# 基础题

## 练习一:高效字节输出流写出字节数据

描述:利用高效字节输出流往C盘下的d.txt文件输出一个字节数。

答案

操作步骤:

1. 创建字节输出流对象关联文件路径
2. 利用字节输出流对象创建高效字节输出流对象
3. 调用高效字节输出流对象的write方法写出一个字节
4. 关闭高效流，释放资源。

代码:

**public class** Test01\_01 {

**public static void** main(String[] args) **throws** IOException {

// 创建字节输出流FileOutputStream对象并指定文件路径。

FileOutputStream fos = **new** FileOutputStream("c:\\d.txt");

// 利用字节输出流创建高效字节输出流对象

BufferedOutputStream bos = **new** BufferedOutputStream(fos);

// 调用高效字节输出流对象的write(int byte)方法写出一个字节数据

bos.write(97);

// 关闭流

bos.close();

}

}

## 练习二:高效字节输出流写出字节数组数据

描述:利用高效字节输出流往C盘下的e.txt文件写出一个字节数组数据，如写出：”i love java”

答案

操作步骤:

1. 创建字节输出流对象关联文件路径
2. 利用字节输出流对象创建高效字节输出流对象
3. 定义字符串存放要输出的数据，然后将字符串转换为字节数组。
4. 调用高效字节输出流对象的write方法将字节数组输出。
5. 关闭高效流。

代码:

**public class** Test01\_02 {

**public static void** main(String[] args) **throws** IOException {

// 创建字节输出流FileOutputStream对象并指定文件路径。

FileOutputStream fos = **new** FileOutputStream("c:\\e.txt");

// 利用字节输出流创建高效字节输出流对象

BufferedOutputStream bos = **new** BufferedOutputStream(fos);

// 调用高效字节输出流对象的write(byte[] buff)方法写出一个字节数据

bos.write("i love java".getBytes());

// 关闭流

bos.close();

}

}

## 练习三:高效流文件复制

描述:利用高效字节输入流和高效字节输出流完成文件的复制。

要求：

1. 将C盘下的c.png文件复制到D盘下
2. 一次读写一个字节数组方式复制

答案

操作步骤:

1. 创建字节输入流对象并关联文件路径
2. 利用字节输入流对象创建高效字节输入流对象
3. 创建字节输出流对象并关联文件路径
4. 利用字节输出流对象创建高效字节输出流对象
5. 创建字节数组用来存放读取的字节数
6. 利用高效字节输入流循环读取文件数据，每读取一个字节数组，利用高效字节输出流对象将字节数组的内容输出到目标文件中。直到读取到文件末尾。
7. 关闭高效流对象

代码:

**public class** Test01\_03 {

**public static void** main(String[] args) **throws** IOException{

// 创建字节输入流对象并关联文件路径

FileInputStream fis = **new** FileInputStream("c:\\c.png");

// 利用字节输出流对象创建高效字节输出流对象

BufferedInputStream bis = **new** BufferedInputStream(fis);

// 创建字节输出流对象并指定文件路径。

FileOutputStream fos = **new** FileOutputStream("d:\\c.png");

// 利用字节输出流创建高效字节输出流对象

BufferedOutputStream bos = **new** BufferedOutputStream(fos);

// 定义字节数组接收读取的字节

**byte**[] buffer = **new byte**[1024];

// 定义变量接收读取的字节数

**int** len = -1;

// 循环读取图片数据

**while**((len = bis.read(buffer)) != -1) {

// 每读取一个字节的数据就写出到目标文件中

bos.write(buffer,0,len);

}

// 关闭流

bis.close();

bos.close();

}

}

## 练习四:高效字符流和集合的综合使用

描述:

分析以下需求，并用代码实现

实现一个验证码小程序，要求如下：

1. 在项目根目录下新建一个文件：data.txt,键盘录入3个字符串验证码，并存入data.txt中，要求一个验证码占一行；

2. 键盘录入一个需要被校验的验证码，如果输入的验证码在data.txt中存在：在控制台提示验证成功，如果不存在控制台提示验证失败

答案代码

public class Test01\_04 {

public static void main(String[] args) throws Exception {

//键盘录入3个字符串并写入项目根路径下的data.txt文件中

writeString2File();

//验证码验证

verifyCode();

}

/\*\*

\* 验证码验证

\* @throws Exception

\*/

private static void verifyCode() throws Exception {

//创建ArrayList集合，用于存储文件中的3个验证码

ArrayList<String>list = new ArrayList<>();

//创建高效字符缓冲输入流对象,并和data.txt文件关联

BufferedReader br = new BufferedReader(new FileReader(new File("data.txt")));

String line = null;

//循环读取每一行

while(null!=(line = br.readLine())) {

//将读到的每一行信息存入到list集合中

list.add(line);

}

//关闭流对象

br.close();

//创建键盘录入对象

Scanner sc = new Scanner(System.in);

//提示用户输入验证码

System.out.println("请输入一个验证码");

String code = sc.nextLine();

if(list.contains(code)) {

System.out.println("验证成功");

} else {

System.out.println("验证失败");

}

}

/\*\*

\* 键盘录入3个字符串并写入项目根路径下的data.txt文件中

\* @throws Exception

\*/

private staticvoid writeString2File() throws Exception {

//创建高效字符缓冲输出流对象并和data.txt文件关联

BufferedWriter bw = new BufferedWriter(new FileWriter(new File("data.txt")));

String line = null;

//创建键盘录入对象

Scanner sc = new Scanner(System.in);

for(inti = 0;i<3;i++) {

System.out.println("请输入第"+(i+1)+"个字符串验证码");

//读取用户键盘录入的一行验证码信息

line = sc.nextLine();

//将读取到的一行验证码写入到文件中

bw.write(line);

//写入换行符

bw.newLine();

}

//关闭流对象

bw.close();

}

}

## 练习五:转换输出流的使用

描述:现有一字符串：”我爱Java”。将该字符串保存到当前项目根目录下的a.txt文件中。

要求：使用gbk编码保存。

注意：idea的默认编码是utf-8,所以可以通过file🡪settings🡪file encodings设置为gbk格式，否则打开a.txt文件看到的将会是乱码。

答案

操作步骤:

1. 创建文件字节输出流关联目标文件
2. 根据文件字节输出流创建转换输出流对象，并指定编码字符集为：gbk
3. 调用流对象的方法将字符串写出到文件中。
4. 关闭流并释放资源。

代码:

**public class** Test01\_05 {

**public static void** main(String[] args) **throws** IOException{

// 要保存的字符串

String content = "我爱Java";

// 创建字节输出流对象

FileOutputStream fos = **new** FileOutputStream("a.txt");

// 创建转换输出流对象

OutputStreamWriter osw = **new** OutputStreamWriter(fos, "gbk");

// 调用方法写出数据

osw.write(content);

// 关闭流释放资源

osw.close();

}

}

## 练习六:转换输入流的使用

描述:利用转换输入流将当前项目根目录下使用gbk编码的a.txt文件的内容读取出来，并打印在控制台上。

要求：不能出现乱码的情况。

答案

操作步骤:

1. 创建字节输入流对象指定文件路径。
2. 根据字节输入流对象创建转换输入流对象并指定字符集编码为：gbk
3. 调用转换输入流对象的读取方法读取内容
4. 关闭流释放资源

代码:

**public class** Test01\_06 {

**public static void** main(String[] args) **throws** IOException{

// 创建字节输入流对象并关联文件

FileInputStream fis = **new** FileInputStream("a.txt");

// 创建转换输入流对象

InputStreamReader isr = **new** InputStreamReader(fis,"gbk");

// 定义字符数组存放读取的内容

**char**[] buffer = **newchar**[1024];

// 定义变量接收读取的字符个数

**int**len = -1;

**while**((len = isr.read(buffer)) != -1) {

System.***out***.print(**new** String(buffer,0,len));

}

// 关闭流

isr.close();

}

}