目录

[1. 所有的对象都有状态，行为，和标识。 1](#_Toc512194835)

[2. 状态：属性或者字段以及它们当前的值构成的集合 1](#_Toc512194836)

[3. 所有的对象都有接口 1](#_Toc512194837)

[4. java类： 1](#_Toc512194838)

[5. 0在java里并不意味着false 1](#_Toc512194839)

[6. String的两类创建 1](#_Toc512194840)

[7. 字面值 literals 和 常量 constant 不是同一个东西。 1](#_Toc512194841)

[8. Boolean Literals 只有两个值：true 和 false 1](#_Toc512194842)

[9. 整型提供八进制，十进制，十六进制，long型（需要在数字后面加上l或L）。 1](#_Toc512194843)

[10. 在java7及以后中，下划线可以加在数字的任何地方。但是不能加在这些地方： 1](#_Toc512194844)

[11. 二进制字面值： 1](#_Toc512194845)

[12. 自从java5之后，字符就开始用unicode表示而不是ASCII了 1](#_Toc512194846)

[13. 数组的初始化： 1](#_Toc512194847)

[14. java 关键字 2](#_Toc512194848)

[15. 变量： 2](#_Toc512194849)

[16. static &&final 2](#_Toc512194850)

[17. final 5](#_Toc512194851)

1. 所有的对象都有状态，行为，和标识。
2. 状态：属性或者字段以及它们当前的值构成的集合

行为：方法的集合

标识：由对象自己表示，与其它相似的对象依靠标识来区别。

1. 所有的对象都有接口
2. java类：

是一种数据类型，即对象类型。

是程序的一个模块。是面向对象程序设计中最基本的程序单元。

1. 0在java里并不意味着false
2. String的两类创建

String s = new String("asdf");

String s = "asdf";

1. 字面值 literals 和 常量 constant 不是同一个东西。
2. Boolean Literals 只有两个值：true 和 false
3. 整型提供八进制，十进制，十六进制，long型（需要在数字后面加上l或L）。

数字后面每年有f、F的是float，有d、D的是double

1. 在java7及以后中，下划线可以加在数字的任何地方。但是不能加在这些地方：
2. 在一个数字的开头或者结尾。
3. 浮点型小数点的附近。
4. F或L下标的前面。
5. 该数值型字面值是字符串类型的时候
6. 二进制字面值：

Byte，short , int, long 都可以用二进制数系统表示。只需要在数字之前加上0b或者0B就可以了。

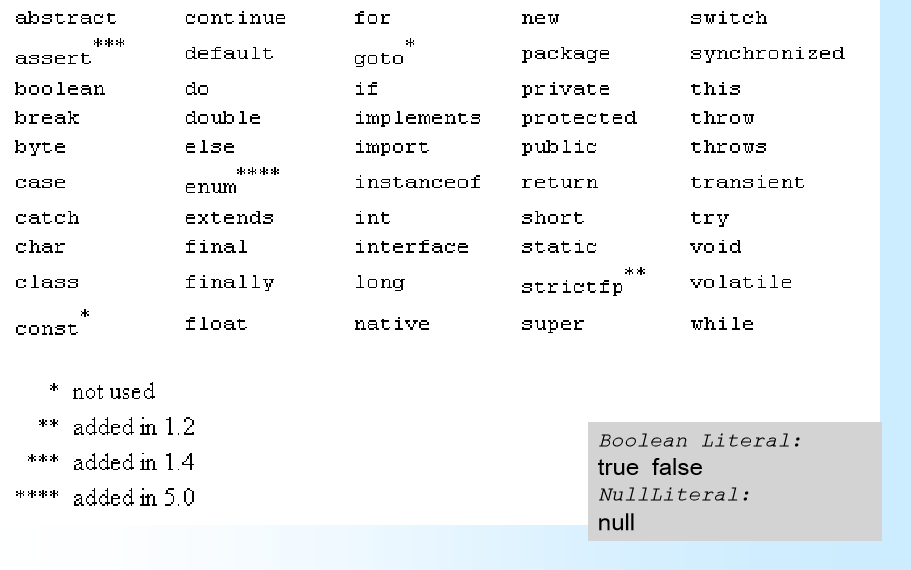
1. 自从java5之后，字符就开始用unicode表示而不是ASCII了

<https://blog.csdn.net/lx697/article/details/5914417> 两者的区别

'\u03df' \u开头的都被看做十六进制

1. 数组的初始化：

Int a[] = {1,3,5,7,9};

1. java 关键字
2. 变量：

成员变量：类里面定义的变量

如果类中的变量有static关键字修饰，叫做静态变量或者类变量；

如果类中的变量没有static关键字修饰，就叫作非静态变量或实例变量。

局部变量：方法内定义的变量，形参、代码块中定义的变量都叫做局部变量。

方法参数

成员变量在所在类被实例化后，存在堆内存中；局部变量在所在方法调用时，存在栈内存空间中。

1. static &&final

两者都有很多种用法，而且在一定环境下使用，可以提高程序的运行性能，优化程序的结构。

1. Static修饰成员变量：

public class Person {

String name;

static int age;

/\* 其余代码不变... \*/

/\*\*Output

\* Name:zhangsan, Age:12

\* Name:lisi, Age:12

\*///~

}

这里的age是static修饰的，age被放在静态存储区中，有类Person直接管理，创建的person对象不再具备这个age属性，而是交给类去管理，多个类实例对象只对应一个age属性，一旦一个age做了改变，其他的对象都会受到影响。

以下面这个程序为例看究竟是什么结果，什么影响：

public class Person {

String name;

int age;

public String toString() {

return "Name:" + name + ", Age:" + age;

}

public static void main(String[] args) {

Person p1 = new Person();

p1.name = "zhangsan";

p1.age = 10;

Person p2 = new Person();

p2.name = "lisi";

p2.age = 12;

System.out.println(p1);

System.out.println(p2);

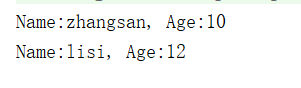
}

/\*\*Output

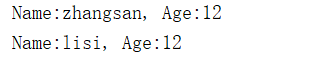
\* Name:zhangsan, Age:10

\* Name:lisi, Age:12

\*///~

}//程序间program中的Person.java

每个实例都有一个age，因为每个对象的创建都为age分配了堆存储。

Age变量前加了static后，后面的赋值语句覆盖了前面的语句。

Age static前再加一个private后，结果还是这样。

1. 修饰成员方法

因为方法本来就是存储在类的定义中，所以加了static也一样，还是在静态存储区中。

Static修饰成员方法的最大作用就是使用“类名.方法名”来操作，避免了要先new出对象的繁琐和资源操作。

1. Static修饰类

这样修饰的方法中，不能使用非static修饰的成员变量和方法。

1. 静态块

class Book{

public Book(String msg) {

System.out.println(msg);

}

}

public class Person {

Book book1 = new Book("book1成员变量初始化");

static Book book2;

static {

book2 = new Book("static成员book2成员变量初始化");

book4 = new Book("static成员book4成员变量初始化");

}

public Person(String msg) {

System.out.println(msg);

}

Book book3 = new Book("book3成员变量初始化");

static Book book4;

public static void funStatic() {

System.out.println("static修饰的funStatic方法");

}

public static void main(String[] args) {

Person.funStatic();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Person p1 = new Person("p1初始化");

}

/\*\*Output

\* static成员book2成员变量初始化

\* static成员book4成员变量初始化

\* static修饰的funStatic方法

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* book1成员变量初始化

\* book3成员变量初始化

\* p1初始化

\*///~

}

在初始化static修饰的成员时，可以将他们统一放在一个以static开始，用花括号包裹起来的块状态语句中。

1. 静态导入包

Eg：

Import static com.study.PrintHelper.\*

这样就可以直接使用printhelper中的方法，而不需要“类.方法名”

16. break，continue，switch

1. 在for循环中

Break的作用是跳出循环，不再执行循环。

Continue的作用格式跳出本次循环，继续下一次循环。

1. 在switch中

Break的作用是跳出循环，继续执行switch后面的业务逻辑。

Continue的作用是跳出switch，不执行switch后面的业务逻辑。

1. Final

总的来说，final关键字指的是“这是不可变的”

在java中，用final关键字修饰的变量，只能进行一次赋值操作，并且在生存期内不可以改变它的值。更重要的是，final会告诉编译器，这个数据是不会修改的，那么编译器就可能会在编译时期就对该数据进行替换甚至执行计算，这样可以对我们的程序起到一点优化。不过在针对基本类型和引用类型时，final关键字的效果存在细微差别。

1. 修饰数据
2. 修饰方法参数

public class FinalTest {

/\* ... \*/

public void finalFunc(final int i, final Value value) {

// i = 5; 不能改变i的值

// v = new Value(); 不能改变v的值

value.v = 5; // 可以改变引用对象的值

}

}//这里的final规定参数的值不能被改变

1. 修饰方法

告诉其他程序员，这个方法不能被覆盖。

1. 修饰类

用final修饰的类是无法被继承的