LambdaDemo

```
package lambda;

import java.io.File;
import java.io.FileFilter;

/**
 * lambda表达式（JDK8推出的新特性）
 * lambda表达式可以让程序员面向函数式编程，使用更精简的语法创建匿名内部类
 * 语法:
 * (参数列表)->{
 * 方法体
 * }
 * 当我们使用匿名内部类创建一个对象时，如果实现的接口中【只有一个抽象方法】时，该操作才能使用lambda表达式做替换，使得代码更简洁优雅。
 */
public class LambdaDemo {
    public static void main(String[] args) {
        //匿名内部类写法创建一个文件过滤器
        FileFilter filter1 = new FileFilter() {
            public boolean accept(File f) {
                return f.length() > 500;
            }
        };
```

```
//lambda表达式写法：去掉接口和方法名部分，在参数列表和方法体大括号之间加上"->"
FileFilter filter2 = (File f) -> {
    return f.length() > 500;
};

//lambda表达式中，可以忽略方法参数类型（仅写参数名即可）
FileFilter filter3 = (f) -> {
    return f.length() > 500;
};

//如果该方法的参数只有【一个】时，参数列表的括号"()"可以忽略不写
//但没参数要写括号，多个参数也要写括号
FileFilter filter4 = f -> {
    return f.length() > 500;
};

//如果方法体中只有一句代码，那么方法体的大括号"{}"可以忽略不写
//注：如果有return关键字，那么在忽略大括号"{}"的同时"return"关键字也必须一同忽略。
FileFilter filter5 = f -> f.length() > 500;

//原本匿名内部类在File中的应用
File dir = new File("./src/file");
if (dir.isDirectory()) {
    //匿名内部类形式
    /*File[] subs = dir.listFiles(new FileFilter() {
        public boolean accept(File file) {
            return file.length()>1000;
```

```
                }
            });*/
            File[] subs = dir.listFiles(f -> f.length() > 1000);
            for (int i = 0; i < subs.length; i++) {
                System.out.println(subs[i].getName() + ":" + subs[i].length());
            }
        }
    }
}
```