January 8, 2002

Chapter 1

Java Basics Quick Reference

Java Basics Quick Reference

Compilation & Interpretation of Application

`> javac Hello.java`	Compile `Hello.java`, produce `Hellow.class`
`> java Hello`	Interpret (run) `Hellow.class`

Application Program Structure

`public class Hello`	Class definition, in file `Hello.java`
`{`	Beginning brace; must have mate
`   public int a = 2;`	Class variable a
`   public static void main(String[] args)`	`main` method
`  {`	Opening brace for `main`
`     double b = 3., c;`	Variables local to `main`
`     int three;`	Integer variable
`     final double HBARC = 197.32;`	Unchangeable constant
`     three = add(1,a);`	Call to method
`     c = Math.sin(b);`	Call to math xlibe
`     System.out.println("sin(b)= "+ c);`	Print to screen
`  }`	Closing brace for `main`
`// ------------end main, begin add-------------------------`	Comment
`   public static int add(int x, int y)`	Stupid method to add to int's
`     {return x + y;}`	Returns value of "add"
`}`	Ending brace for class
`/* Comment field (multiple lines OK) */`	Comment in field
`// One line comment`	One line comment
`/** Documentation comment **/`	Documenting comment
`public static double f(double x)`	Method (function) $f(x)$
`{return x*x ;}`	Multiple lines within brace OK too

Data (Variable) Types

Description	Type	Size/Format
Integer	`byte, short, int, long`	1B (=8b), 2B, 4B, 8B
Floating Point	`float, double`	Single, Double precision (=8B)
Single Character	`char`	16-bit (2B)
Logical	`boolean`	1 bit, true or false

Sample Data Representations

Representation	Meaning	Representation	Meaning
`i = 10 L; i = 10 l;`	long integer	`i = 3.1e+8, 3.1E+08`	Scientific
`i = 10.;`	decimal (=double)	`i = 0xA;`	Hexidecimal
`i = 10.0 F;, 10.0 f;`	float (=single)	`i = 017;`	Octal

Naming Convention

variable, variableName; ClassName, Classname; CONSTANT.

Reserved Words

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

Arrays

`int [] i;`	Declare integer array
`double [] x = new double[10];`	Declare & create (allocate memory) array
`double [][] y = new double [8][9];`	Declare & create 2D array
`arrayname [i][j] = i*j;`	Assign value to array element
`int a[3] = {1, 2, 3};`	Assign values to array elements
`int size = arrayname.length;`	Extract length of array (any array)

Arithmetic Operators

Operator	Example	Description
+	x + y	Add x to y (also concatenates strings)
-	x - y	Subtract y from x
*	x * y	Multiply x by y
/	x / y	Divide x by y
%	x % y	Remainder from x/y; the modular op

Unary Operators

Operator	Example	Description
+	+x	Promotes x to int if it's a byte, short, or char
-	-x	Arithmetically negates x
( )	x = (double) 1	Cast (converts data types)

Shortcuts

Operator	Example	Description
`++`	`x++`	Use x, then set `x = x + 1`
`++`	`++x`	Set `x = x + 1`, use `x`
`--`	`x--`	Use `x`, then set `x = x -1`
`--`	`--x`	Set `x = x -1`, then use `x`

Relational Operators

Operator	Example	Return true if
`>`	`x > y`	`x` is greater than `y`
`>=`	`x >= y`	`x` is greater than or equal to `y`
`<`	`x < y`	`x` is less than `y`
`<=`	`x <= y`	`x` is less than or equal to `y`
`==`	`x == y`	`x` and `y` are same *object*
`!=`	`x != y`	`x` and `y` are not equal

Logical Operators

Operator	Example	Name: Return true if
`&&`	`x && y`	**Logical and:** `x` and `y` both true, conditionally evaluates `y`
`||`	`x || y`	**Logical or:** either `x` or `y` true, conditionally evaluates `y`
`!`	`!x`	**Not:** `x` is false
`&`	`x & y`	**And:** `x` and `y` both true, always evaluates `x` and `y`
`|`	`x | y`	**Or:** either `x` or `y` true, always evaluates `x` and `y`

Bitwise Operators

Operator	Example	Operation
`>>`	`x >> y`	Shift bits of `x` right by distance `y`
`<<`	`x << y`	Shift bits of `x` left by distance `y`
`>>>`	`x >>> y`	Shift bits of `x` right by distance `y` (unsigned)
`&`	`x & y`	Bitwise and
`|`	`x | y`	Bitwise or
`^`	`x ^ y`	Bitwise xor
`~`	`~y`	Bitwise complement

Compound Assignment Operators

Operator	Example	Equivalent to
`+=`	`x += y`	`x = x + y`
`-=`	`x -= y`	`x = x - y`
`*=`	`x *= y`	`x = x * y`
`/=`	`x /= y`	`x = x / y`
`%=`	`x %= y`	`x = x % y`
`&=`	`x &= y`	`x = x & y`
`|=`	`x |= y`	`x = x | y`
`^=`	`x ^= y`	`x = x^y`
`<<=`	`x <<= y`	`x = x << y`
`>>=`	`x >>= y`	`x = x >> y`
`>>>=`	`x >>>= y`	`x = x >>> y`

Order of Precedence

1. Binary ops	2. left to right	3. assignment ops	4. RHS then LHS
5. `x++`	6. `++x`	7. cast	8. `*`
9. `+`	10. `>>>`	11. `==`	12. `=`

Mathematical Function Library [Use: Math.sin(b)]

E	PI	sin	cos	tan	asin	acos	atan
atan2(y,x)	exp	log	pow(x,3.)	sqrt	random	abs	max
min	ceil	floor	rint				

Flow Control

`while (x <= 0.) {y = y * y;`
`              x = x + 2.;}`
	Evaluate as long as true;
	Second line to be evaluated, *etc.*

`if ( (x < 3) && (y == 12) ) z = y * x;`	Evaluate once if true

`if ( x <= 0.  ) { y = y * y; }`
`    else y = 2 * y;`
	Can have one or more lines in {}
	Only one `else` permitted (catchall)

`if ( score >= 90 ) { grade = 'A'; }`
`    else if ( score >= 80 ) { grade = 'B'; }`
`    else if ( score >= 70 ) { grade = 'C'; }`
	The “if” condition
	Any number of `else if`'s OK
	Inaccessible if earlier `else` satisfied

`switch (month) {case 1:  s = "Jan"; break ...`
`    case 12:  s = "Dec"; break;}`
	Fall through if no break
	Can have many cases

`for ( i = 0; i < 100; i++ )`
`    { <statements> }`
	(initial value; repeat for; increment)
	Multi-line code block

`do { <statements> } while ( <boolean> )`
`    <labelname>:  x = x*y; ...`
`    break <labelname>;`
`    if ( i==99 ) continue;`
	Goes through at least once
	Break send control back here
	Use `continue` within loops for new iteration
	Unlabelled continue, jump to loop end

Input and Output, Screen & Keyboard

`System.out.println ( "count = " + j );`	Screen output

`import java.io.*;`
`main(String[] argv) throws IOException,`
`   FileNotFoundException`
	Include input/output package
	Main method to handle exceptions;
	the exception

`BufferedReader b=new Buffered Reader`
`    (new InputStreamReader(System.in));`
`String s = b.readLine();`
`r = Double.valueOf(s).doubleValue();`
	Read via 3 filters;
	”
	Line read stored as string
	Convert string to double

Input and Output, Files

`import java.io.*;`
`main(String[] argv) throws IOException,`
`    FileNotFoundException`
	Need for all but screen & keyboard
	
	`main` with exception throwing

`BufferedReader b = new BufferedReader`
`    (new InputStreamReader`
`    (new FileInputStream("radius.dat")));`
	Open file with JDK 1.1;
	”
	”

`String s = b.readLine();`	Read 1 line, save as string

`double x = Double.parseDouble(s);`
`int i = Integer.parseInt(s);`
	Convert string to double
	Convert string to integer

`PrintWriter q = new PrintWriter`
`    (new FileOutputStream("area.dat", true);`
	Open output file with JDK 1.1;
	appends file

`PrintWriter q = new PrintWriter`
`    (new FileOutputStream("area.dat", false);`
	Open output file with JDK 1.1;
	overwites file

`q.println("radius = " + radius);`	Output word `radius` and its value
`q.close();`	Closes file