1.IO流:将数据从一个地方传输另一个地方.

2.流的分类:

2.1:按输出方向分:以程序为参数物.

输入流:将文件中数据读取到程序中.

注意:如果文件不存在,输入流会抛异常(报错).

输出流:将程序中数据写入到文件中.

注意:如果文件不存在,先创建文件,再向文件中写入数据.

2.2:按单元分:bit(位),byte,kb,mb,g,t,p

字节流:以字节为单元传输的流.

作用:用来传输部分文本文件,图片,视频,二进制文件.

字符流:以字符为单元传输的流.

作用:用来传输文本文件.

2.3:按功能分:

节点流:原始流.

处理流:封装了原始流.

注意:所有处理流都用了装饰者模型.

3.装饰者模式:将原有类的对象进行封装,使其功能更强大.

4.基本字节流(原始字节流)

4.1:字节输入流:InputStream->FileInputStream

4.1:一个字节一个字节的读取

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

```
//声明一个流对象
FileInputStream fis=null;
try {
    //1.创建流对象
    fis=new FileInputStream("bb\\b.txt");
    /*2.用流来读取文件中内容,一个字节一个字节的读取,*/
```

```
            //返回的是读取的内容,先读取一次
            int content=fis.read();
            //接着读
            while (content!=-1) {
                //将读取的字节内容转换为字符输出
                System.out.print((char)content);
                //接着读
                content=fis.read();
            }
            System.out.println("读取完毕!");
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }finally {
            //3.关流
            if (fis!=null) {
                fis.close();
            }
        }
}
```

4.2:按字节数组的读取

```
        eg:public static void main(String[] args) throws IOException {
        //声明流对象
        FileInputStream fis=null;

        try {
            //1.创建流对象
            fis=new FileInputStream("bb\\b.txt");
            /*2.用流对象读取内容*/
            //准备一个数组
            byte[] b=new byte[5];
            //先读取一次,返回的是读取的长度,读取的内容存在数组中.
            int len=fis.read(b);
```

```
            //接着读
            while (len!=-1) {
                System.out.println("读取的长度:"+len);
                //先将读取的内容转换为String类型再输出
                String s1=new String(b, 0, len);
                System.out.println(s1);
                //接着读
                len=fis.read(b);
            }
            System.out.println("读取完毕");
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            //3.关闭对象
            if (fis!=null) {
                fis.close();
            }
        }
}
```

4.2:字节输出流:OutputStream->FileOutputStream

```
    eg:public static void main(String[] args) throws IOException {
    //声明流对象
    FileOutputStream fos=null;

    try {
        //1.创建流对象,第二个参数是否向文件末尾追加内容
        fos=new FileOutputStream("bb\\c.txt",true);
        /*2.用流对象向文件中写入内容*/
        fos.write("Hello Java".getBytes());
        fos.write("Hello Stream".getBytes());
        fos.write("我是千锋人".getBytes());
        System.out.println("写入完毕");
    } catch (Exception e) {
```

```
                e.printStackTrace();
            }finally {
                //3.关流
                if (fos!=null) {
                    fos.close();
                }
            }
        }
```

4.3:文件的拷贝

```
            eg:public static void main(String[] args) throws IOException {
            //声明流对象
            FileInputStream fis=null;
            FileOutputStream fos=null;

            try {
                //1.创建输入流和输出流对象
                fis=new
FileInputStream("C:\\Users\\sx\\Desktop\\image\\4.jpg");
                fos=new FileOutputStream("bb\\a.jpg");
                /*用流边读取边写入*/
                //准备一个数组
                byte[] b=new byte[1024];
                //先读取一次,返回读取的长度,读取的内容存到数组中
                int len=fis.read(b);
                //判断,接着读
                while (len!=-1) {
                    //将读取的内容写入到文件中
                    fos.write(b, 0, len);
                    //接着读
                    len=fis.read(b);
                }
                System.out.println("图片拷贝成功");
            } catch (Exception e) {
```

```
                e.printStackTrace();
            }finally {
                //3.关流
                if (fis!=null) {
                    fis.close();
                }
                if (fos!=null) {
                    fos.close();
                }
            }
        }
```

5.字节缓冲流:处理流封装原始字节流

5.1:字节缓冲输入流:FilterInputStream->BufferedInputStream

```
            eg:public static void main(String[] args) throws IOException {
            //声明流对象
            BufferedInputStream bis=null;

            try {
                //1.创建流对象
                bis=new BufferedInputStream(new
FileInputStream("aa\\a1.txt"));
                /*2.用流对象调用方法读取文件中内容*/
                //准备一个数组
                byte[] b=new byte[10];
                //先读取一次,返回的是读取的长度,读取的内容存在数组中
                int len=bis.read(b);
                //接着读
                while (len!=-1) {
                    //将读取的内容转换为String,再输出
                    String s1=new String(b, 0, len);
                    System.out.println(s1);
                    //接着读取
                    len=bis.read(b);
```

```
            }
            System.out.println("读取完毕!");

        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            //3.关流
            if (bis!=null) {
                bis.close();
            }
        }
    }
```

5.2:字节缓冲输出流:FilterOutputStream->BufferedOutputStream

```
        eg:public static void main(String[] args) throws IOException {
        //声明流对象
        BufferedOutputStream bos=null;

        try {
            //1.创建流对象,可以向文件中追加内容
            bos=new BufferedOutputStream(new
FileOutputStream("aa\\a1.txt",true));
            /*2.用流对象向文件中写入内容*/
            bos.write("今天好像没有太阳".getBytes());
            //刷新缓冲区
            bos.flush();
            bos.write("elephent".getBytes());
            //刷新缓冲区
            bos.flush();
            System.out.println("写入成功");
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            //3.关流
```

```
                if (bos!=null) {
                        bos.close();
                }
        }
    }
```

5.3:字节缓冲流的拷贝

```
        eg:public static void main(String[] args) throws IOException {
        //声明流对象
        BufferedInputStream bis=null;
        BufferedOutputStream bos=null;

        try {
                //1.创建流对象
                bis=new BufferedInputStream(new
FileInputStream("C:\\Users\\sx\\Desktop\\image\\b.jpg"));
                bos=new BufferedOutputStream(new
FileOutputStream("aa\\a2.jpg"));
                /*2.用对象来边读取边写入*/
                //准备一个数组
                byte[] b=new byte[1024];
                //声明一个变量存读取的长度
                int len;
                while ((len=bis.read(b))!=-1) {
                        //将读取的内容写入到文件中
                        bos.write(b, 0, len);
                        //刷新缓存
                        bos.flush();
                }
                System.out.println("拷贝成功");
        } catch (Exception e) {
                e.printStackTrace();
        }finally {
                //3.关流
```

```
                if (bis!=null) {
                        bis.close();
                }
                if (bos!=null) {
                        bos.close();
                }
        }
}
```