后端

Java09

task0：了解Stream

新东西有点多，**这些不算是答题了，当成笔记记的**，字会很多：

笔记part：

1. 有关Sream的使用：在目前看到的所有代码中，stream可附带于List的，Stream类也可以自己定义。

```
List<Integer> motherstream=List.of(1,2,3,4,5,6);
List<Integer> sonList=mother.stream().filter(n-
>n%2==1).collect(Collectors.toList());
```

2. stream中的方法filter：筛选器，在括号里进行语句的设计来筛选出流中符合条件的数据（其中可以任意设置，代码中是“n”来表示流中元素，在->后跟上判断语句）
3. stream中的collect：筛选后收集，很合理，此方法呢会将filter中筛选出的元素重新收集到一个List或者Set中，这两者分别对应在括号里使用Collectors.toList()he1Collectors.toSet()方法。
4. stram中forEach：在括号里定义对每个元素的处理，若是改变数值的处理会改变原流中的数据。那么有没有能只是改变输出不改变原本性质的写法呢？尝试一下。嗷嗷，这是map方法。

```
List<Integer>nums=List.of(1,2,3,4,5,6);
nums.stream().forEach(n->n=n*2);
```

5. stream中的map：到这里应该只看代码也能理解了

```
List<Integer>testone=mother.stream().map(n-
>n*n).collect(Collectors.toList());
System.out.println(testone);
```

6. stream中的limit：用于获取数量确定的流（缺氧.液体流量计）很好理解

```
nums.stream().limit(4).filter(.....);
```

7. stream中的sortrd:对流进行排序

```
nums.stream().limit(10).sorted().foreach(n->.....)
```

8. parallel:文档里看不出个所以然，我们询问ai得知这是多线程处理stream面对大量数据可以提高效率，但要看情况使用。

OK这样写下来我兴许明白了，我们开始答题。

---

task1：流API

懒得到现在才去查到API就是接口的意思

得到了这个（题目里这个“它没有实现Stream”耐人寻味啊……）：

```
limit = java.util.stream.SliceOps$1@27973e9b
```

…结果是输出了stream对象本身吧，而不是其中的东西

再回到题本身：不难理解是要让我们用Collection接口里的方法来返回这个limit

修改代码如下（会传一个整合的代码到github）：

```
Collection<String> a=new ArrayList<>();
    a=List.of("I", "am", "a", "list", "of", "Strings");
    a.stream().limit(4).forEach(n-> System.out.println(n));
```

但这不只是相当于把Collection当成List用吗？

```
java.util.stream.ReferencePipeline$Head@52cc8049
limit = java.util.stream.SliceOps$1@312b1dae
I
am
a
list
```

如何实现管道操作堆叠笔记part：

1. 实现的基本逻辑：内部迭代，Stream中的方法都会对Stream中的元素进行操作，然后返回一个新的Stream，这个Stream则就可以继续调用其它的方法。

总结规律：我们对于一个实现了Stream的类（List，Collection）可以调用各种中间操作和终端操作（forEach，collect，reduce等），其中中间操作后可以继续进行操作，而终端操作则不会再连接其他操作。

Lambda表达式笔记part：

1. 背景：lambda表达式出现是为了代替函数式接口中需要的匿名内部类，从而简化代码
2. 匿名内部类：类似comparator中的compare？
3. 一种理解方式：就是一个没有名字的函数（输入参数）->(内部语句)有几种写法（实际是函数式接口的简写，只能有一个抽象方法）

   详解Java中的Lambda表达式

   Java Lambda 表达式

进阶挑战——应用流API

感觉比较简单，没遇到什么问题，咱就简单点写。（代码已上传github）

---

task2：串行化

笔记part：

1. 控制台的输入输出：用BufferedReader来读取

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
br.read();//读取字符
br.readline();//读取字符串
```

2. 从文件输入：（FileInputStream）有两种创建InputStream定义方式

```
InputStream f=new FileInputStream("文件地址");
//or
File f=new File("文件地址");
InputStream intput=new FileInputStream;
```

（fileinputstream应该是inputstream的一个具体的子类）

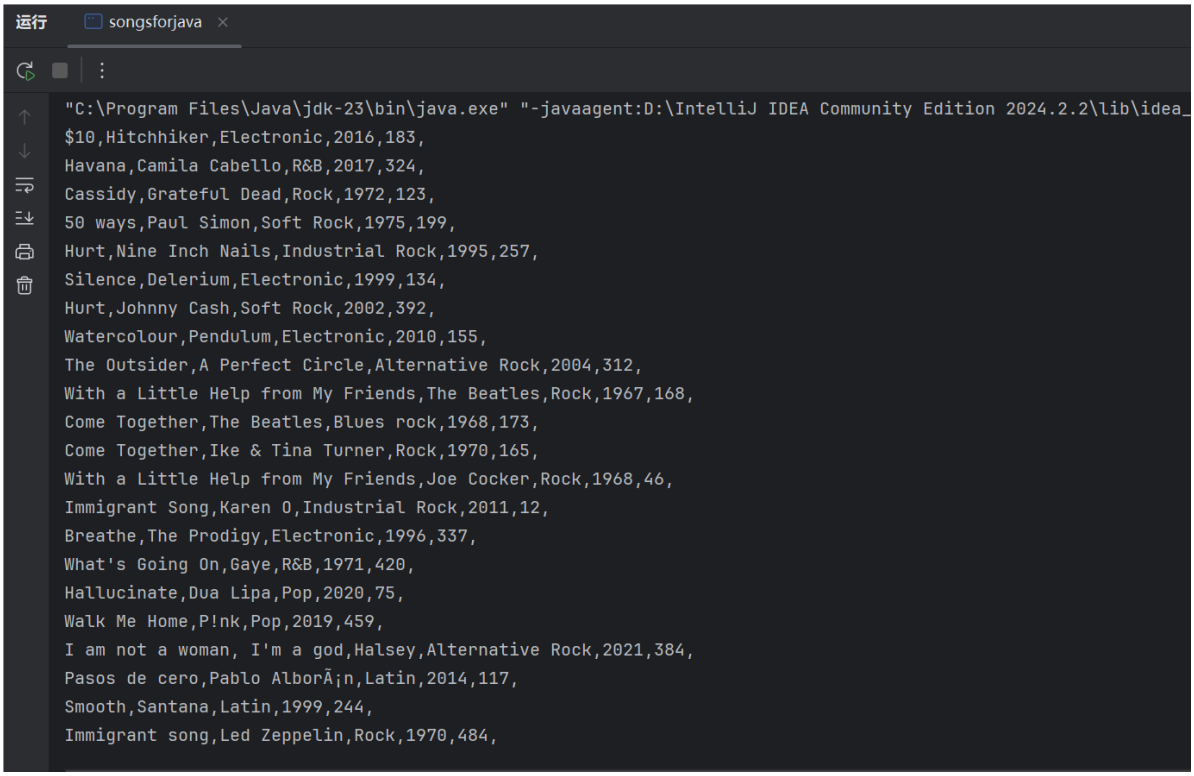
创建完对象后，也有一些方法来进行输入：close()，finalize(),read(int r),read(byte[]r),available()

3. 创建文件并向文件输出：（FileOutputStream）创建方式于输入类似。

其实过程中没有遇到什么太大的障碍，有几个小问题：

- int类型转byte
- 读取时的换行规则
- 读取时每个成员变量的区分

不过都是很简单的问题，想一些办法就解决了。



进阶挑战——文件I/O

emm....感觉上来说这个和写对象差不多，直接干！

是我理解错了吗，为什么感觉简单的有点过了？输出和字符串那种是一样的就不再贴一遍了

（不好意思，代码估计不是很整齐）