IO 流作业

一、填空题

1. IO 的含义是 输入输出流 data source 的含义是 数据源_____。
2. Java IO 流可以分为节点流和处理流两大类，其中前者处于 IO 操作的第一线，所有操作必须通过他们进行。
3. 输入流的唯一目的是提供通往数据的通道，程序可以通过这个通道读取数据，read 方法给程序提供了一个从输入流中读取数据的基本方法。
4. read 方法从输入流中顺序读取源中的单个字节数据，该方法返回字节值(0-255 之间的一个整数)，如果到达源的末尾，该方法返回-1 。
5. Java 系统的标准输入对象是 System.in，标准输出对象有两个，分别是标准输出 System.out 和标准错误输出__System.err_____。
6. Java IO 体系中，___ObjectInputStream____是字节输入流，不仅提供了存取所有 Java 基础类型数据（如：int，double 等）和 String 的方法,也提供了提供存取对象的方法。
7. Java IO 体系中，____DataOutputStream_____是字节输出流，提供了可以存取所有 Java 基础类型数据（如：int，double 等）和 String 的方法,但没有提供存取对象的方法。
8. ____序列化___是指将 Java 对象转换成字节序列，从而可以保存到磁盘上，也可以在网络上传输，使得不同的计算机可以共享对象。
9. transient 的作用是_____标识一个成员变量在序列化子系统中应被忽略_____。

二、选择题

1. 使用 **Java IO** 流实现对文本文件的读写过程中，需要处理下列（B）异常。（选择一项）

A ClassNotFoundException

B IOException

C SQLException

D RemoteException

2. 在 **Java** 的 **IO** 操作中，（D）方法可以用来刷新流的缓冲。（选择两项）

A void release()

B void close()

C void remove()

D void flush()

3. 在 **Java** 中，下列关于读写文件的描述错误的是（B）。（选择一项）

A Reader 类的 read()方法用来从源中读取一个字符的数据

B Reader 类的 read(int n)方法用来从源中读取一个字符的数据

C Writer 类的 write(int n)方法用来向输出流写入单个字符

D Writer 类的 write(String str)方法用来向输出流写入一个字符串

4. 阅读下列文件定入的 **Java** 代码，共有（C）处错误。（选择一项） **import** java.io.*;

```
publicclass TestIO { publicstatic void
    main(String []args){
        String str =" 文 件 写 入 练 习 ";
        FileWriter fw = null;  //1
        try{ fw = new FileWriter("c:\mytext.txt"); //2
            fw.writerToEnd(str); //3
        }catch(IOException e){   //4
            e.printStackTrace();
        }finally{
            //此处省略关闭流
        }
    }
}
```

A 0

B. 1

C. 2

D. 3

5. 分析如下 **Java** 代码，有标注的四行代码中，有错误的是第（D）处。（选择一项）

```
import java.io.FileWriter; import
java.io.IOException; public class Test
{ public static void main(String[ ] args) {
        String str = "Hello World"; FileWriter fw =
        null;  try  {  fw  =  new
        FileWriter("c:\\hello.txt"); // 1 fw.write(str);
         // 2
```

```
        } catch (IOException e) {
            e.printStackTrace();                    // 3
        } finally {
            fw.close();                             // 4
        }
    }
}
```

A. 1

B. 2 **C.** 3

D. 4

6. 以下选项中关于如下代码的说法正确的是（AD）。（选择二项）

```
public class TestBuffered { public static void main(String[]
    args) throws IOException {
        BufferedReader br = new BufferedReader(new
            FileReader("d:/ccjr1.txt"));
        BufferedWriter bw = new BufferedWriter(new
            FileWriter("d:/ccjr2.txt"));
        String str =
        br.readLine();
        while(str !=null){ bw.wri
        te(str); bw.newLine();
        str = br.readLine();
        }
        br.close();
        bw.close()
        ;
    }
}
```

A. 该类使用字符流实现了文件复制，将 d:/ccjr1.txt 复制为 d:/ccjr2.txt

B. FileReader 和 FileWriter 是处理流，直接从文件读写数据

C. BufferedReader 和 BufferedWriter 是节点流，提供缓冲区功能，提高读写效率

D. readLine()可以读取一行数据，返回值是字符串类型，简化了操作

7. InputStreamReader 是转换流，可以将字节流转换成字符流，是字符流与字节流之间的桥梁。它的实现使用的设计模式是（C）。（选择一项）

A. 工厂模式

B. 装饰模式

C. 适配器模式

D. 代理模式

三、判断题

1. 假设文件"a.txt"的长度为 100 字节，那么当正常运行语句"OutputStream f=new FileOutputStream(new File("a.txt"));"之后，文件"a.txt"的长度变为 0 字节。（√）
2. ByteArrayInutStream 和 ByteArrayOutputStream 对内存中的字节数组进行读写操作，属于字节流，属于处理流而不是节点流。（×）
3. 实现 Serializable 接口的可以被序列化和反序列化。该接口中没有定义抽象方法，也没有定义常量。（√）
4. 序列化是指将字节序列转换成 Java 对象，只有实现了 Serializable 接口的类的对象才可以被序列化。（×）
5. 流对象使用完后，一般要调用 close 方法关闭，释放资源。（√）

四、简答题

1. 输入流和输出流的联系和区别，字符流和字节流的联系和区别

输入流是得到数据，输出流是输出数据。字符流和字节流是流的一种划分，按处理照流的数据单位进行的划分。两类都分为输入和输出操作。

在字节流中输出数据主要是使用 OutputStream 完成，输入使 InputStream

在字符流中输出主要是使用 Writer 类完成，输入流主要使用 Reader 类完成

2. 节点流和处理流的联系和区别。

节点流，处理流是流的另一种划分，按照功能不同进行的划分。节点流，

可以从或向一个特定的地方(节点)读写数据。处理流是对一个已存在的流的连接和封装，通过所封装的流的功能调用实现数据读写。如 BufferedReader。

3. 列举常用的字节输入流和字节输出流并说明其特点，至少 5 对。

FileInputStream 从文件系统中的某个文件中获得输入字节。

ByteArrayInputStream 包含一个内部缓冲区，该缓冲区包含从流中读取的字节。

FilterInputStream 包含其他一些输入流，它将这些流用作其基本数据源，它可以直接传输数据或提供一些额外的功能。

ObjectInputStream 对以前使用 ObjectOutputStream 写入的基本数据和对象进行反序列化。

StringBufferInputStream 此类允许应用程序创建输入流，在该流中读取的字节由字符串内容提供

4. 说明缓冲流的优点和原理。

不带缓冲的流的工作原理：它读取到一个字节/字符，就向用户指定的路径写出去，读一个写一个，所以就慢了。带缓冲的流的工作原理：读取到一个字节/字符，先不输出，等凑足了缓冲的最大容量后一次性写出去，从而提高了工作效率

优点：减少对硬盘的读取次数，降低对硬盘的损耗。

5. 序列化的定义、实现和注意事项。反序列化指的是什么?

序列化是指将 Java 对象转换为字节序列的过程，而反序列化则是将字节序列转换为 Java 对象的过程。

6. PrintStream 打印流经常用于什么情况? System.out 是不是打印流?

打印流是输出信息最方便的类，注意包含 PrintStream（字节打印流）和 PrintWriter（字符打印流）。打印流提供了非常方便的打印功能，可以打印任何类型的数据信息，System.out 是打印流

五、编码题

1. 实现字符串和字节数组之间的相互转换。必如将字符串“吉软 java”转换为字节数组，并将字节数组再转换回字符串。

```java
package Homeworks.HomeworkDay12;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/15
 */
0 个用法  新 *
public class Demo01 {
    0 个用法  新 *
    public void test1(){
        String str1 = "吉软java";
        byte[] bytes = str1.getBytes();
        String str2 = new String(bytes, offset: 0,bytes.length);
        System.out.println(str2);
    }
}
```

2. 实现字节数组和任何基本类型和引用类型执行的相互转换

提示：使用 ByteArrayInutStream 和 ByteArrayOutputStream。

```
package Homeworks.HomeworkDay12;

import java.io.*;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/15
 */
0 个用法  新 *
public class Demo02 {
    新 *
    public static void main(String[] args) throws IOException {
        int num1 = 50;
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(baos);
        oos.writeInt(num1);
        byte [] buf = baos.toByteArray();
        baos.close();

        ByteArrayInputStream bais = new ByteArrayInputStream(buf);
        ObjectInputStream ois = new ObjectInputStream(bais);
        int num2 = ois.readInt();
        System.out.println(num2);
        bais.close();
    }
}
```

3. 分别使用文件流和缓冲流复制一个长度大于 100MB 的视频文件，并观察效率的差异。

```
public class Demo03 {
    新 *
    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream(new File( pathname: "F:/1.mp4"));
        FileOutputStream fos = new FileOutputStream( name: "F:/2.mp4");

        byte [] buf = new byte[1024];
        int len = fis.read(buf);
        while(len!=-1){
            fos.write(buf,  off: 0, len);
            len = fis.read(buf);
        }

        fis.close();
        fos.close();
    }
}
```

```java
package Homeworks.HomeworkDay12;

import java.io.*;

/**
 * @author junhaocai
 * @email junhaocai01@gmail.com
 * @date 2023/1/15
 */
0 个用法  新 *
public class Demo03_2 {
    新 *
    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream(new File( pathname: "d:/1.mp4"));
        FileOutputStream fos = new FileOutputStream( name: "d:/2.mp4");
        BufferedInputStream bis = new BufferedInputStream(fis);
        BufferedOutputStream bos = new BufferedOutputStream(fos);

        byte [] buf = new byte[10];
        int len = bis.read(buf);
        while(len!=-1){
            bos.write(buf, off: 0, len);
            len = bis.read(buf);
        }
        bis.close();
        bos.close();
    }
}
```

4. 复制文件夹 d:/ccjrjava 下面所有文件和子文件夹内容到 d:/ccjrjava2。

提示：涉及单个文件复制、目录的创建、递归的使用

```java
public class Demo04 {
    1个用法  新 *
    public static void copyFile(File sourceFile, File targetFile) throws IOException {
        BufferedInputStream inBuff = null;
        BufferedOutputStream outBuff = null;
        try {
            inBuff = new BufferedInputStream(new FileInputStream(sourceFile));
            outBuff = new BufferedOutputStream(new FileOutputStream(targetFile));
            byte[] b = new byte[1024 * 5];
            int len;
            while ((len = inBuff.read(b)) != -1) {
                outBuff.write(b, off: 0, len);
            }
            outBuff.flush();
        } finally {
            if (inBuff != null)
                inBuff.close();
            if (outBuff != null)
                outBuff.close();
        }
    }
    public static void copyDirectiory(String sourceDir, String targetDir)
            throws IOException {
        File fSourceDir = new File(sourceDir);
        if(!fSourceDir.exists() || !fSourceDir.isDirectory()){
            System.out.println("源目录不存在");
            return;
        }
        File fTargetDir = new File(targetDir);
        if(!fTargetDir.exists()){
            fTargetDir.mkdirs();
        }
        File[] file = fSourceDir.listFiles();
        for (int i = 0; i < file.length; i++) {
            if (file[i].isFile()) {
                File sourceFile = file[i];
                File targetFile = new File(fTargetDir, file[i].getName());
                copyFile(sourceFile, targetFile);
            }
            if (file[i].isDirectory()) {
                String subSourceDir = sourceDir + File.separator + file[i].getName();
                String subTargetDir = targetDir + File.separator + file[i].getName();
                copyDirectiory(subSourceDir, subTargetDir);
            }
        }
    }
    新 *
    public static void main(String[] args) throws IOException {
        copyDirectiory( sourceDir: "d:/a.txt", targetDir: "d:/a.txt");
    }
}
```

5. 解释下面代码的含义：

```
File f = new File("d:/a.txt");
FileInputStream fis = new
FileInputStream(f); int m = 0;
while((m=fis.read())!=-1){ char
    c = (char) m;
    System.out.print(c); }
```

读取 d 盘中的 a.txt 中文件，用输入流将文本中的字符打印

```java
public class Demo05 {
    /**
     * 复制单个文件
     * @param sourceFile 源文件
     * @param targetFile 目标文件
     * @throws IOException
     */
    1个用法  新 *
    public static void copyFile(File sourceFile, File targetFile) throws IOException {
        BufferedInputStream inBuff = null;
        BufferedOutputStream outBuff = null;
        try {
            inBuff = new BufferedInputStream(new FileInputStream(sourceFile));
            outBuff = new BufferedOutputStream(new FileOutputStream(targetFile));
            byte[] b = new byte[1024 * 5];
            int len;
            while ((len = inBuff.read(b)) != -1) {
                outBuff.write(b,  off: 0, len);
            }
            outBuff.flush();
        } finally {
            if (inBuff != null) {
                inBuff.close();
            }
            if (outBuff != null) {
                outBuff.close();
            }
        }
    }
    2个用法  新 *
    public static void copyDirectiory(String sourceDir, String targetDir)
            throws IOException {
        File fSourceDir = new File(sourceDir);
        if(!fSourceDir.exists() || !fSourceDir.isDirectory()){
            System.out.println("源目录不存在");
            return;
        }
        File fTargetDir = new File(targetDir);
        if(!fTargetDir.exists()){
            fTargetDir.mkdirs();
        }
        File[] file = fSourceDir.listFiles();
        for (int i = 0; i < file.length; i++) {
            if (file[i].isFile()) {
                File sourceFile = file[i];
                File targetFile = new File(fTargetDir, file[i].getName());
                copyFile(sourceFile, targetFile);
            }
            if (file[i].isDirectory()) {
                String subSourceDir = sourceDir + File.separator + file[i].getName();
                String subTargetDir = targetDir + File.separator + file[i].getName();
                copyDirectiory(subSourceDir, subTargetDir);
            }
        }
    }
    新 *
    public static void main(String[] args) throws IOException {
        copyDirectiory( sourceDir: "d:/a.txt",  targetDir: "d:/a.txt");
    }
}
```

6. 可选题

1. 使用 IO 包中的类读取 D 盘上 exam.txt 文本文件的内容，每次读取一行内容，将每行作为一个输入放入 ArrayList 的泛型集合中并将集合中的内容使用加强 for 进行输出显示。

2. 假设从入学开始所有书写的 Java 类代码都在 d:/ccjrjava 文件夹下，包括多级子文件夹。

   使用 IO 流获取从入学开始，到目前为止已经写了多少行 Java 代码。

   提示：其实就是获取 d:/ccjrjava 文件夹及其子文件夹下的所有.java 文件，使用 readLine() 读取其中每一行，每读取一行，行数加 1。所有的文件读取完毕，得到总共已经写的 Java 代码行数。需要结合递归实现

3. 由控制台按照固定格式输入学生信息，包括学号，姓名，年龄信息，当输入的内容为 exit 退出；将输入的学生信息分别封装到一个 Student 对象中，再将每个 Student 对象加入到一个集合中，要求集合中的元素按照年龄大小正序排序；最后遍历集合，将集合中学生信息写入到记事本，每个学生数据占单独一行。