## 1.Java安装

A.应先安装JDK,后在进行eclipse安装，不然会有一堆懵逼问题在等你。

## 2.Java数据类型

分为**基本数据类型**和**引用数据类型**两大类：

基本数据类型有8种：

**整数类型**{**字节型（byte）,短整型（short）,整型（int），长整型（long）**}; **小数类型**{**单精度（float）,单双精度（double）**};

**字符类型（char）;**

**布尔类型(boolean);**

**(Ps:1.小数类型的数据存储方式和整数不一样；2.小数类型表示的数据范围更比整数的最大范围更大；)**

## java的数据转换

1. **自动转换：小的空间的数据类型向大的空间的数据类型去转换;**

**<1>byte、short、char在做计算的时候会自动的提升数据类型成int;**

**<2>int和其他的数值计算也会溢出，int不会提升而是直接损失精度;**

**<3>任何的整数类型和long类型做运算都会转换成long类型;**

**<4>float、double和所有整数类型计算都会变成对应类型;**

**<5>float和double做计算都会转换成double**

**B.强制转换：大空间的数据类型向小空间的数据类型去转换;**

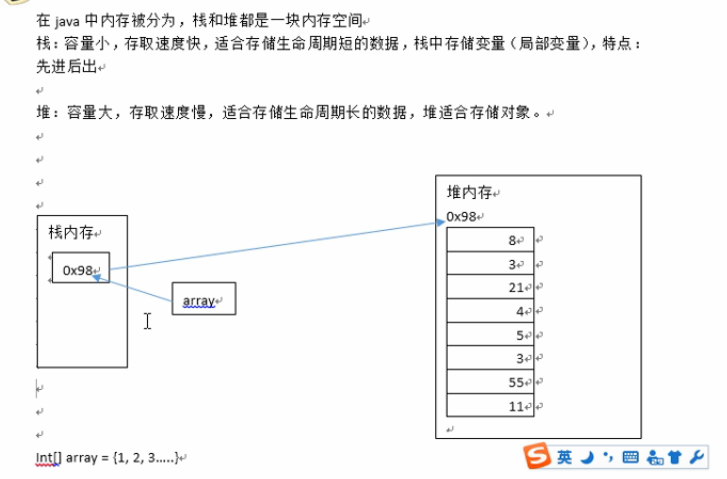
**语法：（目标数据类型）变量名;**

## 位运算符

**PS:计算机中所有的二进制运算都是用的补码; 非负数的反码和补码与其二进制原码一样; 负数的反码是二进制原码的最高位（符号位）不变，其余取反，负数的补码是反码加1;**

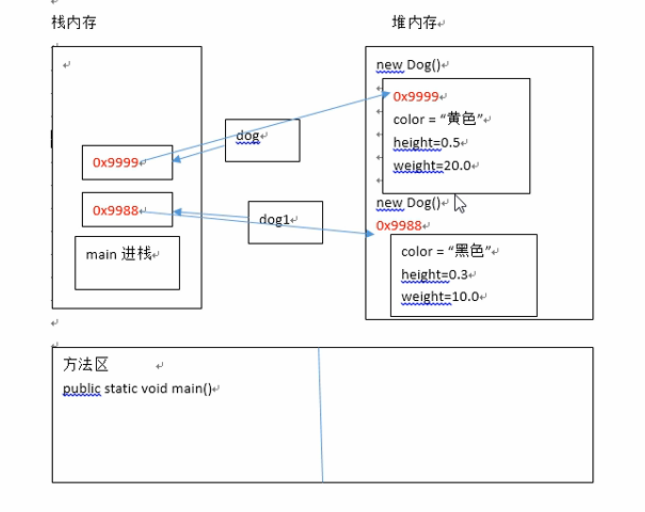
1. **<< 将参与运算的二进制数据向左移动，在低位产生的空位用0来做补齐;**
2. **>> 将参与运算的二进制数据向右移动，在高位产生的空位如果高位是0那么就用0来补齐，如果高位是1就用1来补齐，也就是说右移是带着符号位移动;**
3. **>>> 无符号右移，将参与运算的数据向右移动，在高位所产生的空位0补齐;**

## 数组的内存结构

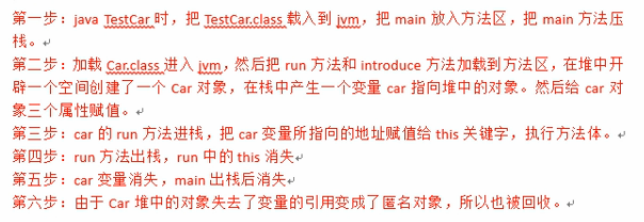


## 对象创建的内存结构

对象存储在堆中。



## 7.程序执行流程说明



## static关键字修饰属性

### 修饰属性

<1>定义类属性：

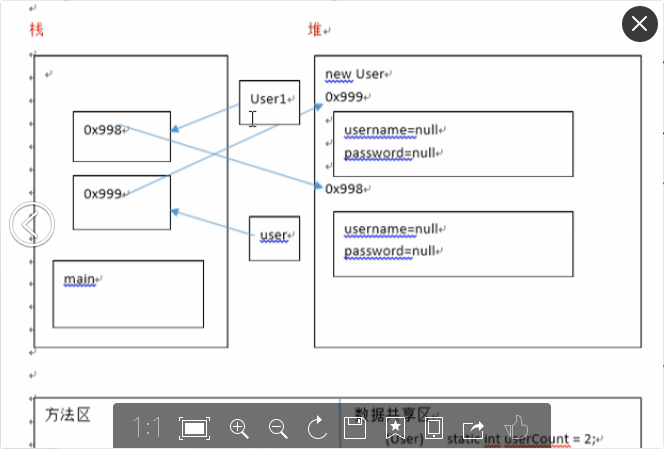
static 数据类型 变量名;

<2>类属性访问：

类名.类属性名 ;（推荐）

对象名.类属性名 ;

<3>static修饰内存结构：





<4>类属性的生命周期：

User.class加载到jvm中时类属性就产生了。

Jvm消失时类属性就消失。

<5>对象属性生命周期：

当对象被创建的时候产生。

当对象所在的方法执行完毕就会被垃圾回收器回收。

### B.修饰方法

<1>语法：

public static 返回值类型 方法名(参数类型 参数，...){

方法体;

return 结果;

}

<2>类的调用：

类名.类方法名;（推荐）

对象名.类方法名;

在类方法中不能使用对象的属性;

类方法只能使用类属性;

## 类的继承

<1>语法：

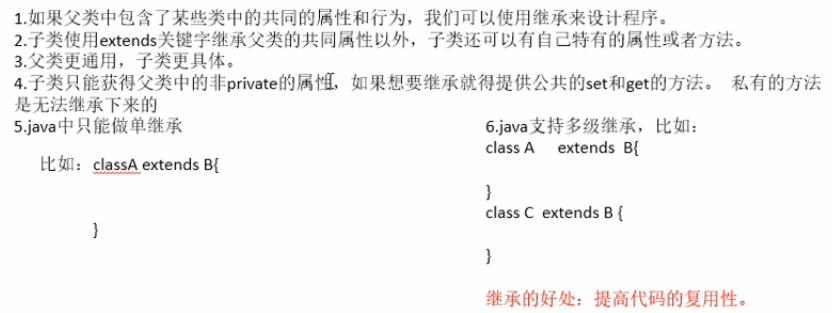
class 子类 extends 父类{

子类的属性

子类的方法

}

<2>特点：



<3>继承的内存结构：

## IMG_25610.关键字：super

<1>自己理解：在子类中的super，相当于其父类的this;

<2>特点：

1. 子类实例化的过程中父类的构造器先被调用，然后再调用子类的构造器。
2. 子类通过构造器实例化时要调用父类的默认构造器，是隐含的调用了super()这个方法。但是如果子类的构造器中调用了父类中的有参数的构造器，那么无参数的构造器就不会被调用。
3. super()的调用必须放在方法的第一行。
4. super可以表示父类的引用，我们可以用super和this来区分父类和子类中同名的属性。在子类中访问属性的查找顺序首先在子类中查找指定的属性名字，如果在子类中没有该属性，那么就去父类中查找该属性。方法也是一样的，当子类和父类有同样的方法时通过子类调用的这个相同方法时调用的是子类的方法。但是我们可以通过super对父类的引用来调用父类的方法。

<3>this关键字：

1. 代表当前类的指定实例的引用。
2. 可以区分同名的属性和局部变量。
3. 通过this可以调用同类中的构造器（this(),this(参数列表)）。
4. 调用本类里面的属性，this.属性名，this.方法()。

## 关键字：final

<1>修饰方法：final修饰的方法不能被重写覆盖。

语法：

权限修饰符 final 返回值类型 方法名(参数列表){

方法体;

}

<2>修饰属性：定义最终常量，无法修改；属性名一般采用大写，多个单词采用下划线隔离，而且这种常量多用于接口。

语法：

权限修饰符 [static] final 数据类型 属性名 = 值;

<3>修饰类:使用final修饰的类是不可以被继承的。

语法：

final class 类名{

属性;

方法;

}

## 抽象类

<1>抽象语法：

抽象方法：public abstract [返回值][void] 方法名(参数名);没有方法体。

抽象类： abstract class 类名{

属性;

方法;

}

Ps:抽象类中可以没有抽象方法，但有抽象方法的类一定是抽象类。

<2>特点：

1. 抽象类不能被实例化。
2. 抽象类必须被子类继承才能使用子类实例化。
3. 继承了抽象类的非抽象类必须实现抽象类的所有抽象方法。
4. 抽象类可以继承抽象类，这样可以不用实现父抽象类的抽象方法。
5. 抽象类的抽象方法不能和private,final,static共存。

## 接口

<1>接口定义： 当一个抽象类中所有方法都是抽象的，那么我们就可以把它定义为一个接口。接口是对行为的抽象，类是对属性和行为的抽象。

<2>接口语法：

interface 接口名{

方法定的定义;

...

}

<3>接口特征：

1. 接口中的方法的定义不需要abstract来修饰，默认是抽象的。
2. 接口是不可以实例化的，需要有类来实现接口，实现接口语法：

class 类名 implements 接口名，接口名，...{

//实现每一个接口中的所有方法；

}。

c.接口中的方法不能和private,static和final共存。

d.在接口中可以定义属性，可以通过接口的实现类的实例来访问，还可以通过接口名来访问（推荐），接口中的属性不能修改，我们接口中的属性默认都是final static 的，通常在接口中来定义属性把它作为常量，常量的名字规范是单词大写，而且多个单词之间用下划线来分隔。

e.接口可以继承接口（单继承）。

## 14.权限修饰符

### A.四种权限的访问修饰符对实体的修饰

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | private | 默认 | protected | public |
| 类（外部类） | No | Yes | No | Yes |
| 方法 | Yes | Yes | Yes | Yes |
| 属性 | Yes | Yes | Yes | Yes |

### 访问权限修饰符对方法和属性的访问限制

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 同类中 | 同一个包（没有父子类关系） | 同一个包（有父子类关系） | 不同包（有父子类关系） | 不同包（没有父子类关系） |
| private | Yes | No | No | No | No |
| 默认 | Yes | Yes | Yes | No | No |
| protected | Yes | Yes | Yes | Yes | No |
| public | Yes | Yes | Yes | Yes | Yes |

## 15.eclipse使用教程

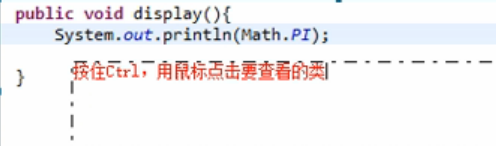
### A.视图的使用

重置视图：Window->Reset Perspective…

### B.项目创建



### 关联源代码：

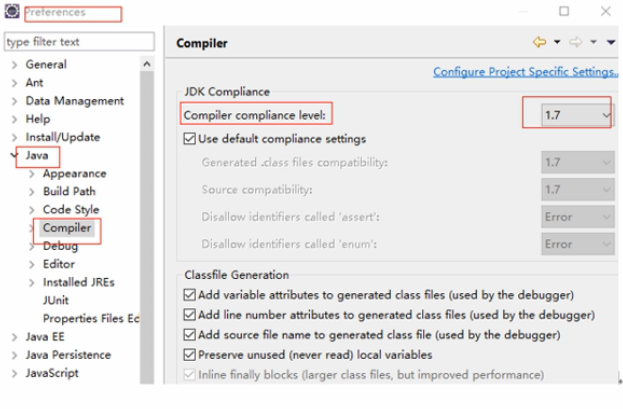


**点击Attach Source…**



### 编译级别的选择

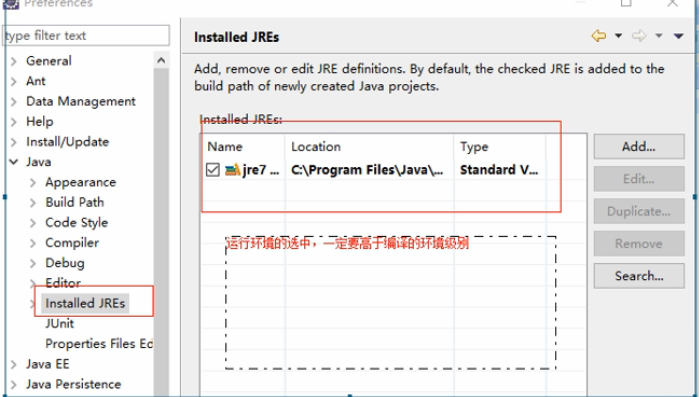
Window --> Preference



所有的软件都是高版本兼容低版本

Jdk1.6编译的class文件肯定能在jdk1.7上运行

Jdk1.7编译的class文件不一定能在jdk1.6上运行



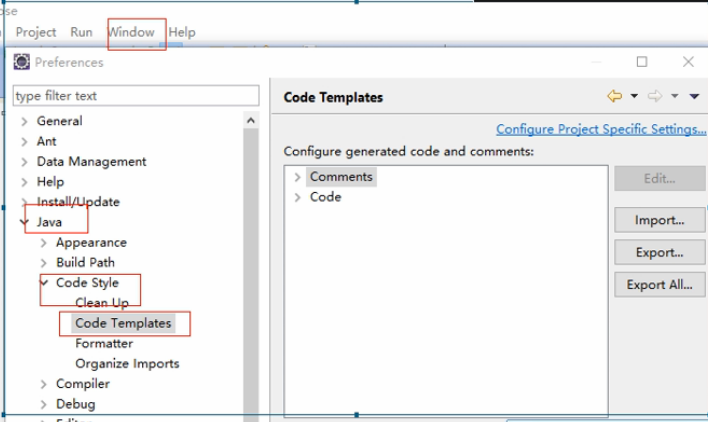
### 调整字体

Window 🡪perference 🡪 General 🡪 Appearance 🡪 Color and Fonts 🡪 Basic 🡪 Text Font

### 注释模板

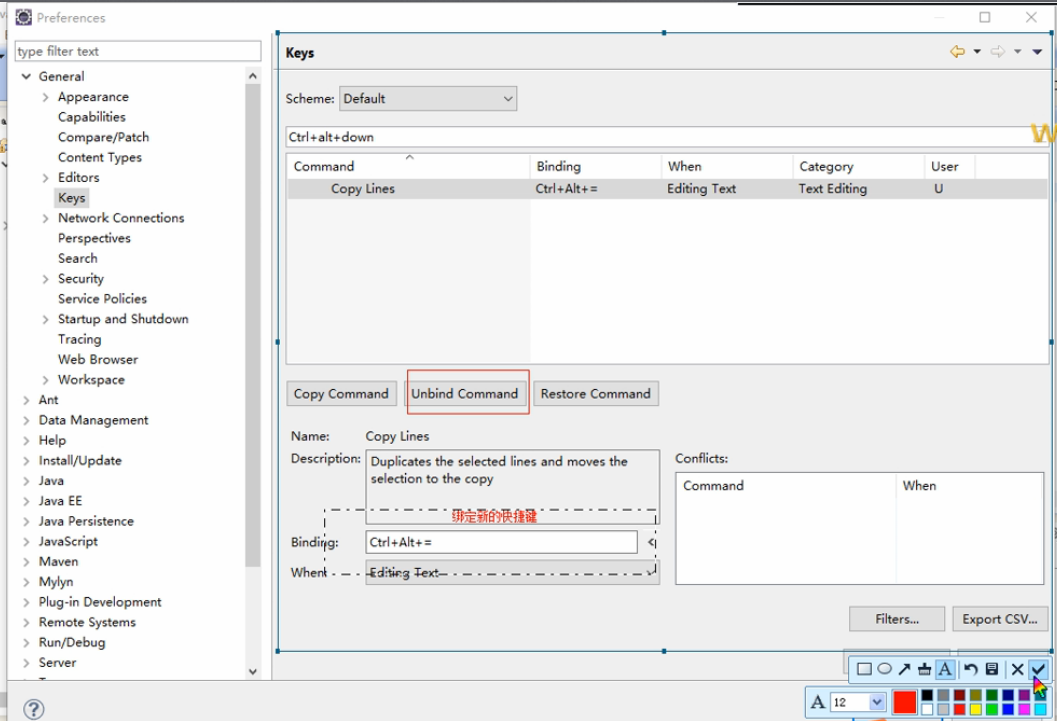
点进去然后自己编写需要的注释模板。

Alt+shift+j使用模板



### 快捷键使用

修改快捷键：



syso:然后alt+/就能出现输出语句;

main:然后alt+/就能出现主方法;

alt+/:补充类或者接口名，帮我们起变量名，new后面直接提示等.

Ctrl+/:单行注释，再按一次，取消单行注释.

Ctrl+shift+/:对选中的区域进行多行注释的封装;

Ctrl+shift+\:用于取消多行注释，只要光标在多行注释中即可;

Ctrl+shift+o:对使用了其他包中的类或者接口进行自动导入，也可以消除没有引用的包。

Ctrl+shift+f:对代码进行格式化;

Alt+向上键:向上移动当前行或者选中行;

Alt+向下键:向下移动当前行或者选中行;

Ctrl+d:删除当前行，或者删除选中行;

Ctrl+shift+x:将选中部分的字母大写;

Ctrl+shift+y:将选中部分的字母小写;

Ctrl+1:光标放在编译中出现红色波浪线提示的内容上,在该快捷键可以获取建议的处理方式;

F3可以查看类或者方法的源码，前提是，该源码已经关联;

Ctrl+Alt+下键:拷贝一行.

Shift+Alt+z:生成try{}catch(){}.

### 代码生成

* + - 1. 在区域中右键（source）
      2. Alt+shift+s

1. 私有字段的get/set

Generate Getters and Setters…

1. 构造方法

Generate Constructor using Fields…

(3)重写父类方法（@Override 注解 表明此方法是重写父类的）

Override/Implement Methods…

## 16.异常

Ps:编译期的异常自行处理即可，运行期异常要向上抛出。

### A.运行期异常及处理

（除了RuntimeException都是编译期的异常）

编译期的异常我们必须处理，如果不处理就会编译失败。

两种解决方案：

**异常中要掌握的方法：**

**getMessage() //返回异常信息**

**printStackTrace() //把异常打印出来**

**toString() //返回具体的异常类和异常信息**

第一种方法：自行解决：

**Try{**

**可能发生异常的代码**

**}catch(异常类型 变量名){**

**处理异常的代码**

**}**

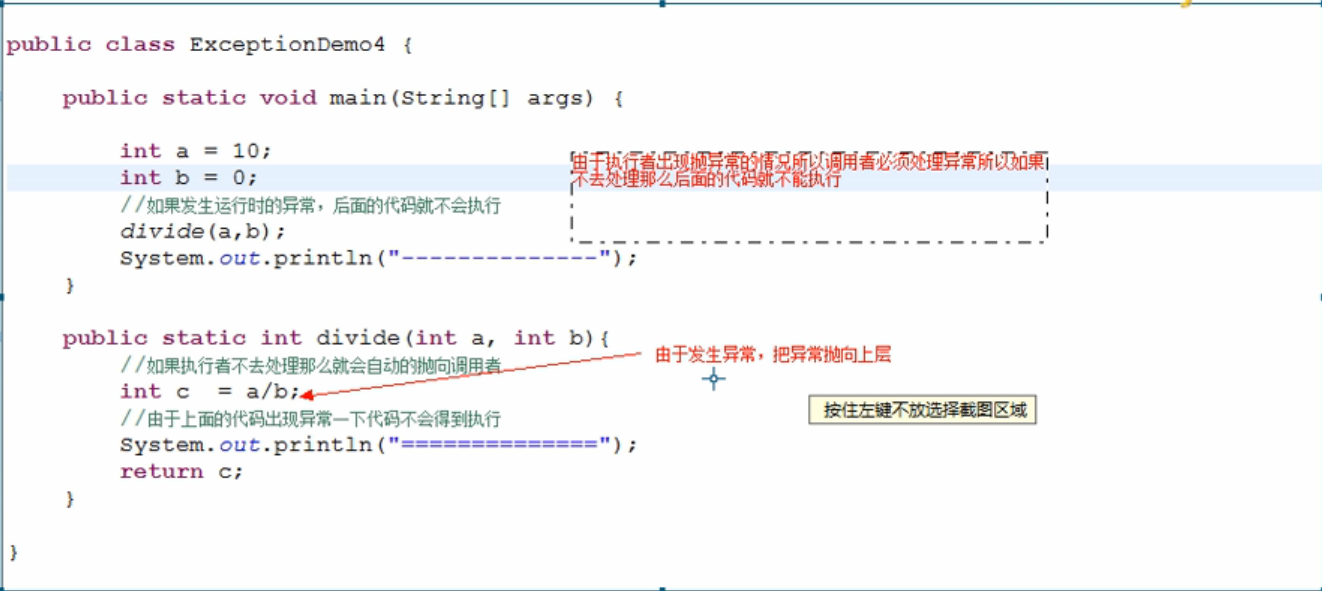
**如果不发生异常就不会产生异常的对象也不会进入catch**

**如果发生异常就会产生一个相应的异常，这个异常的实例会传入到catch中，走catch中的代码**

**如果有异常得到处理（通过catch来处理了）**



第二种方法：向上抛出，如果不处理就自动向上抛。



final,finally,finalize区别？

1.final可以修饰类，方法，属性，如果在类上，类不能被继承，如果在方法上不能重写，如果在属性上值不能改。

2.finally是和try…catch异常处理模块一同使用的语句块，它在最后一定会被执行。

3.finalize当堆中的对象没有任何引用时这个方法会被调用来做垃圾回收。

### B.编译期异常及处理

1.自己处理

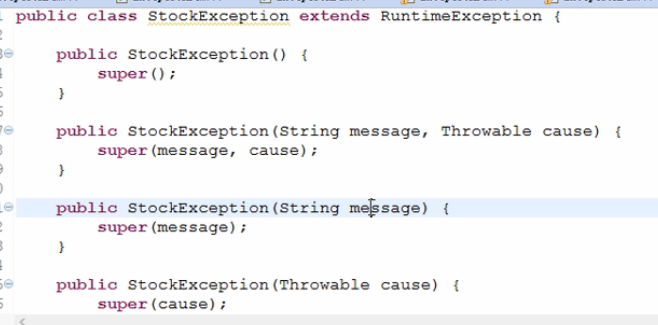
和运行期一样

2.向上抛出

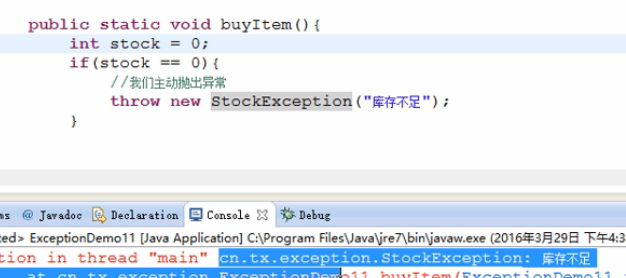


### C.自定义异常（都是运行期异常）

1.自定义的异常：直接继承父类的构造器就可以了。



2.自定义异常的使用：



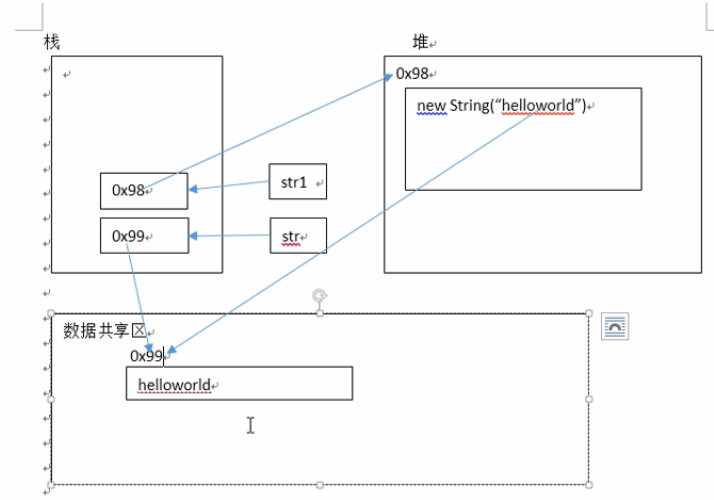
## 17.字符串

A.定义：在定义String str = “abc”的时候，”abc”是常量存储数据共享区，定义时首先会到数据共享区中去查找是否存在”abc”这样的字符串，如果存储在的直接把”abc”的地址赋值给str，如果数据共享区中不存在”abc”那么就去创建一个”abc”。

B.特点：1.常量字符串的特点：不能被改变。

2.String str = “helloworld”在数据共享区中创建的是一个对象

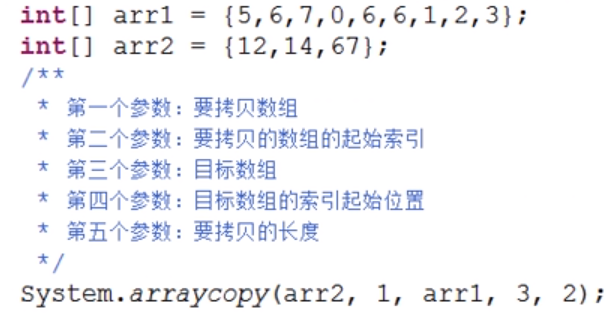
String str1 = new String(“helloworld”),在堆中创建一个字符串对象，然后到数据共享区中



* + - 1. 两个引用字符串拼接后一定会产生新的字符串（即使在数据共享区中存在拼接后值相等的字符串）。如果两个没有引用的字符串常量（就是直接双引号内容拼接）拼接那么就会去查找数据共享区是否有相等的字符串，如果有相等字符串就不创建新的字符串了，直接使用。
      2. String一旦被创建后，值不能改变，如果参与了操作，引用发生了变化，不是在原有的字符串上操作，而是创建了一个新字符串。StringBuffer是线程安全的，效率稍低。
      3. StringBuffer插入字符时，在指定的所因为位置插入字符串，注意不要出现字符串的索引越界的问题。

## 18.System类常用方法：

1.数组拷贝：替换要拷贝的原先元素



2.获取当前时间：从1970年到现在的毫秒数：System.currentTimeMillis()

3.退出jvm：System.exit(0)

4.调用垃圾回收器：System.gc()

## 19.数组与集合：

数组：长度固定

可以存储基本类型数据，也能存储对象。

集合：长度可变

只能存储对象类型（由于有包装类的存在，集合可以存储任何类型）。

### 1.ArrayList

List接口的大小可变数组的实现。实现了所有可选列表操作，并允许包括null在内的所有元素。除了实现List接口外，此类还提供一些方法来操作内部用来存储列表的数组的大小。

特点：ArrayList中的元素和可以重复。

是有序的集合，长度不固定。

不是线程安全的。

效率高。

### **2.LinkedList**

List接口的链接列表实现。实现所有可选的列表操作，并且允许所有元素（包括null）。除了实现List接口外，LinkedList类还为在列表的开头及结尾get,remove和insert元素提供了统一的命名方法。这些操作允许将链接列表用作堆栈、队列或双端队列。

不是线程安全的。

### 3.Set

特点：1.允许有null，但是只能有一个；

2.无序性；

3.不能重复。

Set集合不会出现ConcurrentModificationException;

但不是线程安全的。

### HashSet

特点：1.元素唯一性

2.无序性

3.允许一个null存在

4.不是线程安全（效率高）

### LinkedHashSet

特点：1.元素唯一性

2.有序性

3.允许一个null存在

4.不是线程安全（效率高）

### TreeSet

特点：1.元素唯一性

2.可自定义排序

3.不允许null存在

4.不是线程安全

### 7.TreeMap

特点：1.可以按着key来做排序

2.key不能null,key不能重复，值可以有多个null

3.不是线程安全的

### 8.HashTable

特点：1.HashTable是map的实现类

2.不允许任何null值和null键

3.Hashtable中的元素没有顺序（跟添加的顺序无关）

4.HashTable是线程安全的。

### 9.LinkedHashMap

特点：1.LinkedHashMap是map的实现类

2.允许多个null值和一个null键

3.LinkedHashMap有顺序（添加的顺序）

4.LinkedHashMap不是线程安全

## 20.增强的for循环：

for(数据类型 变量 ：集合){ //增强的for循环有并发问题存在

}

for(String s : sc){ //数组从上往下将元素依次赋给s。

System.out.println(s);

}

## 21.IO流

流的介绍：字符输出流

字符输入流

高效缓冲流

### 字符流

字符流writer后一定要刷新（flush），close自带刷新效果。

先关输出流再关输入流。

### 字符流与字节流转换

1. 字节流转换为字符流（只能是输入的时候）.
2. 字符流转换为字节流（只能是输出的时候）

### 打印流（PrintStream）

1. 可以打印各种数据类型。
2. 封装了字符输出流，还可以字符流和字节流的转换。
3. 可以通过构造器实现自动刷新。
4. 可以直接向文件中写数据。

### Properties

1. 表示了一个持久的属性集，可保存在流中或从流中加载，属性里表中每个键及其对应值都是一个字符串。
2. 继承于HashTable，是线程安全的键值对存储结构。
3. Properties可保存在流中或从流中加载。
4. 只能保存字符串的键值对。

## 22.线程

### A.线程的实现方式：

1.方式一：继承于jdk提供的java.lang.Thread，必须重写run方法，run方法是线程体。

2.方法二：声明实现Run接口的类，然后该类实现run方法，然后可以分配该类 的实例，在创建Thread时座位一个参数来传递并启动。

### B.synchronized(锁对象){}加载地方？

1.代码被多个线程访问

2.代码中有共享的数据

3.共享数据被多条语句操作

### C.并发

1.Synchronized同步代码块的锁对象可以是任意类对象（线程的实现方法是使用继承于Thread），这个对象必须是线程类共享的（静态的）。

2.Synchronized是可以加在方法上，如果是静态方法Synchronized的锁对象就是类的类对象，如果不是静态的方法，Synchronized如果在对象方法上，那么他的锁就是this.

### 休眠

1. 如果休眠是在同步代码块中执行，休眠不会让出锁对象。

### 线程间的通信

1. 线程间的通信共享数据一定要有同步代码块synchronized。
2. 一定要有wait和notify,而且二者一定是成对出现。
3. 生产者和消费者的线程实现一定是在while(true)里面。

## 23.网络编程

### A.三要素

IP、端口、协议；

### UDP和TCP

UDP：

将数据源和目的地封装到数据包中，不需要建立连接。

每个数据包的大小限制在64k。

因为不建立连接，所以安全性差，速度快。

TCP:

建立连接形成传输数据的通道，然后再传输数据，通过三次的握手。

安全性高，效率低。

### InetAddress类

用于管理IP的类，没有构造器。

1. 单例模式。
2. 根据静态的方法来返回该对象。

### UDP开发步骤

1. UDP服务器端的步骤
2. 创建socket服务对象：

UDP的服务端的构造器

DatagramSocket(int port)：创建数据报套接字并将其绑定到本地主机上的指定端口。

1. 接收数据：使用数据包的方式来接收，用到了DataGramPacket此类表示数据报包。

DatagramPacket(byte[] buf,int length)：构造DatagramPacket，用来接收长度为length的数据包。

1. 解析数据

解析数据包：

InetAddress->getAddress():返回某台机器的ip地址，此数据报将要发往该机器或者是从该机器接收到的。

byte[]->getData():返回数据缓冲区。

int->getLength():返回将要发送或接收到的数据的长度。

1. 释放资源

关闭套接字。

### TCP开发步骤

1. TCP服务端：
   1. 创建服务器Socket对象，指定端口。ServerSocket(int port):创建绑定到特定端口的服务器套接字。
   2. 2.监听并获得客服端的对象Socket。Socket🡪accpet():监听并接受到此套接字的连接。