Java-08

一、Task1.异常

1. **Error与Exception的区别**

	Exception	Error
本质	可以处理的意外情况	无法处理的严重问题
来源	程序逻辑或外部环境错误	JVM或系统底层错误
处理方式	可通过`try-catch`捕获或`throws`声明	无法处理，因此无需捕获
对程序的影响	处理后可继续执行	程序终止

2. **两种异常**
    - **Checked Exception**
        - 在编译期就需要处理的异常，否则编译不通过

          通过 **try-catch** 块进行捕获并处理

          或者通过 **throws** 声明异常

          (注意:子类方法声明中的 throws 子句不能出现父类对应方法的 throws 子句中没有的异常类型)
        - **原因: 外部因素**：文件、网络、数据库等问题
        - **类型:** 除了 **RuntimeException** 及其子类之外的所有 **Exception** 的子类
    - **Unchecked Exception**
        - 编译器不要求强制处理的异常，可以编译通过
        - **原因:** 程序逻辑错误引起的

          如 访问null对象的成员，数组越界，错误的类型转换

          或 给方法传递了不合法的参数
        - **类型: RuntimeException** 及其子类

二、Task2.处理

定义要求的异常后读取文件并 **try-catch** 分类处理即可

```
import java.io.*;

//先定义要求的异常类
class FileNotFoundException extends Exception {
    public FileNotFoundException() {
        super();
    }
    public FileNotFoundException(String message) {
        super(message);
    }
}
```

```
class EmptyFileException extends Exception {
    public EmptyFileException() {
        super();
    }
    public EmptyFileException(String message) {
        super(message);
    }
}

public class Readfiles {
    static void readFile(String path) throws
FileNotFoundException,EmptyFileException,IOException {
        File file = new File(path);
        //判断文件是否存在
        if (!file.exists()) {
            throw new FileNotFoundException("未找到文件:"+path);
        }
        int sum = 0;
        int quantity = 0;
        //使用try-with-resources读取，无需手动关闭
        try (BufferedReader br = new BufferedReader(new FileReader(path))) {
            String line;
            while ((line = br.readLine()) != null) {
                try {
                    sum += Integer.parseInt(line);
                    quantity++;
                } catch (NumberFormatException e) {
                    //判断是否存在非整数
                    System.out.println("读取数据中存在非整数数据"+line);
                }
            }
            if (quantity == 0) {
                //判断文件是否为空
                throw new EmptyFileException("文件为空");
            }
        }
        double result = (sum)/quantity;
        System.out.println("平均值为:"+result);
    }

    public static void main(String[] args) {
        try {

 readFile("F:\\Idea\\Code\\Test\\Zhaoxing\\src\\cn\\org\\glimmer\\Java08\\Read.t
xt");
        } catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
        } catch (EmptyFileException e) {
            System.out.println(e.getMessage());
        }
        //其他情况
        catch (IOException e) {
            System.out.println("读取文件发生错误");
        }
    }
}
```

输入:
```
a
45
3
4
```
输出:
```
读取数据中存在非整数数据a
平均值为:17.0
```

输入:
```
1
```
输出:
```
文件为空
```

找不到文件时:
```
未找到文件:F:\Idea\Code\Test\Zhaoxing\src\cn\org\glimmer\Java08\Reaad.txt
```

三、Task3.Stream流

1. 输出结果:

```
limit = java.util.stream.SliceOps$1@3f99bd52
```

原因:直接打印流对象并不会输出流中的内容

因为Stream并不是一个集合，它并不存储元素，而是对元素进行各种计算操作

打印实际上是调用了对应的**Object类的toString方法** 返回对象的**类名**和**哈希码**

2. 代码修改如下:

```java
import java.util.*;
import java.util.stream.Collectors;

public class Main {

    public static void main(String[] args) {
        // 测试数据：学生列表
        List<Student> students = Arrays.asList(
                new Student("Alice", 85),
                new Student("Bob", 58),
                new Student("Charlie", 90),
                new Student("David", 45),
                new Student("Eve", 72),
                new Student("Frank", 60),
                new Student("Grace", 55),
                new Student("Heidi", 95)
        );

        // 请在这里补充代码，完成以下任务：
        // 1. 过滤分数≥60的学生
        // 2. 姓名转换成大写
        // 3. 按姓名字母顺序排序
        // 4. 收集成 List<String> 返回并打印

        // --- 你的代码开始 ---

        List<String> passingStudents = students.stream()
                .filter(a -> a.score >=60)
                .sorted(Comparator.comparing(Student::getName))
                .map(a -> a.name.toUpperCase())
```

```
                .collect(Collectors.toList());


        // TODO: 补充流操作链

        // --- 你的代码结束 ---

        // 打印结果
        System.out.println(passingStudents);
    }
}
```