Q1

Error与Exception的区别:

Error：
表示严重的系统级错误，通常由JVM抛出
多数情况下程序无法恢复，如 OutOfMemoryError、StackOverflowError
属于Throwable的子类，但不建议应用程序捕获处理
Exception：
表示程序运行过程中可以预期的异常情况
可分为检查异常(checked)和非检查异常(unchecked)
应用程序可以通过捕获和处理Exception来恢复执行

处理态度:

对于Exception：
应该积极捕获并处理
根据业务需求选择合适的处理方式（恢复、重试、记录日志等）
可以通过try-catch块进行异常处理
对于Error：
通常不应该尝试捕获或处理
发生Error时应让程序终止执行
重点应放在预防Error的发生，如优化内存使用、避免无限递归等

q2

Checked异常（检查异常）：

继承自 Exception 但不继承 RuntimeException
编译器强制要求处理，要么用 try-catch 捕获，要么用 throws 声明抛出
通常表示可预见的外部错误情况

Unchecked异常（运行时异常）：

继承自 RuntimeException
编译器不强制要求处理，可选择性捕获
通常表示程序逻辑错误

发生原因

Checked异常发生原因：

外部资源问题（如 IOException 文件不存在、网络连接失败）
环境因素（如 SQLException 数据库连接问题）

可预见到的异常情况，需要强制处理

Unchecked异常发生原因：

程序逻辑错误（如 NullPointerException 空指针引用）
编程失误（如 ArrayIndexOutOfBoundsException 数组越界）
违反约定条件（如 IllegalArgumentException 参数不合法）

Q3

自定义异常FileNotFoundException类：

```
1. public class FileNotFoundException extends RuntimeException {
2. public FileNotFoundException(String message) {
3. super(message);
4. }
5. }
```

自定义异常EmptyFileException:

```
1. public class EmptyFileException extends RuntimeException {
2. public EmptyFileException(String message) {
3. super(message);
4. }
5. }
```

主要程序：

```
1. import java.io.*;
2. import java.lang.NumberFormatException;
3. import java.util.ArrayList;
4.
5. public class exception01   {
6.     public static void main (String[] args) {
7.         String filepath="data.txt";
8.         try {
9.            processfile(filepath);
10.         } catch (Exception e) {
11.             System.err.println("程序执行出错: " + e.getMessage());
12.         }
13.     }
14.
15.     public static void processfile (String filepath) throws EmptyFileException
    ,FileNotFoundException{
16.     BufferedReader bis= null;
```

```
17.
18.     String line;
19.     ArrayList<Integer> numbers = new ArrayList<>();
20.     try {
21.         bis=new BufferedReader(new FileReader(filepath));
22.         long sum=0;
23.         while((line=bis.readLine())!=null){
24.             try {
25.                 int number=Integer.parseInt(line);
26.                 numbers.add(number);
27.                 sum+=number;
28.             } catch (NumberFormatException e) {
29.                 System.err.println("格式错误: 无法解析内容 " );
30.                 return;
31.             }
32.         }
33.         // 检查文件是否为空
34.         if (numbers.isEmpty()) {
35.             throw new EmptyFileException("文件为空");
36.         }
37.
38.         double average=(double) sum / numbers.size();
39.         System.out.println("读取到 " + numbers.size() + " 个整数");
40.         System.out.println("这些整数的平均值是: " + average);
41.     }
42.     catch (FileNotFoundException e) {
43.         System.err.println("文件未找到: " + filepath);}
44.     catch (IOException e) {
45.         System.err.println("文件读取错误: " + e.getMessage());
46.     }
47.     finally {
48.         try {
49.             if (bis != null) {
50.                 bis.close();
51.             }
52.         } catch (IOException e) {
53.             System.out.println("文件关闭失败");
54.         }
55.     }
56. }
57. }
```

Q4

结果说明:

得到的是一个 Stream 对象的内存地址引用，和我预期的字符串内容不一样

原因分析：

这是因为直接打印 Stream 对象只会显示其内部实现类的引用信息

Q5

```java
import java.util.Arrays;
import java.util.List;
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(s-> s.getScore()>=60)
                .map(s -> s.getName().toUpperCase())
                .sorted().
                collect(Collectors.toList());
        // TODO: 补充流操作链
        // --- 你的代码结束 ---

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

```
39. }
```