

目录

[Day13. Java](#)

[1 回顾](#)

[2 异常](#)

[2.1 异常继承结构](#)

[2.2 异常捕获](#)

[2.3 throws](#)

[2.4 throws 和 catch](#)

[2.5 throw](#)

[2.6 异常包装](#)

[2.7 自定义异常](#)

[2.8 RuntimeException 和其他Exception](#)

[3 IO](#)

[4 java.io.File](#)

[5 流 Stream](#)

[6 InputStream / OutputStream](#)

[7 FileInputStream / FileOutputStream](#)

[8 作业](#)

Day13. Java

1 回顾

- HashMap的哈希算法
 - key.hashCode() 得到哈希值
 - 用哈希值和数组长度计算下标i
 - 新建Entry对象放入 i 位置
 - ◆ 空位置, 直接放入
 - ◆ 有数据, 依次用equals()比较是否相等
 - 找到相等的, 覆盖值
 - 没有相等的, 链表连接在一起
 - ◆ 负载率、加载因子 0.75
 - 新建翻倍长度新数组
 - 所有数据, 重新执行哈希运算, 存入新数组
 - ◆ jdk1.8
 - 链表长度到8, 转成红黑树
 - 树上的数据减少到6, 转回链表

2 异常

- 封装错误信息的对象
- 错误信息
 - 类型
 - 提示消息
 - 行号

2.1 异常继承结构

```
Throwable
|- Error 系统级错误
|- Exception 可修复的错误
    |- 其他Exception
    |- RuntimeException
        |- NullPointerException
        |- ArrayIndexOutOfBoundsException
        |- NumberFormatException
        |- ArithmeticException
        |- ClassCastException
        |- ...
```

2.2 异常捕获

```
try {
    可能出现异常的代码
} catch(AException e) {

} catch(BException e) {

} catch(父类型Exception e) {

} finally {
    不管出不出错，都会执行
}
```

异常

项目: day1301_异常
类: day1301.Test1

package day1301;

```

import java.util.Scanner;

public class Test1 {
    public static void main(String[] args) {
        while(true) {
            try {
                f();
                break;
            } catch (ArrayIndexOutOfBoundsException e) {
                System.out.println("输入两个, 两个! ");
            } catch (ArithmeticException e) {
                System.out.println("不能除零! ");
            } catch (Exception e) {
                System.out.println("出错, 请重试! ");
            } finally {
                System.out.println("-----");
            }
        }
    }

    private static void f() {
        /*
         * "3345,64"
         * --> ["3345", "64"]
         * 3345 / 64
         */
        System.out.print("输入逗号隔开的两个整数: ");
        String s = new Scanner(System.in).nextLine();
        String[] a = s.split(",");
        int n1 = Integer.parseInt(a[0]);
        int n2 = Integer.parseInt(a[1]);
        int r = n1/n2;
        System.out.println(r);
        /*
         * 3453,64
         * 6543
         * entr,jrtr
         * 3434,0
         */
    }
}

```

2.3 throws

- 在方法上设置异常的抛出管道

```
void f() throws A,B,C {
```

```
}
```

- RuntimeException 及其子类型异常，存在默认异常管道

```
void f() throws RuntimeException {

}
```

- 其他异常
 - 要抛出，必须手动添加管道
 - 不抛出，必须catch 捕获

throws

Test2

```
package day1301;

import java.io.File;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class Test2 {
    public static void main(String[] args) {
        try {
            f();
        } catch (ParseException e) {
            System.out.println("日期格式错误");
        } catch (IOException e) {
            System.out.println("不能创建文件");
        }
    }

    private static void f() throws ParseException, IOException {
        /*
         * "1999-8-21"
         * --> Date --> 153456236623
         *
         * 在d盘根目录, 新建文件
         * "d:\153456236623.txt"
         */

        System.out.print("生日 (yyyy-MM-dd) : ");
        String s = new Scanner(System.in).nextLine();
        SimpleDateFormat sdf =
            new SimpleDateFormat("yyyy-MM-dd");
        Date d = sdf.parse(s);
        long t = d.getTime();
    }
}
```

```
String path = "d:\\"+t+".txt";
File file = new File(path);
file.createNewFile();
System.out.println("文件已创建: "+path);
}
}
```

2.4 throws 和 catch

- 底层异常，一般要抛到上层进行处理
- 在方法调用路径中，可以选择一个合适的位置，来捕获、修复异常
- 如果不知道该 throws 还是 catch，就选择throws

2.5 throw

- 手动抛出异常对象，执行异常的抛出动作
- 当逻辑判断，发现错误情况时，可以选择手动创建异常对象，并手动抛出异常

```
if(....) {
    AException e = new AException("...");
    throw e;
}
```

throw

Test3

```
package day1301;

import java.util.Scanner;

public class Test3 {
    public static void main(String[] args) {
        f();
    }

    private static void f() {
        /*
         * 63453.354
         * 34.3636
         *
         * 63453.354 / 34.3636
         */
        System.out.println("输入两个浮点数: ");
        double a = new Scanner(System.in).nextDouble();
        double b = new Scanner(System.in).nextDouble();
    }
}
```

```
try {
    double r = divide(a, b);
    System.out.println(r);
} catch (ArithmeticException e) {
    System.out.println("不能除零，是我们的错，请鞭答我们吧！");
    e.printStackTrace();
}

private static double divide(double a, double b) {
    if(b == 0) {
        ArithmeticException e =
            new ArithmeticException("/ by zero");
        //"不能除零，是我们的错，请鞭答我们吧！");
        throw e;
    }
    return a/b;
}
```

2.6 异常包装

- 捕获的异常对象，包装成另一种类型，再抛出
- 使用场景
 - 不能抛出的类型，包装成能跑出的类型，再抛
 - 异常的简化
- 重写方法时，异常管道，不能比父类方法多

异常包装

Test4

```
package day1301;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Date;

public class Test4 {
    public static void main(String[] args) {
        f();
    }

    private static void f() {
```

```
/*
 * "2018-11-16"
 * "2018-11-20"
 * "2018-11-16"
 * "2018-11-1"
 * "2018-11-2"
 * "2018-11-10"
 * "2018-11-30"
 * "2018-11-3"
 */
ArrayList<String> list = new ArrayList<>();

Collections.addAll(
    list,
    "2018-11-16", "2018-11-20", "2018-11-16",
    "2018-11-1", "2018-11-2", "2018-11-10",
    "asdfafgsdfgsdfg",
    "2018-11-30", "2018-11-3");

Collections.sort(list, new Comparator<String>() {
    @Override
    public int compare(String o1, String o2) {
        SimpleDateFormat sdf =
            new SimpleDateFormat("yyyy-MM-dd");
        try {
            Date d1 = sdf.parse(o1);
            Date d2 = sdf.parse(o2);
            return d1.compareTo(d2);
        } catch (ParseException e) {
            throw new RuntimeException(e);
        }
    }
});
System.out.println(list);
}
```

2.7 自定义异常

- 现有异常类型，无法表示业务中的错误情况

转账失败

ZhuanZhangShiBaiException

用户名错误

UsernameNotFoundException

密码错误

WrongPasswordException

- 自定义异常
 - 起一个合适的类名
 - 继承一个合适的父类
 - 添加构造方法

自定义异常

UsernameNotFoundException

```
package day1301;

public class UsernameNotFoundException extends Exception {

    public UsernameNotFoundException() {
        super();
    }

    public UsernameNotFoundException(String message, Throwable cause,
boolean enableSuppression,
        boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }

    public UsernameNotFoundException(String message, Throwable cause) {
        super(message, cause);
    }

    public UsernameNotFoundException(String message) {
        super(message);
    }

    public UsernameNotFoundException(Throwable cause) {
        super(cause);
    }

}
```

WrongPasswordException

```
package day1301;

public class WrongPasswordException extends Exception {

    public WrongPasswordException() {
        super();
    }

}
```



```

    public WrongPasswordException(String message, Throwable cause, boolean
enableSuppression,
        boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }

    public WrongPasswordException(String message, Throwable cause) {
        super(message, cause);
    }

    public WrongPasswordException(String message) {
        super(message);
    }

    public WrongPasswordException(Throwable cause) {
        super(cause);
    }
}

```

Test5

```

package day1301;

import java.util.Scanner;

public class Test5 {
    public static void main(String[] args) {
        System.out.print("用户名: ");
        String n = new Scanner(System.in).nextLine();
        System.out.print("密码: ");
        String p = new Scanner(System.in).nextLine();
        try {
            login(n,p);
            System.out.println("欢迎登录");
        } catch (UsernameNotFoundException e) {
            System.out.println("用户名错误");
        } catch (WrongPasswordException e) {
            System.out.println("密码错误");
        }
    }

    private static void login(String n, String p) throws
UsernameNotFoundException, WrongPasswordException {
        //abc, 123
        if(! "abc".equals(n)) {
            throw new UsernameNotFoundException();
        }

        if(! "123".equals(p)) {
            throw new WrongPasswordException();
        }
    }
}

```

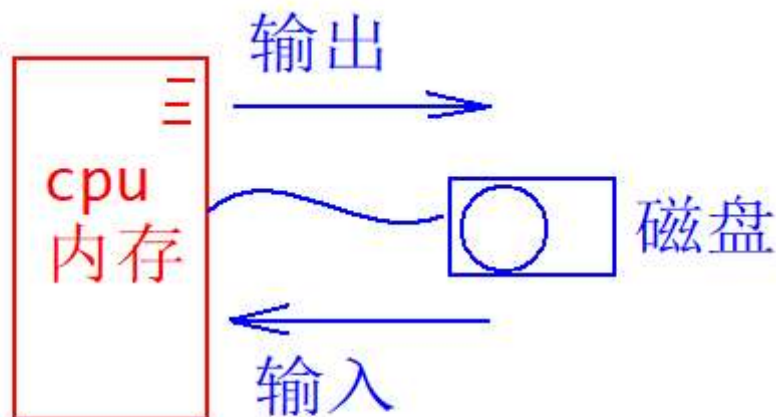
```
}
```

2.8 RuntimeException 和 其他Exception

- RuntimeException
 - 非检查异常
 - 编译器不会检查这种异常
 - 存在默认抛出管道
- 其他Exception
 - 检查异常
 - 编译器会检查，这种异常有没有处理代码
 - ◆ throws
 - ◆ try-catch
 - 器强制开发者，在编码时，就要考虑错误如何处理

3 IO

- IO - Input/Output
- 输入和输出



程序运行时，
数据在内存中

- java.io 包
 - File
 - InputStream/OutputStream
 - FileInputStream/FileOutputStream
 - BufferedInputStream/BufferedOutputStream
 - ObjectInputStream/ObjectOutputStream

- Reader/Writer
- InputStreamReader/OutputStreamWriter
- BufferedReader
- PrintWriter、PrintStream

4 java.io.File

- 封装磁盘路径字符串
 - 存在的文件路径
 - 存在的文件夹路径
 - 不存在路径

d:\abc

d:\a.txt

k:\xxx\xx

- 方法
 - 文件或文件夹的属性
 - ◆ length()
文件的字节量
对文件夹无效
 - ◆ exists()
判断路径是否存在
 - ◆ lastModified()
最后修改时间，毫秒值
 - ◆ getName()
获得文件、文件夹的名字
 - ◆ getParent()
获得父文件夹
 - ◆ getAbsolutePath()
获得绝对路径，完整路径
 - 创建、删除文件、文件夹
 - ◆ createNewFile()
创建文件
文件已经存在，不会创建
文件夹不存在，会出现异常
 - ◆ mkdirs()
创建多层文件夹

- ◆ delete()
删除文件，或删除空文件夹
- ◆ File.createTempFile()
在系统的临时目录中，创建临时文件
- 文件夹列表
 - ◆ list()
获得 String[] 数组，包含文件或文件夹名称
 - ◆ listFiles()
获得 File[] 数组，包含文件或文件夹的 File 封装对象

File

项目: day1302_File
类: day1302.Test1

Test1

```
package day1302;

import java.io.File;

public class Test1 {
    public static void main(String[] args) {
        String path;

        path = "D:\\\\home\\java\\eclipse\\eclipse.exe";//存在的文件
        //path = "D:\\\\home\\java\\eclipse";//存在的文件夹
        //path = "x:\\\\ee\\gg";//不存在的路径

        File f = new File(path);
        System.out.println(f.length());//文件字节量，文件夹无效
        System.out.println(f.isFile());//是否是文件
        System.out.println(f.isDirectory());//是否是文件夹
        System.out.println(f.lastModified());//最后修改时间毫秒值
        System.out.println(f.exists());//是否存在
        System.out.println(f.getName());//获得文件名
        System.out.println(f.getParent());//获得父文件夹
        System.out.println(f.getAbsolutePath());//获得完整路径
    }
}
```

Test2

```
package day1302;

import java.io.File;
import java.io.IOException;
import java.util.Scanner;

public class Test2 {
    public static void main(String[] args) throws IOException {
        /*
         * d:\abc\aa
         * d:\abc\aa\fl
         * 删除f1
         * 删除aa文件夹
         */
        File dir = new File("d:/abc/aa");
        File file = new File("d:/abc/aa/fl");
        System.out.println("按回车继续");
        new Scanner(System.in).nextLine();
        dir.mkdirs();

        new Scanner(System.in).nextLine();
        file.createNewFile();

        new Scanner(System.in).nextLine();
        file.delete();

        new Scanner(System.in).nextLine();
        dir.delete();
    }
}
```

Test3

```
package day1302;

import java.io.File;
import java.util.Scanner;

public class Test3 {
    public static void main(String[] args) {
        System.out.println("文件夹路径: ");
        String s = new Scanner(System.in).nextLine();
        File dir = new File(s);
        if(! dir.isDirectory()) {
            System.out.println("输入的不是文件夹");
            return;
        }
    }
}
```

```

/*
 * 文件夹如果不存在,
 * 或文件夹没有权限进入,
 * 会得到 null 值
 */
String[] names = dir.list();
File[] files = dir.listFiles();

if(null == names) {
    System.out.println(
        "不能获取该文件夹列表");
    return;
}

for(String n : names) {
    System.out.println(n);
}
System.out.println("-----");
for(File f : files) {
    System.out.println(
        f.getName()+" : "+f.length());
}
}
}

```

文件夹的总大小

```

[a]
|- b
|- c
|- [d]
    |- e
    |- [f]
        |- h
        |- i
    |- [g]
        |- j
        |- k

```

Test4

```

package day1302;

import java.io.File;
import java.util.Scanner;

public class Test4 {
    public static void main(String[] args) {

```

```
System.out.println("文件夹路径: ");
String s = new Scanner(System.in).nextLine();
File dir = new File(s);
if(! dir.isDirectory()) {
    System.out.println("输入的不是文件夹");
    return;
}

long size = dirLength(dir);
System.out.println(size);

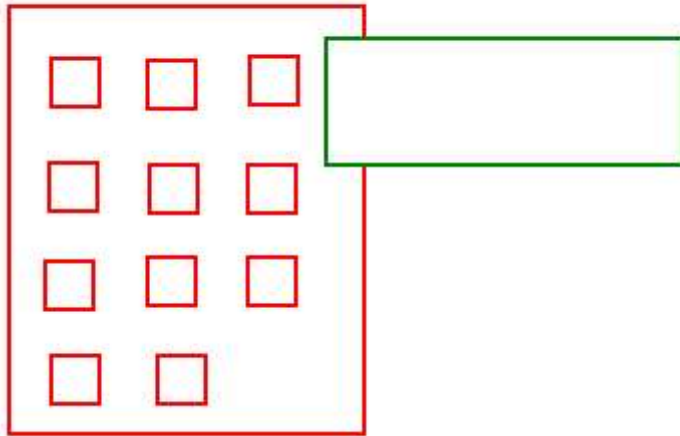
}

private static long dirLength(File dir) {
    /*
     * *)对参数dir文件夹列表, 存到files
     * *)如果files是null, 返回0
     *
     * *)定义累加变量 sum = 0L
     * *)遍历files数组, 取出文件存到 f
     *     *)如果f是文件
     *         *)向sum累加 f 文件的大小
     *     *)否则
     *         *)向sum累加 f 文件夹的大小(递归求出)
     *
     * *)循环结束, 返回 sum 的值
     */
    File[] files = dir.listFiles();
    if(files == null) {
        return 0;
    }
    long sum = 0;
    for (File f : files) {
        if(f.isFile()) { //f是文件
            sum += f.length();
        } else { //f是文件夹
            sum += dirLength(f); //递归
        }
    }
    return sum;
}
}
```

5 流 Stream


- 把数据的读、写, 抽象成数据在管道中流动

- 流是单向流动
 - 输入流
 - 输出流
- 数据只能从头到尾，顺序流动一次



内存

6 InputStream / OutputStream

- 字节流的抽象父类
- 方法
 - write(int b)
把int四个字节中，末尾的一个字节输出到文件

 - write(byte[] buff)
输出数组中全部字节值
 - write(byte[] buff, start, length)
输出数组中从start开始的length个字节值

7 FileInputStream / FileOutputStream

- 直接插在文件上，读写文件数据
- 创建对象
 - new FileOutputStream(文件)
不管文件是否存在，都新建空文件
如果文件夹不存在，会出现异常
 - new FileOutputStream(文件, true)

文件不存在, 新建;
文件已经存在, 追加。

文件流

项目: day1303_文件流

类: day1303.Test1

```
package day1303;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;

public class Test1 {
    public static void main(String[] args) throws Exception {
        /*
         * d:/abc/f1
         *
         * FOS--f1
         */
        FileOutputStream out =
            new FileOutputStream("d:/abc/f1");
        out.write(97);//00 00 00 61 --> 61
        out.write(98);//00 00 00 62 --> 62
        out.write(99);//00 00 00 63 --> 63
        out.write(356);//00 00 01 64 --> 64
        byte[] a = {
            101,102,103,104,105,
            106,107,108,109,110
        };
        out.write(a);//全部10个
        out.write(a, 3, 4);//下标3开始的4个

        out.close();//释放系统资源
    }
}
```

8 作业

- 输入文件夹, 查找所有的图片
递归到深层目录查找
判断文件名, 是否是图片

```
name.toLowerCase().matches(".*\\.(jpg|png|gif|bmp)")
```

- 重写
 - day1103_丑数
 - day1205_作业_学生
- day1104_猜游戏
 - 在GuessGame中, 处理异常, 提示用户输入有误, 重新输入
- 高阶作业
 - 手写双向链表

```
class MyList {
    Node head;
    Node tail;
    int size;

    public void add(Object value) {
    }

    public void get(int i) {
    }

    public int size() {
    }

    class Node {
        Object value;
        Node prev;
        Node next;
    }
}
```