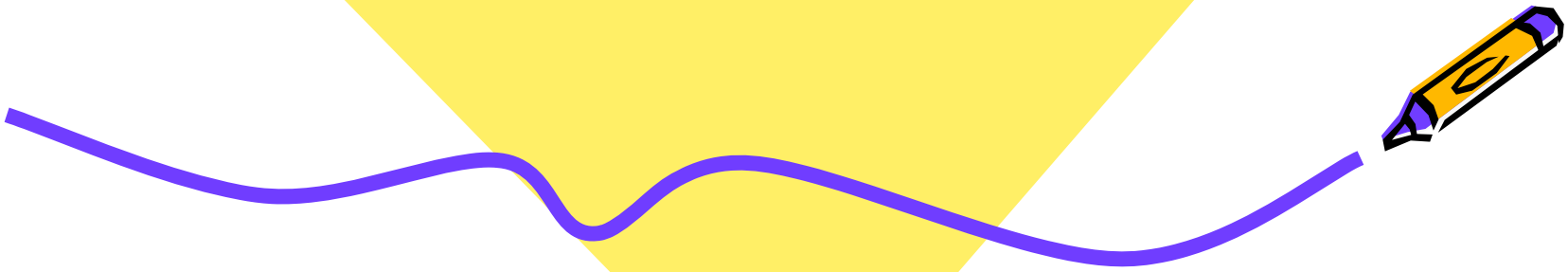
字符串、数组及多态



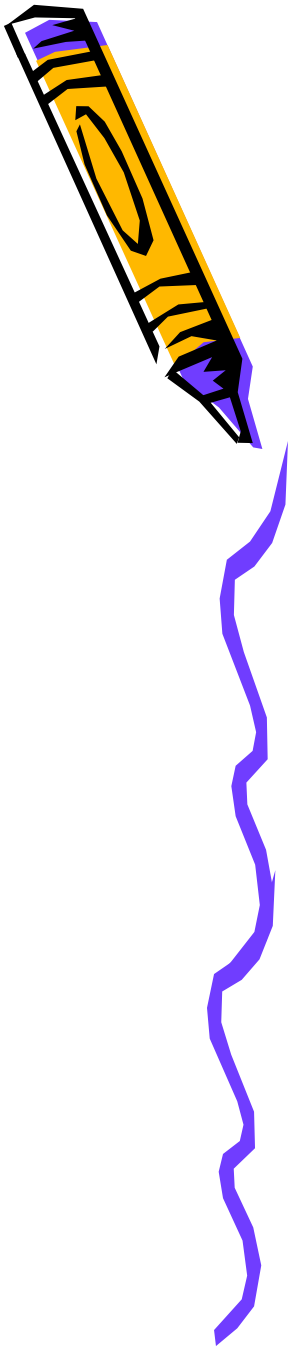
——江健

2015/11/20

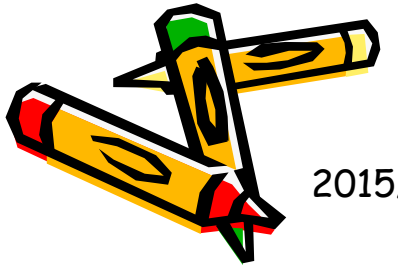
作业讲解

- 定义一个SavingAccount类。类中有一个静态变量annualInterestRate以存储年利率，一个实例变量savingsBalance以存储当前的储蓄额。该类提供一下方法：
- void calculateinterest(int n); //计算n年的复利
- static void modifyinterestrate(double rate); //设置存款利率

作业讲解

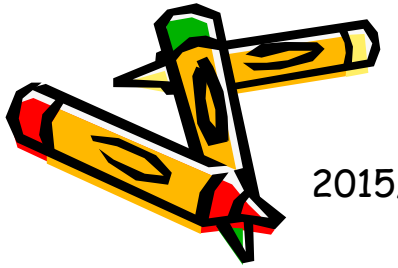


- 要点：
  - 静态变量
  - 实例变量
  - 静态方法
  - 实例方法



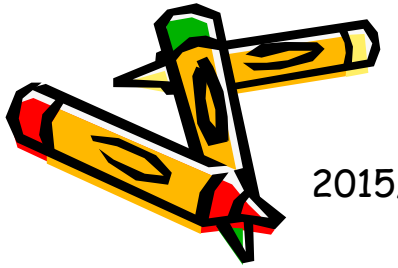
主讲内容

- 软件准备
- 字符串
- 数组
- 多态
- Coding Assignment

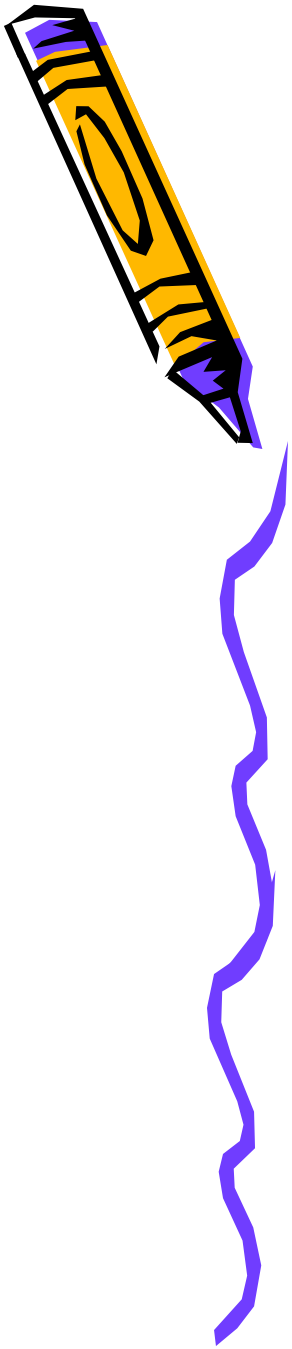


零、软件准备

- jdk 
- Spring Tools Suit 

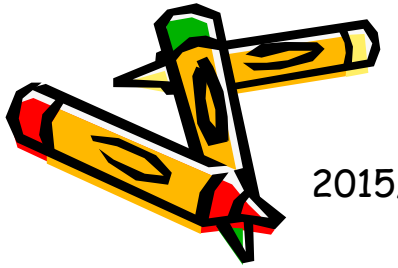


2015/11/20



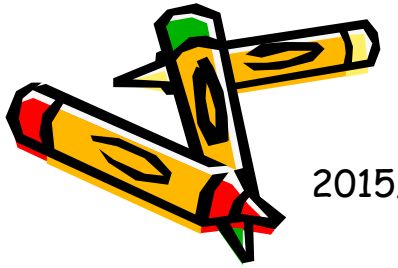
一、字符串

- 创建字符串
- 提取与定位
- 字符串的比较
- valueOf
- 其他方法
- StringBuffer类
- 字符串的特殊处理



1. 创建字符串

```
String s1 = "java语言";
String s2 = new String();
char[] value = {'J', 'a', 'v', 'a', '语', '言'};
String s3 = new String(value);
String s4 = new String(s3);
```



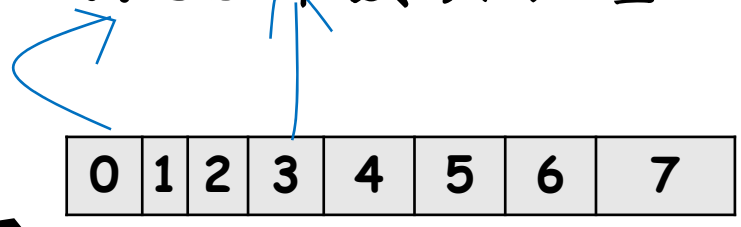
2. 提取与定位

定位

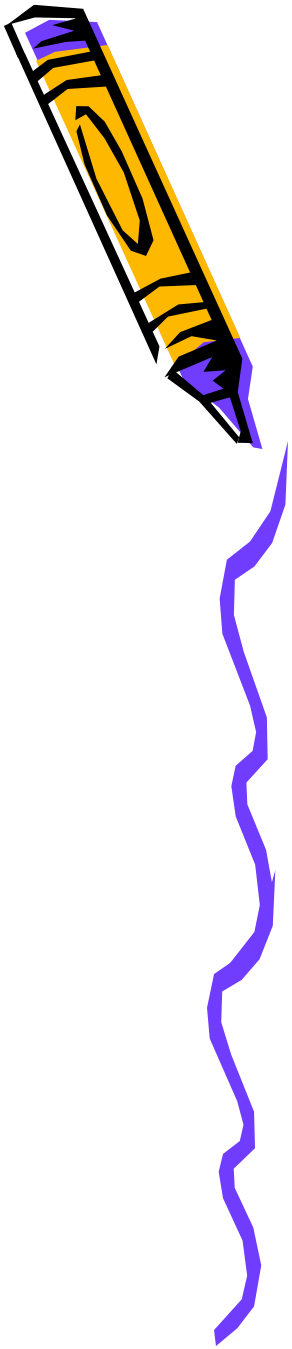
以“Web开发实验室”为例

'开'位于第几位呢?

“Web开发实验室”

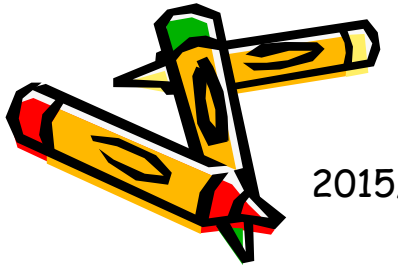


2. 提取与定位

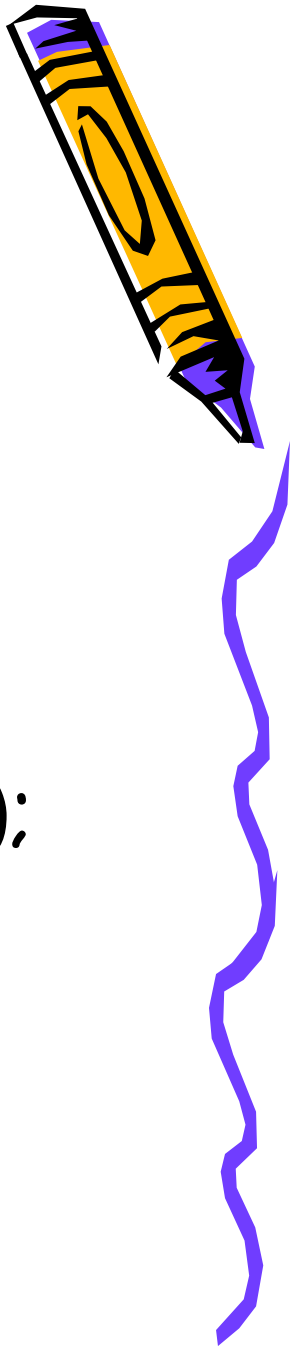


定位

```
String s = "Web开发实验室";
int i = s.indexOf('开');
int j = s.indexOf("实验室");
```



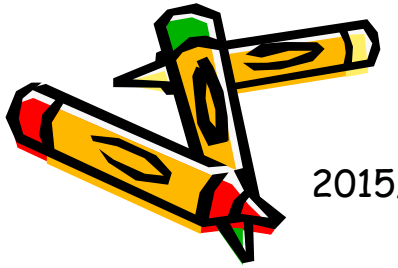
2. 提取与定位



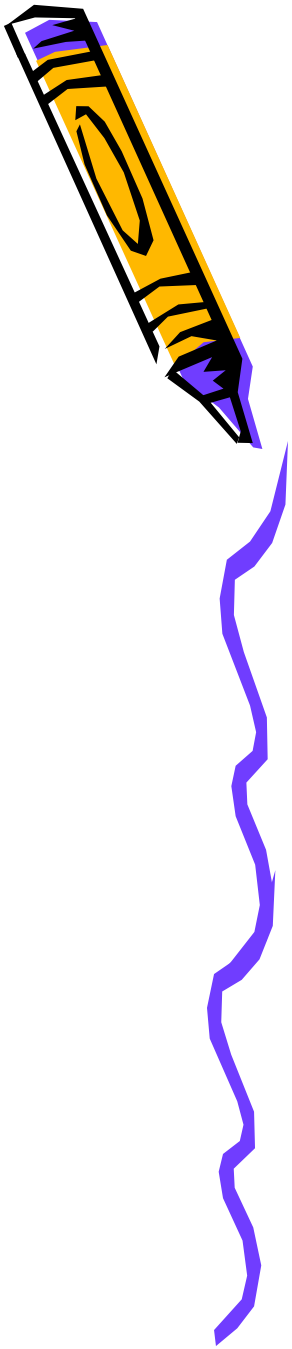
- 提取 (substring方法)
- String substring(int begin);

  截取子串（从begin开始直至末尾）
- String substring(int begin, int end);

  截取子串（从begin开始直至end-1）

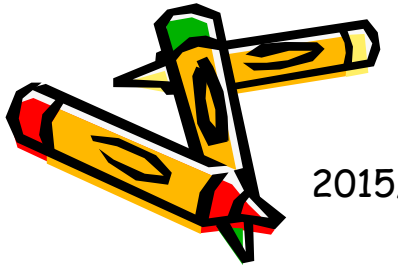


2. 提取与定位

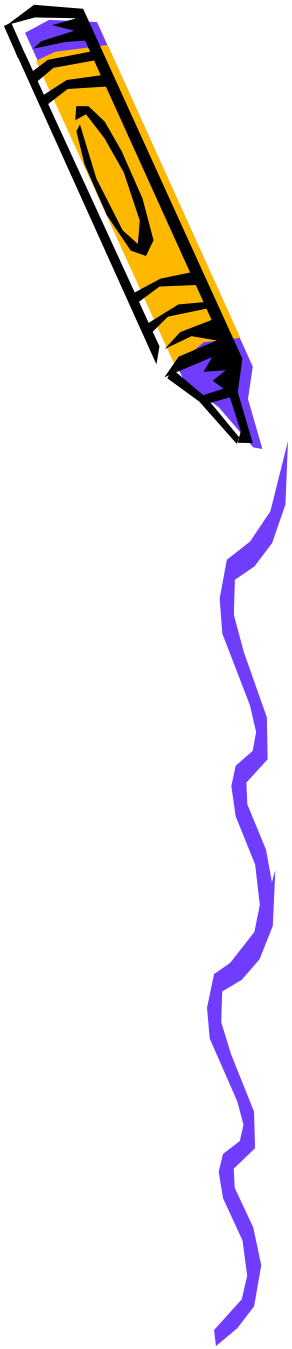


- 提取
  - “2015-11-14”从中提取月份
  - “艾欧尼亚#影流#祖安#德玛西亚”

  提取所有的服务器名称



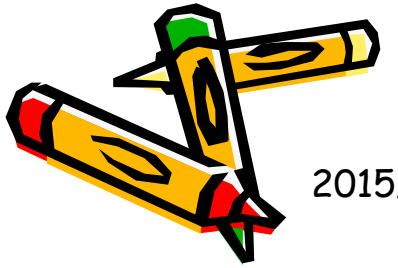
3. 字符串比较



```
String s1 = "abc";
String s2 = new String("abc");
String s3 = s1;
```

s1==s2

s1==s3



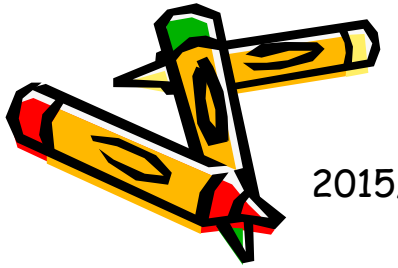
3. 字符串比较

注意：

==用于判断两个引用类型变量是否指向同一个对象

equals方法则用于判断两个字符串是否具有完全相同的内容

s1.equals(s2);



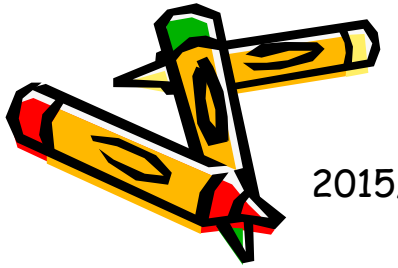
4. valueOf

valueOf是类方法，它可以将其他的基本数据类型或者引用类型转化成String型

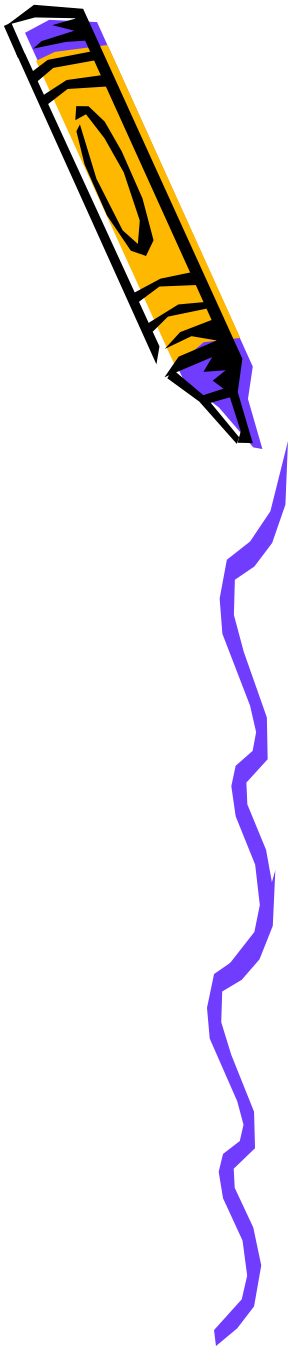
例：

String.valueOf(true);

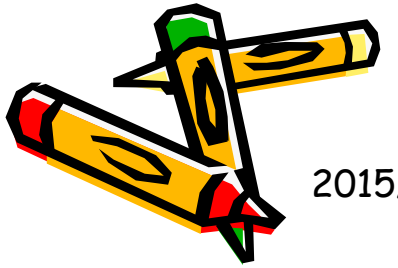
String.valueOf(1024);



5. 其他方法



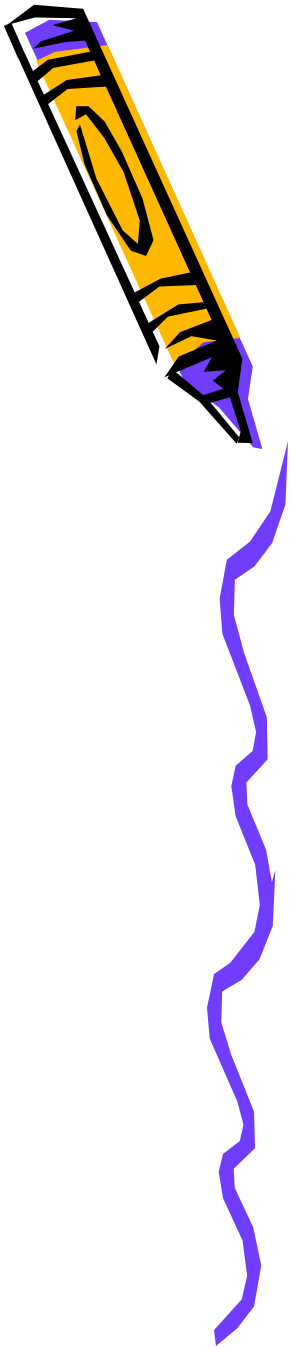
- 连接concat(String str);
- 替换replace(char old, char new);
- 去空trim()
- 大小写转换
  - toLowerCase()
  - toUpperCase()



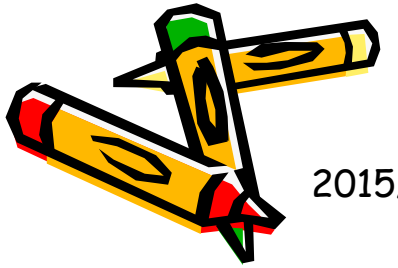
6. StringBuffer类

- StringBuffer对象的内容可以修改；而String对象一旦产生后就不可以被修改，重新赋值其实是两个对象。
- StringBuffer的内部实现方式和String不同，StringBuffer在进行字符串处理时，不生成新的对象，在内存使用上要优于String类。所以在实际使用时，如果经常需要对一个字符串进行修改，例如插入、删除等操作，使用StringBuffer要更加适合一些。

6. StringBuffer类

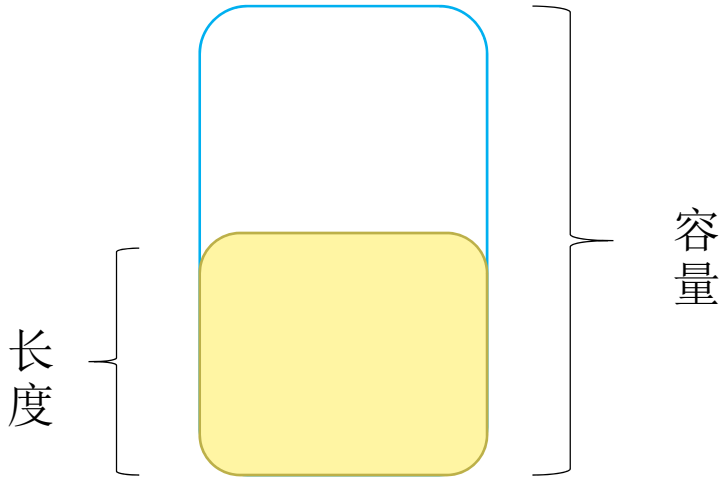


- 构造方法
- StringBuffer()
  - 初始容量16，内容为空
- StringBuffer(int length)
  - 初始容量为length，内容为空
- StringBuffer(String str)
  - 内容为str字符串，初始容量为str的长度+16

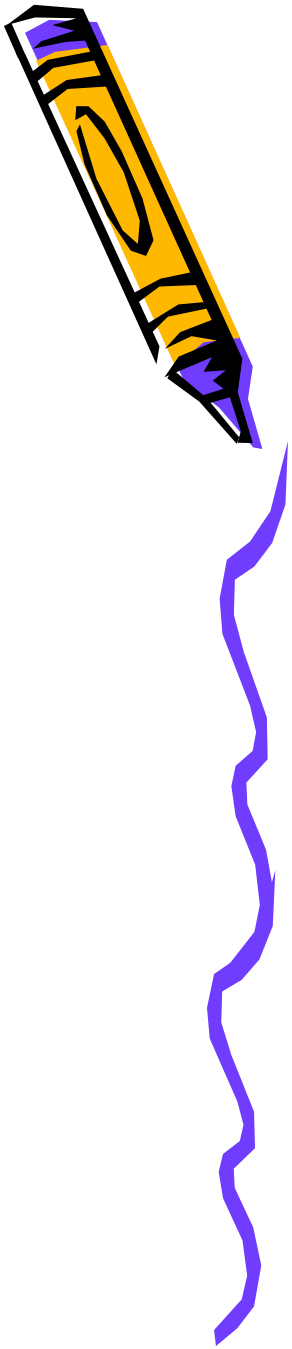


6. StringBuffer类

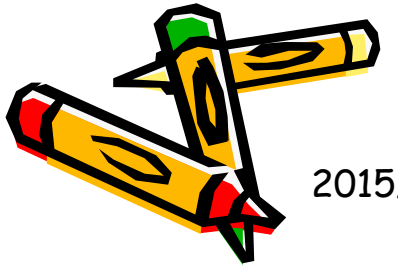
- 长度与容量
  - 长度指的是StringBuffer类包含的字符串的长度
  - 容量指的是该实例当前占用的内存空间能够表示的最长字符串的长度



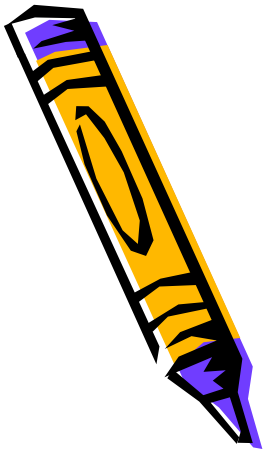
6. StringBuffer类



- 基本方法
- append(String str)
- insert(int offset, String str);
- delete(int start, int end)
- replace(int start, int end, String str)
- reverse()



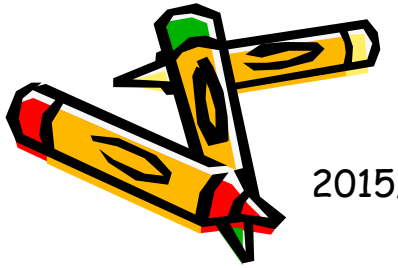
7. 字符串的特殊处理



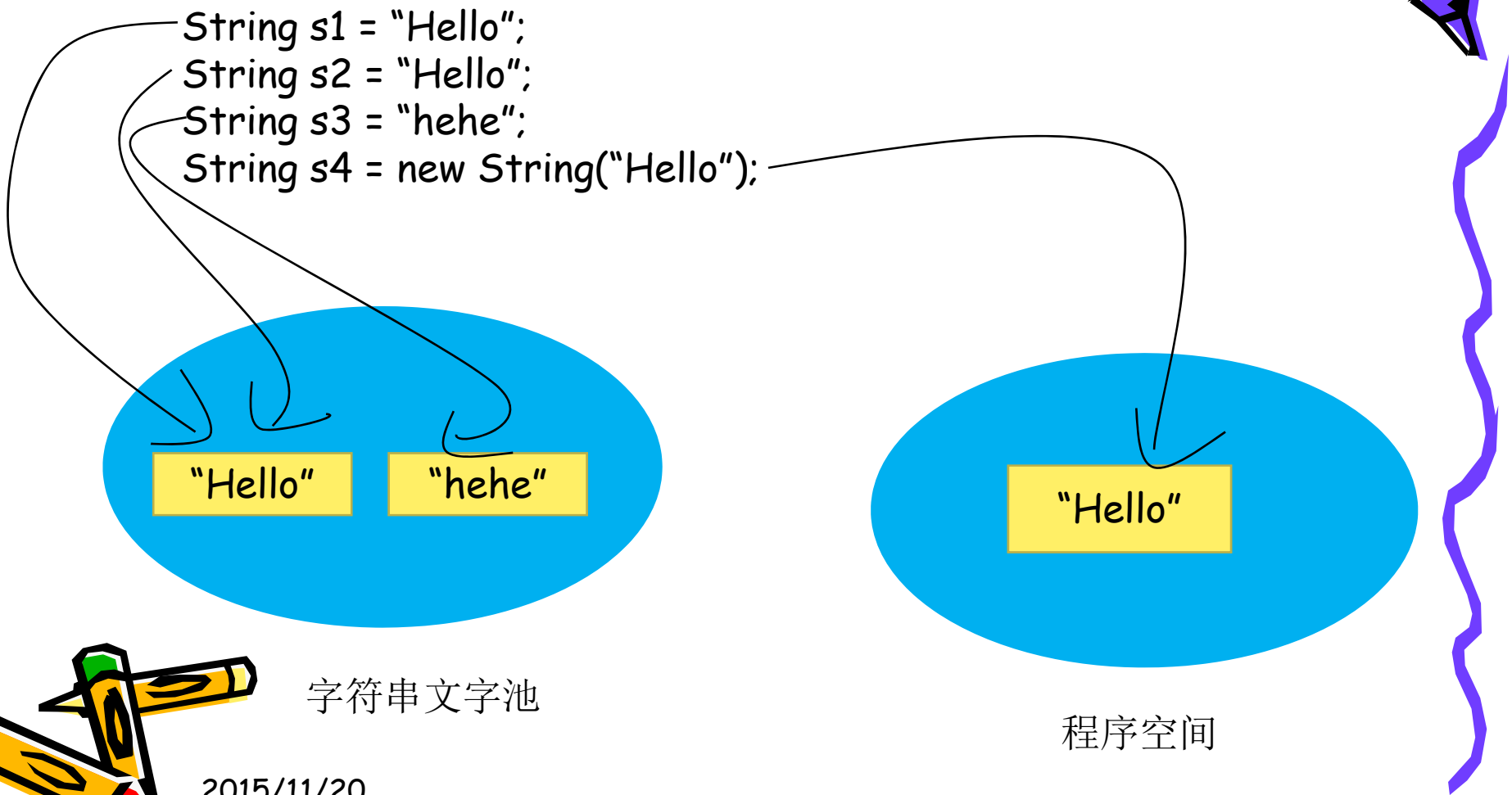
- 字符串文字池
- 以字符串文字创建的字符串都会存入字符串文字池。所谓字符串文字就是以下面这种方式创建的字符串：

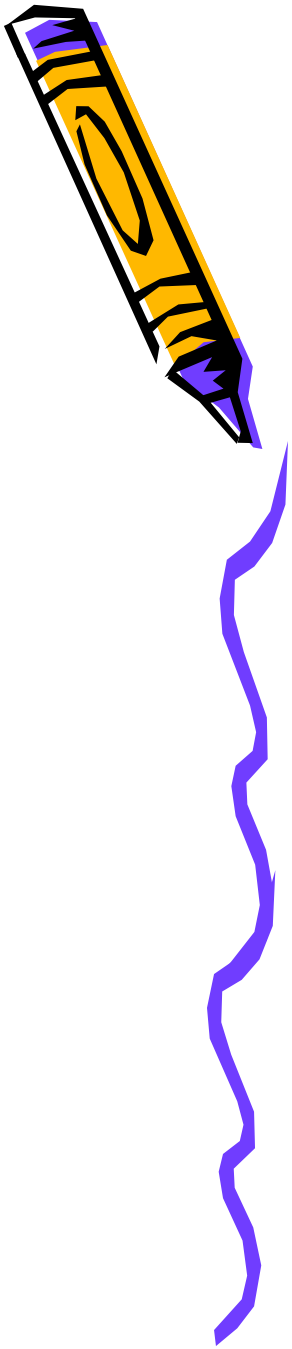
  String s = "web开发实验室";

  字符串文字池中不会出现内容完全相同的字符串



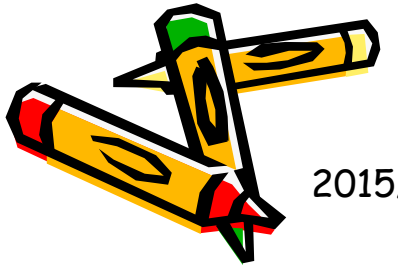
7. 字符串的特殊处理


String s1 = "Hello";
String s2 = "Hello";
String s3 = "hehe";
String s4 = new String("Hello");
"Hello"
"hehe"
"Hello"
字符串文字池
程序空间

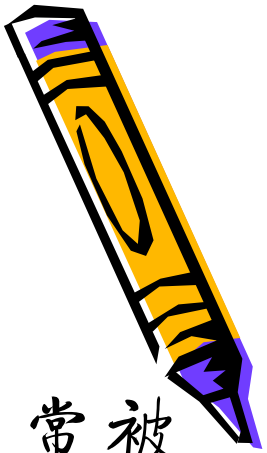


二、数组

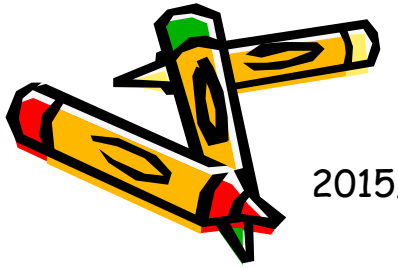
- 数组的概念
- 数组的创建
- 数组成员的访问
- 别样的for循环
- 二维数组
- 实例：全排列



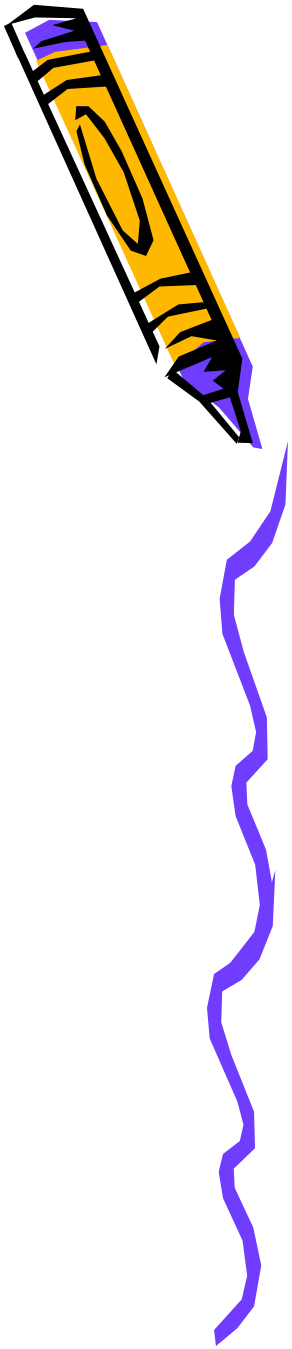
1. 数组的概念



- 一个数组包含一组变量，这些变量通常被称为数组元素，数组元素的数目称为数组长度。数组元素没有自己的名字，而是通过数组变量名和一个下标来标识。
- 一个数组中的每个元素必须具有相同的数据类型，可以是基本数据类型，也可以是引用类型

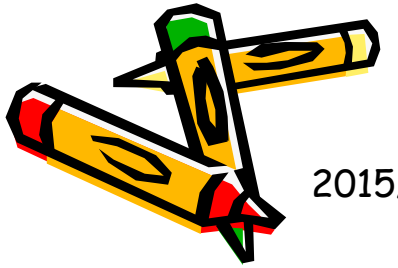


2. 数组的创建

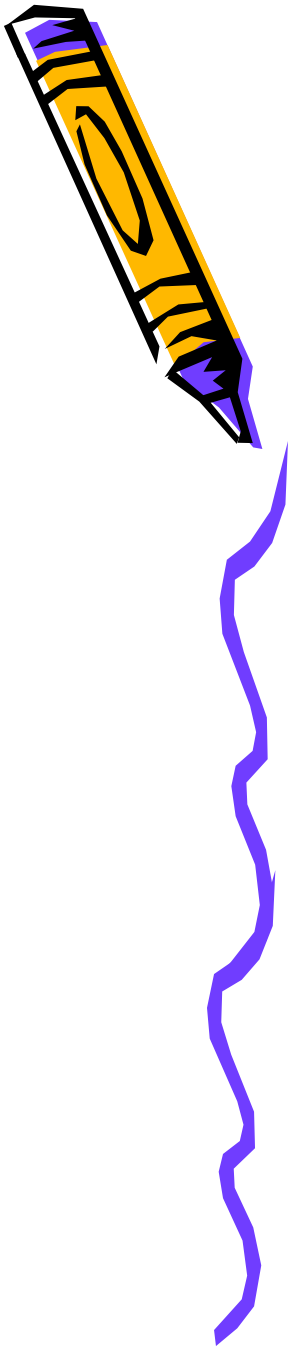


数组的声明：

- 元素类型[] 数组名称
- int[] nums;
- String[] strs;
- String[] args;



2. 数组的创建

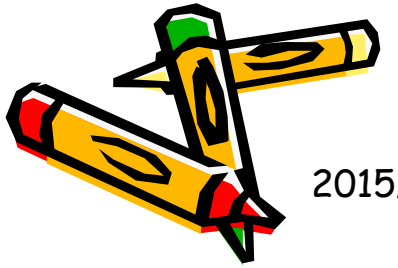


数组的创建：

- nums = new int[10];

也可以将数组的声明和创建合并成一步

int[] nums = new int[10];



3. 数组成员的访问

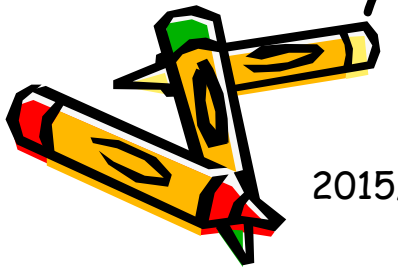
格式：<数组对象引用>[<下标>]

如：nums[0];

注意：数组的第一个元素是通过下标0访问

下标不能超出允许的取值范围，否则将引发运行时例外

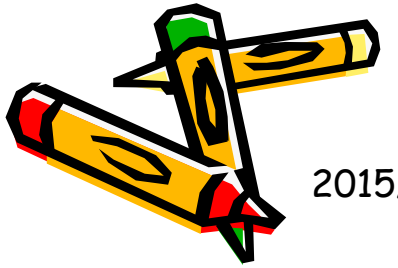
ArrayIndexOutOfBoundsException



3. 数组成员的访问

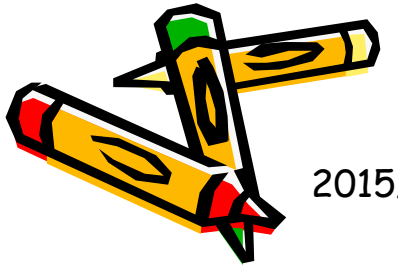
打印一个数组的所有值

```
int[] nums = {1,1,2,3,5,8,13,21};
for(int i = 0; i<nums.length; i++){
        System.out.println(nums[i]);
}
```



4. 别样的for循环

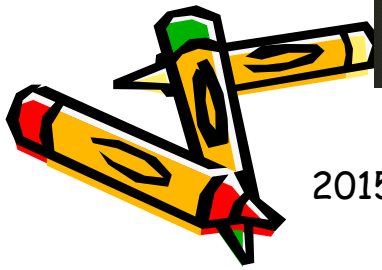
```
for(int x: nums){
        System.out.println(x);
}
```



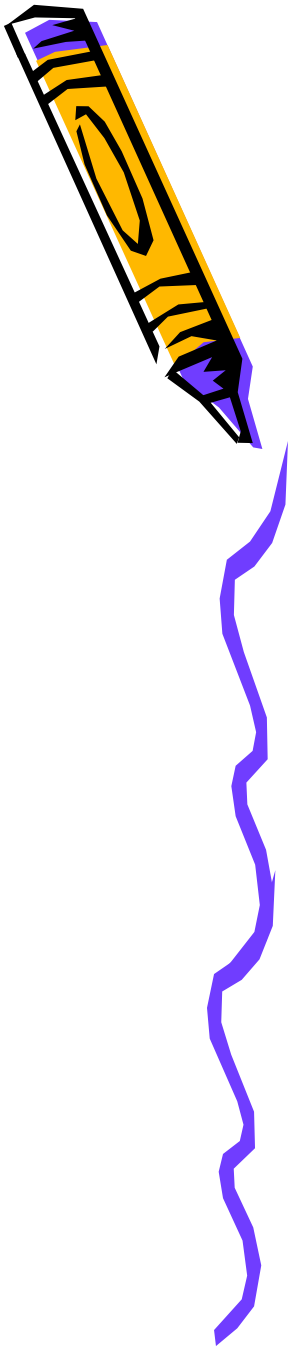
5. 二维数组

所谓的二维数组是指数组的嵌套，即数组中的每个元素都是数组。

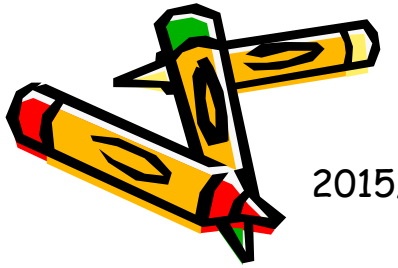
```
int[][] maze = {
        {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
        {1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,1,1,1,1,1},
        {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,1},
        {1,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,1,0,0,0,0,0,0,0,0,1},
        {1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1},
        {1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,1,0,0,1,1,0,0,0,1},
        {1,1,0,0,0,0,0,0,1,1,1,0,0,0,1,0,0,0,0,1,1,0,1,1,1,1,0,0,1},
        {1,0,0,1,1,1,1,1,1,1,1,0,1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,1},
        {1,0,0,1,1,1,0,0,1,1,1,0,0,0,1,0,0,0,0,1,1,0,0,1,1,0,0,0,1},
        {1,0,0,1,1,1,0,0,1,1,1,0,0,0,1,1,0,0,0,0,1,0,0,0,1,0,0,0,1},
        {1,0,0,1,1,1,0,0,1,1,1,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,1},
        {1,0,0,1,1,1,0,0,1,1,1,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,1},
        {1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,1},
        {1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,1,0,0,1},
        {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}};
```



6. 全排列

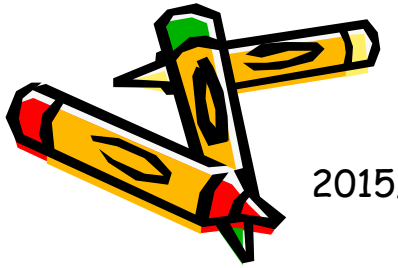


- 用数组存储字符
- 交换数字
- 打印
- 详见Allcombo.java

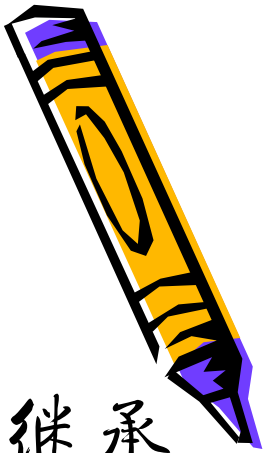


三、多态

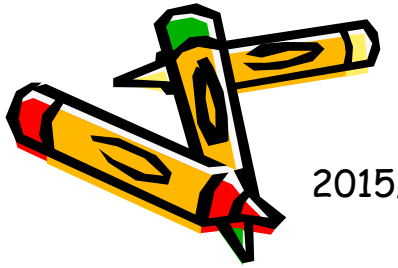
- 继承
- 子类超类的概念
- 类成员
- is-a关系
- 重新认识构造方法
- Object类
- 接口
- instanceof



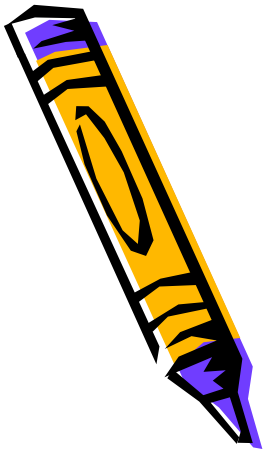
1. 继承



- 继承是面向对象最显著的一个特性。继承是从已有的类中派生出新的类，新的类能吸收已有类的数据属性和行为，并能扩展新的能力
- Java中用extends关键词表示继承

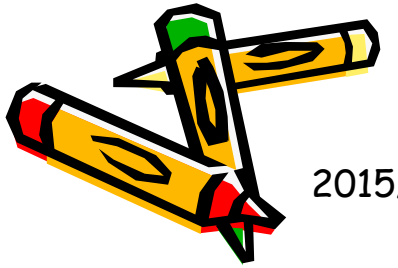


2. 子类和超类



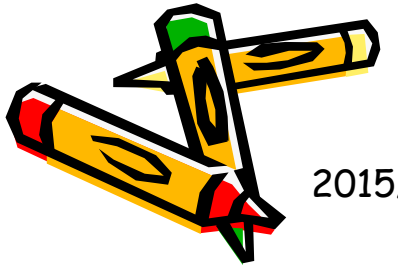
```
class X{int x;}
class Y extends X{ int y;}
class Z extends Y{ int z;}
```

- 类Y是类X的直接子类，类X是类Y的直接超类
- 类Z是类Y的直接子类，类Y是类Z的直接超类
- 类X是类Z的超类

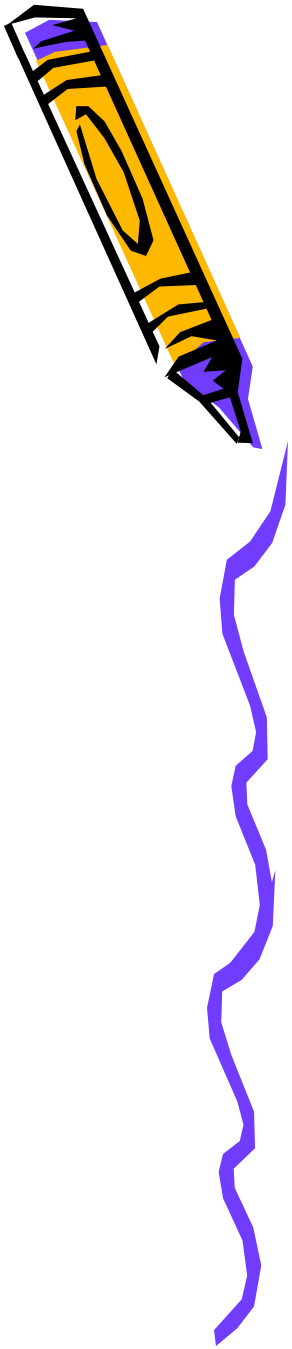


3. 类成员

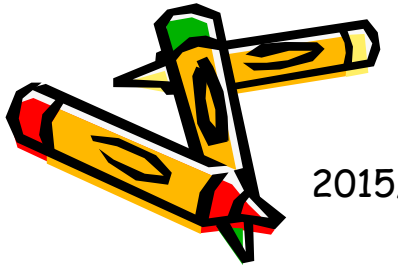
- 超类中声明为private的成员不会被子类继承。
- 若子类和超类不在同一个包中，则超类中没有用protected和public修饰的成员不会被子类继承



4. is-a关系

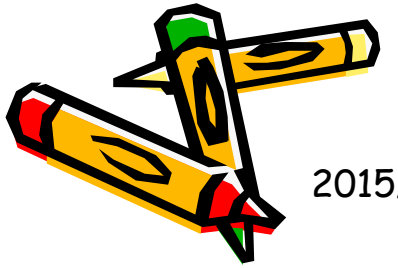


- Z o1 = new Z();
- Y o2 = o1;
- X o3 = o1;
- "is-a"关系是“特殊-一般”的关系
- 手机-智能手机



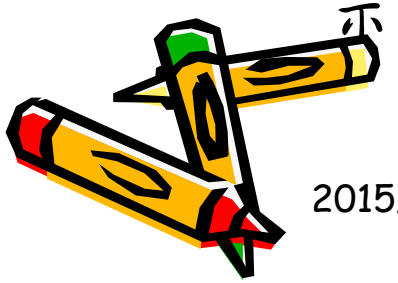
5. 重新认识构造方法

- 如果当前类不是Object类，而某个构造方法体的第一行代码没有调用父类的构造方法，那么编译器会在构造方法体的开始处自动插入一下代码：
  - super();
- 也可以在构造方法体的第一行显式调用超类的构造方法



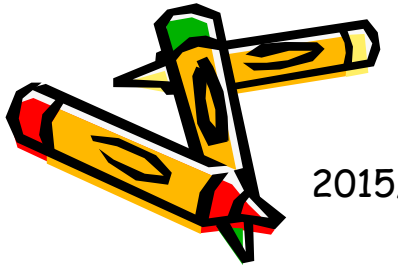
6. Object类

- Java中的所有其他类都是Object的子类
- Object类中定义了以下方法：
  - equals(Object obj) //判断给定对象和当前对象是否相等
  - clone() //生成一个和当前对象**内容一样**的对象
  - toString() //返回当前对象的一个字符串表示

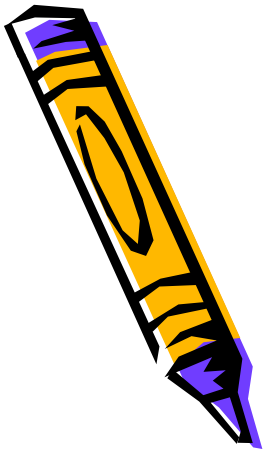


7. 接口

- 接口是一种引用类型，接口中只能定义又名常量和抽象方法，而不能定义普通的成员变量，也不能给出方法的具体实现。
- 方法代码可以再实现该接口的具体类中提供。



7. 接口



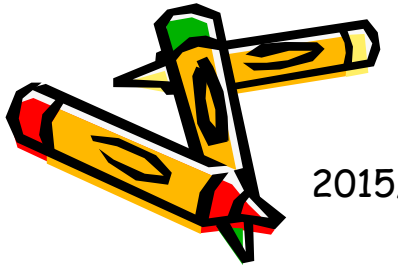
- 接口的定义格式

  [public] interface <接口名> [extends <超接口名表>]{

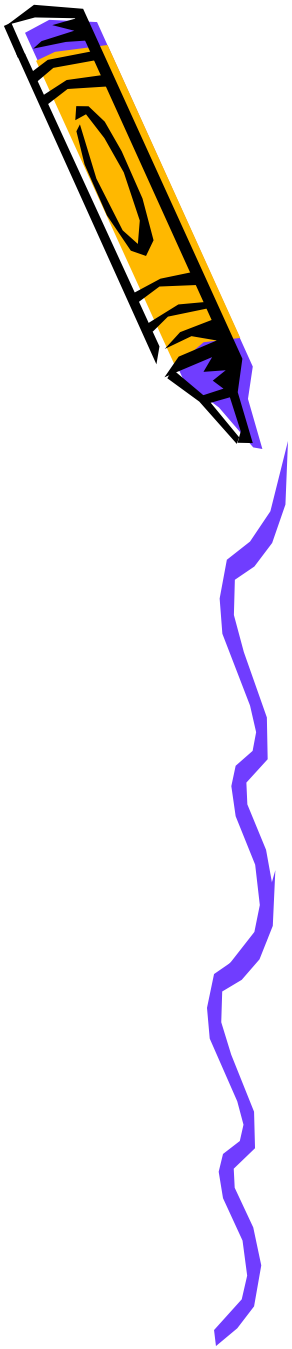
  [<类型> <常量名> = <初始表达式>;]...

  [<返回类型> <方法名> (<形参表>);]...

  }

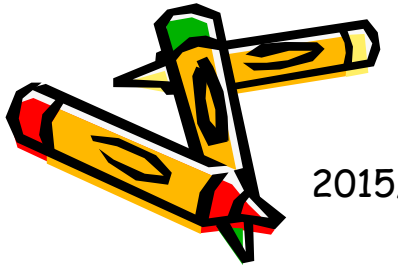


7. 接口

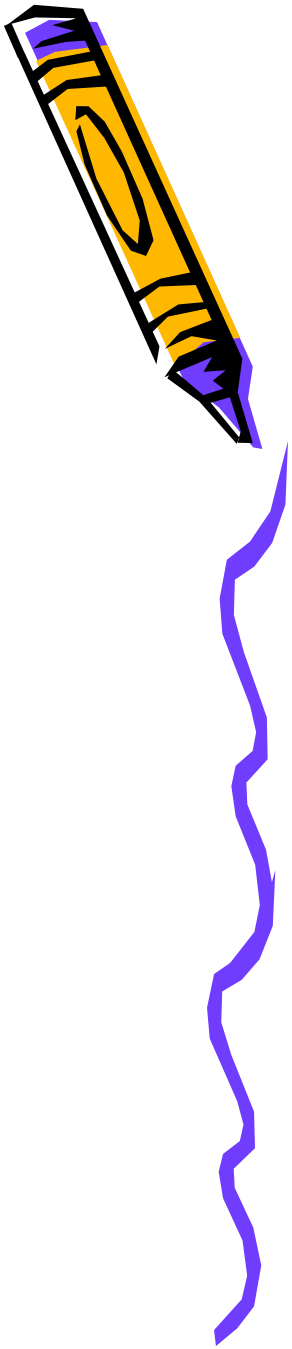


- 例：

```
interface shape{
    double getArea();
    double getParameter();
}
```



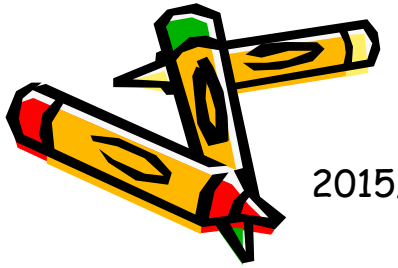
8. instanceof



- 这是一个运算符
- 用于检测某个实例是否属于某个类型
- 如：

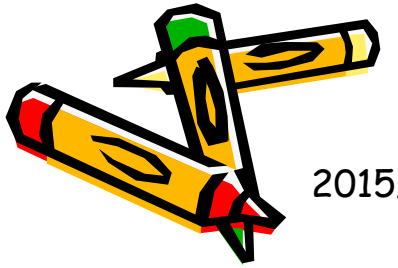
  boolean flag = a instanceof Square;

  该运算返回的类型是boolean型

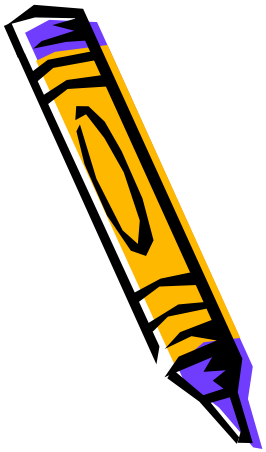


实例：多态实例

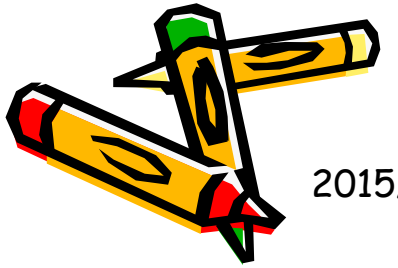
- Shape
- Rectangle
- Square



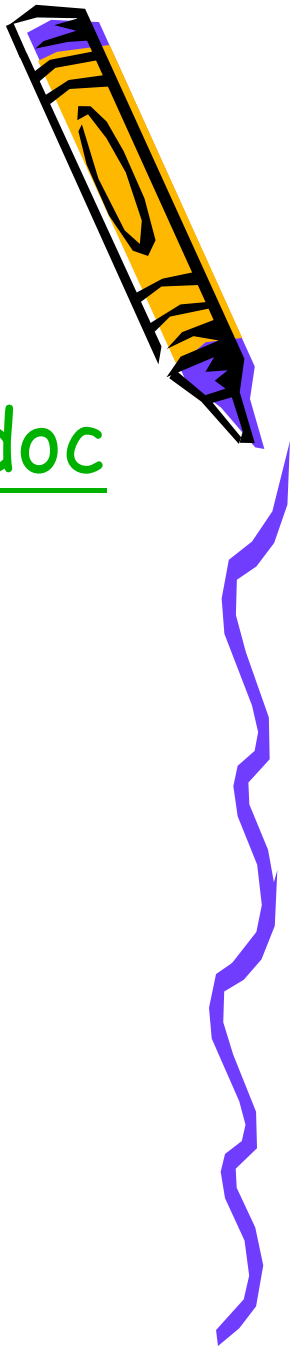
Code Assignment



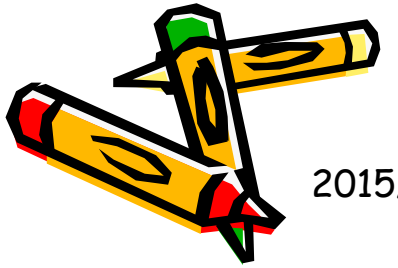
- 判断一串字符串是不是回文
  - 回文：字符串正向打印和逆向打印的结果一样，如："abcdcba"
- 实现二维数组的转置



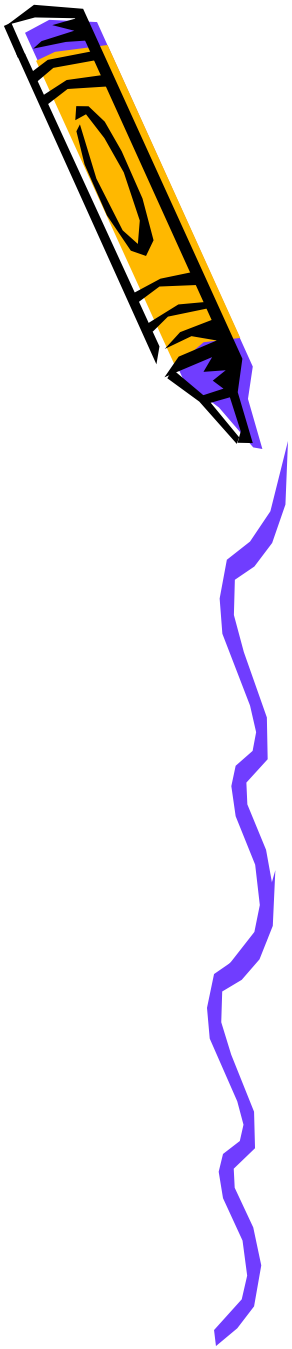
参考网站



- http://docs.oracle.com/javase/8/docs/api/
- http://www.baidu.com



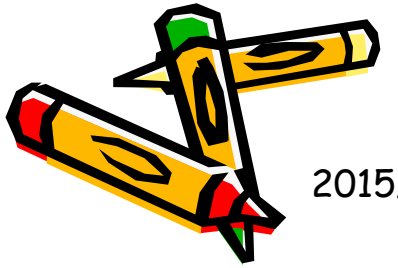
公邮

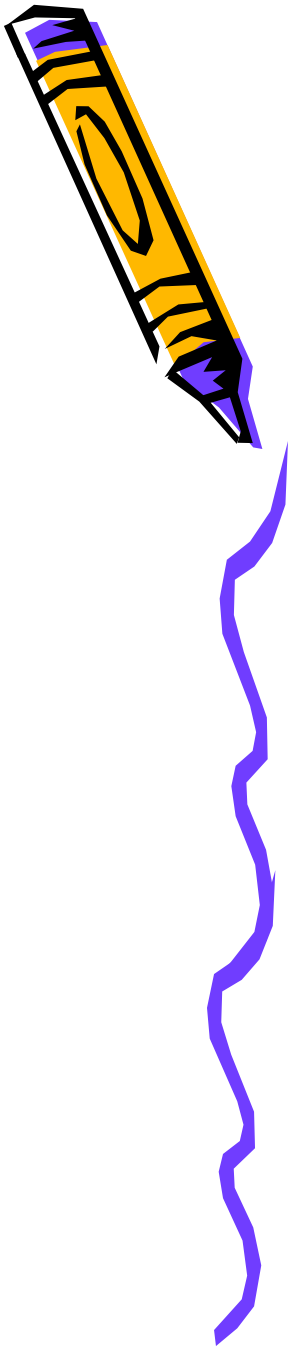


- Email:

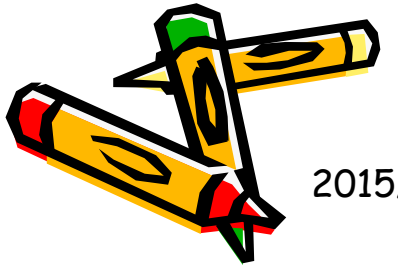
  weblab409@163.com
- Password:

  irm409





THANK YOU



2015/11/20