java.io.File类：文件和目录路径名的抽象表达形式

与操作系统无关的类

file文件，directory文件夹/目录，path路径

```
/*
    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 与系统有关的路径分隔符。

    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 与系统有关的默认名称分隔符。
 */
/*
    路径:
        绝对路径:是一个完整的路径
            以盘符(c:,D:)开始的路径
                c:\\a.txt
                C:\\Users\itcast\\IdeaProjects\\shungyuan\\123.txt
                D:\\demo\\b.txt
        相对路径:是一个简化的路径
            相对指的是相对于当前项目的根目录(C:\\Users\itcast\\IdeaProjects\\shungyuan)
            如果使用当前项目的根目录,路径可以简化书写
            C:\\Users\itcast\\IdeaProjects\\shungyuan\\123.txt-->简化为: 123.txt(可以省略项目的根目录)
        注意:
            1.路径是不区分大小写
            2.路径中的文件名称分隔符windows使用反斜杠,反斜杠是转义字符,两个反斜杠代表一个普通的反斜杠
 */
/*
    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。
    参数:
        String pathname:字符串的路径名称
        路径可以是以文件结尾,也可以是以文件夹结尾
        路径可以是相对路径,也可以是绝对路径
        路径可以是存在,也可以是不存在
        创建File对象,只是把字符串路径封装为File对象,不考虑路径的真假情况
 */
private static void show01() {
    File f1 = new File( pathname: "C:\\Users\\itcast\\IdeaProjects\\shungyuan\\a.txt");
    System.out.println(f1);//重写了Object类的toString方法 C:\Users\itcast\IdeaProjects\shungyuan\a.txt

    File f2 = new File( pathname: "C:\\Users\\itcast\\IdeaProjects\\shungyuan");
    System.out.println(f2);//C:\Users\itcast\IdeaProjects\shungyuan

    File f3 = new File( pathname: "b.txt");
    System.out.println(f3);//b.txt
}
```

```
public class Demo02File {
    public static void main(String[] args) {
        /*
            File类的构造方法
         */
        //show02("c:\\","a.txt");//c:\a.txt
        //show02("d:\\","a.txt");//d:\a.txt
    }

    /*
        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。
        参数:把路径分成了两部分
            String parent:父路径
            String child:子路径
        好处:
            父路径和子路径,可以单独书写,使用起来非常灵活;父路径和子路径都可以变化
     */
```

```java
/*
    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。
    参数:把路径分成了两部分
        File parent:父路径
        String child:子路径
    好处:
        父路径和子路径,可以单独书写,使用起来非常灵活;父路径和子路径都可以变化
        父路径是File类型,可以使用File的方法对路径进行一些操作,再使用路径创建对象
 */
private static void show03() {
    File parent = new File( pathname: "c:\\");
    File file = new File(parent, child: "hello.java");
    System.out.println(file);//c:\hello.java
}
```

public String getAbsolutePath(): 返回此File的绝对路径名字符串
public String getPath(): 将此File转换为路径名字符串
public String getName(): 返回此File表示的文件或目录名称(路径结尾名)
public long length(): 返回此File表示的文件的大小(文件夹结尾或路径不存在,返回0,字节为单位)

判断方法:

public boolean exists(): 此File表示的文件或目录是否存在
public boolean isDirectory(): 此File表示的是否为目录
public boolean isFile(): 此File表示的是否为文件

创建,删除方法

public boolean createNewFile(): 当且仅当具有该名称的文件尚不存在时,创建一个新文件.
public boolean delete(): 删除File表示的文件或目录
public boolean mkdir(): 创建由此File表示的单级文件夹 (若已存在,返回false;否则创建并返回true)
public boolean mkdirs(): 创建由此File表示的单/多级文件夹.
(路径必须存在,否则抛出异常IOException)

遍历方法:

public String[] list(): 返回表示该File目录中所有子文件或目录的名称的String数组
public File[] listFiles(): 返回表示该File目录中所有子文件或目录的File数组.
(若路径不存在,会报NullPointerException).

文件过滤器优化

java.io.FileFilter 接口,是File的过滤器.
File[] listFiles(FileFilter filter): 用来过滤File对象
抽象方法: boolean accept(File pathname): 测试指定对象名是否应该在某个路径名列表中.

File[] listFiles(FilenameFilter filter): 用来过滤File名称
抽象方法: boolean accept(File dir, String name): 测试指定文件是否应该在某一文件列表中.

需求:
遍历c:\\abc文件夹,及abc文件夹的子文件夹
只要.java结尾的文件
c:\\abc
c:\\abc\\abc.txt
c:\\abc\\abc.java
c:\\abc\\a
c:\\abc\\a\\a.jpg
c:\\abc\\a\\a.java
c:\\abc\\b
c:\\abc\\b\\b.java
c:\\abc\\b\\b.txt

必须明确两件事情:
1.过滤器中的accept方法是谁调用的
2.accept方法的参数pathname是什么?

```java
public class Demo01Filter {
    public static void main(String[] args) {
        File file = new File( pathname: "c:\\abc");
        getAllFile(file);
    }

public static void getAllFile(File dir){
    File[] files = dir.listFiles(new FileFilterImpl());//传递过滤器对象

    for (File f : files) {
        //对遍历得到的File对象f进行判断,判断是否是文件夹
        if(f.isDirectory()){
            //f是一个文件夹,则继续遍历这个文件夹
            //我们发现getAllFile方法就是传递文件夹,遍历文件夹的方法
            //所以直接调用getAllFile方法即可:递归(自己调用自己)
            getAllFile(f);
        }else{
            //f是一个文件,直接打印即可
            System.out.println(f);
        }
    }
```

File
c:\\abc\\abc.txt

true

c:\\abc\\abc.txt true

accept方法返回值是一个布尔值
true:就会把传递过去的File对象保存到File数组中
false:就不会把传递过去的File对象保存到File数组中

```java
/*
    创建过滤器FileFilter的实现类,重写过滤方法accept,定义过滤规则
 */
public class FileFilterImpl implements FileFilter{
    @Override
    public boolean accept(File pathname) {
        return true;
    }
}
```

c:\\abc\\abc.txt

listFiles方法一共做了3件事情:
1.listFiles方法会对构造方法中传递的目录进行遍历,获取目录中的每一个文件/文件夹-->封装为File对象
2.listFiles方法会调用参数传递的过滤器中的方法accept
3.listFiles方法会把遍历得到的每一个File对象,传递过accept方法的参数pathname

过滤的规则:
在accept方法中,判断File对象是否是以.java结尾
是就返回true
不是就返回false