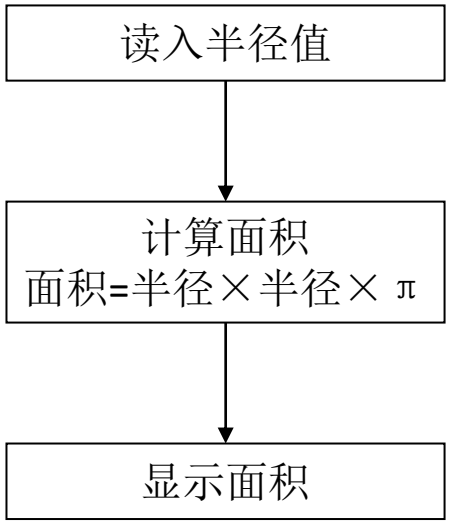
第2章 基本程序设计

目 录

contents

2.1 编写简单的程序

◆编写程序，给定半径，计算圆的面积




```
package  testProgram;  //java的类都在包中
//默认会import java.lang.System类
public class ComputeArea {
        public static void main (String[] args) {
                //变量定义
                double radius;
                double area;
                //指定半径
                radius = 10.0;
                //计算面积
                area = radius * radius * 3.14159;
                //显示结果
                System.out.println("The area for the circle of radius " +
                                radius + " is " + area);
        }
}
```

2.2 从控制台读取输入

◆标准输入/输出流

- System.out：标准输出流类OutputStream的对象
- System.in：标准输入流类InputStream的对象

◆Scanner类（java.util.Scanner）

```
Scanner scanner = new Scanner(System.in);
//构造函数Scanner的参数类型也可为java.io.File
//这是Scanner就从文件而不是标准输入流读取数据
double d = scanner.nextDouble( );
```

方法：

nextByte( )、nextShort( )、nextInt( )
nextLong( )、nextFloat( )、nextDouble( )
next( ) 读入一个字符串

2.2 从控制台读取输入

```
package  test;  //类必须在一个包中
import java.util.Scanner;
public class TestScanner {
        public static void main(String[ ] args) {
                // Create a scanner
                Scanner scanner = new Scanner(System.in);

                // Prompt the user to enter an integer
                System.out.print("Enter an integer: ");
                int intValue = scanner.nextInt();
                System.out.println("You entered the integer: " + intValue);

                // Prompt the user to enter a double value
                System.out.print("Enter a double value: ");
                double doubleValue = scanner.nextDouble();
                System.out.println("You entered the double value: " + doubleValue);

                System.out.print("Enter a string without space: ");
                String string = scanner.next();
                System.out.println("You entered the string: " + string);
        }
}
```

注意如果输入的不是一个合法的整数，该语句会抛出异常

2.3 标识符，变量，常量

标识符

◆Java中使用标识符(identifier)来命名变量、常量、方法、类、包等实体。

◆标识符命名规则

- 标识符是由字母、数字、下划线(_)、美元符号($)组成的字符序列。
- 标识符必须以字母、下划线(_)、美元符号($)开头。不能以数字开头。
- 标识符不能是保留字。
- 标识符不能为true、false或null等事实上的保留字（参见英文维基网）。
- 标识符可以为任意长度，但编译通常只接受前128字符。

例如：$2, area, radius, showMessageDialog是合法的标识符；2A, d+4是非法的标识符

2.3 标识符，变量，常量

Java保留字：

abstract	default	if	package	this
assert	do	goto	private	throw
boolean	double	implements	protected	throws
break	else	import	public	transient(非序列化)
byte	enum	instanceof	return	true
case	extends	int	short	try
catch	false	interface	static	void
char	final	long	strictfp(严格浮点)	volatile
class	finally	native(本地方法)	super	while
const	float	new	switch	
continue	for	null	synchronized	

Java保留字中：红色的为修饰符，可一个或多个用于修饰类型、变量、参数和返回值。例如public static 一起修饰一个方法

2.3 标识符，变量，常量

变量

◆变量(variable)用于保存数据输入、数据输出和中间值。可以向变量赋予类型匹配的值。

◆声明变量语法

```
datatype variableName;
或者
datatype v1, v2, ... , v3;
例如:
int x;
double radius, area;
```

2.3 标识符，变量，常量

常量

◆常量(constant)是一旦初始化后就不能再改变的数据。

◆语法

```
final datatype CONSTANT_NAME = value;
//注意常量的声明和初始化必须同时完成
final double PI = 3.14159;
```

◆使用常量的好处

避免重复输入

便于程序修改

便于程序阅读

2.3 标识符，变量，常量

常量

```
public class ComputeAreaConst
{
    public static void main(String[] args) {
        double radius;
        double area;
        final double PI = 3.14159; // 常量

        // 指定半径
        radius = 20;

        // 计算面积
        area = radius * radius * PI;

        // 显示结果
        System.out.println("The circle for the circle of radius " + radius + " is " + area);
    }
}
```

2.4 赋值语句和赋值表达式

赋值语句

◆语法

variable = expression;

其中expression是包含字面量（literal）、变量、方法调用和操作符的表达式。赋值语句的结果是将表达式的值赋值给左边的变量。

例如：

```
x = 1;  //右边表达式是整数字面量
x = 5 * (3 / 2) + 3 * 2; //右边表达式是整数字面量+操作符组成的算术表达式
x = y + 1; //右边表达式是整数字面量+变量+操作符组成的算术表达式
area = radius * radius * 3.14159; //右边表达式是变量+操作符组成的算术表达式
y = Math.random() ;  //右边表达式是方法调用组成的表达式
```

2.4 赋值语句和赋值表达式

赋值表达式

◆语法

variable = expression

赋值表达式的结果等于表达式的值。赋值表达式是右结合的。

例如：

```
i = j = k = 1;
```

等价于 //不要认为i, j, k的值不变，volatile类型的变量值可变

```
k = 1;
j = k;
i = j;
```

2.4 赋值语句和赋值表达式

同时完成变量声明和初始化

◆语法

datatype variable = expression;

例如:

int x = 1; //某些变量在申明时必须同时初始化：final int m=0;

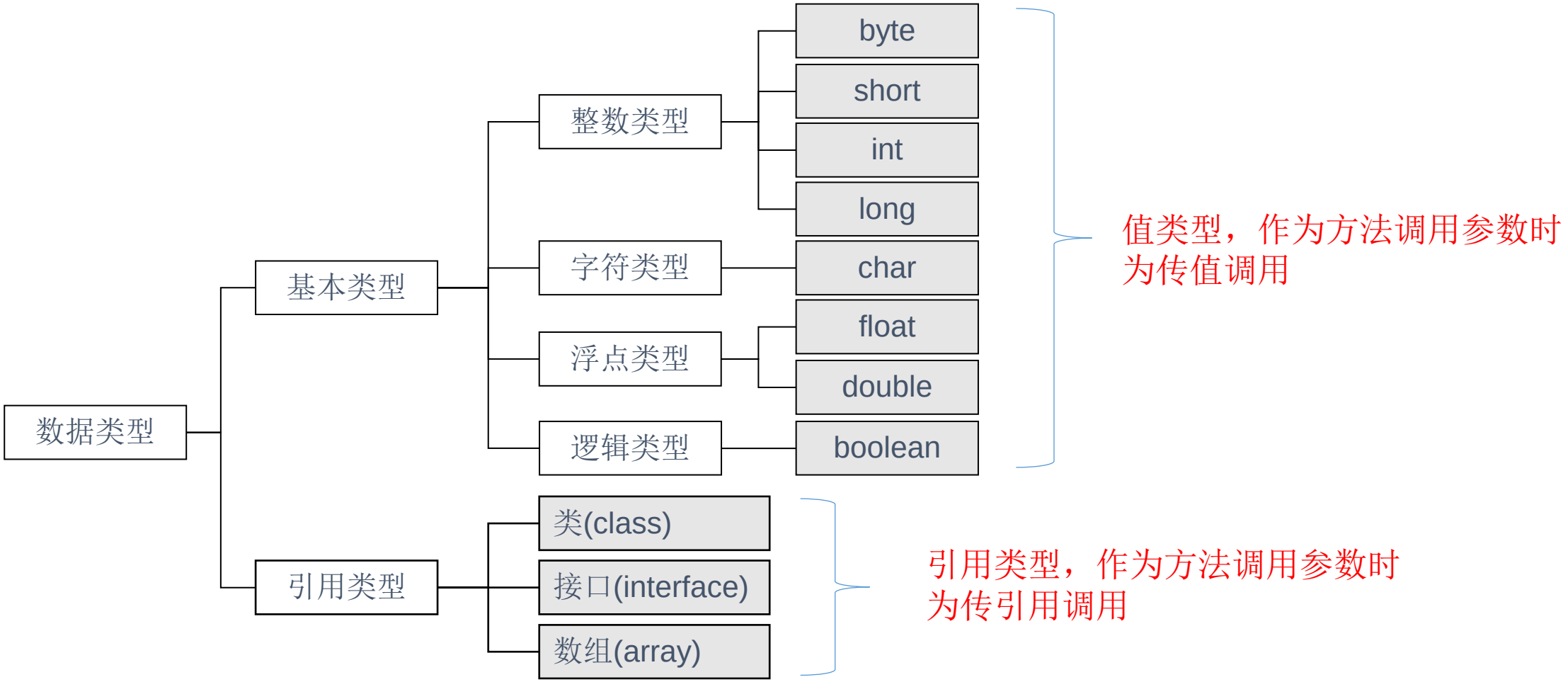
int x = 1, y = 2;

◆局部变量在使用前必须赋值。

int x, y; //若是类的成员变量，x, y有默认值=0

y = x + 1; //局部变量无默认值则错error

2.5 JAVA数据类型



传值调用(Call by value)和引用调用(Call by reference)的性质和C++完全一样。由于传值调用和传引用调用的性质不一样，因此在Java里区分值变量和引用变量非常重要

2.5 JAVA数据类型

- 为什么Java采用这种规定的方式来区分值变量和引用变量?
  - 因为Java没有专门的语法来区别值变量和引用变量
  - 在C++，通过语法来区分(符号&)值变量和引用变量
    - int i;　　//值变量
    - int &i;　　//引用变量
- 为了使得语法尽量简单，Java采用规定的方式来区别值变量和引用变量
  - 变量为值变量或引用变量取决于变量的声明类型Type

byte
short
int
long
char
float
double
boolean

Type　x;　x为值变量

类(class)
接口(interface)
数组(array)

Type　x;　x为引用变量

2.5 JAVA数据类型

- ◆Java引用和C++的引用一样，本质上都是指针
- ◆但是为了避免指针(p++,p- -这种操作）带来的风险，Java完全取消了指针
  - ◆通过引用变量可以让你访问对象，但不能对引用变量本身进行任何操作
  - ◆通过引用变量进行的操作实际上是被引用对象上的操作

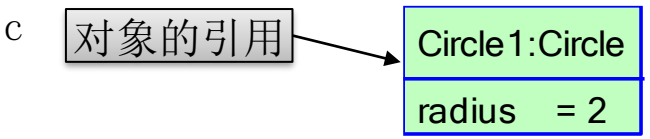
引用类型

```
Circle c;  //c=null
```

示意图里看出，c本身存贮的是对象的地址。

- ◆ 当c没有引用（指向）任何对象时，c的值为null；
- ◆ 当c指向对象时，c的值是对象的引用（地址）
- ◆ 在Java里我们不能对c本身的值进行任何操作（甚至不能读）

```
Circle c=new Circle(); //实例化对象，引用变量c指向这个对象
```



- ◆ 对引用变量c执行的操作c.findArea()实际上是调用被引用对象上的findArea()方法。

2.5 JAVA数据类型

数值数据类型

◆整数

- byte　　8位带符号整数(-128 到 127)
- short　　16位带符号整数(-32768 到 32767)
- int　　32位带符号整数(-2147483648 到 2147483647)
- long　　64位带符号整数(-9223372036854775808 到 9223372036854775807)

◆浮点数

- float　　32位浮点数(负数 -3.4×10^{38}到-1.4×10^{-45} 正数 1.4×10^{-45}到3.4×10^{38})
- double　64位浮点数(负数 -1.8×10^{308}到-4.9×10^{-324} 正数 4.9×10^{-324}到1.8×10^{308})

2.5 JAVA数据类型

数值运算符

◆加(+)、减(-)、乘(*)、除(/)、求余(%)：注意+，-的优先级较低

```
int a = 34 + 1;                 // 35
double b = 34.0 – 0.1;          // 33.9
long c = 300 * 30;              // 9000
double d = 1.0 / 2.0;           // 0.5: 此处为浮点除
int e = 1 / 2;                  // 0: 此处为整除
byte f = 20 % 3;                // 2: 取余数
```

◆整数相除的结果还是整数，省略小数部分。

```
int i = 5 / 2                   // 2
int j = -5 / 2                  // -2
```

2.5 JAVA数据类型

数值字面值（literal）

◆字面值是直接出现在程序中的常量值。

```
int i = 34;
long k = 100000L;
```

◆整数字面值

➢以0开头表示八进制，如035；以0x或0X开头表示十六进制，如0x1D,0X1d；以1-9开头表示十进制，如29

➢后缀字母：以l或L结尾表示long类型，如29L；无后缀表示int类型。

◆浮点数字面值

➢浮点数是包含小数点的十进制数，后跟可选的指数部分。如

18. 1.8e1 .18E2

➢后缀字母：以d或D结尾或者无后缀表示double类型；以f或F结尾表示float类型

2.5 JAVA数据类型

简洁操作符和递增递减操作符

◆常用简洁操作符，简洁操作符组成的表达式求值结果均为右值。

操作符	举例	等价于
+=	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

◆递增和递减运算符：++，--。结果均为右值。

- 前缀表示先加(减)1后使用
- 后缀表示先使用后加(减) 1

```
int i =10;              //i=++i + ++i; 结果为23
int newNum= 10 * i++;           //newNum = 100, i = 11
int newNum= 10 * ++i;           //newNum = 110, i = 11
```

2.5 JAVA数据类型

数值类型转换

◆如果二元操作符的两个操作数的数据类型不同，那么根据下面的规则对操作数进行转换：

➢数据转换总是向较大范围的数据类型转换，避免精度损失

```
byte i = 10;
long k = i * 3 + 4; //i变成int参与右边表达式计算，计算结果转long
double d = i * 3.1 + k / 2; //i转double， k/2转double
```

2.5 JAVA数据类型

赋值与强制类型转换

◆将值赋值给较大取值范围的变量时，自动进行类型转换。

byte < char < short < int < long < float < double

◆将值赋值给较小取值范围的变量时，必须使用强制类型转换(type casting)。语法：

(datatype)variableName

例如：

```
float f = (float)10.1;          // 10.1是double类型 ,double > float
int i = (int)f;                 // 10
int j = (int)-f;                // -10
double d = 10;          //自动进行类型转换：10是int型字面量，int < double
```

2.5 JAVA数据类型

注意

◆整数操作时，除数不能为0。

整数除0产生ArithmeticException异常。

◆浮点数操作上溢至Infinity（正无穷和下无穷）,下溢至0（数值绝对值太小而无法表示）。

浮点数除0等于Infinity(Java 预定义的符号）。

0.0除0.0等于NaN（Not a Number，Java预定义的符号）

2.5 JAVA数据类型

字符数据类型

◆char表示16位的单个Unicode字符（Java里字符都是Unicode编码）。

◆char类型的字面值

- 以两个单引号界定的单个Unicode字符。如:'男','女'
- 可以用\uxxxx形式表示， xxxx为十六进制。如:'\u7537', '\u5973'
- 转义字符表示：\n \t \b \r \f \\ \' \"

例如：

```
char letter = 'A';
char numChar = '4';
```

如果想打印带”“的信息 He said “Java is fun “

```
System.out.println(“He said \”Java is fun \”“ );
```

2.5 JAVA数据类型

字符串

◆String表示一个字符序列，注意字符串是String类（不是Java内置的数据类型）实现的，是引用类型

字符串的字面值是由双引号界定的零个或多个字符。

"Welcom to java!"　　""

◆连接运算：+，+=

➢加号用于连接两个字符串。如果二个操作数一个是字符串，另一个不是字符串，则先将非字符串操作数转换成字符串，再执行连接操作。

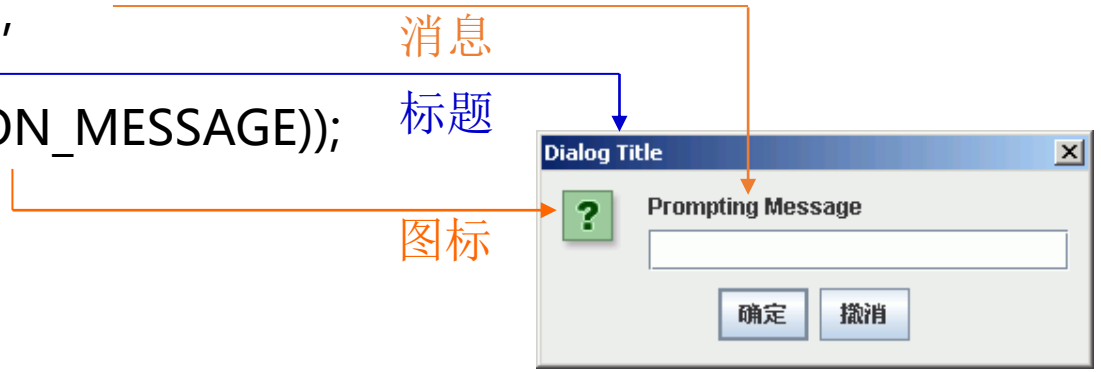
```
String message = "Welcome " + "to " + "java";  // Welcome to Java
String s =  “Chapter”  + 2; // Chapter2：2，自动转成字符串
//String s1 += "Supplement" + 'B';                    // SupplementB
String s1 = "Supplement";                    // SupplementB
s1 += 'B';                    // SupplementB
message += " and Java is fun";                // Welcome to Java and Java is fun
int i = 1;
int j = 2;
System.out.println("i + j = "  + i + j);                // i+j=12
System.out.println("i + j = "  + (i + j));                // i+j = 3
```

2.5 JAVA数据类型

从输入对话框获得输入

◆获取输入字符串：import javax.swing.JOptionPane

```
String string = JOptionPane.showInputDialog(
    null,  //Parent参数
    "Prompting Message",
    "Dialog Title",
    JOptionPane.QUESTION_MESSAGE));
```




```
String string = JOptionPane.showInputDialog("Prompting Message");
```

◆字符串转换成数字类型

```
int  i = Integer.parseInt(string);
double d = Double.parseDouble(string);
```

2.6 编程风格和Java常见错误类型

编程风格

- ◆良好的编程风格有利于减少错误，产生容易阅读、易于理解的代码。
- ◆注释
  - ➢类和方法前使用文档注释
  - ➢方法步骤前使用行注释。
- ◆命名：
  - ➢类名的每个单词的首字母大写：使用UpperCamelCase(驼峰式命名法)
    - ➢StudentInfomation
  - ➢变量和方法名使用小写，如果有多个单词，第一个单词首字母小写，其它单词首字母大写。
  - ➢常量使用大写，单词间以下划线分隔。
- ◆缩进、空格、块样式（在eclipse中使用ctrl+shift+f)

2.6 编程风格和Java常见错误类型

编程风格

- 有用的资源
  - CheckStyle：开源的代码风格检查工具， http://checkstyle.sourceforge.net/checks.html
  - Google官方代码规范， https://google.github.io/styleguide/javaguide.html
  - Sun官方代码规范， http://java.sun.com/docs/books/jls/second_edition/html/index.html
  - IDEA Checkstyle插件主页，http://plugins.jetbrains.com/plugin/1065-checkstyle-idea

www.apache.org:最大的Java开源社区网站

2.6 编程风格和Java常见错误类型

Java常见错误类型

◆语法错误(syntax error)

在编译期间产生的错误，编译报错。

◆运行时错误(runtime error)

导致程序非正常终止的错误，编译不会报错。

◆逻辑错误(logic error)

程序不能按预期的方式执行，编译不会报错。