JAVA：作业4

刘泓尊 2018011446 计84 liu-hz18@mails.tsinghua.edu.cn

1.文本统计（2）

为了维护单一性与输入顺序，使用`LinkedHashSet<String>`即可满足需求。输出时先输出`size`再遍历输出内容即可。

```
private LinkedHashSet<String> mLineSet = new LinkedHashSet<>();
public void addLine(final String line) {
    mLineSet.add(line);
}
public final void printInfo() {
    System.out.println(mLineSet.size());
    for(String line: mLineSet) {
        System.out.println(line);
    }
}
```

2.小明的农场计划

使用`TreeMap<Integer, Integer>`来维护小明拥有的贝壳数。键为贝壳面额，值为贝壳数量。

当遇到买方优势时，只需要使用`floorEntry`方法便可以获知是否有满足条件的贝壳。同理，买方劣势只需要使用`ceilingEntry`方法。

当不存在满足条件的贝壳时，只需要抛出`NullPointerException`异常，并由`main`函数输出`-1`即可。

以**买方优势**为例：

```
private void buyInAdvantage(int value) throws NullPointerException {
Entry<Integer, Integer> e = tmap.floorEntry(value); //or `ceilingEntry(value)`
for DisAdvantage
    if (e == null) {
        throw new NullPointerException();
    }
    update(e);
}
```

3.城市规划

本质上这是一个无向图的连通性判定问题。

为了使用OOP的风格，图的存储使用邻接链表，封装了`Node`类，成员包括节点访问状态和邻居节点列表（以构成邻接表）。

```
class Node {
    public int state;
    public ArrayList<Integer> neighbors = new ArrayList<Integer>();
    public Node () {
        state = 0;
    }
    public void addNeighbor(int n) {
        neighbors.add(n);
    }
}
```

封装了`Graph`类，提供了`addEdge()`方法，向邻接链表加入对应的信息。同时避免自环。

```
public void addEdge (int from, int to) {
    if (from != to) {
        from --;
        to --;
        nodes[from].addNeighbor(to);
        nodes[to].addNeighbor(from);
    }
}
```

最终连通性的判定使用`bfs`，并维护了节点访问计数器`visited`。

当`bfs`结束时，若`visited == 节点总数`， 说明图的连通的。

`bfs`的辅助队列使用`ArrayDeque<Integer>`结构。

`bfs`的过程如下：

```
//bfs
int visited = 0;
ArrayDeque<Integer> queue = new ArrayDeque<>();
nodes[source].state = DISCOVERED;
queue.add(source);
while (!queue.isEmpty()) {
    int v = queue.poll();
    for (int u: nodes[v].neighbors) {
        if (nodes[u].state == UNDISCOVERED) {
            nodes[u].state = DISCOVERED;
            queue.add(u);
        }
    }
    nodes[v].state = VISITED;  //设置节点访问信息
    visited++; //计数器++
}
if (visited == this.n) {
    return true;
} else {
    return false;
}
```