Problems

1. `+=, StringBuilder, StringBuffer`. 使用 += 会生成新的字符串实例，在循环中使用 += 会加大性能开销，尤其是字符串较大时。StringBuilder 在添加新子字符串时不会创建新字符串。StringBuilder 和 StringBuffer 的区别同学们可以进行探索。
2. **deleteMin**。当删除一个元素对 heap 进行处理时，**如果 start index == 0，那么 2 * index + 1 才是左子元素，2 * index 是自己本身**。

```
int hole = 0;
int child;
// 注意，这里栈顶索引是 0，所以 hole * 2 指向自身，hole * 2 + 1 才是指向左子元素。
for(; hole * 2 <= currentSize; hole = child){
    child = hole * 2;
    // ...
}
```

3. **Debug**。
    - **sout**。即使用 `System.out.println` 方法查看是否存在死循环、分支代码是否能够运行等。
    - **breakpoint**。使用 IDE 提供的常规断点，条件断点，异常断点等，配合程序中的变量值的变化，快速定位 bug。
4. **Submit**。提交时只提交 **src 目录中的代码**即可，*.iml，out，test 等配置文件和中间结果文件（夹）、测试文件等不必提交。
5. **modCount**。继承抽象类 AbstractList 的类 （ArrayList 等）会保持 modCount 属性，在使用 **forEach** 遍历方式时，modCount 不能被修改（增删等），否则会抛出异常。

String	StringBuffer	StringBuilder
String的值是不可变的，这就导致每次对String的操作都会生成新的String对象，不仅效率低下，而且浪费大量优先的内存空间	StringBuffer是可变类，和线程安全的字符串操作类，任何对它指向的字符串的操作都不会产生新的对象。每个StringBuffer对象都有一定的缓冲区容量，当字符串大小没有超过容量时，不会分配新的容量，当字符串大小超过容量时，会自动增加容量	可变类，速度更快
不可变	可变	可变
	线程安全	线程不安全
	多线程操作字符串	单线程操作字符串