

目录

[Day14. Java](#)

[1 回顾](#)

[2 InputStream / OutputStream](#)

[3 高级流、操作流](#)

[4 BufferedInputStream/BufferedOutputStream](#)

[5 ObjectInputStream / ObjectOutputStream](#)

[6 作业](#)


Day14. Java

1 回顾

- Object
 - toString()
"类名@地址"
 - equals(obj)
比较地址 this == obj
- String
 - 封装char[]数组
 - 常量池
 - 不可变, 字符串连接效率低
- StringBuilder / StringBuffer
 - append() 高效率字符串连接
 - StringBuilder 线程不安全, 效率高
 - StringBuffer 线程安全
- 正则表达式
- 基本类型包装类
 - Integer.valueOf(5)
256个缓存对象, -128到127
- BigDecimal / BigInteger
 - 精确浮点数运算
 - 超大的整数运算
 - BigDecimal.valueOf(2)
- Date
- SimpleDateFormat
- 集合
 - ArrayList
 - LinkedList
 - HashMap
 - 哈希算法
 - ◆ 用key的哈希值计算下标 i

- ◆ 空位置，直接放入
- ◆ 有数据，用equals()依次比较是否相等
 - 找到相等的，覆盖值
 - 没有相等的，链表连接在一起
- ◆ 负载率、加载因子 0.75
 - 新建翻倍长度新数组
 - 所有数据，重新执行哈希运算，放入新数组
- ◆ jdk1.8
 - 链表长度到8，转成红黑树
 - 红黑树数据减少到6，转回成链表
- Iterator
 - ◆ hasNext(),next(),remove()
- Collections 工具类
- 异常
 - 其他异常
 - RuntimeException
- io
 - File
 - InputStream / OutputStream
 - FileInputStream / FileOutputStream

2 InputStream / OutputStream

- 字节流的抽象父类
- 方法
 - write(int b)
输出末尾的一个字节
 - write(byte[] buff)
输出全部字节值
 - write(byte[] buff,start,length)
输出从start开始的length个字节值
 - read()
读取一个字节值，补三个0字节，转成int


读取结束后，再读取，返回 -1
 - read(byte[] buff)
根据数组长度，读取一批字节值，放入数组，并返回这一批的字节数量
最后一批如果放不满，返回值可能小于数组长度

读取结束后，再读取，返回 -1

文件流

day1303_文件流
Test2

package day1303;

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;

public class Test2 {
    public static void main(String[] args) throws Exception {
        /*
         * FIS--f1
         */
        FileInputStream in =
            new FileInputStream("d:/abc/f1");

        //单字节循环读取，标准格式
        int b;
        while((b = in.read()) != -1) {
            System.out.println(b);
        }

        in.close();
    }
}
```

Test3

```
package day1303;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.Arrays;

public class Test3 {
    public static void main(String[] args) throws Exception {
        FileInputStream in =
            new FileInputStream("d:/abc/f1");

        //批量循环读取标准格式
        byte[] buff = new byte[5];
        int n; //保存每一批的数量
        while((n = in.read(buff)) != -1) {
            System.out.println(
                n + "个: " + Arrays.toString(buff));
        }

        in.close();
    }
}
```

文件复制

项目: day1401_文件复制

类: day1401.Test1

```
package day1401;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Scanner;

public class Test1 {
    public static void main(String[] args) {
        System.out.println("原文件: ");
        String s1 = new Scanner(System.in).nextLine();
        File from = new File(s1);
        if(! from.isFile()) {
            System.out.println("输入的不是文件");
            return;
        }
        System.out.println("目标文件: ");
        String s2 = new Scanner(System.in).nextLine();
        File to = new File(s2);
        if(to.isDirectory()) {
            System.out.println(
                "不能输入文件夹, 请输入具体文件");
            return;
        }
        try {
            copy(from, to);
            System.out.println("完成");
        } catch (Exception e) {
            System.out.println("失败");
            e.printStackTrace(); //打印完整的异常信息
        }
    }

    private static void copy(
        File from, File to) throws Exception {
        /*
         * *) 新建流
         *   FIS--from 存到变量in
         *   FOS--to 存到变量out
         * *) 单字节循环读取标准格式
         *   *) 读取的字节值, 想输出流输出
         *
         * *) 关闭输入流和输出流
         */
        FileInputStream in =
            new FileInputStream(from);
        FileOutputStream out =
            new FileOutputStream(to);

        /*int b;
        while((b = in.read()) != -1) {
            out.write(b);
        }*/
    }
}
```

```
byte[] buff = new byte[8192];
int n; // 每一批的数量
while((n = in.read(buff)) != -1) {
    out.write(buff, 0, n);
}

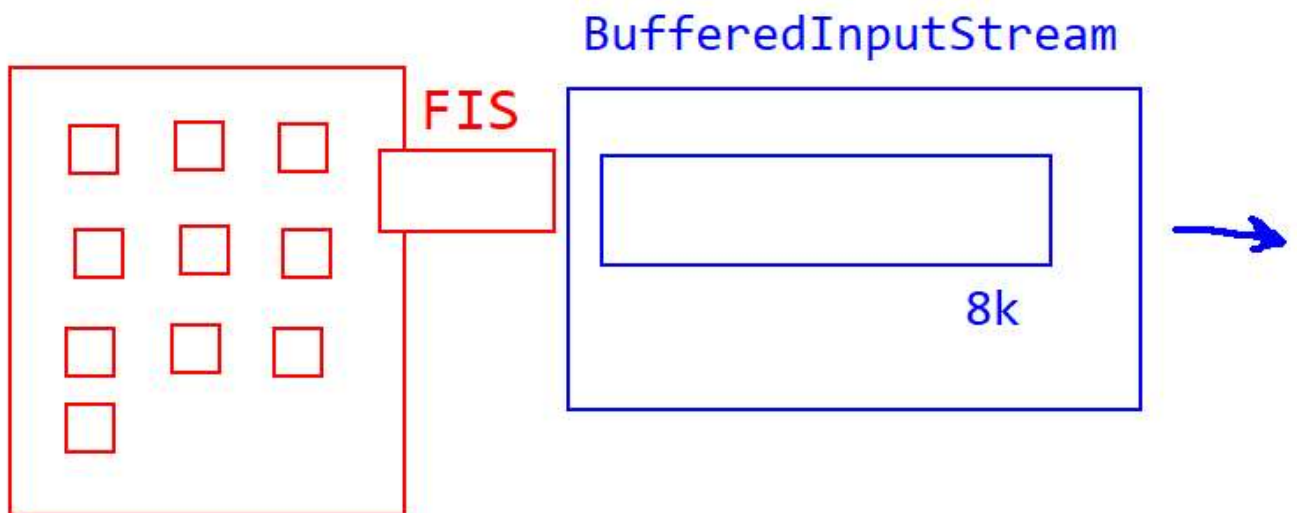
in.close();
out.close();
}
```

3 高级流、操作流

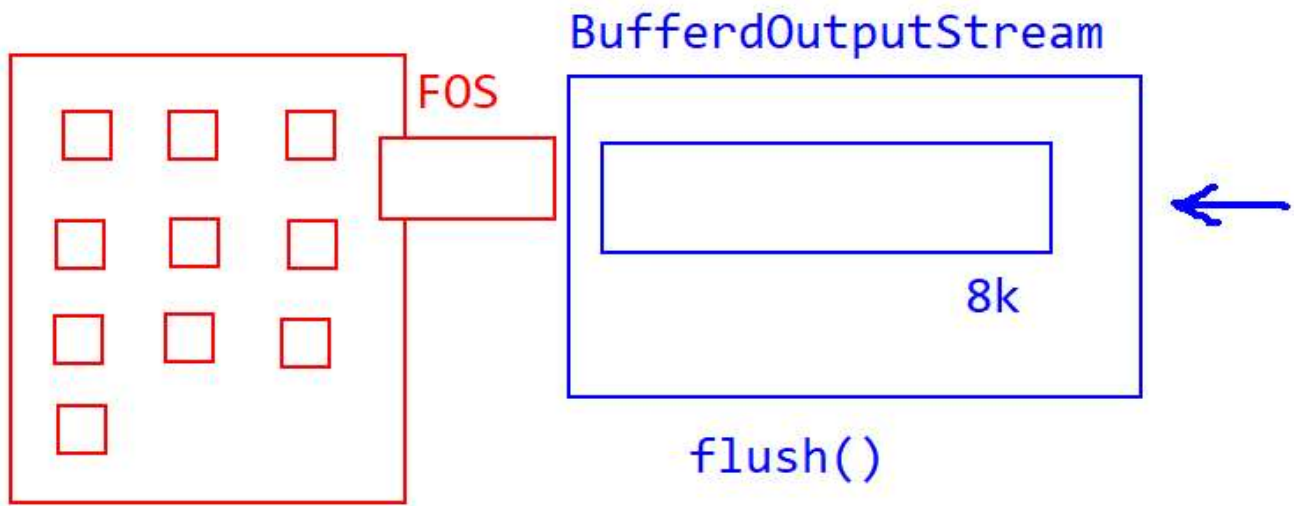
- 与其他流相接，提供特定的数据处理功能
- 对高级流的操作，会对相接的流执行相同操作

4 BufferedInputStream/BufferedOutputStream

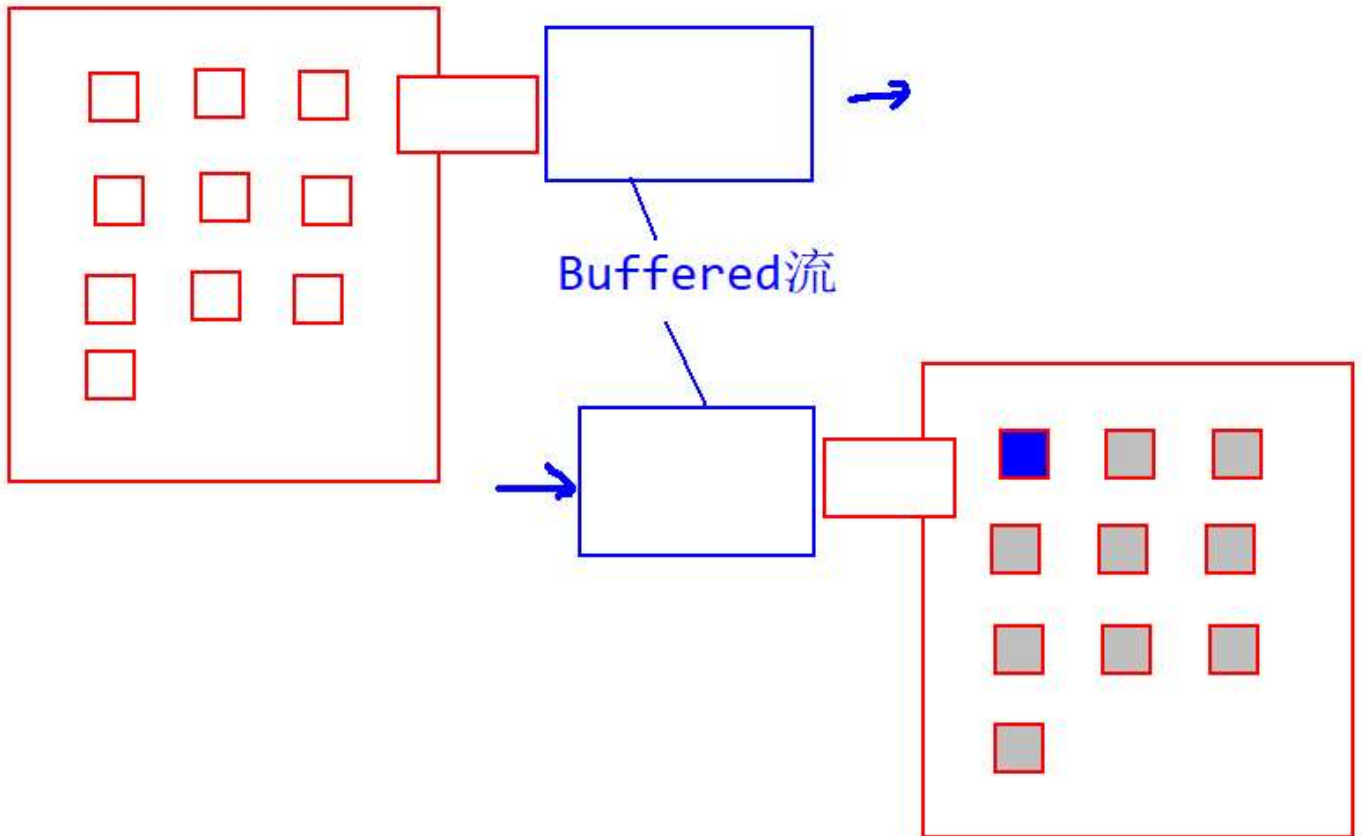
- 提供一个内存缓冲区，提高单字节的读写效率
- BufferedInputStream
单字节读取时，缓冲流帮助做批量读取，读取一批数据缓存在它内部数组中



- BufferedOutputStream
向输出流输出的数据，会暂时缓存在它内部数组中
数组存满，自动刷出
可以手动刷出缓存 flush()



加密复制



```
package day1401;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.util.Scanner;

public class Test2 {
    public static void main(String[] args) {
        System.out.println("原文件: ");
        String s1 = new Scanner(System.in).nextLine();
```

```
File from = new File(s1);
if(! from.isFile()) {
    System.out.println("输入的不是文件");
    return;
}
System.out.println("目标文件: ");
String s2 = new Scanner(System.in).nextLine();
File to = new File(s2);
if(to.isDirectory()) {
    System.out.println(
        "不能输入文件夹, 请输入具体文件");
    return;
}

System.out.println("KEY:");
int key = new Scanner(System.in).nextInt();

try {
    encryptCopy(from, to, key);
    System.out.println("完成");
} catch (Exception e) {
    System.out.println("失败");
    e.printStackTrace();//打印完整的异常信息
}

}

private static void encryptCopy(
    File from, File to, int key) throws Exception {
    /*
     * BIS--FIS--from
     * BOS--FOS--to
     */
    BufferedInputStream in =
        new BufferedInputStream(
            new FileInputStream(from));
    BufferedOutputStream out =
        new BufferedOutputStream(
            new FileOutputStream(to));

    int b;
    while((b = in.read()) != -1) {
        b ^= key;
        out.write(b);
    }

    in.close();
    out.close();
}

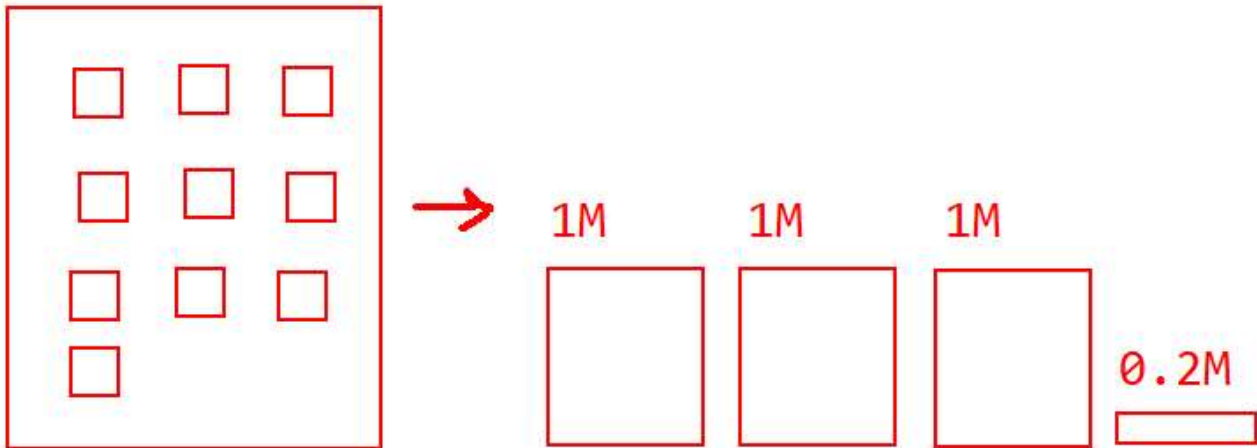
}
```

文件拆分合并

项目: day1402_文件拆分合并

类: day1402.Test1

3.2M



```
package day1402;
```

```
import java.io.BufferedReader;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.util.Scanner;
```

```
public class Test1 {
    public static void main(String[] args) {
        /*
         * 文件: d:/abc/a.mp4
         * 拆分文件的大小(Kb): 1024
         *
         * d:/abc/a.mp4
         * [d:/abc/a.mp4_split]
         *     a.mp4.1
         *     a.mp4.2
         *     a.mp4.3
         *     ...
         */
        System.out.println("文件: ");
        String s = new Scanner(System.in).nextLine();
        File file = new File(s);
        if(! file.isFile()) {
            System.out.println("输入的不是文件");
            return;
        }
        System.out.print("拆分文件大小(Kb): ");
        long size =
            1024*new Scanner(System.in).nextLong();
        try {
            split(file, size);
            System.out.println("完成");
        } catch (Exception e) {
            System.out.println("失败");
            e.printStackTrace();
        }
    }
}
```

```
private static void split(
```



```
File file, long size) throws Exception {
//d:/abc/a.mp4
//[d:/abc/a.mp4_split]
//准备文件夹
File dir = zbwjj(file);
//原文件文件名
String name = file.getName();
//两个计数变量
long byteCount = 0;
int fileCount = 0;

BufferedInputStream in =
    new BufferedInputStream(
        new FileInputStream(file));
BufferedOutputStream out = null;
int b;
while((b = in.read()) != -1) {
    //如果没有输出流, 或起一个文件满了
    if(out == null || byteCount == size) {
        if(byteCount == size) {
            out.close();//关闭前一个输出流
        }
        out = new BufferedOutputStream(
            new FileOutputStream(
                new File(dir,
                    name+"."+(++fileCount))));
        byteCount = 0;
    }
    out.write(b);
    byteCount++;
}
in.close();
out.close();
}
```

```
private static File zbwjj(File file) {
    File dir = new File(
        file.getAbsolutePath()+"_split");

    if(! dir.exists()) { //文件夹不存在
        dir.mkdirs();//创建
    } else { //文件夹已经存在
        clear(dir);//清空
    }
    return dir;
}

private static void clear(File dir) {
    File[] files = dir.listFiles();
    if(files == null) {
        return;
    }
    for (File f : files) {
        if(f.isFile()) { //f是文件
            f.delete();
        } else { //f是文件夹
            clear(f);//清空f文件夹
            f.delete();//删除f文件夹
        }
    }
}
```

```
}  
}
```

5 ObjectOutputStream / ObjectOutputStream

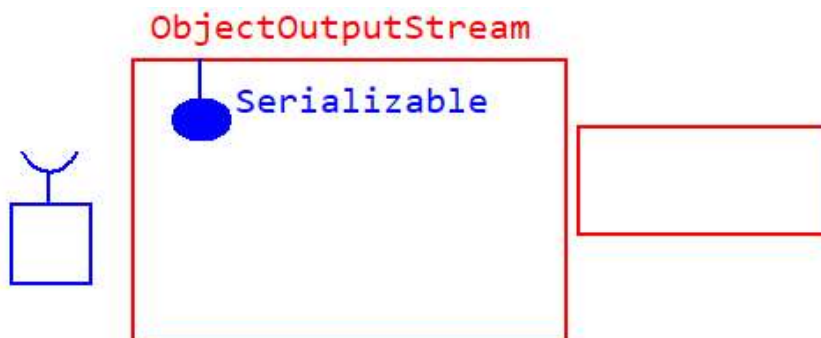
- 对象的序列化、反序列化
- 序列化
把对象的信息，按照固定的字节格式，转成一串字节值，进行输出

day1404.Student

id	5
name	zs
gender	M
age	12



- 方法
 - writeObject(Object obj)
把对象，变成一串字节序列，输出
 - readObject()
读取序列化数据，反序列化恢复对象
- 被序列化的对象，必须实现 Serializable 接口



- 不序列化的成员
 - static，属于类，不属于对象，不会随对象一起被序列化输出
 - transient，临时，只在程序运行期间，在内存中临时存在，不会随对象序列化，被持久的保存
- 序列化版本号
static final long serialVersionUID
 - 用来控制，旧版本的数据，不允许恢复成新版本的类型
 - 如果不定义版本号，编译器会根据类的定义信息，自动生成一个版本号

项目: day1403_序列化
类: day1403.Test1
Student

Test1

```
package day1403;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;

public class Test1 {
    public static void main(String[] args) throws Exception {
        Student s =
            new Student(9527, "唐伯虎", "男", 19);

        /*
         * OOS--FOS--f2
         */
        ObjectOutputStream out =
            new ObjectOutputStream(
                new FileOutputStream("d:/abc/f2"));
        //序列化输出学生对象
        out.writeObject(s);

        out.close();
    }
}
```

Test2

```
package day1403;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.ObjectInputStream;

public class Test2 {
    public static void main(String[] args) throws Exception {
        /*
         * OIS--FIS--f2
         */
        ObjectInputStream in =
            new ObjectInputStream(
                new FileInputStream("d:/abc/f2"));
        //反序列化, 读取恢复学生对象
        Student s = (Student) in.readObject();

        in.close();
        System.out.println(s);
    }
}
```

Student

```
package day1403;

import java.io.Serializable;
/*
 * Serializable 接口
 * 空接口,
 * 标识接口, 标识一个对象, 允许被序列化
 */
public class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id;
    private String name;
    private static String gender;
    private transient int age;

    public Student() {
    }
    public Student(int id, String name, String gender, int age) {
        this.id = id;
        this.name = name;
        this.gender = gender;
        this.age = age;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getGender() {
        return gender;
    }
    public void setGender(String gender) {
        this.gender = gender;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + ", gender=" + gender + ", age=" +
age + "]\n";
    }
}
```

6 作业

- 重写
 - day1402_文件拆分合并
- 完成文件合并
 - 输入拆分文件的存放文件夹: d:/a.jpg_split
 - 输入合并的目标文件: d:/a2.jpg
- ◆ 文件列表之后, 要对文件新型重新排序, 按数字后缀顺序排列

