二维数组和异常

回顾

1.数组的初始化方式
2.数组的使用（元素访问，元素修改，遍历）
3.数组的应用（排序和查找）
4.选择排序、冒泡排序、顺序查找和二分法查找的算法以及代码实现
5.Arrays工具类中的常用方法
6.可变参数使用过程中的注意事项

今天内容

教学目标

1.了解二维数组的概念和定义
2.掌握二维数组的初始化和遍历
3.了解异常的概念和分类
4.掌握异常的处理方式
5.了解自定义异常的使用

第一节 二维数组

1.1 二维数组的概念

本质上还是一个一维数组，只是其数组元素又是一个一维数组

举例说明:变量，一维数组，二维数组之间的关系
吸烟:

没钱	1根	一个变量
稍微有钱 一包		一维数组【20根】
有钱	一条	二维数组【10包】

1.2 二维数组的定义

```
方式一：元素类型[][] 数组名称；

方式二：元素类型 数组名称[][]；
推荐使用方式一
```

1.3 数组的初始化

```
静态初始化：

语法：元素类型[][] 数组名称 = new 元素类型[][]{{一维数组1,一维数组2,一维数组3....};
简化：元素类型[][] 数组名称 ={{一维数组1,一维数组2,一维数组3....};

举例：int[][] arr = new int[][]{{2,3},{5,2,1},{10,45,22,54}};
      int[][] arr = {{2,3},{5,2,1},{10,45,22,54}};

动态初始化：

语法：元素类型[][] 数组名称 = new 元素类型[二维数组的长度][一维数组的长度]
举例：int[][] arr = new int[3][4];
说明：定义一个数组arr，二维数组中一维数组的个数为3个，每个一维数组中元素的个数为4个
```

1.4 二维数组的访问

通过下标访问指定元素

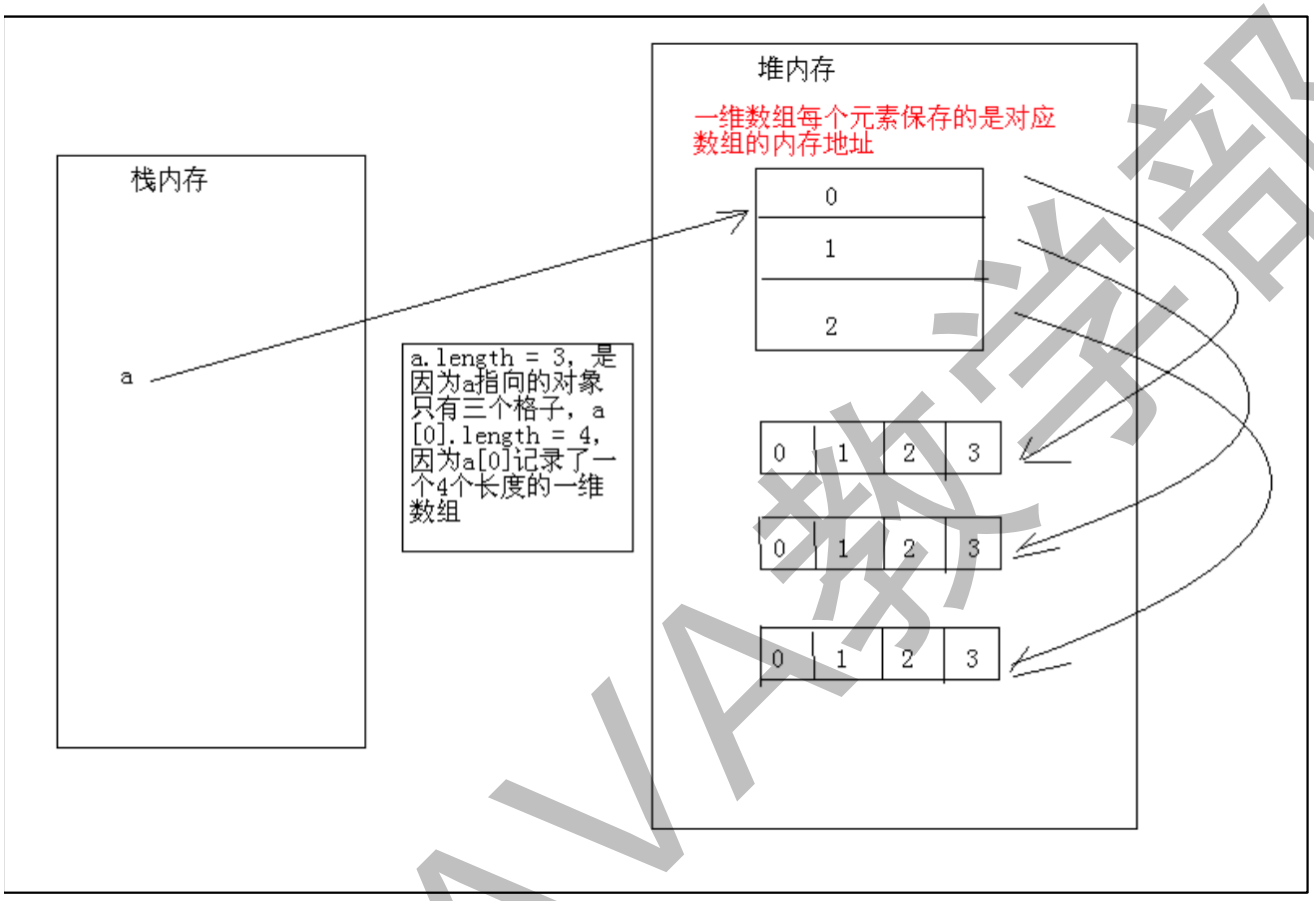
```
class TwiceArrayDemo01
{
    public static void main(String[] args)
    {
        int[][] arr = new int[3][4];

        System.out.println(arr);//[[I@15db9742
        System.out.println(arr.length);//3
        System.out.println(arr[0]);//[I@6d06d69c
        System.out.println(arr[0].length);//4
        System.out.println(Arrays.toString(arr));//[[I@6d06d69c, [I@7852e922, [I@4e25154f]
        System.out.println(Arrays.toString(arr[0]));//[0, 0, 0, 0]

        /*
        [[I@15db9742
        3
        [I@6d06d69c
        4
        [[I@6d06d69c, [I@7852e922, [I@4e25154f]
        [0, 0, 0, 0]
        */
    }
}
```

遍历数组

```java
//常见的操作：遍历二维数组
class TwiceArrayDemo02
{
    public static void main(String[] args)
    {
        //如果二维数组中一维数组的元素个数不确定
        //int[][] arr = new int[3][];

        int[][] arr = new int[][]{{2,3},{5,2,1},{10,45,22,54}};

        //遍历arr
        for(int i = 0;i < arr.length;i++) {
            System.out.println(arr[i]);
        }

        //赋值：给arr中的第1个元素修改值
        arr[1] = new int[2];

        //给arr[0]中的第0个元素修改值
        arr[0][0] = 10;

        //遍历arr[0]
        for(int i = 0;i < arr[0].length;i++) {
            System.out.println(arr[0][i]);
        }

        //二维数组的遍历：嵌套for循环
        //简单for循环
        for(int i = 0;i < arr.length;i++) {
            int[] subArr = arr[i];
            for(int j = 0;j < subArr.length;j++) {
                System.out.println(subArr[j]);
            }
        }

        //增强for循环
        for(int[] subArr1:arr) {
            for(int n:subArr1) {
                System.out.println(n);
            }
        }
    }
}
```

1.5 内存中的二维数组

画图分析：



第二节 异常

2.1 异常的概念

> 在程序执行过程中由于设计或设备原因导致的程序中断的异常现象叫做异常

2.2 异常的分类

运行时异常

> 运行时异常：（RuntimeException）在编译过程不会发现（没有语法错误），但是在执行程序程中，由于重大的逻辑错误导致的程序中断

常见的运行时异常

```
NullPointerException          空指针异常（一个对象没有初始化调用方法）
IndexOutOfBoundsException     下标越界异常
ClassCastException            类型转换异常（对象类型转换时）
NumberFormatException         数字格式异常
ArithmeticException           算术异常
```

非运行时异常

> 非运行时异常：编译异常或检查异常，在程序设计过程中，编译时就会被发现，但是执行时可能发生也可能不发生的异常，为了程序不报错可以执行，那么这一类异常必须进行相应的处理

2.3 异常的处理方式

抛出异常：throws和throw

> 1.throw关键字：抛出异常
>
> 语法：
>
> ```
> //使用格式：throw 异常对象;
> throw new Exception();
> ```

> 2.throws关键字：上抛异常
>
> 上抛异常，将异常抛到方法的签名后，表示提示方法的调用者此方法具有异常没有处理，你若想使用我，需要对此异常进行处理。
>
> 语法：
>
> ```
> public static void fun()throws 异常A或异常A的父类{
>     throw new 异常A;
> }
> ```

捕获异常：try..catch..finally语句

语法：

```
try{
    //功能代码
}catch(异常类型1 异常引用1){
    //异常处理代码1
}catch(异常类型2 异常引用2){
    //异常处理代码2
}
......
finally{
    //最终执行的代码
}
```

说明：
功能代码：需要将所有可能发生异常的代码放进try块中
若try块中的代码发生异常，将会指向相应的catch块
当try块中的代码出现了异常类型1的异常，则执行异常处理代码1
finally块：最终被执行的代码，无论之前执行的是try块还是catch块，方法结束之前finally块中的代码一　定会执行

```
public static int fun1() {
    try {
        throw new Exception();
    }catch(Exception e) {
        System.out.println("出现异常");
        return 1;
    }finally {
        System.out.println("finally");//这条语句在方法返回1之后执行（最后执行的代码）
        //return 2;//若打开此语句注释，则这个方法的返回值一定是2
    }
}
```

3.使用try-catch结构捕获异常时的注意事项：

1）catch块可以存在多个，但是不能存在try块中不存在的异常
2）上面的catch块不能是下面catch块捕获异常的父类
3）finally块可以省略，根据需求决定是否需要finally块

2.4 自定义异常类

1.过程：自定义非运行时异常

i.定义一个类，继承Exception类

ii.在此类中定义构造方法，调用父类中的带字符串参数的构造方法（此字符串表示对异常的描述）

iii.使用异常

```
public class ZDYException extends Exception {
    public ZDYException() {}
    //此构造方法中的字符串参数表示对此异常的说明描述
    public ZDYException(String masage) {
        super(masage);
    }
}
public class TestZDYException {
    public static void main(String[] args) throws ZDYException {
        check("123456");
    }
    //设计一个方法，完成邮箱格式的初步验证（判断传入的字符串是否包含@符）
    //若字符串包含@返回true，否则返回false并抛出自定义异常
    public static void check(String email){
        if(email.contains("@")) {
            System.out.println(true);
        }else {
            System.out.println(false);
            try {
                throw new ZDYException("没有@符");
            } catch (ZDYException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

2.自定义运行时异常：

i.定义一个类，继承RuntimeException类
ii.在此类中定义构造方法，调用父类中的带字符串参数的构造方法（此字符串表示对异常的描述）
iii.使用异常

```
public class ZDYRuntime extends RuntimeException {
    public ZDYRuntime() {}
    public ZDYRuntime(String masage) {
        super(masage);
    }
}
public class TestZDYRuntime {
    public static void main(String[] args) {
        throw new ZDYRuntime("hello");
    }
}
```

课前默写

根据下面要求完成题目：
1.分别使用静态初始化和动态初始化的方式定义一个数组
2.对静态初始化的数组分别使用冒泡和选择进行排序，其中，冒泡实现升序，选择实现降序
3.使用for循环和foreach遍历排好序的数组

作业

1.自定义一个运行时异常
2.班上有3个学生，每个学生都参加了三门功课的考试，其中第二个学生是特长生，上级要求给他每门功课都+5.【要求：使用二维数组做，并且分别使用for循环和增强for循环遍历二维数组】
3.求一个3*3矩阵对角线元素之和
4.打印出杨辉三角形（要求打印出10行如下图）
程序分析：

```
        1

       1 1

      1 2 1

     1 3 3 1

    1 4 6 4 1

  1 5 10 10 5 1
```

面试题

1.二维数数组在内存中的存储方式是怎样的？
2.什么是异常？常见的异常有哪些？
3.异常有哪些处理方式，分别需要注意什么问题？