IO流：文件和字节流

回顾

```

```

今天任务

```
1.File类的使用
        1.1 File类的作用
        1.2 File类的构造方法
        1.3 File类文件属性方法
        1.4 File类的使用练习
2.IO流
        2.1 什么是IO流
        2.2 流的作用和原理
        2.3 IO流的分类
        2.4 字节输入流
        2.5 字节输出流
        2.6 字符输入流
        2.7 字符输出流
```

教学目标

```
1.了解File类的作用
2.掌握File类的构造方法和常用成员方法
3.掌握流的作用和原理
4.了解流的分类
5.掌握字符流和字节流的使用
```

第一节 File类

1.1 File类作用

```
在java程序中，对磁盘文件进行描述的类。文件和目录路径名的抽象表示形式。
```

1.2 File类的常用构造方法

方法名	描述
File(File parent, String child)	根据 parent 抽象路径名和 child 路径名字符串创建一个新 File 实例。
File(String pathname)	通过将给定路径名字符串转换为抽象路径名来创建一个新 File 实例。
File(String parent, String child)	根据 parent 路径名字符串和 child 路径名字符串创建一个新 File 实例。

1.3 File类文件属性方法

属性：

```
static String pathSeparator：与系统有关的路径分隔符，为了方便，它被表示为一个字符串（;）。
static char pathSeparatorChar：;
static String Separator：与系统有关的默认名称分隔符，为了方便，它被表示为一个字符串（\）。
static char SeparatorChar：\
```

```
    //分号
    System.out.println(File.pathSeparator);
    System.out.println(File.pathSeparatorChar);
    //反斜杠
    System.out.println(File.separator);
    System.out.println(File.separatorChar);
```

方法：

返回值	方法名/描述
boolean	canExecute() 测试应用程序是否可以执行此抽象路径名表示的文件。
boolean	canRead() 测试应用程序是否可以读取此抽象路径名表示的文件。
boolean	canWrite() 测试应用程序是否可以修改此抽象路径名表示的文件。
int	compareTo(File) 按字母顺序比较两个抽象路径名。
boolean	exists() 测试此抽象路径名表示的文件或目录是否存在。
boolean	createNewFile() 当且仅当不存在具有此抽象路径名指定名称的文件时，不可分地创建个新的空文件。
boolean	delete() 删除此抽象路径名表示的文件或目录。
File	getAbsoluteFile() 返回此抽象路径名的绝对路径名形式。
String	getAbsolutePath() 返回此抽象路径名的绝对路径名字符串。
File	getCanonicalFile() 返回此抽象路径名的规范形式。
String	getCanonicalPath() 返回此抽象路径名的规范路径名字符串。
String	getName() 返回由此抽象路径名表示的文件或目录的名称。
String	getParent() 返回此抽象路径名父目录的路径名字符串；如果此路径名没有指定父目录，则返回 null。
File	getParentFile() 返回此抽象路径名父目录的抽象路径名；如果此路径名没有指定父目录则返回 null。
String	getPath() 将此抽象路径名转换为一个路径名字符串。
boolean	isDirectory() 测试此抽象路径名表示的文件是否是一个目录。
boolean	isFile() 测试此抽象路径名表示的文件是否是一个标准文件。
boolean	isHidden() 测试此抽象路径名指定的文件是否是一个隐藏文件。
String[]	list() 返回一个字符串数组，这些字符串指定此抽象路径名表示的目录中的文件和目录。
boolean	mkdir() 创建此抽象路径名指定的目录。
boolean	mkdirs() 创建此抽象路径名指定的目录，包括所有必需但不存在的父目录。
boolean	renameTo(File dest) 重新命名此抽象路径名表示的文件。
File[]	listFiles() 返回一个抽象路径名数组，这些路径名表示此抽象路径名表示的目录中的文件。

代码实现一：

```
        File file = new File("d:\\a.txt");//创建一个文件对象，表示d盘的a.txt文件
        System.out.println(file.canExecute());
        System.out.println(file.canWrite());
        System.out.println(file.canRead());
        System.out.println(file.isHidden());
```

代码实现二：

```
        File file = new File("d:\\a");
        File f = new File("d:\\b.txt");
        System.out.println(file.compareTo(f));
        //exists方法：判断文件抽象路径表示的文件或目录是否存在
        System.out.println(file.exists());
        //createNewFile方法：创建一个新的空文件（若存在则创建失败）
        System.out.println(f.createNewFile());
        //delete方法：只能删除文件和空文件夹，非空文件夹不能使用delete方法删除
        System.out.println(f.delete());
        System.out.println(file.delete());
```

代码实现三：

```java
public class Demo {
    public static void main(String[] args) throws IOException {
        //现有d:\\a\\b\\hello.txt
        //要求：在hello.txt相同的目录下创建出一个world.txt文件
        File f1 = new File("d:\\a\\b\\hello.txt");
        System.out.println(f1.getPath());
        fun(f1);
    }
    //设计一个方法，在某个文件的相同目录下创建出一个新文件（新建文本文档.txt）
    /*
     * 分析：
     *       返回值：不需要
     *       参数：一个File对象
     * getParent方法
     * createNewFile方法
     * 构造方法
     */
    public static void fun(File file) throws IOException {
        //获取父路径
        String parent = file.getParent();
        //创建File对象
        File f = new File(parent, "新建文本文档.txt");
        //创建新文件
        f.createNewFile();
    }
}
```

代码实现四：

```
File file = new File("d:\\a\\hello.txt");
System.out.println(file.exists());
System.out.println(file.isDirectory());//判断一个File对象是否是文件夹
System.out.println(file.isFile());//判断一个File对象是否是文件
String[] files = file.list();//获取文件夹中所有子文件夹和文件的名称（字符串形式）
System.out.println(files.length);
for(String s:files) {
    System.out.println(s);
}
File[] fs = file.listFiles();//获取文件件中所有子文件夹和文件的抽象路径（File对象）
System.out.println(fs.length);
for(File f:fs) {
    System.out.println(f);
}
```

```
File file = new File("d:\\aa\\bb\\cc\\dd\\a.txt");
//在创建一个文件时，需要先判断父目录是否存在，若不存在则创建父目录
File parent = file.getParentFile();
if(!parent.exists()) {
    System.out.println(parent.mkdirs());//创建一个新的空文件夹
}
System.out.println(file.createNewFile());

//对文件重命名
File file = new File("d:\\a.txt");
File f = new File("a.txt");
File f1 = new File("hello\\hello.txt");
System.out.println(file.renameTo(f));//将file表示文件重命名为f时，必须保证file是存在的文件
System.out.println(f.renameTo(f1));
```

1.4 File类的使用练习

代码实现：

```java
public class FileUsageDemo02 {

    public static void main(String[] args) {
        String string  = "C:\\Users\\Administrator\\Desktop\\HZ-J2ee1709\\Day16";
        method1(string);
       method2(string);
    }

    //需求一：列出指定目录下所有子文件夹以及子文件
    public static void method1(String path) {
        File file = new File(path);

        //list
        //子文件或者子文件夹的名称
        String[] arr = file.list();

        for(String str:arr) {
            System.out.println(str);
        }
    }

    /**
     * 需求二：列出指定目录下所有子文件夹以及子文件，要求格式如下：
     * 子文件：    isFile()
     * 。。。。
     * 子文件夹：isDirectory()
     * 。。。。
     */
    public static void method2(String path) {
        File file = new File(path);

        //获取指定路径下所有File对象
        //listFiles()
        File[] files = file.listFiles();

        for(File f:files) {
            if(f.isFile()) {
                System.out.println("子文件：");
                //获取每个文件的名字
                System.out.println(f.getName());
            } else if(f.isDirectory()) {
                System.out.println("子文件夹：");
                System.out.println(f.getName());
            }
        }
    }

    //需求：列出指定目录下所有后缀为.java的文件
    public static void method3(String path) {

        //endsWith
```

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

        String[] arr = file.list();

        for(String str:arr) {
            if(str.endsWith(".java")) {
                System.out.println(str);
            }
        }
    }
}
```

第二节 IO流

2.1 什么是IO流

在工作中，经常回去操作磁盘上的资源，这个过程中实现了数据的输入和输出操作，磁盘上的文件和内存之间进行交互，数据的交互需要有一个媒介或者管道，把这个媒介或者管道就称为IO流，也被称为输入输出流【I:Input O:Output】

2.2 流的作用和原理

流是一组有顺序的，有起点和终点的字节集合，是对数据传输的总称或抽象。即数据在两设备间的传输称为流，流的本质是数据传输，根据数据传输特性将流抽象为各种类，方便更直观的进行数据操作。

2.3 IO流的种类

2.3.1 按照流的流向分：输入流、输出流

输入流：表示将数据读取到java程序（内存）中使用的流。

输出流：表示从java程序（内存）向外传输使用使用的流。

2.3.2 按照数据单位分：字符流、字节流

字节流：一次性传输一个字节数据，将数据以字节的形式传输。

字符流：一次性传输一个字符数据，将数据以字符的形式传输。

2.3.3 按照层次分：节点流、处理流

节点流：可以从或向一个特定的地方(节点)读写数据。

处理流：是对一个已存在的流的连接和封装，通过所封装的流的功能调用实现数据读写。

2.4 字节输入流

2.4.1 InputStream类的常用方法

`InputStream`是一个抽象类，不能实例化对象。

方法名	描述
void close()	关闭此输入流并释放与该流关联的所有系统资源。
int read()	从输入流中读取数据的下一个字节。
int read(byte[] b)	从输入流中读取一定数量的字节，并将其存储在缓冲区数组b中。
int read(byte[] b,int off, int len)	将输入流中最多len个数据字节读入 byte 数组。

2.4.2 文件输入流FileInputStream

```java
public class DemoFileInputStream {
    public static void main(String[] args) {
        //创建被操作文件：此文件必须存在，否则读取时，抛出文件找不到异常
        File file = new File("test\\hello.txt");
        //声明流，不初始化
        FileInputStream fis = null;
        try {
            //初始化流
            fis = new FileInputStream(file);
            //准备数组用来存储数据
            byte[] b = new byte[3];
            //先定义一个变量，初始值是-1
            int i = -1;
            //定义一个计数循环的变量
            int count = 0;
            //while循环读取
            while((i=fis.read(b))!=-1) {
                count++;
                for(int j=0; j<i; j++) {
                    System.out.print((char)b[j]);
                }
            }
            System.out.println(count);//计数：计算循环读取的次数
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally {
            if(fis!=null) {
                try {
                    fis.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
    }
}
```

2.5 字节输出流

2.5.1 OutputStream类的常用方法

`OutputStream`是抽象类，不能实例化对象。

方法名	描述
void close()	关闭此输出流并释放与此流有关的所有系统资源。
void flush()	刷新此输出流并强制写出所有缓冲的输出字节。
void write(byte[] b)	将 b.length 个字节从指定的 byte 数组写入此输出流。
void write(byte[] b,int off, int len)	将指定 byte 数组中从偏移量 off 开始的 len 个字节写入此输出流。
void write(int b)	将指定的字节写入此输出流。

2.5.2 文件输出流FileOutputStream

```
public class TestFileOutputStream {
    public static void main(String[] args) {
        //向文件中写入数据
        //在工程中创建一个test文件夹
        //设计程序，向test\\hello.txt中写入hello world
        //第一步：创建被操作文件对象
        //当向一个文件中写入数据时，若文件不存在，程序会自动创建
        File file = new File("test\\hello.txt");
        FileOutputStream fos = null;
        try {
            //第二步：创建流对象
            fos = new FileOutputStream(file, true);
            //第三步：准备数据
            String str = "hello world";
            byte[] b = str.getBytes();
            System.out.println(b.length);
            //第四步：使用流写入
            fos.write(b);
        }catch(IOException e) {
            e.printStackTrace();
        } finally {
            if(fos!=null) {
                try {
                    //第五步：刷新流，关闭流
                    fos.flush();
                    fos.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
    }
}
```


2.6 字符输入流

Reader类

`Reader`：是所有字符输入流的父类，为一个抽象类，不能实例化对象，使用它的子类`FileReader`类

```java
public class FileReaderUsageDemo {
    public static void main(String[] args) {
        //1.将文件的路径转换为File对象
        File file = new File("file/input1.txt");
        Reader reader = null;

        try {
            //2.实例化一个FileReader的对象
            reader = new FileReader(file);

            //3.定义一个数组
            char[] arr = new char[8];

            //4.定义一个int的变量
            int hasRead = 0;

            //5.循环读取
            while((hasRead = reader.read(arr)) != -1) {
                String result = new String(arr, 0, hasRead);
                System.out.println(result);
            }

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        finally {
            //避免出现空指针异常
            if(reader != null) {
                try {
                    reader.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```


2.7 字符输出流

Writer类

Writer：是所有字符输出流的父类，为一个抽象类，不能实例化对象，使用它的子类FileWriter类

```
public class FileWriterUsageDemo {

    public static void main(String[] args) throws IOException {
        File file = new File("file/output1.txt");

        Writer writer = new FileWriter(file);

        //写入
        //注意：区别于字节流，可以直接传字符串
        writer.write("天边最美的云彩");

        writer.flush();

        writer.close();
    }
}
```

总结

课前默写

1.自定义一个类Person，将多个Person对象添加到TreeSet中，并按照年龄进行降序排序
2.向一个HashMap集合中添加几对键值对，使用增强for循环将key和value遍历出来
3.简述HashMap和HashSet之间的区别

作业

1.总结集合中的内容，完善笔记
2.创建HashMap对象，向集合中加入5个键值对，采用课上的三种方式遍历集合，打印出所有的键值对
3.创建TreeMap对象，key值采用Student自定义类型，value值为String类型，表示学生的喜好，要求按照年龄升序排序，如果年龄相同，则按照成绩降序排序
4.在控制台输入一句英语， 获得每个字母出现的次数，注：每个字符作为key，出现的次数作为value
7.列出指定目录下所有扩展名为.txt的文件
8.递归删除文件，如果文件不存在，则抛出一个文件不存在的自定义异常
9.利用流实现文件内容的拷贝

面试题

1.什么是IO流，有什么作用
2.IO流都有哪些分类？依据是什么？