词出现的行号集合>构成的 **HashMap**，其中行号从 1 开始计数。其中“每个单词出现的行号集合”可使用 **HashSet** 存储。

(2). 打印索引 printIndex：按照字母表顺序依次输出每个单词及其出现的位置，每个单词出现的位置则按行号升序输出。例如，如果“created”出现在第 3, 20 行，“dead”分别出现

在 14, 20, 22 行。则输出结果如下（冒号和逗号后面都有一个空格，行号不重复）：

…
created: 3, 20
dead: 14, 20, 22
…

(3). 检索 queryIndex：接下来输入查询(Query)字符串，每行包含一个查询，每个查询由若干关键字(Keywords)组成，每个关键字用空格分隔且全部为小写字母单词。要求输出包含全部单词行的行号（升序排列），每个查询输出一行。若某一关键字在全部行中从没出现过或没有一行字符串包含全部关键字，则输出“None”。遇到空行表示查询输入结束。如对于上面创建的索引，当查询为“created”时，输出为“3, 20”；当查询为“created dead”时，输出为“20”；当查询为“abcde dead”时，输出为“None”；

实验说明：可以使用“倒排索引测试数据.txt”文件中的内容进行测试，文件读写方法参见实验一中“Scanner 类的用法.txt”。如果不会读取文件中的内容，也可使用 ArrayList<String> 来存储自己输入的测试数据。
程序测试要求请参考“倒排索引说明.txt”。

题目 6：逆向最大匹配分词算法(特别加分)

实现逆向最大匹配分词算法，即从右向左扫描，找到最长的词并切分。如句子“研究生命的起源”，逆向最大匹配分词算法的输出结果为“研究 生命 的 起源”。

实验说明：
输入格式:
第一行是词表，每个词之间以空格分隔。
接下来是若干行中文句子。

输出格式：
逆向最大匹配的分词结果，每个词之间使用空格分隔。每个输入对应一行输出。

输入样例：
词表：你 我 他 爱 北京 天安门 研究 研究生 命 生命 的 起源
句子 1:研究生命的起源
句子 2:我爱北京天安门
句子 3:好朋友

输出样例：
句子 1 分词结果：研究 生命 的 起源
句子 2 分词结果：我 爱 北京 天安门
句子 3 分词结果：好 朋 友

提示：
使用 HashSet 存储词表

题目 7:无重复范围固定的数据集排序
BitSet