**java摘要**

目录

[编程思想： 2](#_Toc467859601)

[编码 2](#_Toc467859602)

[标识符： 3](#_Toc467859603)

[类名. 和 this.的区别 3](#_Toc467859604)

[关键字&及延伸的概念 4](#_Toc467859605)

[final 最终的（版本） 作为一个修饰符 4](#_Toc467859606)

[static 静态 作为一个修饰符 4](#_Toc467859607)

[extends 继承 5](#_Toc467859608)

[abstract 抽象 5](#_Toc467859609)

[抽象类的特点 5](#_Toc467859610)

[implements 实现，实施，执行 6](#_Toc467859611)

[instanceof 判断某一类型引用指向的对象到底符合什么类型 6](#_Toc467859612)

[private 封装，私有 7](#_Toc467859613)

[reference 参考参照 7](#_Toc467859614)

[笔记 9](#_Toc467859615)

[Day07 9](#_Toc467859616)

[子类的实例化过程（解释什么是实例化） 9](#_Toc467859617)

[继承 extends 9](#_Toc467859618)

[接口 10](#_Toc467859619)

[接口的特点 10](#_Toc467859620)

[聚集关系 11](#_Toc467859621)

[多线程 12](#_Toc467859622)

[理解 14](#_Toc467859623)

[为什么字符串可以直接点方法 14](#_Toc467859624)

[内存图（左栈，右堆） 14](#_Toc467859625)

[作用域 15](#_Toc467859626)

[Java EE 16](#_Toc467859627)

[我的理解 16](#_Toc467859628)

[1、编码简介 16](#_Toc467859629)

# 编程思想：

累加思想

计数器思想

string 定义字符串类型

char 字符型，只是单个字符

# 编码

asc码: 美国 1个字节的7位 -128-127

iso-8859-1： 欧洲 latin-1

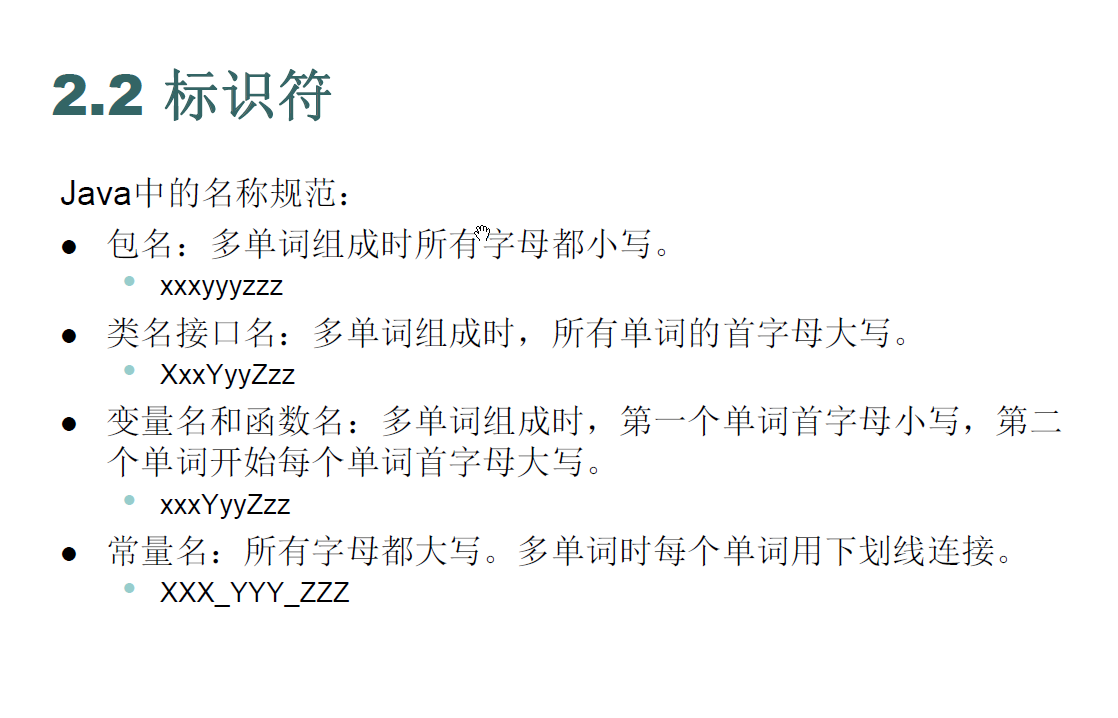
gb2312: 字母用1个字节，中文用两个字节

gbk：是在gb2312基础上进行进一步扩展和完善 中文用两个字节

utf-8：万国码 ,因为浏览器是面向世界的，浏览器默认解析的是utf-8，中文在网络中传输会产生问题，所以只要是web就必须用utf-8的编码，所以工作空间要改为utf-8的编码否则中文在浏览器显示就会出现乱码

GBK是本地的码，是中国的专用码

# 标识符：



三元运算符

语法格式：条件表达式？表达式1：表达式2

public 公共

private 私有

# 类名. 和 this.的区别

为什么静态调用要用类名. 非静态要用this.？

我们用大腿想一想  
this是要new出来的  
static是不用new的  
就是当你static存在的时候  
那个this没有new  
是不存在的

this引用当前对象，但是static方法并不依赖某个对象，通俗点就是类方法，并不需要创建对象就可以调用的。而this没有对象就根本不存在，你让方法怎么执行，

所以this不能用作静态的调用 如果this.一个静态的成员

那么在没有建立对象的情况下，this是不存在的，根本就不能实现调用，因为静态不需要对象就先存在了，这是先后的问题

非静态前面省略的是this. 静态前面省略的是 类名.

# 关键字&及延伸的概念

## final 最终的（版本） 作为一个修饰符

1、可以修饰类，函数，变量

2、被final修饰的类，不可以被继承，为了避免被继承，被子类重写功能

3、被final修饰的方法，不可以被重写

4、被final修饰的变量是一个常量只能赋值一次，既可以修饰成员变量，也可以修饰局 部变量

5、内部类定义在类中的局部位置上时，只能访问该局部被final修饰的局部变量

（当在描述事物时，一些数据的出现是固定的，这时为了增强阅读性，都给这些值起个名字，方便阅读。而这个值不需要改变，所以加上final修饰）

一般定义了final的变量都会这样定义

（当然也可以根据需要定义其他的成员，如函数，不能定义这样定义类，因为修饰类的关键字目前学到的只有，public 和 final 这两个，像private是修饰成员的，static等也是修饰成员的，不是用来修饰类的）

public static final double PI = 3.14 ; （可以称之为 全局常量）

因为这个值已经被固定了，永远都不会被更改，那么就算是给他最大的权限，并且将它用static共享了，也无所谓，反正他们也改变了这个值，这样给足了权限反而让其他函数调用更加的方便；当然如果不希望被调用和共享，就不需要加；

## static 静态 作为一个修饰符

1、用于修饰成员（成员变量，成员函数），不能修饰局部

2、当成员被静态修饰后，多了一种调用方式，除了被对象调用，还可以直接被类名调 用格式类名.静态成员

被它定义的成员会在类加载的时候就在方法区（共享区，数据区）产生

它的数据时被共享的，因为共享区嘛，所有的对象中，都会有这个成员，多个对象中有相同的成员，就可以将它提取出来，也就用static修饰，这样他就不用每一个对象中都在堆内存中生成这个数据，只用在方法区存在一个就好了，这样就可以共享出去了

**作为常量：**

常量的书写规范是，所有的字母都大写，如果由多个单词组成，单词间通过 \_ 连接

## extends 继承

用于子类继承父类

## abstract 抽象

1、抽象：笼统的，模糊的，不具体的，看不懂的东西就是抽象的

2、对于这种，完全看不懂的方法，我们用一个关键字来标识一下

用他标识的标识，抽象的方法，看不懂的方法

3、抽象方法，必须放在抽象类中。因为普通类调用抽象的方法没有意义 ，方法都没有主体你调用他有什么意义呢。

4、一般含有抽象方法的类都是抽象类，所以，这个类中有看不懂的抽象方法，它没有主体，所以不要创建这个类的对象，因为调用这个类中的某些方法是没有意义的，不能去创建一个你看不懂的一个类的对象。编译也不会通过，会报错

5、只能修饰类和方法，你见过抽象变量嘛？

### 抽象类的特点

抽象：看不懂

1、抽象方法一定定义在抽象类中

2、抽象方法和抽象类，都必须被abstract 关键字修饰

3、抽象类不可以用new 对象，因为调用抽象方法没意义

4、抽象类中的抽象方法要被使用，必要由子类复写其所有的抽象方法后，建立子类对象调用，如果子类只覆盖了部分抽象方法，那么该子类也会成为一个抽象类

（因为子类继承抽象类以后，会在子类中将抽象类的所有方法都产生在子类中，（因为父类是将相同的功能向上抽取都到的，所以子类中就会自动产生父类所有的成员）那么你复写了一个，另一个还存在在子类中，并且没有被复写，所以另一个方法是抽象的，含有抽象方法的类也就会变成抽象类）

5、抽象类既可以有抽象方法，也可以有非抽象方法

抽象类和一般类，没有太大的不同。改怎样描述事物，就怎样描述。只不过该事物中出现了一些看不懂的东西，这些不确定的部分，也是该事物的功能，需要明确出来，但是无法定义主体，通过抽象方法来表示

抽象类比一般类，多了个抽象方法（函数）

抽象类不可以实例化，也就是不能用new创建对象，不能产生实例

特殊：

抽象类中可以不定义抽象方法，这样做仅仅是不让该类建立对象

## implements 实现，实施，执行

类与类之间的关系是继承，extends

类与接口之间的关系是实现 implements

## instanceof 判断某一类型引用指向的对象到底符合什么类型

**英译：实例，运算符**

也就是判断引用数据类型直接的关系

## private 封装，私有

只能修饰成员，不能修饰成员函数内的东西，也就是不能修饰局部了

## reference 参考参照

# 笔记

## Day07

### 子类的实例化过程（解释什么是实例化）

实例化说白了，就是将子类new为对象，实例化的过程，就是他是怎么初始化的

这个过程呢：就是在子类的每一个函数的第一行都有一句隐式的语句super()

用来调用父类中的空参数的构造函数，用作初始化，现有父类，也就是这么来的

没有空参数就需要手动在子类中定义super来初始化父类中对应的构造函数，和this用法基本一致

### 继承 extends

1、

当多个类中出现相同的功能，但功能主体不同，这时可以进行向上抽取，只抽取功能定义，不抽取功能主体。

**代码体现：**

abstract class Student

{

[abstract](#_abstract_抽象) void study( ) ; //后面不用加 { }，只抽取功能定义

}

class BaseStudent extends Student

{

void study()

{

S.o.p(“study”);

}

}

class AdvStudent extends Student

{

void study()

{

S.o.p(“study”);

}

}

## 接口

接口的其他笔记请见 [Day07\InterfaceDemo.java](../../workspace/Day07/src/InterfaceDemo.java)

### 接口的特点

1、接口是对外暴露的规则

2、接口是程序的功能扩展

3、接口可以用来多实现

4、类与接口之间是实现关系，而且类可以继承一个类的同时实现多个接口

5、接口与接口之间可以有继承关系

## 聚集关系

除了继承和实现，其实还有很多的关系存在

比如.聚集

聚集分两种：has a 谁里边有谁

聚合：紧密程度一般

组合：紧密程度更高

## 多线程

# 理解

## 为什么字符串可以直接点方法

一直在说字符串本身即是一个对象，但是不知道为什么，下面给解释

因为java语言有一个特性，字符串可以用带引号的文本初始化。通常，必须对对象采用一种更通用的初始化方式（new，通过构造函数）。

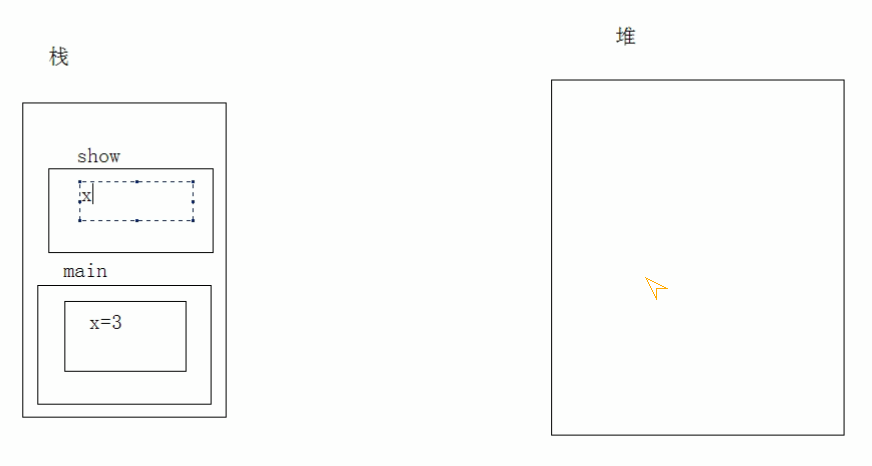
而这里直接用引号初始化，所以字符串就被初始化了，初始化后就是一个对象，所以字符串本身就是一个对象，必须要引号的那种，因为是用引号初始化后，才是对象

一次他即是一个对象也是一个字符串

还可以这么理解

string 的包装类也是string，包装类就是创对象的，所以字符串也就是一个string，这里的string指的是包装类型，所以字符串直接就是一个string类型的对象

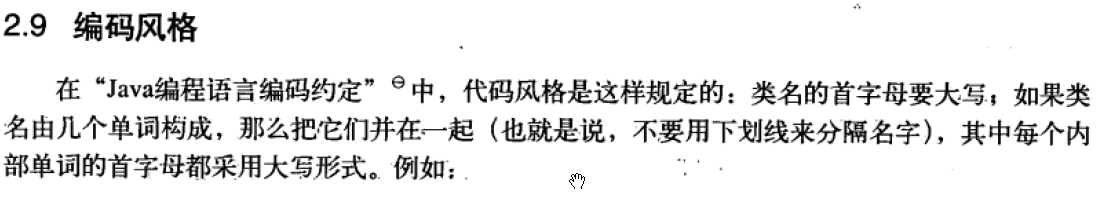
## 内存图（左栈，右堆）



## 作用域

基本数据类型的作用域由花括号决定

对象的作用域，是可以存在于作用域之外的，这也是为什么你在任何文件中都可以调用java给的静态方法，因为对象可以在作用域外存在，哪怕没有在java的那个对象所在的文件中，在其他文件中照样可以使用



# Java EE

## 我的理解

### 1、编码简介

asc码:美国 1个字节的7位 -128-127

iso-8859-1：欧洲 latin-1

gb2312:字母用1个字节，中文用两个字节

gbk：是在gb2312基础上进行进一步扩展和完善 中文用两个字节

utf-8：万国码