Q1

```
package com.Example.IO;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class FileCopyUtil {
    public static void main(String[] args) {
        // 将一个文件复制到另一个位置
        //源文件
        String sourcePath = "C:\\Users\\ZZF\\AppData\\Local\\Temp\\Java09-1.9f07a976.jpg";
        //目标文件
        String targetPath =
"C:\\Users\\ZZF\\IdeaProjects\\Java02\\src\\com\\Example\\IO\\doro_copy.jpg";
        //用try-catch捕获处理异常
        try {
            copyFile(new File(sourcePath), new File(targetPath));
            System.out.println("文件复制成功! ");
        } catch (IOException e) {
            System.err.println("文件复制失败: " + e.getMessage());
            e.printStackTrace();/*e.getMessage() 获取异常的具体描述（如文件不存在、无法读取等）。
                                  e.printStackTrace() 打印完整的异常堆栈信息，方便调试。*/
        }
    }


    public static void copyFile(File source, File target) throws IOException {
        // 检查源文件是否存在且可读
        if (!source.exists() || !source.isFile()) {//使用 || (逻辑或) 连接，意味着只要有一个条件成
立，就会进入判断
            throw new IOException("源文件不存在或不是文件: " + source.getAbsolutePath());
        }

        // 确保目标文件的父目录存在
        File parentDir = target.getParentFile();
        if (parentDir != null && !parentDir.exists()) {
            parentDir.mkdirs();
        }

        // 使用try-with-resources自动关闭流
        try (FileInputStream fis = new FileInputStream(source);
             FileOutputStream fos = new FileOutputStream(target)) {

            // 使用1024字节的缓冲区
            byte[] buffer = new byte[1024];/*读取时：fis.read(buffer)将磁盘文件数据批量读入
buffer，返回实际读取的字节数；
                                                写入时：fos.write(buffer, 0, bytesRead)将 buffer
中已读取的有效数据（而非整个数组）批量写入目标文件，避免无效数据干扰。*/
            int bytesRead;
```

```
            // 循环读取数据直到文件末尾
            while ((bytesRead = fis.read(buffer)) != -1) {
                // 写入读取到的字节数
                fos.write(buffer, 0, bytesRead);
            }

            // 确保所有数据都被写入磁盘
            fos.flush();
        }
    }


}
```

Q2

```
package com.Example.IO;
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
public class NameSorter {
    public static void main(String[] args) {
        // 源文件和目标文件路径
        String sourcePath =
"C:\\Users\\ZZF\\IdeaProjects\\Java02\\src\\com\\Example\\IO\\name.txt";
        String targetPath =
"C:\\Users\\ZZF\\IdeaProjects\\Java02\\src\\com\\Example\\IO\\name_sorted.txt";

        // 使用try-with-resources自动关闭流
        try (BufferedReader reader = new BufferedReader(
                new InputStreamReader(
                        new FileInputStream(sourcePath),
                        StandardCharsets.UTF_8//指定编码格式
                )
        );
             BufferedWriter writer = new BufferedWriter(
                     new OutputStreamWriter(
                             new FileOutputStream(targetPath),
                             StandardCharsets.UTF_8
                     )
             )
        ) {
            // 1. 读取并处理数据
            List<String> names = new ArrayList<>();
            String line;
            while ((line = reader.readLine()) != null) {
                // 去除首尾空格
                String trimmedLine = line.trim();
                // 跳过空行
                if (!trimmedLine.isEmpty()) {
                    names.add(trimmedLine);
```

```
                }
            }

            // 2. 按Unicode自然排序
            Collections.sort(names);

            // 3. 写入排序后的文件
            for (String name : names) {
                writer.write(name);
                writer.newLine();//不同操作系统的换行符不同，像Windows 使用 \r\n而Linux/macOS 使用
\n
            /*可以用系统属性获取平台换行符但会增加代码复杂度
            String lineSeparator =System.getProperty("line.separator")；
            writer.write(lineSeparator);*/
                                /*这个方法会自动使用当前平台的正确换行符，让代码更简洁、可移植性更好*/
            }

            System.out.println("排序完成! 已生成文件: " + targetPath);

        } catch (FileNotFoundException e) {
            System.err.println("文件未找到: " + e.getMessage());
        } catch (IOException e) {
            System.err.println("IO错误: " + e.getMessage());
        }
    }

}
```

Q3

```
package com.Example.IO;
import java.io.Serializable;
import java.io.*;
public class Student implements Serializable  {
    // 添加serialVersionUID,表示这个类的对象可以被序列化
    private static final long serialVersionUID = 1L;

    private Integer id;
    private String name;
    private Integer gender; //1表示男性 2表示女性
    private String phone;

    public Student(Integer id, String name, Integer gender, String phone) {
        this.id = id;
        this.name = name;
        this.gender = gender;
        this.phone = phone;
    }

    public Integer getId() {
        return id;
    }
```

```
    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

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

    public Integer getGender() {
        return gender;
    }

    public void setGender(Integer gender) {
        this.gender = gender;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    @Override
    public String toString() {
        return "Student{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", gender=" + gender +
                ", phone='" + phone + '\'' +
                '}';
    }

    public static void main(String[] args) {
        //创建Student对象
        Student student = new Student(1, "张三", 1, "13800138000");

        String fileName = "student.dat";

        //序列化对象到文件
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(fileName))) {
            oos.writeObject(student);//ObjectOutputStream 用于将对象转换为字节流
            System.out.println("对象已成功序列化到文件: " + fileName);
        } catch (IOException e) {
            System.err.println("序列化失败: " + e.getMessage());
            e.printStackTrace();
        }
```

```
        //从文件反序列化,ObjectInputStream 用于将字节流还原为对象
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fileName))) {
            Student doro = (Student) ois.readObject();
            System.out.println("对象已成功从文件反序列化");

            //输出反序列化后的对象信息
            System.out.println("反序列化得到的对象信息:");
            System.out.println(doro);
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("反序列化失败: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```