今日目标

能够使用字节输出流写出数据到文件

能够使用字节输入流读取数据到程序

能够说出读取数据read(byte[])方法的原理

能够使用字节流完成文件的复制

能够使用字节缓冲流读取数据到程序

能够使用字节缓冲流写出数据到文件

能够完成单级文件夹复制

# 字节流

## ## 回顾I/O流的字符流

1.字符流作用

1.将程序中的数据写入文件

2.将文件中的数据读取到程序中

2.字符流继承体系

输入流 输出流

Reader Writer

| |

InputStreamReader OutputStreamWriter

| |

FileReader FileWriter

3.字符流简单练习

3.1. 向a.txt中写入中国

3.2. 将a.txt中的内容读取到程序中

代码

public class Demo01 {

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

// testWrite();

*testRead*();

}

private static void testRead() throws Exception {

FileReader fr = new FileReader("abc.txt");

int ch;

while ((ch = fr.read()) != -1) {

System.*out*.print((char)ch);

}

fr.close();

}

public static void testWrite() throws IOException {

FileWriter fw = new FileWriter("abc.txt");

fw.write("中国");

fw.close();

}

}

## ## 计算机数据的本质

存储在计算机中的所有数据(文本,图片,音频,视频)都是以**二进制**保存在硬盘中的(最小单位是字节 1个字节=8位)

## ## IO流的分类

1.按数据流向分(以我们的程序作为参考的)

1. 输入流

2. 输出流

2.按数据类型分

1. 字符流(操作的单位字符)

2. 字节流(操作的单位是字节)(通用)

## ## 字节输出流OutputStream

1.OutputStream作用

此抽象类是表示输出字节流的所有类的超类

2.OutputStream的普通方法

|  |  |
| --- | --- |
| void | **[write](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/OutputStream.html" \l "write(byte[]))**(byte[] b)            将字节数组写入到流中 |
| void | **[write](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/OutputStream.html" \l "write(byte[], int, int))**(byte[] b, int off, int len)            将字节数组的一部分写入到流中,从off开始写len的长度 |
| abstract  void | **[write](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/OutputStream.html" \l "write(int))**(int b)           写一个字节到流中 |

3.OutputStream子类

OutputStream是一个抽象类我们不能直接使用,使用子类**FileOutputStream**

## ## FileOutputStream类

FileOutputStream: 将数据写入文件

1.构造方法

1.FileOutputStream(String name)

通过字符串路径创建FileOutputStream对象

2.FileOutputStream(File file)

通过File对象创建FileOutputStream对象

## ## FileOutputStream的3个写数据方法

|  |  |
| --- | --- |
| void | **[write](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/OutputStream.html" \l "write(byte[]))**(byte[] b)            将字节数组写入到流中 |
| void | **[write](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/OutputStream.html" \l "write(byte[], int, int))**(byte[] b, int off, int len)            将字节数组的一部分写入到流中,从off开始写len的长度 |
| void | **[write](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/OutputStream.html" \l "write(int))**(int b)           写一个字节到流中 |

代码

public class Demo02 {

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

// 创建对应的文件,如果文件已经存在会先删除在创建

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

// File f = new File("a.txt");

// FileOutputStream fos = new FileOutputStream(f);

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

// 写一个字节

// fos.write(100); // 01100100 -> 文本编辑器转换的 d

// 写一个字节数组

byte[] buf = {97, 98, 99, 100};

// fos.write(buf);

// 写一个字节数组的一部分

// fos.write(buf, 1, 2);

// 写中文

byte[] bytes = "中国".getBytes();

fos.write(bytes);

fos.close();

}

}

## ## 给文件中续写和换行

1.我们直接new FileOutputStream(file)这样创建对象，写入数据，会覆盖原有的文件

2.FileOutputStream另外2个构造方法可以追加写入

FileOutputStream(File file, boolean append)

FileOutputStream(String name, boolean append)

当append为true时,就是自动追加到后面

3.内容换行

"\r\n".getBytes()

代码

public class Demo03 {

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

// test01();

*test02*();

}

// 文件换行

public static void test02() throws Exception {

FileOutputStream fos = new FileOutputStream("c.txt", true);

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

fos.write(100);

fos.close();

}

// 文件续写

public static void test01() throws Exception {

FileOutputStream fos = new FileOutputStream("b.txt", true);

fos.write(100);

fos.close();

}

}

## ## IO异常的处理

IO异常的处理

代码

public class Demo05 {

public static void main(String[] args) {

FileOutputStream fos = null;

try {

fos = new FileOutputStream("d.txt");

fos.write(100);

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally {

if (fos != null) {

try {

fos.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

}

## ## 字节输入流InputStream

1.InputStream的作用

此抽象类是表示字节输入流的所有类的超类

2.InputStream普通方法

|  |  |
| --- | --- |
| abstract  int | **[read](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/InputStream.html" \l "read())**()            读取一个字节 |
| int | **[read](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/InputStream.html" \l "read(byte[]))**(byte[] b)            将读取到的数据存到字节数组中b |
| (用的少) int | **[read](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/InputStream.html" \l "read(byte[], int, int))**(byte[] b, int off, int len)            将输入流中最多 len 个数据字节读入 byte 数组。 |

3.InputStream的子类

InputStream是一个抽象类,我们使用子类FileInputStream

## ## FileInputStream类

FileInputStream类: 从文件中读取字节数据

1.构造方法

1.FileInputStream(String name)

通过字符串路径创建FileInputStream对象

2.FileInputStream(File file)

通过File对象创建FileInputStream对象

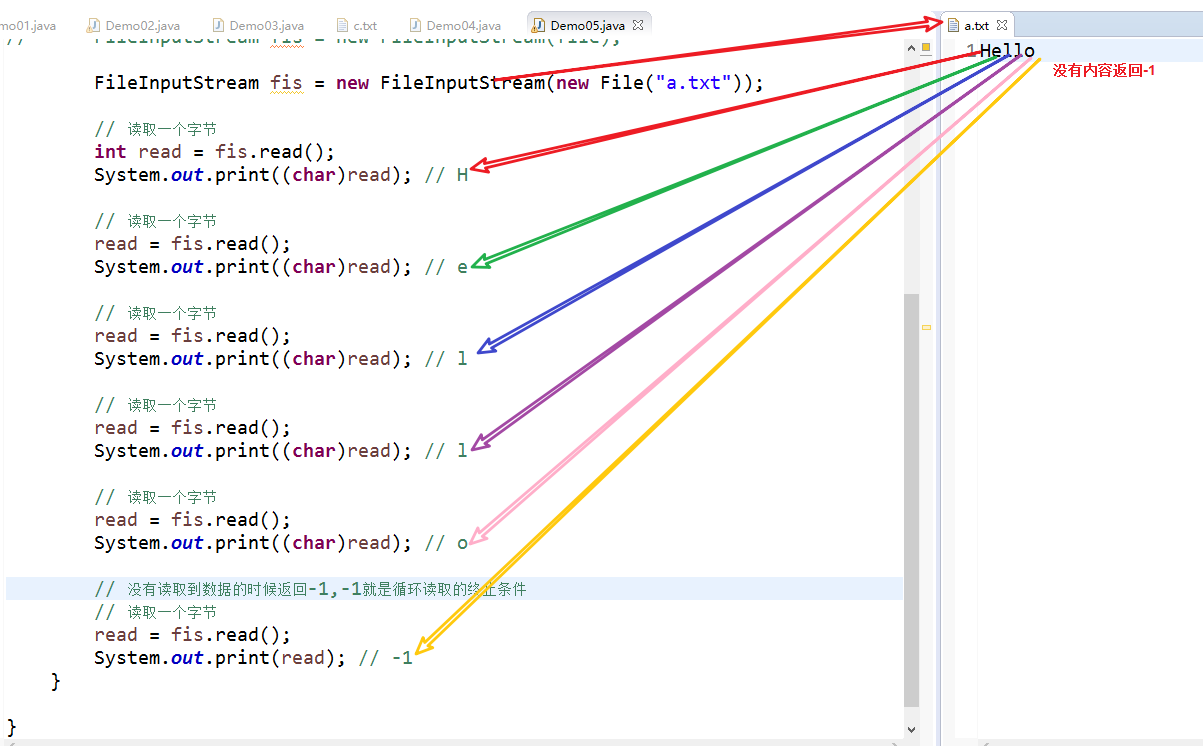
2.普通方法

|  |  |
| --- | --- |
| int | **[read](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/InputStream.html" \l "read())**()            读取一个字节 |
| int | **[read](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/InputStream.html" \l "read(byte[]))**(byte[] b)            将读取到的数据存到字节数组中b |
| (用的少) int | **[read](mk:@MSITStore:C:\\develop\\手册\\JDK_API_1_6_zh_CN.CHM::/java/io/InputStream.html" \l "read(byte[], int, int))**(byte[] b, int off, int len)            将输入流中最多 len 个数据字节读入 byte 数组。 |

## ## FileInputStream类读取数据read方法

1.int read()

读取一个字节



2.如何使用

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

// 使用循环解决,-1就是循环读取的终止条件

**int** by;

// 1.将读取到的字节数据赋值给by;

// 2.使用by和-1比较

**while** ((by = fis.read()) != -1) {

System.***out***.print((**char**)by);

}

fis.close();

代码

public class Demo05 {

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

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

// File file = new File("a.txt");

// FileInputStream fis = new FileInputStream(file);

// FileInputStream fis = new FileInputStream(new File("a.txt"));

// // 读取一个字节

// int read = fis.read();

// System.out.print((char)read); // H

//

// // 读取一个字节

// read = fis.read();

// System.out.print((char)read); // e

//

// // 读取一个字节

// read = fis.read();

// System.out.print((char)read); // l

//

// // 读取一个字节

// read = fis.read();

// System.out.print((char)read); // l

//

// // 读取一个字节

// read = fis.read();

// System.out.print((char)read); // o

//

// // 没有读取到数据的时候返回-1,-1就是循环读取的终止条件

// // 读取一个字节

// read = fis.read();

// System.out.print(read); // -1

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

// 使用循环解决,-1就是循环读取的终止条件

int by;

// 1.将读取到的字节数据赋值给by;

// 2.使用by和-1比较

while ((by = fis.read()) != -1) {

System.*out*.print((char)by);

}

fis.close();

}

}

## ## 字节流复制图片

1.需求:

复制工程下的图片.JPG到工程下名为图片Copy.JPG

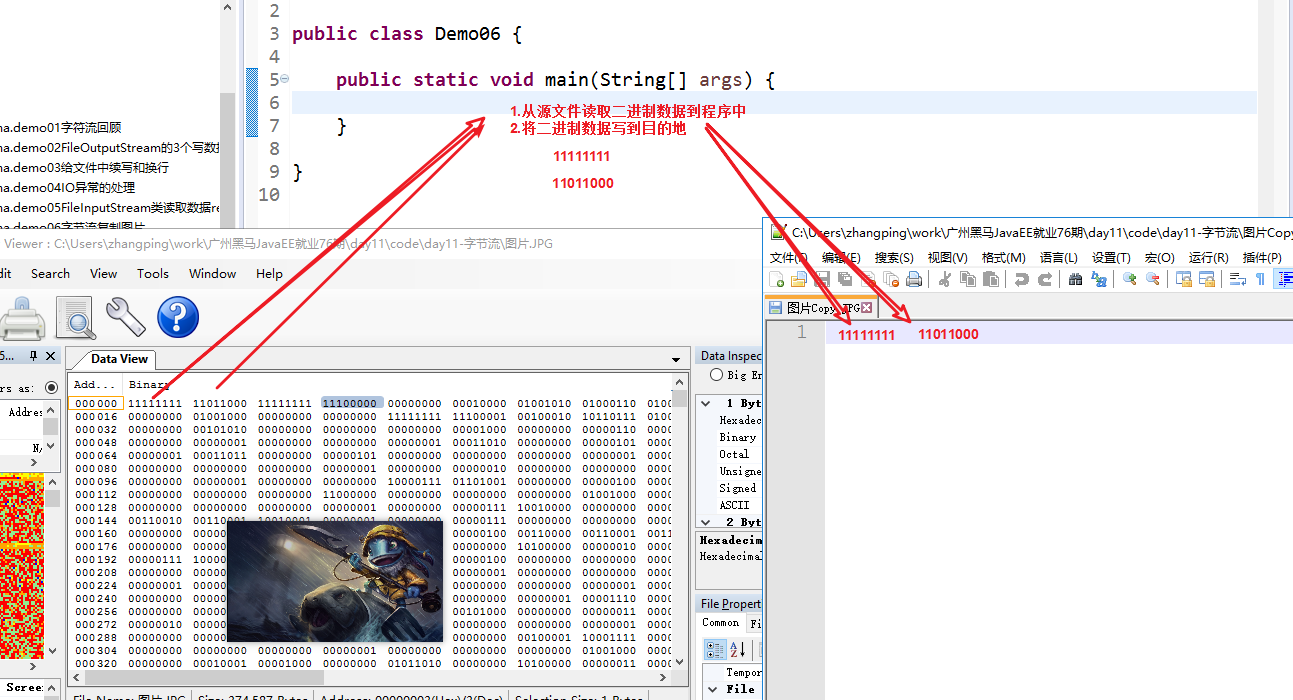
2.分析

数据来源

图片.JPG 读取数据 FileInputStream int read()

数据目的地

图片Copy.JPG 写数据 FileOutputStream void write(int b)



代码

/\*

1.需求:

复制工程下的图片.JPG到工程下名为图片Copy.JPG

2.分析

数据来源

图片.JPG 读取数据 FileInputStream int read()

数据目的地

图片Copy.JPG 写数据 FileOutputStream void write(int b)

\*/

**public** **class** Demo06 {

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

FileInputStream fis = **new** FileInputStream("图片.JPG");

FileOutputStream fos = **new** FileOutputStream("图片Copy.JPG");

// 循环读写

**int** by;

**while** ((by = fis.read()) != -1) { // 读取到内容

// 写入到目的地

fos.write(by);

}

fis.close();

fos.close();

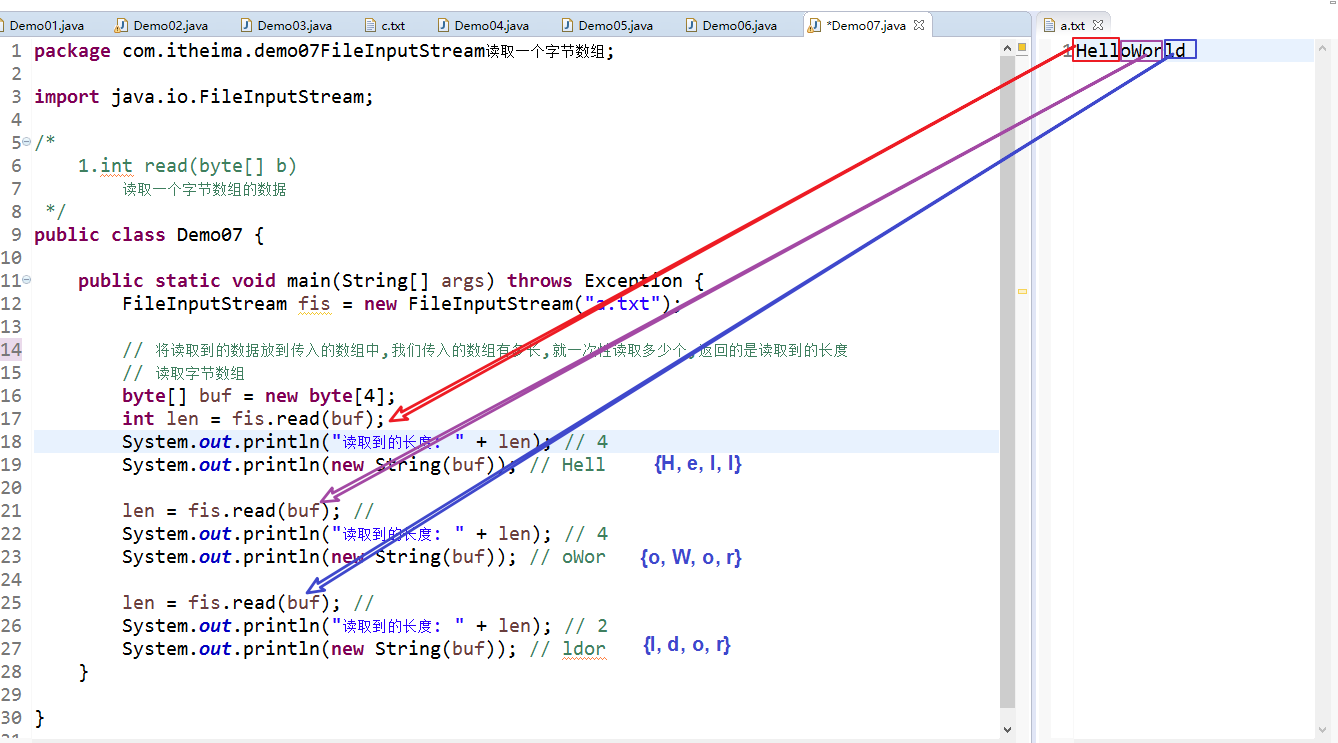
}

}

## ## FileInputStream读取一个字节数组

1.int read(byte[] b)

读取一个字节数组的数据



能够说出读取数据read(byte[])方法的原理

2.int read(byte[] b)读取数据的原理

将读取到的数据放到传入的数组中, 我们传入的数组有多长,就一次性读取多少个, 返回的是读取到的长度

3.好处

**一次性读取多个数据,读取文件的次数就减少了.读取文件是耗时的操作**

4.如何使用

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

**byte**[] buf = **new** **byte**[1024]; // 1k的整数倍

**int** len;

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

// 有内容

// 读取到多少转多少

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

}

fis.close();

代码

public class Demo07 {

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

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

//

// // 将读取到的数据放到传入的数组中,我们传入的数组有多长,就一次性读取多少个,返回的是读取到的长度

// // 读取字节数组

// byte[] buf = new byte[4];

// int len = fis.read(buf);

// System.out.println("读取到的长度: " + len); // 4

// System.out.println(new String(buf, 0, len)); // Hell

//

// len = fis.read(buf); //

// System.out.println("读取到的长度: " + len); // 4

// System.out.println(new String(buf, 0, len)); // oWor

//

// len = fis.read(buf); //

// System.out.println("读取到的长度: " + len); // 2

// System.out.println(new String(buf, 0, len)); // ldor

//

// // 没有读取到内容返回-1

// len = fis.read(buf);

// System.out.println("读取到的长度: " + len); // -1

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

byte[] buf = new byte[1024]; // 1k的整数倍

int len;

while ((len = fis.read(buf)) != -1) {

// 有内容

// 读取到多少转多少

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

}

fis.close();

}

}

## ## 字节流数组复制图片

1.需求

复制工程下的图片.JPG到工程下名为图片Copy2.JPG

2.分析

1.数据来源

图片.JPG 读取数据 FileInputStream int read(byte[] b)

2.数据目的地

图片Copy2.JPG 写数据 FileOutputStream void write(byte[] b, int off, int len)

代码

public class Demo08 {

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

FileInputStream fis = new FileInputStream("图片.JPG");

FileOutputStream fos = new FileOutputStream("图片Copy2.JPG");

// 循环读写数据

byte[] buf = new byte[1024];

int len;

while ((len = fis.read(buf)) != -1) {

// 将读取到的数据写入到目的地

fos.write(buf, 0, len);

}

fis.close();

fos.close();

}

}

## ## 字节缓冲流介绍

我们发现操作字节数组的速度明显快于操作一个字节,所以Java在设计的时候就为我们提供好了带数组的缓冲流

字符缓冲流(BufferedReader/BufferedWriter)

2.字节输入缓冲流(BufferedInputStream)

内部有一个数组,提高读取效率

继承结构

InputStream

|

FilterInputStream

|

BufferedInputStream

3.字节输出缓冲流: (BufferedOutputStream)

内部有一个数组,提高读取效率

继承结构

OutputStream

|

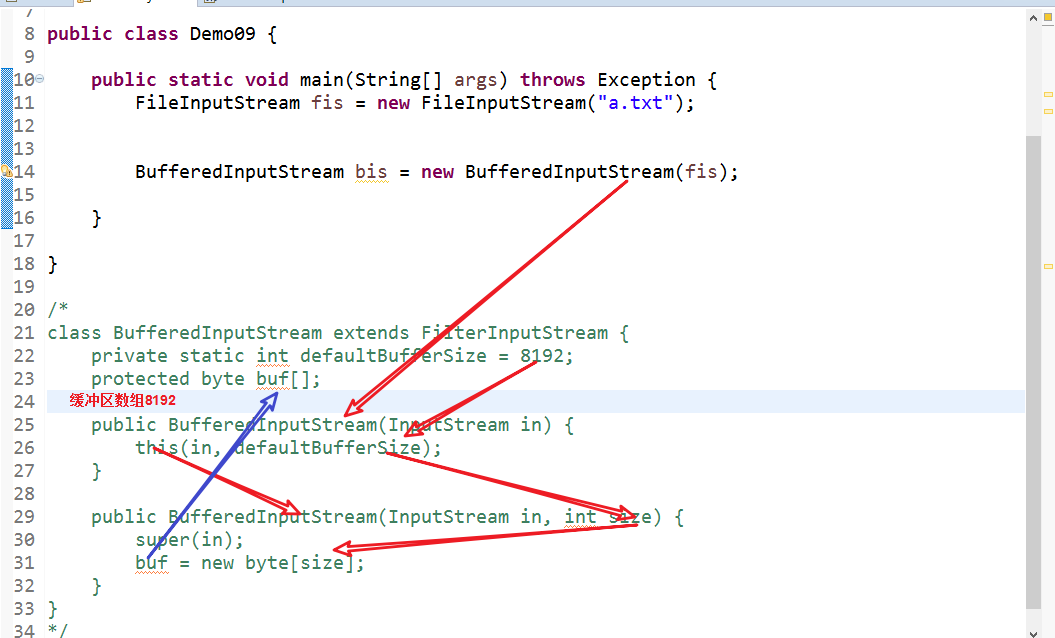
FilterOutputStream

|

BufferedOutputStream

4.字节缓冲流默认缓冲区大小

8192



## ## BufferedOutputStream写数据

1.构造方法

BufferedOutputStream(OutputStream out)

通过OutputStream对象来创建BufferedOutputStream

2.普通方法

1.void write(int b)

写入一个字节数据

2.void write(byte[] b)

写入一个字节数组

\*\*\*\*\* 3.void write(byte[] b, int off, int len)

写入一个字节数组的一部分,从off开始写,写len个字节

3.练习: 往文件中写"HelloBoss"



代码

public class Demo10 {

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

// 3.练习: 往文件中写"HelloBoss"

// BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream("bos.txt"));

FileOutputStream fos = new FileOutputStream("bos.txt");

// 创建缓冲流

BufferedOutputStream bos = new BufferedOutputStream(fos);

// 写数据

bos.write("HelloBoss".getBytes());

// 关流

bos.close(); // 实际上关的是我们构造方法传入的fos

}

}

/\*

class FilterOutputStream extends OutputStream {

protected OutputStream out = fos;

public FilterOutputStream(OutputStream out) {

this.out = out;

}

}

class BufferedOutputStream extends FilterOutputStream {

public BufferedOutputStream(OutputStream out) {

this(out, 8192);

}

public BufferedOutputStream(OutputStream out, int size) {

super(out);

buf = new byte[size];

}

public void close() throws IOException {

out.close();

}

}

\*/

## ## BufferedInputStream读数据

1.构造方法

BufferedInputStream(InputStream in)

通过InputStream对象创建BufferedInputStream

2.普通方法

1. int read()

读取一个字节数据

\*\*\*\*\* 2.int read(byte[] b)

读取一个字节数组的数据

3.int read(byte[] b, int off, int len)

读取一个字节数组的数据,从off开始放,放len个

3.练习: 读取文件中的内容

代码

public class Demo11 {

// 3.练习: 读取文件中的内容

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

// 创建字节缓冲输入流

FileInputStream fis = new FileInputStream("bos.txt");

BufferedInputStream bis = new BufferedInputStream(fis);

byte[] buf = new byte[1024];

int len;

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

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

}

bis.close();

}

}

## ## 缓冲流复制图片

1.需求

复制工程下的图片.JPG到工程下名为图片Copy3.JPG

2.分析

1.数据来源

图片.JPG 读取数据 BufferedInputStream int read(byte[] b)

2.目的地

图片Copy3.JPG 写数据 BufferedOutputStream void write(byte[] b, int off, int len)

代码

public class Demo12 {

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

// 创建输入输出缓冲流

BufferedInputStream bis = new BufferedInputStream(new FileInputStream("图片.JPG"));

BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream("图片Copy3.JPG"));

// 循环读写

byte[] buf = new byte[1024];

int len;

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

// 将读取到的数据写入目的地

bos.write(buf, 0, len);

}

bis.close();

bos.close();

}

}

## ## 四种复制文件方式

需求

将工程目录下的小苹果.mp3复制到工程目录下小苹果Copy.mp3

使用4中方式复制

1.字节流一次复制一个字节

2.字节流一次复制一个字节数组

3.缓冲字节流一次复制一个字节

4.缓冲字节流一次复制一个字节数组

代码

## ## 复制单级文件夹

需求

将 C:\abc 文件夹中的所有文件复制到 C:\test 文件夹

分析

1. 定义一个File关联到对应的文件夹C:\abc
2. listFiles列出里面的所有内容
3. 遍历每个File对象
4. 如果是文件夹,就不管
5. 如果是文件就复制

来源:遍历到的File对象

目的地: 目的地文件夹+文件名(C:\abc\1.jpg)

代码

## ## 加密

可逆(加密后能还原)

对称加密(用什么加密就用什么解密)

123 -> %^加密 -> @@@@ -> %^解密 -> 123

非对称加密(加密和解密使用不同的密码)

123 -> \*&加密 -> \*\*\*\* -> $$解密 -> 123

不可逆(加密后不能还原)

MD5/SHA-1/SHA-2/SHA-5 (数字摘要)

20G -> 加密 -> 32位的字符串

可以用来判断内容是否被改过

人 -> 指纹

对密码加密

加密的核心是对每个字节进行修改,保存到一个新文件

^boolean类型

true ^ true = false

false ^ true = true

ture ^ false = true

false ^ false = false

相同为假,不同为真

0代表false

1代表true

100 0110 0100

^ 5 0000 0101

97 0110 0001 加密

^ 5 0000 0101

0110 0100 解密

一个数异或另一个数2次会等于这个数本身

代码

public class Demo15 {

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

*test*();

}

// 读取加密前的内容 -> 程序中修改字节 -> 写到加密后的文件中

public static void test() throws Exception {

// 加密前的内容

// 加密后的内容

// FileInputStream fis = new FileInputStream("图片.JPG");

// FileOutputStream fos = new FileOutputStream("图片-加密.JPG");

FileInputStream fis = new FileInputStream("图片-加密.JPG");

FileOutputStream fos = new FileOutputStream("图片-解密.JPG");

// 读取加密前的内容 -> 程序中修改字节 -> 写到加密后的文件中

byte[] buf = new byte[1024];

int len;

while ((len = fis.read(buf)) != -1) {

// 读取到了内容,在buf中,长度是len

// 对buf中的每个字节数据都加密 {11, 12, 13, 14} ^ 5-> {33, 44, 55, 99} ^ 5-> {11, 12, 13, 14}

for (int i = 0; i < len; i++) {

// 取出数据

byte b = buf[i];

// 加密

byte newB = (byte)(b ^ 5);

buf[i] = newB;

}

// 将加密好的数据写入文件中

fos.write(buf, 0, len);

}

fis.close();

fos.close();

}

}