Java08-异常处理与Stream流

国庆节快乐!!!

Task1

Q1. 请你简单概述一下Error与Exception的区别 当发生Exception和Error时 程序的处理态度分别应该是什么?
区别:
表示程序可预期、可处理的异常，通常由代码或外部环境引发，可以通过 try-catch 捕获并恢复。

Error 表示严重错误，通常是系统级或虚拟机层面的问题，如内存溢出，无法通过代码处理，程序一般会终止。
态度:
Exception：应主动捕获并合理处理，保证程序健壮性和用户体验。 Error：通常不捕获，发生时让程序终止，需通过优化代码或环境避免。
Q2. 你知道这两种异常有什么区别吗 他们发生的原因分别是什么?
区别： Unchecked Exception 编译器不会强制要求处理，可以选择捕获或不处理。
通常是程序逻辑错误或开发者疏忽，如空指针、数组越界、类型转换错误等。 Checked Exception
编译器强制要求处理，必须用 try-catch 或 throws 显式声明。
多为外部环境或资源相关问题，如文件未找到、网络异常、数据库操作失败等，程序无法完全预知和避免。

Task2

Q3. 题目: 文件读取和数据处理

代码详情请见EighthQ3文件夹

下面给出运行代码:

```java
public class Average {
    Run | Debug
    public static void main(String[] args) throws Exception {
        Reader reader = null;
        BufferedReader bufferReader = null;
        try {

            File file = new File(pathname:"C:\\Users\\29867\\Desktop\\daima\\EighthQ3\\data.txt");
            if (!file.exists()) {
                throw new FileNotFoundException(message:"File not found");
            }
            reader = new FileReader(fileName:"C:\\Users\\29867\\Desktop\\daima\\EighthQ3\\data.txt");
            bufferReader = new BufferedReader(reader);
            Double sum = 0.0;
            int count = 0;
            String line;
            while ((line = bufferReader.readLine()) != null) {
                Integer num;
                try {
                    num = Integer.parseInt(line);
                    sum = sum + num;
                } catch (Exception e) {
                    throw new NumberFormatException("Invalid number format: " + line);
                }
                count++;
            }
            if (count == 0) {
                throw new EmptyFileException(message:"File is empty");
            }
            System.out.println("Average is:" + (sum / count));
        } finally {
            if (bufferReader != null) {
                bufferReader.close();
            }
            if (reader != null) {
                reader.close();
            }
        }
    }
}
```

Task4

Q4. 尝试执行上面代码 会得到什么结果? 这和你预想的一样吗? 为什么会出现这种结果?

会得到类型和一串哈希码

与我想的不一样

应为没有使用stream流的终止方法,limit返回的仍然是stream流

Q5. 完成下述题目

```
//定义一个学生类
java
public class Student {
    String name;
    int score;

    public Student(String name, int score) {
        this.name = name;
```

```
        this.score = score;
    }

    public String getName() {
        return name;
    }

    public int getScore() {
        return score;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setScore(int score) {
        this.score = score;
    }
}
```

改后代码

```
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(student -> student.getScore() >= 60) // 过滤分数≥60的学生
        .map(student -> student.getName().toUpperCase()) // 姓名转换成大写
        .sorted()//排序
        .toList(); // 收集返回

        // TODO: 补充流操作链
```

```
        // --- 你的代码结束 ---

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