**每日作业卷**

**JavaSE第11天**

# 关卡1

## 训练案例1

### 训练知识点

1.字节输出流写出字节数据

### 训练描述

利用字节输出流一次写一个字节的方式向C盘的a.txt文件输出内容。

### 操作步骤描述

1. 创建字节输出流FileOutputStream对象并指定文件路径。
2. 调用字节输出流的write(int byte)方法写出数据

**public** **class** Test {

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

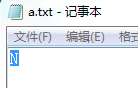
FileOutputStream fileOutputStream = **new** FileOutputStream("c:/a.txt");

fileOutputStream.write(78);

fileOutputStream.close();

}

}



## 训练案例2

### 训练知识点

1. 字节输出流写出字节数组数据

### 训练描述

利用字节输出流一次写一个字节数组的方式向C盘的b.txt文件输出内容。

### 操作步骤描述

1. 创建字节输出流FileOutputStream对象并指定文件路径。
2. 调用字节输出流的write(byte[] buf)方法写出数据。

**public** **class** Test {

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

FileOutputStream fileOutputStream = **new** FileOutputStream("c:/a.txt");

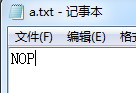
**byte**[] bytes = {78,79,80};

fileOutputStream.write(bytes);

fileOutputStream.close();

}

}



## 训练案例3

### 训练知识点

1.文件的续写和换行输出

### 训练描述

利用字节输出流对象往C盘下c.txt文件输出5句：”i love java”

要求：

1. 不能覆盖文件中原有的内容
2. 每一句输出各占一行

### 操作步骤描述

1. 利用两个参数的构造方法创建字节输出流对象，参数一指定文件路径，参数二指定为true
2. 调用字节输出流的write()方法写入数据，在每一行后面加上换行符:”\r\n”

**public** **class** Test {

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

FileOutputStream fileOutputStream = **new** FileOutputStream("c:/a.txt",**true**);

fileOutputStream.write("\r\n".getBytes());

**for** (**int** i = 0; i < 5; i++) {

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

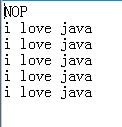
fileOutputStream.write("\r\n".getBytes());

}

fileOutputStream.close();

}

}



## 训练案例4

### 训练知识点

1.字节输入流一次读取一个字节数据

### 训练描述

利用字节输入流读取C盘文件a.txt的内容，使用循环读取，一次读取一个字节，直到读取到文件末尾。将读取的字节输出到控制台

### 操作步骤描述

1. 创建字节输入流对象指定文件路径。
2. 调用read(byte b)方法循环读取文件中的数据
3. 直到读取到-1时结束读取

**public** **class** Test {

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

FileInputStream fileInputStream = **new** FileInputStream("c:/a.txt");

**byte** b;

**while**((b=(**byte**) fileInputStream.read())!=-1){

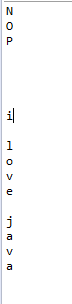
System.***out***.println((**char**)b);

}

fileInputStream.close();

}

}



## 训练案例5

### 训练知识点

1. 字节输入一次读取一个字节数组数据

### 训练描述

利用字节输入流读取C盘文件b.txt的内容，使用循环读取，一次读取一个字节数组，直到读取到文件末尾，将读取到的字节数组转换成字符串输出到控制台。

### 操作步骤描述

1. 创建字节输入流对象指定文件路径。
2. 定义一个字节数数组，用来存放读取的字节数
3. 调用read(byte[] buf)方法传入字节数组，循环读取文件中的数据
4. 直到读取到-1时结束读取

**public** **class** Test {

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

FileInputStream fileInputStream = **new** FileInputStream("c:/a.txt");

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

**int** len = -1;

**while** ((len = fileInputStream.read(b)) != -1) {

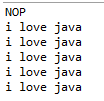
System.***out***.println(**new** String(b, 0, len));

}

fileInputStream.close();

}

}



## 训练案例6

### 训练知识点

1. 字节流复制文件

### 训练描述

利用字节流将C盘下的a.png图片复制到D盘下(文件名保存一致)

要求：

1.一次读写一个字节的方式

### 操作步骤描述

1. 创建字节输入流对象关联文件路径：C盘下的a.png
2. 创建字节输出流对象关联文件路径：D盘下的a.png
3. 使用循环不断从字节输入流读取一个字节，每读取一个字节就利用输出流写出一个字节。
4. 关闭流，释放资源

**public** **class** Test {

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

FileInputStream fileInputStream = **new** FileInputStream("c:/a.png");

FileOutputStream fileOutputStream = **new** FileOutputStream("d:/a.png");

**byte** b ;

**while** ((b = (**byte**) fileInputStream.read()) != -1) {

fileOutputStream.write(b);

}

fileInputStream.close();

fileOutputStream.close();

}

}

## 训练案例7

### 训练知识点

1.字节流复制文件

### 训练描述

用字节流将C盘下的a.png图片复制到D盘下(文件名保存一致)

要求：一次读写一个字节数组的方式进行复制

### 操作步骤描述

1. 创建字节输入流对象关联文件路径：C盘下的a.png
2. 创建字节输出流对象关联文件路径：D盘下的a.png
3. 定义一个字节数组，用来存放读取到文件字节数
4. 使用循环不断从字节输入流读取一个字节数组，每读取一个字节数组就利用输出流写出一个字节数组。
5. 关闭流，释放资源

**public** **class** Test {

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

FileInputStream fileInputStream = **new** FileInputStream("c:/a.png");

FileOutputStream fileOutputStream = **new** FileOutputStream("d:/a.png");

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

**int** len = -1;

**while** ((len = fileInputStream.read(b)) != -1) {

fileOutputStream.write(b);

}

fileInputStream.close();

fileOutputStream.close();

}

}

## 训练案例8

### 训练知识点

1.高效字节输出流写出字节数据

### 训练描述

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。

### 操作步骤描述

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。

**public** **class** Test {

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

BufferedOutputStream bOutputStream =

**new** BufferedOutputStream(**new** FileOutputStream("c:/d.txt"));

bOutputStream.write(78);

bOutputStream.close();

}

}

## 训练案例9

### 训练知识点

1. 高效字节输出流写出字节数组数据

### 训练描述

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”

### 操作步骤描述

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。

**public** **class** Test {

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

BufferedOutputStream bOutputStream =

**new** BufferedOutputStream(**new** FileOutputStream("c:/e.txt"));

String str = "i love java";

bOutputStream.write(str.getBytes());

bOutputStream.close();

}

}

## 训练案例10

### 10.1训练知识点

1.高效流文件复制

### 10.2训练描述

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b.png文件复制到D盘下
2. 一次读写一个字节方式复制

### 10.3操作步骤描述

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. 利用高效字节输入流循环读取文件数据，每读取到一个字节，利用高效字节输出流对象将读取到字节输出到目标文件中。直到读取到文件末尾。
6. 关闭高效流对象

**public** **class** Test {

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

FileInputStream fileInputStream = **new** FileInputStream("c:/b.png");

BufferedInputStream bStream = **new** BufferedInputStream(fileInputStream);

FileOutputStream fileOutputStream = **new** FileOutputStream("d:/b.png");

BufferedOutputStream bStream2 = **new** BufferedOutputStream(fileOutputStream);

**byte** b;

**while**((b=(**byte**) bStream.read())!=-1){

bStream2.write(b);

bStream2.flush();

}

fileInputStream.close();

fileOutputStream.close();

}

}

## 训练案例11

### 11.1训练知识点

1.高效流文件复制

### 11.2训练描述

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

### 11.3操作步骤描述

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

**public** **class** Test {

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

FileInputStream fileInputStream = **new** FileInputStream("c:/b.png");

BufferedInputStream bStream = **new** BufferedInputStream(fileInputStream);

FileOutputStream fileOutputStream = **new** FileOutputStream("d:/b.png");

BufferedOutputStream bStream2 = **new** BufferedOutputStream(fileOutputStream);

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

**int** len = -1;

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

bStream2.write(b);

bStream2.flush();

}

fileInputStream.close();

fileOutputStream.close();

}

}

## 训练案例12

### 12.1训练知识点

1.文件复制效率比较

### 12.2训练描述

四种复制文件方式比较

1. 字节流复制文件一次读写一个字节
2. 字节流复制文件一次读写一个字节数组
3. 高效流复制文件一次读写一个字节
4. 高效流复制文件一次读取一个字节数组

利用上面四种方式复制同一个文件，输出每一种复制方式花费的时间。

### 12.3操作步骤描述

1.在复制开始前记录开始时间

2.在复制完成后记录结束时间

3.两个时间相减即可









# 关卡2

## 训练案例1

### 训练知识点

1. 键盘录入数据
2. 字节输出流使用

### 训练描述

从控制台循环接收用户录入的学生信息，输入格式为：学号-学生名字

将学生信息保存到C盘下面的stu.txt文件中，一个学生信息占据一行数据。

当用户输入end时停止输入。

\* 要求使用字节输出流写出数据

### 操作步骤描述

1. 使用Scanner类进行键盘录入数据
2. 创建字节输入流对象关联目标文件
3. 使用死循环不停的接收用户输入的学生数据
4. 接收用户输入的学生数据
5. 判断输入的内容是否是end，是则终止循环，停止输入。否则就数据写出到文件中。

**public** **class** Test {

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

FileOutputStream fileOutputStream = **new** FileOutputStream("c:/stu.txt");

Scanner scanner = **new** Scanner(System.***in***);

String str = scanner.next();

**while**(!str.equals("end")){

fileOutputStream.write(str.getBytes());

fileOutputStream.write("\r\n".getBytes());

str = scanner.next();

}

fileOutputStream.close();

}

}

## 训练案例2

### 训练知识点

1. 高效输入流的使用

### 训练描述

使用高效流输入读取上一题保存学生信息的文件，每次读取一行信息。

每读取到一行信息就创建一个学生对象并添加到集合中。

遍历集合打印学生信息到控制台。

### 操作步骤描述

1. 创建高效流字符输入流对象关联到C盘的stu.txt文件中。
2. 定义一个学生类，包含姓名和学号两个属性。
3. 创建一个集合对象用来存放学生对象。
4. 循环读取文件中内容，一次读取一行数据。
5. 每读取一行数据使用-符合分隔字符串，根据字符串数组的内容创建一个学生对象
6. 将学生对象添加到集合中
7. 遍历集合打印学生信息。

**public** **class** Test {

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

BufferedReader bufferedReader = **new** BufferedReader(**new** FileReader("c:/stu.txt"));

ArrayList<Student> arrayList = **new** ArrayList<>();

String str = **null**;

**while**((str=bufferedReader.readLine())!=**null**){

Student student = **new** Student();

String[] strArr = str.split("-");

student.setId(strArr[0]);

student.setName(strArr[1]);

arrayList.add(student);

}

**for**(Student student:arrayList){

System.***out***.println(student.getId()+"-"+student.getName());

}

bufferedReader.close();

}

}

## 训练案例3

### 训练知识点

1. 字节输入输出流复制文件(不能使用高效流)

### 训练描述

在C盘下创建一个目录d1,在目录d1下创建创建若干个文本文件，并在每一个文本文件中添加若干内容。利用字节高效流将该目录下的所有文件复制到C盘下的d2目录下。

### 操作步骤描述

1. 创建文件对象关联路径C:/d1。
2. 通过文件对象获得该目录下的所有文件，得到一个文件数组对象。
3. 遍历文件数组，每得到一个文件就将该文件复制C:/d2目录下。

**public** **class** Test {

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

File file = **new** File("c:/d1");

File[] files = file.listFiles();

**for** (File f:files) {

FileInputStream fileInputStream = **new** FileInputStream("c:/d1/"+f.getName());

FileOutputStream fileOutputStream = **new** FileOutputStream("c:/d2/"+f.getName());

**int** len = -1;

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

**while**((len=fileInputStream.read(bytes))!=-1){

fileOutputStream.write(bytes, 0, len);

}

fileInputStream.close();

fileOutputStream.close();

}

}

}

## 训练案例4

### 训练知识点

1. 高效流复制文件

### 训练描述

在C盘下创建一个目录d1,在目录d1下创建若干个txt文件，并在每一个txt文件中添加若干内容。将所有txt文件的内容复制到D盘下的copy.txt中。

\* 要求所有txt的文件内容都复制到同一个文件中

### 操作步骤描述

1. 创建文件对象关联目录C:/d1
2. 通过文件对象获得文件数组
3. 遍历文件数组获得每一个文件，每获取一个就文件就将其内容以追加的形式复制到D盘下copy.txt文件中。

**public** **class** Test {

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

File file = **new** File("c:/d1");

File[] files = file.listFiles();

**for** (File f:files) {

FileInputStream fileInputStream = **new** FileInputStream("c:/d1/"+f.getName());

FileOutputStream fileOutputStream = **new** FileOutputStream("d:/copy.txt",**true**);

**int** len = -1;

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

**while**((len=fileInputStream.read(bytes))!=-1){

fileOutputStream.write(bytes, 0, len);

}

fileInputStream.close();

fileOutputStream.close();

}

}

}

## 训练案例5

### 训练知识点

1. 字节流读取文件数据

### 训练描述

在C盘下有一个文本文件test.txt(里面的内容由数字和字母组成)

定义一个方法统计test.txt文件中指定字符出现的次数。

比如a字符在文件中出现了10次则调用方法传入a后，方法内部输出：a出现10次

### 操作步骤描述

1. 定义一个方法接收一个参数，参数类型为char
2. 创建字节输入流对象，循环从文件中读取一个字节
3. 定义一个整数变量用来统计字符出现的次数。
4. 将读取的字节转换字符跟传入的字符进行比较，相同则计数加一。
5. 输出结果。

**public** **class** Test {

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

*count*('a');

}

**public** **static** **void** count(**char** ch) **throws** IOException {

BufferedInputStream bufferedInputStream =

**new** BufferedInputStream(**new** FileInputStream("c:/test.txt"));

**byte** b;

**int** sum = 0;

**while**((b=(**byte**) bufferedInputStream.read())!=-1){

**if**((**char**)b==ch){

sum++;

}

}

System.***out***.println(ch+"出现了"+sum+"次");

}

}

## 训练案例6

### 训练知识点

1. 集合的使用
2. 随机数的生成
3. 高效流的使用

### 训练描述

使用集合存储10个1-50(含50)的随机偶数元素,要求数字不能重复,按指定格式输出到C盘的num.txt中,例如: 48,44,40,38,34,30,26......

### 操作步骤描述

1. 创建集合对象HashSet
2. 创建随机数对象Random
3. 使用循环随机生成10个1-50(含50)的偶数
4. 将生成的偶数添加到集合中
5. 创建高效流关联目标文件，将集中的元素按指定格式拼接成字符串输出到文件中。

**public** **class** Test {

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

HashSet<Integer> hashSet = **new** HashSet<>();

Random random = **new** Random();

**while**(hashSet.size()<10){

**int** i = random.nextInt(50)+1;

**if**(i%2==0){

hashSet.add(i);

}

}

BufferedOutputStream bufferedOutputStream =

**new** BufferedOutputStream(**new** FileOutputStream("c:num.txt"));

StringBuffer sb = **new** StringBuffer();

**for**(**int** i:hashSet){

sb.append(i).append(",");

}

bufferedOutputStream.write(sb.toString().getBytes());

bufferedOutputStream.close();

}

}

# 关卡3

## 训练案例1

### 训练知识点

1.键盘录入数据

2.文件内容的读取

3.保存内容到文件

### 训练描述

模拟用户登录和注册功能实现。

接收用户输入的用户名和密码，然后和文件中存储的用户名、密码匹配。

如果文件已经存在用户名并且密码也是正确的，提示登录成功，否则提示登录失败。

如果文件中不存在该用户名，则使用该用户名和密码注册一个新的账号，并提示注册成功。

**public** **class** Test {

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

BufferedReader bufferedReader = **new** BufferedReader(**new** FileReader("user.txt"));

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.println("请输入用户名");

String username = scanner.next();

System.***out***.println("请输入密码");

String password = scanner.next();

String str = **null**;

**boolean** flag = **true**;

**while**((str=bufferedReader.readLine())!=**null**){

String[] strArr = str.split(" ");

**if**(username.equals(strArr[0])){

flag = **false**;

**if**(password.equals(strArr[1])){

System.***out***.println("登陆成功");

}**else** {

System.***out***.println("登陆失败");

}

}

}

**if**(flag){

BufferedWriter bufferedWriter = **new** BufferedWriter(**new** FileWriter("user.txt",**true**));

bufferedWriter.newLine();

bufferedWriter.write(username+" "+password);

System.***out***.println("注册成功");

bufferedWriter.close();

}

}

}







## 训练案例2

### 训练知识点

1. 文件的读写操作
2. Map集合的使用

### 训练描述

在d盘目录下有一个加密文件a.txt（文件里只有英文和数字），密码是“igeek”

当密码输入正确时才能读取文件里的数据。现要求用代码来模拟读取文件的过程，并统计文件里各个字母出现的次数，并把统计结果按照如下格式输出到d盘的count.txt中。

a：2个

b：3个

c：4个

............

**public** **class** Test {

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

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.println("请输入密码");

String password = scanner.nextLine();

**while**(!password.equals("igeek")){

System.***out***.println("密码错误，重新输入");

password = scanner.next();

}

BufferedInputStream bufferedInputStream =

**new** BufferedInputStream(**new** FileInputStream("d:/a.txt"));

**int** c = -1;

HashMap<Character, Integer> hashMap = **new** HashMap<>();

**while**((c=bufferedInputStream.read())!=-1){

System.***out***.println((**char**)c);

**if**(!hashMap.containsKey((**char**)c)){

hashMap.put((**char**) c, 1);

}**else** {

hashMap.put((**char**)c, hashMap.get((**char**)c)+1);

}

}

bufferedInputStream.close();

BufferedWriter bufferedWriter = **new** BufferedWriter(**new** FileWriter("d:/count.txt"));

**for**(**char** ch:hashMap.keySet()){

bufferedWriter.write(ch+":"+hashMap.get(ch)+"个");

bufferedWriter.newLine();

}

bufferedWriter.close();

}

}

## 训练案例3

### 训练知识点

1. 文件的获取
2. 文件过滤器
3. 递归的使用
4. 字节流的使用

### 训练描述

从控制台获取输入的文件目录然后将该目录(包含子目录)下的.java文件复制到D:/java文件夹中,并统计java文件的个数.

提示:如果有相同的名称的文件,如果两个Test01.java,则拷贝到目标文件夹时只能有一个Test01.java,另一个Test01.java要修改为另一个名称:该名称可随机生成,只要不重复即可.

**public** **class** Test {

**static** HashSet<String> *name* = **new** HashSet<>();

**static** **int** *sum* = 0;

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

Scanner scanner = **new** Scanner(System.***in***);

String path = scanner.nextLine();

File file = **new** File(path);

**int** sum = *copy*(file);

System.***out***.println(sum);

}

**private** **static** **int** copy(File file) **throws** IOException {

TestFileFilter filter = **new** TestFileFilter();

File[] files = file.listFiles(filter);

**for** (File f : files) {

**if** (f.isFile()) {

*sum*++;

String newName;

**if**(*name*.contains(f.getName())){

UUID uuid = UUID.*randomUUID*();

newName = uuid.toString().substring(0, 5)+".java";

}**else**{

newName = f.getName();

}

*name*.add(newName);

FileInputStream bufferedInputStream = **new** FileInputStream(f.getAbsolutePath());

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

**int** len = -1;

**while** ((len = bufferedInputStream.read(bytes)) != -1) {

BufferedOutputStream bufferedOutputStream = **new** BufferedOutputStream(

**new** FileOutputStream("d:/java/"+newName));

bufferedOutputStream.write(bytes,0,len);

bufferedOutputStream.close();

}

bufferedInputStream.close();

} **else** {

*copy*(f);

}

}

**return** *sum*;

}

}