北華航天工業學院
NORTH CHINA INSTITUTE OF AEROSPACE ENGINEERING

第2章 JAVA语言基础-I

本章目标

- Java程序的构成
- 标识符和保留字
- 常量、变量
- JAVA数据类型
- 运算符与表达式

Java程序的构成

逻辑构成

Java源程序逻辑构成分为两大部分：程序头包的引用和类的定义。

1. 程序头包的引用

主要是指引用JDK软件包自带的包，也可以是自己定义的类。引用之后程序体中就可以自由应用包中的类的方法和属性等。

2. 类的定义

Java源程序中可以有多个类的定义，但必须有一个主类，这个主类是Java程序运行的入口点。在应用程序中，主类为包含main方法的类；在Java源程序中，主类的名字同文件名一致。

类的定义又包括类头声明和类体定义。类体中包括属性声明和方法描述。下面来看一个例子，其中斜体表示的语句行为主类类头，主类类头下面从大括号“{”开始到“}”结束的部分称为主类类体。

【例2.1】下面是一个应用程序，也是一个Applet，既可以在命令行下运行，也可以嵌入到HTML网页中用appletviewer命令运行。运行时在界面上的第一个文本框中输入你的名字，按回车键后，在第二个文本框中会显示“XXX，欢迎你来到Java世界！”，运行结果如图2.1所示。

```
//程序文件名称为WelcomeApplet.java          注释语句

import java.applet.*;
import java.awt.*;                          引入包
import java.awt.event.*;

public class WelcomeApplet extends Applet implements ActionListener  主类类
{
```

```
Label lblName;
TextField txtName;
TextField txtDisp;
```

属 性

```
public void init()
{
    lblName = new Label("请输入您的名字");
     txtName = new TextField(8);
     txtDisp = new TextField(20);
    add(lblName);
    add(txtName);
    add(txtDisp);
     txtName.addActionListener(this);
}
```

init方法

```
public void actionPerformed(ActionEvent e)
{
    txtDisp.setText(txtName.getText() + "欢迎你来到Java世界");
}
```

actionPerformed方法

```
public static void main(String args[])
{
        Frame f = new Frame("欢迎");
        f.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent evt)
       {
              System.exit(0);
        }
        }
       ;
       WelcomeApplet a = new WelcomeApplet();
       a.init();
       f.add("Center", a);
       f.setSize(400,300);
       f.show();
       a.start();
}
```

main主方法

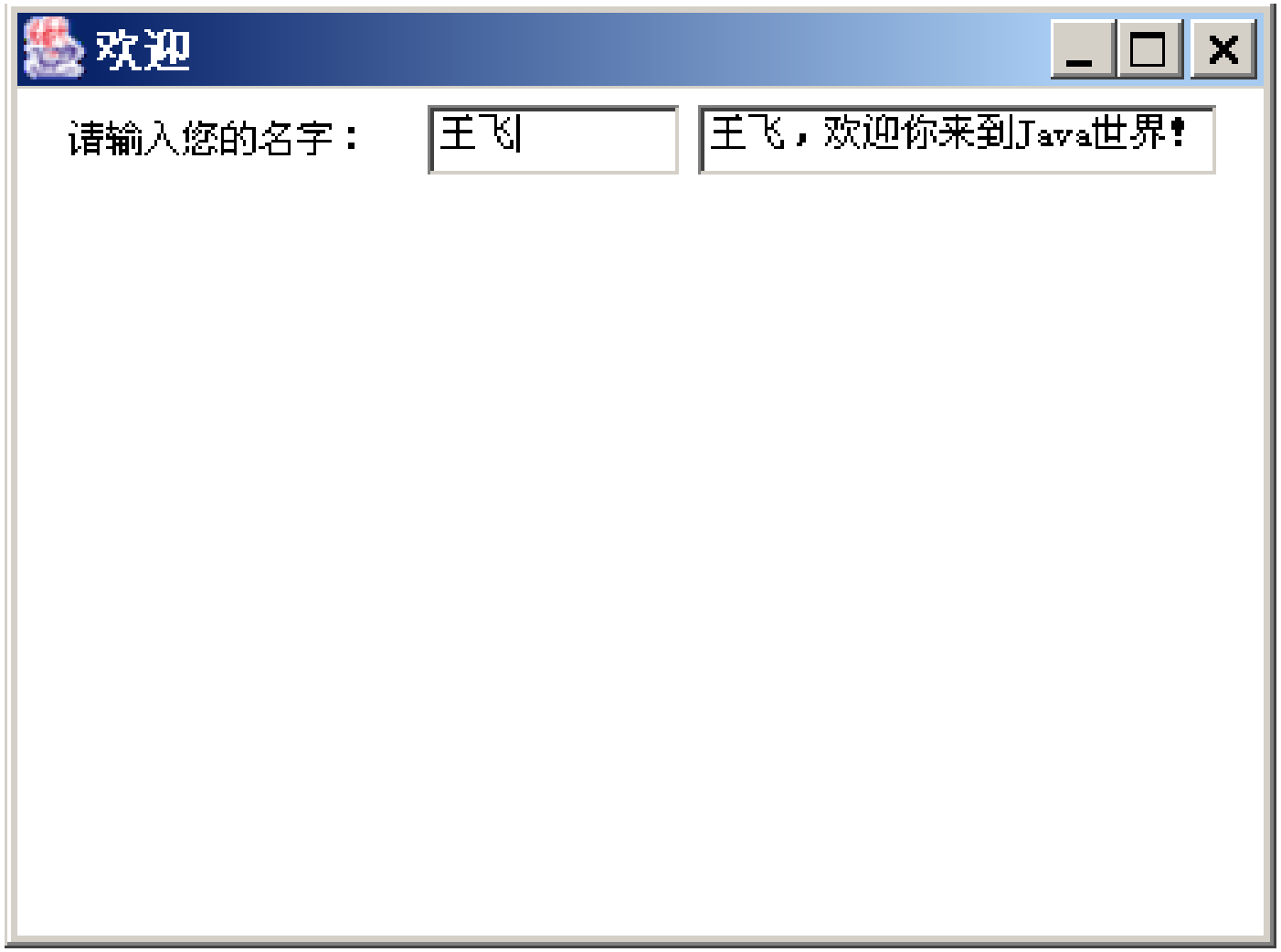


图2.1 程序界面

物理构成

Java源程序物理上由三部分构成，分别为语句、块和空白。

(1) 语句指一行以分号“;”结束的语句。

(2) 块指用括号对{}界定的语句序列，块可以嵌套使用。

(3) 空白指语句之间、块内部或者块之间的空白行。空白不影响Java源程序的编译和运行，适当地运用空白，可以形成良好的代码风格。

标识符

◆标识符

程序员对程序中的各个元素加以命名时使用的命名记号称为标识符（identifier），使用标识符来给类、变量、方法等命名

◆命名规则：

- 第一个字符必须是字母、下划线_、美元符$，中间也不能有空格；后面可以跟字母、下划线、美元符、数字。
- 标识符不能与关键字（keyword）重复
- 标识符大小写敏感，长度无限制

合法标识符	不合法标识符
userName	false
$change	room#
_endline	2mail
$persons	54.5

保留字

abstract	break	byte	boolean	catch
case	class	char	continue	default
double	do	else	extends	false
final	float	for	finally	if
import	implements	int	interface	instanceof
long	native	null	package	private
const	new	protected	public	return
switch	synchronized	short	static	super
try	true	this	throw	throws
volatile	transient	void	while	goto
strictfp				

注释

- 对于Java注释我们主要了解三种：
- 1,　// 注释一行
- 2,　/* ..... */ 注释若干行
- 3,　/**……*/文档注释
- 通过注释提高Java源程序代码的可读性；使得Java程序条理清晰，易于区分代码行与注释行。

注释

- /** ..... */ 注释若干行，并写入 javadoc文档通常这种注释的多行写法如下：
- /**
- * .....
- * .....
- */

- 注：注释内容即不会被编译的内容，只是解释说明

- 类（模块）注释：

- 类（模块）注释采用 /** …… */，在每个类（模块）的头部要有必要的注释信息，包括：工程名；类（模块）编号；命名空间；类可以运行的JDK版本；版本号；作者；创建时间；类（模块）功能描述（如功能、主要算法、内部各部分之间的关系、该类与其类的关系等，必要时还要有一些如特别的软硬件要求等说明）；主要函数或过程清单及本类（模块）历史修改记录等。

```
/**
* 文 件 名 :
* CopyRright
* 文件编号:
* 创 建 人:
* 日 期:
* 修 改 人:
* 日 期:
* 描 述:
* 版 本 号:
*/
```

Java常量

- JAVA的常量值用字符串表示，区分为不同的数据类型
  - 如整型常量23
  - 实型常量3.13
  - 字符常量‘a’
  - 逻辑常量true、false
  - 字符串常量“helloworld”
- “常量”还表示值不可变的变量
  - 如final关键字

```
final typeSpecifier
                    varname=value[,varname[=value]….]
```

如：final int NUM=100;

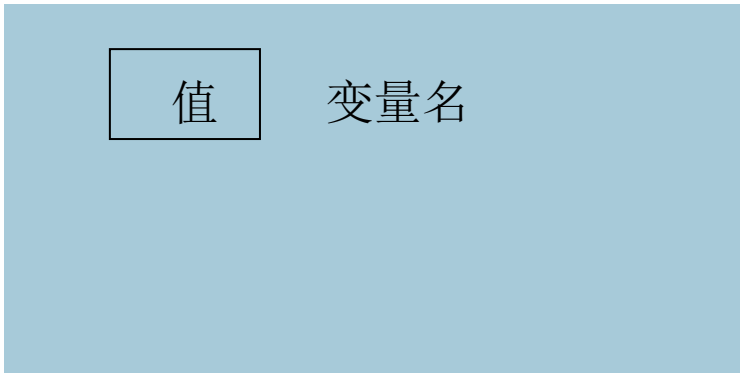
Java变量

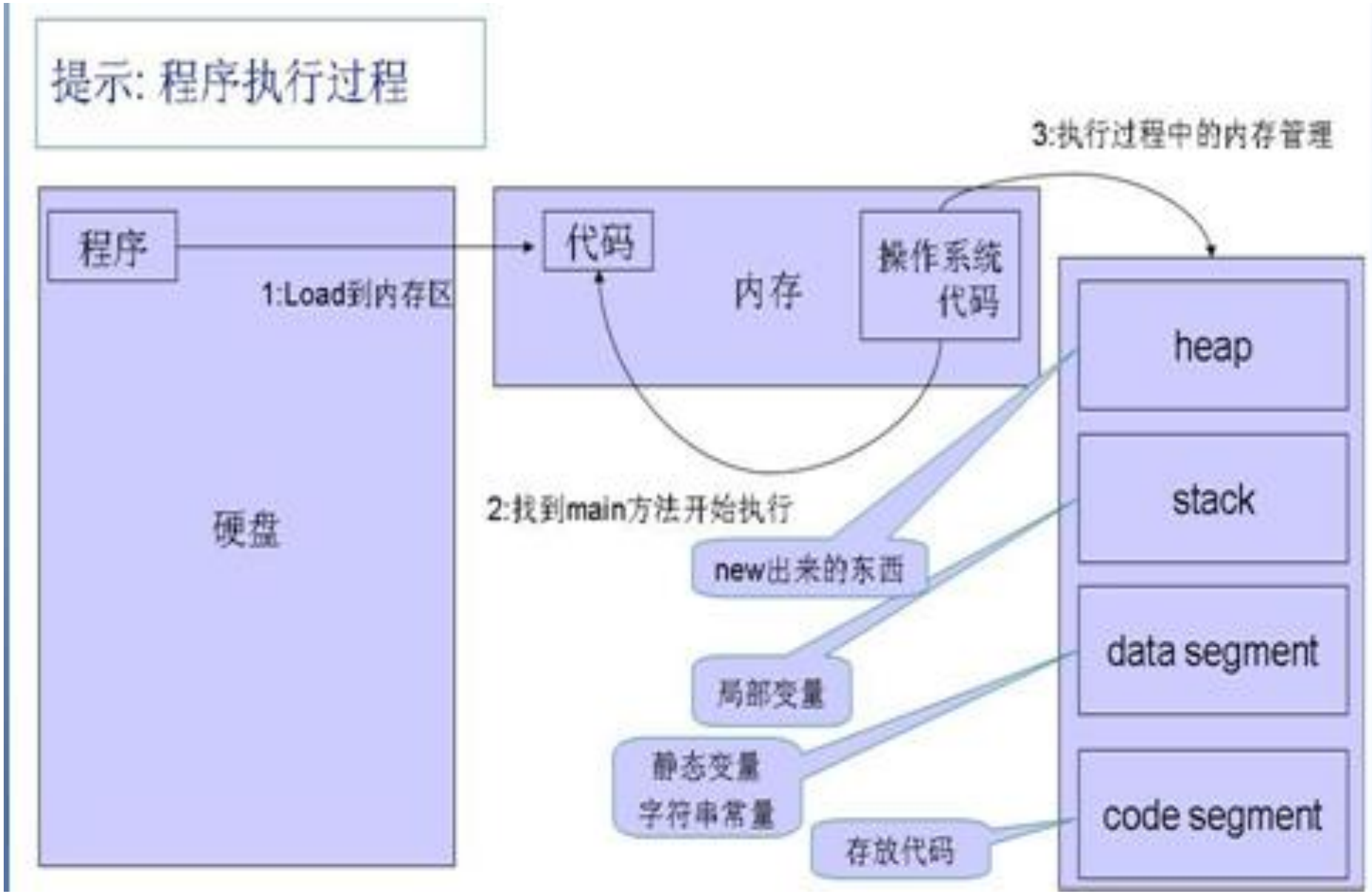
- 变量：基本的存储单元，包括变量名、变量类型和作用域
- 格式：

```
type identifier[ = value][,varname[=value]….];
```

如：int count;　　char c= ‘a’ ;

- 每个变量使用前，必须先声明，然后赋值，才能使用




提示: 程序执行过程
3:执行过程中的内存管理
程序
1:Load到内存区
硬盘
代码
内存
操作系统
代码
2:找到main方法开始执行
new出来的东西
局部变量
静态变量
字符串常量
存放代码
heap
stack
data segment
code segment

Java变量

- **变量作用域：指明可访问该变量的一段代码，声明一个变量的同时也就指明了变量的作用域。**
- **按作用域来分：**
  - 局部变量

    局部变量在方法或方法的一个块代码中声明，它的作用域为所在的代码块（整个方法或方法中的某块代码）。
  - 成员变量

    方法外部、类的内部定义的变量
- **按所属的数据类型划分为：**
  - 基本数据类型变量
  - 引用数据类型变量

Java的局部变量和成员变量

- **方法体内部声明的变量称为局部变量**
  - 方法体内部是指与方法对应的大括号内部

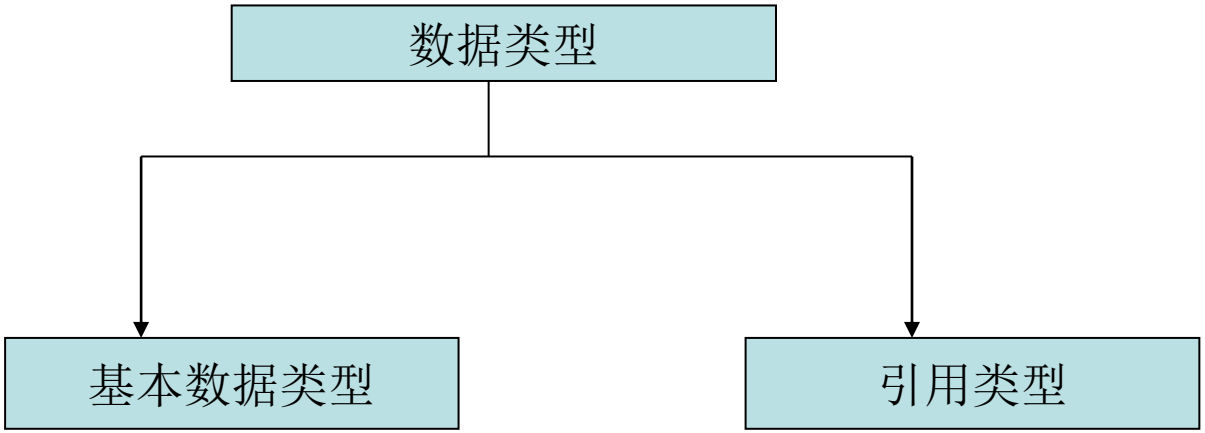
```
public static int add(int num1,int num2){
int sum=num1+num2;
return sum; //return语句返回值
}
```

- **在方法体外，类体内声明的变量为成员变量**

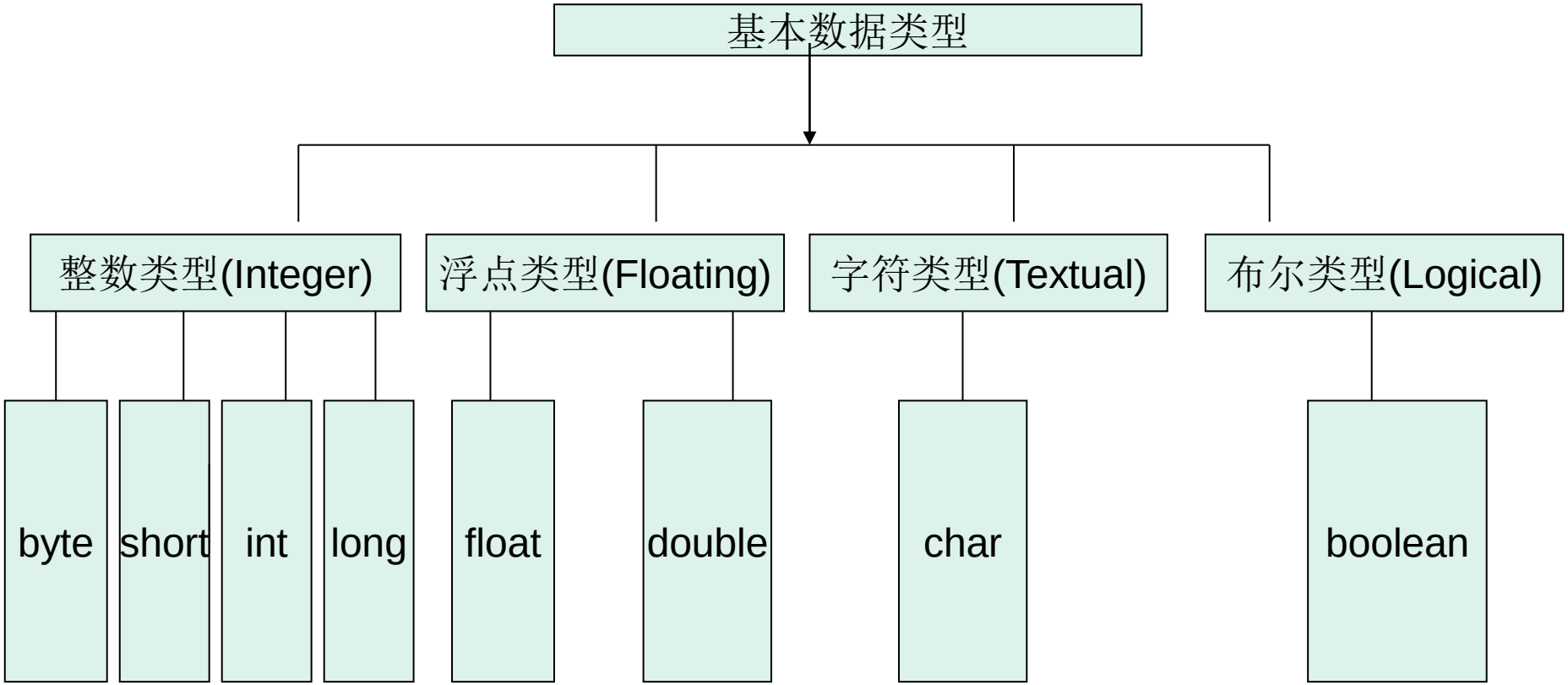
```
public class Student{
    private int stu_name;
}
```

Java数据类型

◆数据类型划分

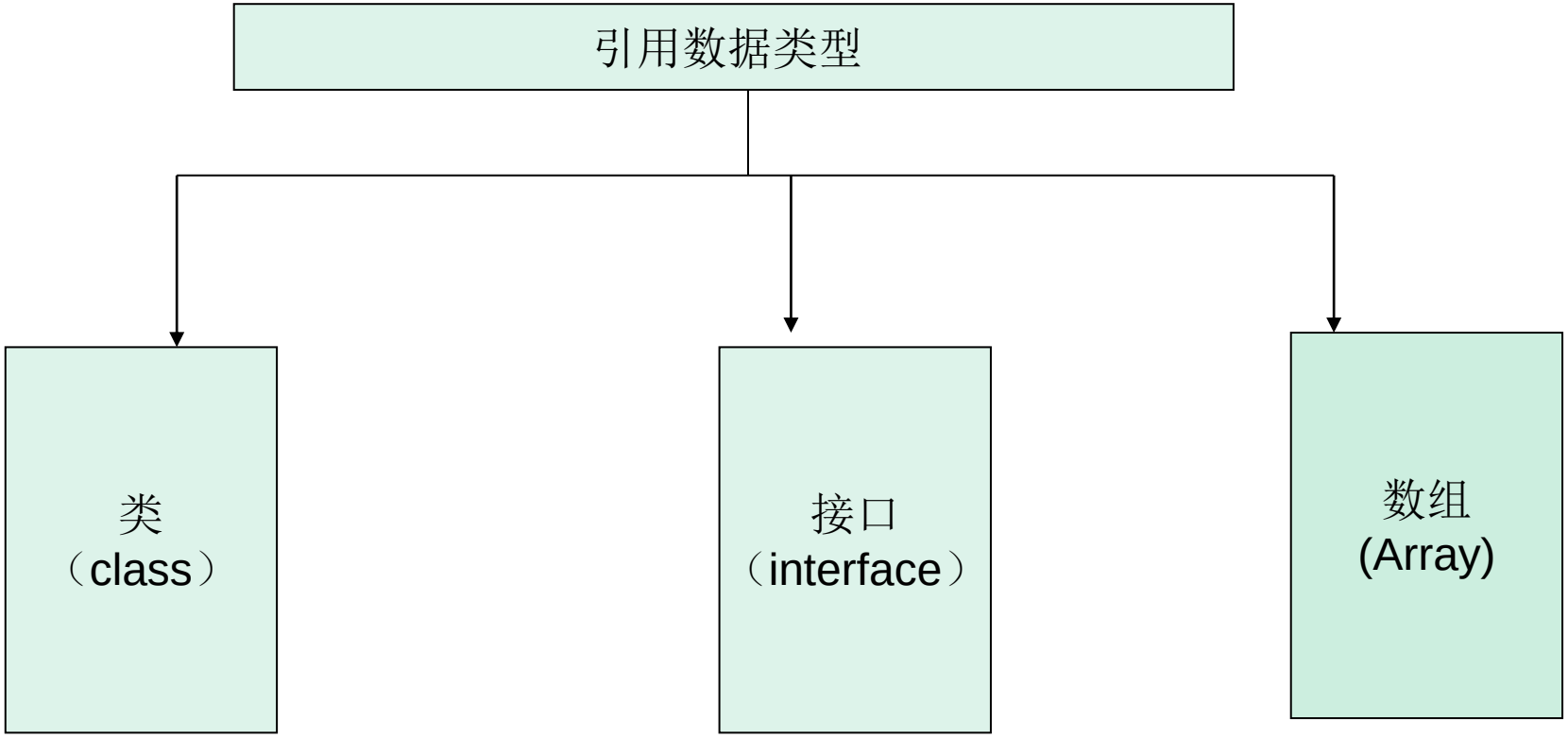


Java数据类型



4类8种基本数据类型

Java数据类型



基本数据类型-布尔类型

- 布尔类型：boolean
  - 取值只有true、false
  - 1和0不能用作boolean值
  - boolean值只能做布尔运算
- 例如：

```
boolean flag=true;
 if(flag){
     System.out.println(“flag is true”);
  }
```

基本数据类型-字符类型

- 字符类型：char
- 字符类型分类：字符常量和字符变量
  - 字符常量：用单引号括起来的一个字符，如'a'，'A'
  - 字符变量：在机器中占16位 ，其范围为0～65535

    如： char c='a'; /*指定变量c为char型，且赋初值为'a'*/
- Java字符采用Unicode编码(Unicode是全球语言统一编码)，每个字符占两个字节。

  例如：

  char c1=‘\u0061’;

基本数据类型-字符类型

- 转义符是指一些有特殊含义的、很难用一般方式表达的字符，如回车、换行等。所有的转义符以反斜线(\)开头，后面跟着一个字符来表示某个特定的转义符，如下表所示。

基本数据类型-字符类型

- 使用转移字符’\’来将其后的字符转变为其它含义

转义序列	描述
\b	Backspace后退一格
\t	Tab制表符
\n	Linefeed换行
\f	Formfeed换页
\r	Carriage Return返回

基本数据类型-整型类型

- 各整数类型有固定的表数范围和字段长度，其不受具体操作系统影响，以保证java程序的可移植性
- 整型常量表示形式：
  - 十进制整数：如123，－456，0
  - 八进制整数：以0(零)开头，如0123表示十进制数83
  - 十六进制整数：以0x(零x)或0X开头，如0x123表示十进制数291，－0X12表示十进制数－18。

数据类型	所占位数	数的范围
byte （字节）	8 （1B）	$-2^7 \sim 2^7-1$
short（短整型）	16 （2B）	$-2^{15} \sim 2^{15}-1$
int（整型）	32 （4B）	$-2^{31} \sim 2^{31}-1$
long（长整型）	64 （8B）	$-2^{63} \sim 2^{63}-1$

基本数据类型-浮点型类型

- **浮点型（实型）：float、double**
- 实型常量表示形式：
  - 十进制数形式

    由数字和小数点组成，如0.123，1.23，123.0
  - 科学计数法形式

    如：123e3或123E3，其中e或E之前必须有数字，且e或E后面的指数必须为整数。
  - float型的值,必须在数后字加f或F,如1.23f。

基本数据类型-浮点型类型

◆实型变量：

数据类型	所占位数	数的范围
float（单精度实型）	32(4B)	含字节数为4，数值范围为-3.4E38~3.4E38（7位精度）
double（双精度实型）	64（8B)	含字节数为8，数值范围-1.7E308~1.7E308（14位精度）

注意：当精度比较高，数值比较大时用double,而且double在CPU上处理速度更快，所以sin()等函数都返回的是double型。当声明浮点型的变量时，则默认为double型。所以变量都用double型

一、判断对错

```
//错误，字面量自动扩展为double
float f=900.09;
//正确
float f=900.09f;
//正确，为什么？
float f1=5;
```

基本数据类型-类型转换

◆基本数据类型中各类型数据间的优先关系和相互转换

- 不同类型数据间的优先关系如下：

低----------->高

byte,short,char-> int -> long -> float -> double

- 自动类型转换规则

整型、实型、字符型数据可以混合运算。运算中，不同类型的数据先转化为同一类型，然后进行运算，转换从低级到高级。

注意： byte,short,char之间不会互相转换，在计算时三者都先转换为INT类型。实数常量默认为double,整数常量默认为int.

操作数1类型	操作数2类型	转换后的类型
byte, short , char	int	int
byte, short, char , int	long	long
byte, short , char , int , long	float	float
byte , short , char , int, long, float	double	double

- 注意：
- 1、只要整型表达式中包含byte/short/int和字面量值如2，都被升级为int类型。
- 2、高级数据要转换成低级数据，需用到强制类型转换。
-

运算符

- **运算符**：表示各种不同运算的符号称为运算符
- **按操作数的数目分：**
  - 一元运算符：＋＋ －－
  - 二元运算符：＋ － ＞ ＜ ＝
  - 三元运算符：？ ：
- **按功能划分：**
  - 算数运算符：+, −, *, /, %, ++, −−
  - 位运算符：&, |, ^, ~, >>, <<, >>>
  - 逻辑运算符：!, ^, &&, ||
  - 关系运算符：>, <, >=, <=, ==, !=
  - 赋值运算符：=
  - 扩展赋值运算符：+=, −=, *=, =
  - 字符串连接运算符:+

算数运算符

◆自加(++)、自减(--)运算符

```
int i=0;
int j= i++ + ++i;
int k= --i + i--;
System.out.println(j+"  "+k);
```

输出：j=2 ,k=2

◆注意：

- ++(--)在前时先运算再取值
- ++(--)在后时先取值再运算

逻辑运算符

- **逻辑运算符：**
  - !-逻辑非，&-逻辑与，^-逻辑异或，|-逻辑或
  - &&-短路与，||-短路或
- **运算规则：**

a	b	!a	a&&b	a||b	a|b	a&b	a^b
true	true	false	true	true	true	true	false
true	false	false	false	true	true	false	true
false	true	true	false	true	true	false	true
false	false	true	false	false	false	false	false

赋值运算符及扩展赋值运算符

- 赋值运算符 =
- 扩展赋值运算符： +=，—=，*=，/=

运算符	示例	等效表达式
+=	i += 8	i = i + 8
-=	f -= 8.0	f = f - 8.0
*=	i*= 8	i = i * 8
/=	i /= 8	i = i / 8
%=	i %= 8	i = i % 8

例如：

```
String s = "Hello";
int i = 10;
String s1 = "hh";
s += I;     s += s1;
System.out.println(s);
```

字符串连接运算符

- 字符串连接运算符： +
- “+”除用于算数加法运算外，还可用于对字符串进行连接操作

```
int id=90+9;  String s=“hello”+”world”;
```

- “+”运算符两侧的操作数中只要有一个是字符串类型，系统会自动将另一个操作数转换为字符串然后再进行连接。

```
int c=12;
System.out.println(“c=”+c);
```

条件运算符

- **条件运算符：　？：**
- **例如：**

```
result=(sum= =0 ? 1 : num/sum);
```

解析过程：如果sum==0为true，则返回-1，否则返回num/sum。

等价于

```
if(sum= =0 ){
    result=1;
 } else{
    result= num/sum;
 }
```

表达式

- 表达式：由操作数、运算符和方法调用按一定的语法形式组成的符号序列。
- 表达式类型：由运算以及参与运算的操作数的类型决定，可以是简单类型，也可以是复合类型。
- 例如：

布尔型表达式：　x&&y||z;

整型表达式：　num1+num2;

表达式

优先次序	运算符
7	> < >= <=
8	= = !=
9	&
10	^
11	|
12	&&
13	||
14	?:
15	= += -= *= /= %= ^=
16	&= |= <<= >>= >>>=

总结

- 标识符和关键字
- 常量、变量
- java数据类型
  - 基本数据类型:

    char, byte, short, int, long, boolean, float, double
  - 复杂数据类型

    类，接口，数组
- 运算符及表达式

谢谢！