1.File类

1.1File类概述和构造方法【应用】

- File类介绍
  - 它是文件和目录路径名的抽象表示
  - 文件和目录是可以通过File封装成对象的
  - 对于File而言，其封装的并不是一个真正存在的文件，仅仅是一个路径名而已。它可以是存在的，也可以是不存在的。将来是要通过具体的操作把这个路径的内容转换为具体存在的
- File类的构造方法

方法名	说明
File(String pathname)	通过将给定的路径名字符串转换为抽象路径名来创建新的 File实例
File(String parent, String child)	从父路径名字符串和子路径名字符串创建新的 File实例
File(File parent, String child)	从父抽象路径名和子路径名字符串创建新的 File实例

- 示例代码

```
public class FileDemo01 {
    public static void main(String[] args) {
        //File(String pathname)：通过将给定的路径名字符串转换为抽象路径名来创建新的 File实例。
        File f1 = new File("E:\\itcast\\java.txt");
        System.out.println(f1);

        //File(String parent, String child)：从父路径名字符串和子路径名字符串创建新的File实例。
        File f2 = new File("E:\\itcast","java.txt");
        System.out.println(f2);

        //File(File parent, String child)：从父抽象路径名和子路径名字符串创建新的 File实例。
        File f3 = new File("E:\\itcast");
        File f4 = new File(f3,"java.txt");
        System.out.println(f4);
    }
}
```

1.2File类创建功能【应用】

- 方法分类

方法名	说明
public boolean createNewFile()	当具有该名称的文件不存在时，创建一个由该抽象路径名命名的新空文件
public boolean mkdir()	创建由此抽象路径名命名的目录
public boolean mkdirs()	创建由此抽象路径名命名的目录，包括任何必需但不存在的父目录

- 示例代码

```
public class FileDemo02 {
    public static void main(String[] args) throws IOException {
        //需求1：我要在E:\\itcast目录下创建一个文件java.txt
        File f1 = new File("E:\\itcast\\java.txt");
        System.out.println(f1.createNewFile());
        System.out.println("--------");

        //需求2：我要在E:\\itcast目录下创建一个目录JavaSE
        File f2 = new File("E:\\itcast\\JavaSE");
        System.out.println(f2.mkdir());
        System.out.println("--------");

        //需求3：我要在E:\\itcast目录下创建一个多级目录JavaWEB\\HTML
        File f3 = new File("E:\\itcast\\JavaWEB\\HTML");
//        System.out.println(f3.mkdir());
        System.out.println(f3.mkdirs());
        System.out.println("--------");

        //需求4：我要在E:\\itcast目录下创建一个文件javase.txt
        File f4 = new File("E:\\itcast\\javase.txt");
//        System.out.println(f4.mkdir());
        System.out.println(f4.createNewFile());
    }
}
```

1.3File类判断和获取功能【应用】

- 判断功能

方法名	说明
public boolean isDirectory()	测试此抽象路径名表示的File是否为目录
public boolean isFile()	测试此抽象路径名表示的File是否为文件
public boolean exists()	测试此抽象路径名表示的File是否存在

- 获取功能

方法名	说明
public String getAbsolutePath()	返回此抽象路径名的绝对路径名字符串
public String getPath()	将此抽象路径名转换为路径名字符串
public String getName()	返回由此抽象路径名表示的文件或目录的名称
public String[] list()	返回此抽象路径名表示的目录中的文件和目录的名称字符串数组
public File[] listFiles()	返回此抽象路径名表示的目录中的文件和目录的File对象数组

- 示例代码

```
public class FileDemo04 {
    public static void main(String[] args) {
        //创建一个File对象
        File f = new File("myFile\\java.txt");

//        public boolean isDirectory()：测试此抽象路径名表示的File是否为目录
//        public boolean isFile()：测试此抽象路径名表示的File是否为文件
//        public boolean exists()：测试此抽象路径名表示的File是否存在
        System.out.println(f.isDirectory());
        System.out.println(f.isFile());
        System.out.println(f.exists());

//        public String getAbsolutePath()：返回此抽象路径名的绝对路径名字符串
//        public String getPath()：将此抽象路径名转换为路径名字符串
//        public String getName()：返回由此抽象路径名表示的文件或目录的名称
        System.out.println(f.getAbsolutePath());
        System.out.println(f.getPath());
        System.out.println(f.getName());
        System.out.println("--------");

//        public String[] list()：返回此抽象路径名表示的目录中的文件和目录的名称字符串数组
//        public File[] listFiles()：返回此抽象路径名表示的目录中的文件和目录的File对象数组
        File f2 = new File("E:\\itcast");

        String[] strArray = f2.list();
        for(String str : strArray) {
            System.out.println(str);
        }
        System.out.println("--------");

        File[] fileArray = f2.listFiles();
        for(File file : fileArray) {
//            System.out.println(file);
//            System.out.println(file.getName());
            if(file.isFile()) {
                System.out.println(file.getName());
            }
```

```
        }
    }
}
```

1.4File类删除功能【应用】

- 方法分类

方法名	说明
public boolean delete()	删除由此抽象路径名表示的文件或目录

- 示例代码

```
public class FileDemo03 {
    public static void main(String[] args) throws IOException {
//        File f1 = new File("E:\\itcast\\java.txt");
        //需求1：在当前模块目录下创建java.txt文件
        File f1 = new File("myFile\\java.txt");
//        System.out.println(f1.createNewFile());

        //需求2：删除当前模块目录下的java.txt文件
        System.out.println(f1.delete());
        System.out.println("--------");

        //需求3：在当前模块目录下创建itcast目录
        File f2 = new File("myFile\\itcast");
//        System.out.println(f2.mkdir());

        //需求4：删除当前模块目录下的itcast目录
        System.out.println(f2.delete());
        System.out.println("--------");

        //需求5：在当前模块下创建一个目录itcast,然后在该目录下创建一个文件java.txt
        File f3 = new File("myFile\\itcast");
//        System.out.println(f3.mkdir());
        File f4 = new File("myFile\\itcast\\java.txt");
//        System.out.println(f4.createNewFile());

        //需求6：删除当前模块下的目录itcast
        System.out.println(f4.delete());
        System.out.println(f3.delete());
    }
}
```

- 绝对路径和相对路径的区别
  - 绝对路径：完整的路径名，不需要任何其他信息就可以定位它所表示的文件。例如：E:\itcast\java.txt
  - 相对路径：必须使用取自其他路径名的信息进行解释。例如：myFile\java.txt

2.递归

2.1递归【应用】

- 递归的介绍
  - 以编程的角度来看，递归指的是方法定义中调用方法本身的现象
  - 把一个复杂的问题层层转化为一个与原问题相似的规模较小的问题来求解
  - 递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算
- 递归的基本使用

```
public class DiGuiDemo {
    public static void main(String[] args) {
        //回顾不死神兔问题，求第20个月兔子的对数
        //每个月的兔子对数：1,1,2,3,5,8，...
        int[] arr = new int[20];

        arr[0] = 1;
        arr[1] = 1;

        for (int i = 2; i < arr.length; i++) {
            arr[i] = arr[i - 1] + arr[i - 2];
        }
        System.out.println(arr[19]);
        System.out.println(f(20));
    }

    /*
        递归解决问题，首先就是要定义一个方法：
            定义一个方法f(n)：表示第n个月的兔子对数
            那么，第n-1个月的兔子对数该如何表示呢？f(n-1)
            同理，第n-2个月的兔子对数该如何表示呢？f(n-2)

        StackOverflowError:当堆栈溢出发生时抛出一个应用程序递归太深
     */
    public static int f(int n) {
        if(n==1 || n==2) {
            return 1;
        } else {
            return f(n - 1) + f(n - 2);
        }
    }
}
```

- 递归的注意事项
  - 递归一定要有出口。否则内存溢出
  - 递归虽然有出口，但是递归的次数也不宜过多。否则内存溢出

2.2递归求阶乘【应用】

- 案例需求

  用递归求5的阶乘，并把结果在控制台输出

- 代码实现

```java
public class DiGuiDemo01 {
    public static void main(String[] args) {
        //调用方法
        int result = jc(5);
        //输出结果
        System.out.println("5的阶乘是：" + result);
    }

    //定义一个方法，用于递归求阶乘，参数为一个int类型的变量
    public static int jc(int n) {
        //在方法内部判断该变量的值是否是1
        if(n == 1) {
            //是：返回1
            return 1;
        } else {
            //不是：返回n*(n-1)!
            return n*jc(n-1);
        }
    }
}
```

2.3递归遍历目录【应用】

- 案例需求

  给定一个路径(E:\itcast)，通过递归完成遍历该目录下所有内容，并把所有文件的绝对路径输出在控制台

- 代码实现

```java
public class DiGuiDemo02 {
    public static void main(String[] args) {
        //根据给定的路径创建一个File对象
//        File srcFile = new File("E:\\itcast");
        File srcFile = new File("E:\\itheima");

        //调用方法
        getAllFilePath(srcFile);
    }

    //定义一个方法，用于获取给定目录下的所有内容，参数为第1步创建的File对象
    public static void getAllFilePath(File srcFile) {
        //获取给定的File目录下所有的文件或者目录的File数组
        File[] fileArray = srcFile.listFiles();
        //遍历该File数组，得到每一个File对象
        if(fileArray != null) {
            for(File file : fileArray) {
                //判断该File对象是否是目录
                if(file.isDirectory()) {
                    //是：递归调用
                    getAllFilePath(file);
                } else {
                    //不是：获取绝对路径输出在控制台
                    System.out.println(file.getAbsolutePath());
```

```
                }
            }
        }
    }
}
```

3.IO流

3.1 IO流概述和分类【理解】

- IO流介绍
  - IO：输入/输出(Input/Output)
  - 流：是一种抽象概念，是对数据传输的总称。也就是说数据在设备间的传输称为流，流的本质是数据传输
  - IO流就是用来处理设备间数据传输问题的。常见的应用：文件复制；文件上传；文件下载
- IO流的分类
  - 按照数据的流向
    - 输入流：读数据
    - 输出流：写数据
  - 按照数据类型来分
    - 字节流
      - 字节输入流
      - 字节输出流
    - 字符流
      - 字符输入流
      - 字符输出流
- IO流的使用场景
  - 如果操作的是纯文本文件，优先使用字符流
  - 如果操作的是图片、视频、音频等二进制文件。优先使用字节流
  - 如果不确定文件类型，优先使用字节流。字节流是万能的流

3.2字节流写数据【应用】

- 字节流抽象基类
  - InputStream：这个抽象类是表示字节输入流的所有类的超类
  - OutputStream：这个抽象类是表示字节输出流的所有类的超类
  - 子类名特点：子类名称都是以其父类名作为子类名的后缀
- 字节输出流
  - FileOutputStream(String name)：创建文件输出流以指定的名称写入文件
- 使用字节输出流写数据的步骤
  - 创建字节输出流对象(调用系统功能创建了文件，创建字节输出流对象，让字节输出流对象指向文件)
  - 调用字节输出流对象的写数据方法
  - 释放资源(关闭此文件输出流并释放与此流相关联的任何系统资源)
- 示例代码

```
```

```java
public class FileOutputStreamDemo01 {
    public static void main(String[] args) throws IOException {
        //创建字节输出流对象
        //FileOutputStream(String name)：创建文件输出流以指定的名称写入文件
        FileOutputStream fos = new FileOutputStream("myByteStream\\fos.txt");
        /*
            做了三件事情：
                A:调用系统功能创建了文件
                B:创建了字节输出流对象
                C:让字节输出流对象指向创建好的文件
         */

        //void write(int b)：将指定的字节写入此文件输出流
        fos.write(97);
//        fos.write(57);
//        fos.write(55);

        //最后都要释放资源
        //void close()：关闭此文件输出流并释放与此流相关联的任何系统资源。
        fos.close();
    }
}
```

3.3字节流写数据的三种方式【应用】

- 写数据的方法分类

方法名	说明
void write(int b)	将指定的字节写入此文件输出流 一次写一个字节数据
void write(byte[] b)	将 b.length字节从指定的字节数组写入此文件输出流 一次写一个字节数组数据
void write(byte[] b, int off, int len)	将 len字节从指定的字节数组开始，从偏移量off开始写入此文件输出流 一次写一个字节数组的部分数据

- 示例代码

```java
public class FileOutputStreamDemo02 {
    public static void main(String[] args) throws IOException {
        //FileOutputStream(String name)：创建文件输出流以指定的名称写入文件
        FileOutputStream fos = new FileOutputStream("myByteStream\\fos.txt");
        //new File(name)
//        FileOutputStream fos = new FileOutputStream(new File("myByteStream\\fos.txt"));

        //FileOutputStream(File file)：创建文件输出流以写入由指定的 File对象表示的文件
//        File file = new File("myByteStream\\fos.txt");
//        FileOutputStream fos2 = new FileOutputStream(file);
//        FileOutputStream fos2 = new FileOutputStream(new File("myByteStream\\fos.txt"));
```

```
12
13          //void write(int b)：将指定的字节写入此文件输出流
14  //          fos.write(97);
15  //          fos.write(98);
16  //          fos.write(99);
17  //          fos.write(100);
18  //          fos.write(101);
19
20  //          void write(byte[] b)：将 b.length字节从指定的字节数组写入此文件输出流
21  //          byte[] bys = {97, 98, 99, 100, 101};
22          //byte[] getBytes()：返回字符串对应的字节数组
23          byte[] bys = "abcde".getBytes();
24  //          fos.write(bys);
25
26          //void write(byte[] b, int off, int len)：将 len字节从指定的字节数组开始，从
    偏移量off开始写入此文件输出流
27  //          fos.write(bys,0,bys.length);
28          fos.write(bys,1,3);
29
30          //释放资源
31          fos.close();
32      }
33  }
```

3.4字节流写数据的两个小问题【应用】

- 字节流写数据如何实现换行
  - windows:\r\n
  - linux:\n
  - mac:\r
- 字节流写数据如何实现追加写入
  - public FileOutputStream(String name,boolean append)
  - 创建文件输出流以指定的名称写入文件。如果第二个参数为true ，则字节将写入文件的末尾而不是开头
- 示例代码

```
1   public class FileOutputStreamDemo03 {
2       public static void main(String[] args) throws IOException {
3           //创建字节输出流对象
4   //          FileOutputStream fos = new FileOutputStream("myByteStream\\fos.txt");
5           FileOutputStream fos = new
    FileOutputStream("myByteStream\\fos.txt",true);
6
7           //写数据
8           for (int i = 0; i < 10; i++) {
9               fos.write("hello".getBytes());
10              fos.write("\r\n".getBytes());
11          }
12
13          //释放资源
14          fos.close();
15      }
```

```
}
```

3.5字节流写数据加异常处理【应用】

- 异常处理格式
  - try-catch-finally

```
try{
    可能出现异常的代码;
}catch(异常类名 变量名){
    异常的处理代码;
}finally{
    执行所有清除操作;
}
```

  - finally特点
    - 被finally控制的语句一定会执行，除非JVM退出
- 示例代码

```
public class FileOutputStreamDemo04 {
    public static void main(String[] args) {
        //加入finally来实现释放资源
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream("myByteStream\\fos.txt");
            fos.write("hello".getBytes());
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if(fos != null) {
                try {
                    fos.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

3.6字节流读数据(一次读一个字节数据)【应用】

- 字节输入流
  - FileInputStream(String name)：通过打开与实际文件的连接来创建一个FileInputStream ，该文件由文件系统中的路径名name命名
- 字节输入流读取数据的步骤
  - 创建字节输入流对象
  - 调用字节输入流对象的读数据方法
  - 释放资源

- 示例代码

```
public class FileInputStreamDemo01 {
    public static void main(String[] args) throws IOException {
        //创建字节输入流对象
        //FileInputStream(String name)
        FileInputStream fis = new FileInputStream("myByteStream\\fos.txt");

        int by;
        /*
            fis.read()：读数据
            by=fis.read()：把读取到的数据赋值给by
            by != -1：判断读取到的数据是否是-1
         */
        while ((by=fis.read())!=-1) {
            System.out.print((char)by);
        }

        //释放资源
        fis.close();
    }
}
```

3.7字节流复制文本文件【应用】

- 案例需求

  把“E:\itcast\窗里窗外.txt”复制到模块目录下的“窗里窗外.txt”

- 实现步骤
  - 复制文本文件，其实就把文本文件的内容从一个文件中读取出来(数据源)，然后写入到另一个文件中(目的地)
  - 数据源：

    E:\itcast\窗里窗外.txt --- 读数据 --- InputStream --- FileInputStream
  - 目的地：

    myByteStream\窗里窗外.txt --- 写数据 --- OutputStream --- FileOutputStream

- 代码实现

```
public class CopyTxtDemo {
    public static void main(String[] args) throws IOException {
        //根据数据源创建字节输入流对象
        FileInputStream fis = new FileInputStream("E:\\itcast\\窗里窗外.txt");
        //根据目的地创建字节输出流对象
        FileOutputStream fos = new FileOutputStream("myByteStream\\窗里窗
外.txt");

        //读写数据，复制文本文件(一次读取一个字节，一次写入一个字节)
        int by;
        while ((by=fis.read())!=-1) {
            fos.write(by);
```

```
        }

        //释放资源
        fos.close();
        fis.close();
    }
}
```

3.8字节流读数据(一次读一个字节数组数据)【应用】

- 一次读一个字节数组的方法
  - public int read(byte[] b)：从输入流读取最多b.length个字节的数据
  - 返回的是读入缓冲区的总字节数,也就是实际的读取字节个数
- 示例代码

```
public class FileInputStreamDemo02 {
    public static void main(String[] args) throws IOException {
        //创建字节输入流对象
        FileInputStream fis = new FileInputStream("myByteStream\\fos.txt");

        /*
            hello\r\n
            world\r\n

            第一次：hello
            第二次：\r\nwor
            第三次：ld\r\nr

         */

        byte[] bys = new byte[1024]; //1024及其整数倍
        int len;
        while ((len=fis.read(bys))!=-1) {
            System.out.print(new String(bys,0,len));
        }

        //释放资源
        fis.close();
    }
}
```

3.9字节流复制图片【应用】

- 案例需求

  把“E:\itcast\mn.jpg”复制到模块目录下的“mn.jpg”

- 实现步骤
  - 根据数据源创建字节输入流对象
  - 根据目的地创建字节输出流对象
  - 读写数据，复制图片(一次读取一个字节数组，一次写入一个字节数组)
  - 释放资源

- 代码实现

```java
public class CopyJpgDemo {
    public static void main(String[] args) throws IOException {
        //根据数据源创建字节输入流对象
        FileInputStream fis = new FileInputStream("E:\\itcast\\mn.jpg");
        //根据目的地创建字节输出流对象
        FileOutputStream fos = new FileOutputStream("myByteStream\\mn.jpg");

        //读写数据，复制图片(一次读取一个字节数组，一次写入一个字节数组)
        byte[] bys = new byte[1024];
        int len;
        while ((len=fis.read(bys))!=-1) {
            fos.write(bys,0,len);
        }

        //释放资源
        fos.close();
        fis.close();
    }
}
```