

1.00/1.001 - Session 2 Fall 2005

Basic Java Data Types, Control Structures

Java Data Types

- **8 primitive or built-in data types**
 - 4 integer types (byte, short, int, long)
 - 2 floating point types (float, double)
 - Boolean (boolean)
 - Character (char)
- **These are not objects**

Java Data Types

- These are defined (almost) identically on every machine on which Java runs, unlike other programming languages
- Java is a strongly typed language:
 - Every variable must have a declared type

Java Data Types

Type	Size (bits)	Range
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2,147,483,648 to 2,147,483,647
long	64	-9,223,372,036,854,775,808L to 9,223,372,036,854,775,807L
float	32	+/- 3.4E+38F (6-7 significant digits)
double	64	+/- 1.8E+308 (15 significant digits)
char	16	ISO Unicode character set
boolean	1	true or false

What data type would you use?

- What would you use to store:
 - Speed of light
 - Your grade in this course
 - Your grade point average this term
 - Number of refrigerators in a room
 - Location of a point on a screen
 - 2^{65}
 - \$234.77
 - Half of \$234.77
 - Bits per second transmitted by modem

What data type would you use?

- What would you use to store:

– Speed of light	double
– Your grade in this course	char
– Your grade point average this term	double/float
– Number of refrigerators in a room	int
– Location of a point on a screen	float/int
– 2^{65}	BigInteger
– \$234.77	double/int
– Half of \$234.77	double/int
– Bits per second transmitted by modem	int/float

Using Java Data Types

```

public class DataTypes {
    public static void main(String[] args) {
        boolean isReal=true;    // Names are case sensitive,
                                // start w/letter, have nos,_, $
        byte d= 122;            // Must be less than 127
        short e= -29000;        // Must be less than 32767
        int f= 100000;          // Must be less than 2.1 billion
        long g= 999999999999L;  // Must put L on end
        float h= 234.99F;       // Must be < 3E38; F on end
        double i= 55E100;
        char cvalue= '4';      // char '4' is not integer 4

        // Strings are objects, not primitives. Example:
        String name= "Cl aud i us";
    }
}

```

Arithmetic Operators

Table in precedence order, highest precedence at top

Operators	Meaning	Associativity
++	increment	Right to left
--	decrement	
+ (unary)	unary + (x = +a)	
- (unary)	unary - (x = -a)	
*	multiplication	Left to right
/	division	
%	modulo	
+	addition	Left to right
-	subtraction	

Using Arithmetic Operators

```
public class DataType2 {
    public static void main(String[] args) {
        int j, k, m;
        int d= 122;
        j= d++; // j is 122 and d is 123
        System.out.println("j= " + j);
        k= ++d; // k is 124 and d is 124
        System.out.println("k= " + k);
        m= --d; // m is 123 and d is 123
        System.out.println("m= " + m);
        m= k % j; // Remainder op for int types
                // k=124, j=122, so m= 2
        System.out.println("m= " + m);
        j= 5; k= 3; m= j/k; // Integer division: m= 1
        System.out.println("m= " + m);
        System.exit(0);
    }
}
```

Logical Operators

- Produce results of type boolean
- Comparisons use 8 operators:

Equal	==	Not equal	!=
Less than	<	Less than or equal	<=
Greater than	>	Greater than or equal	>=
Logical and	&&	Logical or	

Logical Operators

- **Example:**

```
double c= 0.0, b= 3.0;
if (c != 0.0 && b/c > 5.0) System.out.println("Boo");
// Never use == with float or double (this is bad
//   example)
// Short circuit evaluation: quit after false
//   subexpression
```

- **There are also “bitwise” operators; we won’t use these much (if at all)**

Assignment Operators

- **Assignment is not the same as equality!!!**

= is not the same as ==

- **Assignments are expressions:**

```
int x, y;
x = y = 5; // Same as x = (y= 5); assoc from R to L
```

- **Short cut forms exist:**

```
int x = 5, y = 3;
x += y; // Same as x= x + y;
```

- **Forms include +=, -=, *=, /=, &=, ^=, |=, %=**

Exercises

- **Get the percent grad students in 1.00:**

```
int students= 240;  
int grads= 35;  
_____;
```

- **Represent 15^i correctly:**

```
int i= 10000000 + 100000000;  
_____;
```

- **Write expression to test if int x greater than double y and x less than y^2 and x not equal x^2 :**

```
_____ // Decl are x, y  
if ( _____ // Wri te logical expressi on
```

- **Increment int z by int a:**

```
_____ // Decl are a, z  
_____ // Increment z by a
```

Exercises

- **Get the percent grad students in 1.00:**

```
int students= 240;  
int grads= 35;  
double pctGrad= grads/(double) students;
```

- **Represent 15^i correctly:**

```
int i= 10000000 + 100000000;  
long j= 15L*i;
```

- **Write expression to test if int x greater than double y and x less than y^2 and x not equal x^2 :**

```
int x; double y;  
if (x > y && x < y*y && x != x*x) ...
```

- **Increment int z by int a:**

```
int a=5, z=2;  
int z += a;
```

Control Structures: Branch

General form	Example
if (boolean) statement;	if (x == y) a = 20; if (x == z) { b = 10; c = 20; }
if (boolean) statement1; else statement2;	if (x == y) { a = 10; b = 20; } else x = y;
if (boolean1) statement1; ... else if (booleanN) statementN; else statement;	if (x > 60) y = 20; else if (x < 30) { z += y; y = 25; } else y = 40;

Control Structures: Branch

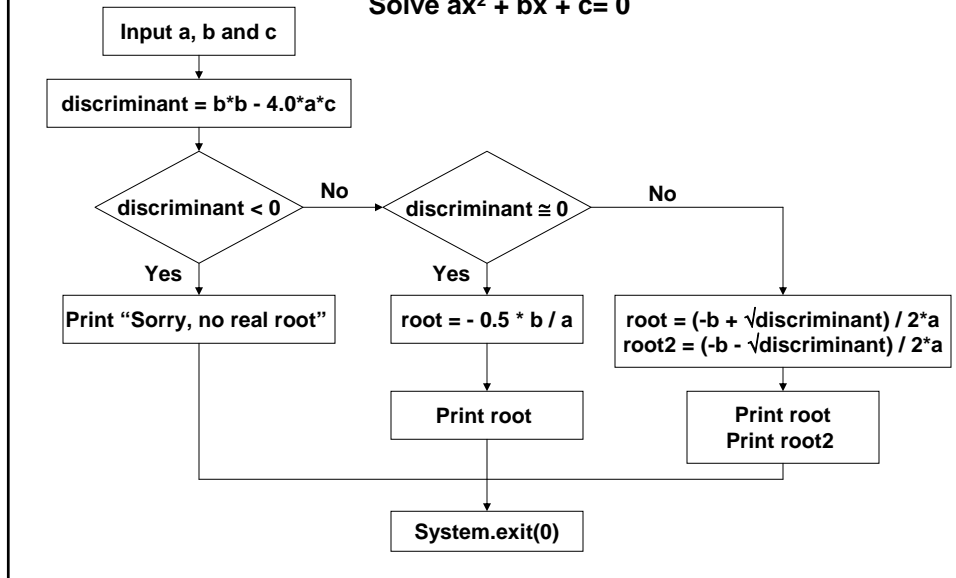
- A statement can be replaced by a block or set of statements enclosed in a pair of braces: { }
- An else clause belongs to its nearest if statement.
 - If you want to associate it with a farther if statement, put intervening if statements within { }

```
if (x > 3) {
    if (y < 7) z = 5; }
else // This else belongs to if (x > 3)
    z = y;
```
- Ternary comparison statement
 - (boolean ? expr_if_true : expr_if_false)

```
boolean isCelsius = true;
double tC = 10.0;
double displayTemp = (isCelsius ? tC : 1.8*tC + 32.0);
```

Control example

Solve $ax^2 + bx + c = 0$



Control example

```
import javax.swing.*; // To support simple input
public class Control { // Quadratic formula
    public static void main(String[] args) {
        final double TOL= 1E-15; // Constant(use 'final')
        String input;
        input= JOptionPane.showInputDialog("Enter a");
        double a= Double.parseDouble(input);
        input= JOptionPane.showInputDialog("Enter b");
        double b= Double.parseDouble(input);
        input= JOptionPane.showInputDialog("Enter c");
        double c= Double.parseDouble(input);
```

Control example

```
double discriminant= b*b - 4.0*a*c;
if ( discriminant < 0)
    System.out.println("Sorry, no real root");
else if (Math.abs(discriminant) <= TOL) {
    double root= -0.5 * b / a;
    System.out.println("Root is " + root); }
else { // Redefine 'root'; blocks have own scopes
    double root=(-b + Math.sqrt(discriminant))/
        (2.0*a);
    double root2=(-b- Math.sqrt(discriminant))/
        (2.0*a);
    System.out.println("Roots" + root + ", " +
        root2); }
System.exit(0); }
```

Control example

- **The previous program has a deliberate, subtle bug**
 - Can you see it?
 - Is it likely that you'd find it by testing?
 - Is it likely you'd find it by using the debugger and reading the code?
- **Fix the error by rearranging the order of the if-else clauses**

Control structure: Iteration

General form	Example
<pre>while (boolean) statement;</pre>	<pre>while (x > 0) { System.out.println("x= " + x); x--; }</pre>
<pre>do statement; while (boolean); // Always executes stmt at least once</pre>	<pre>do { System.out.println("x=" + x); x--; } while (x > 0)</pre>
<pre>for (start_expr; end_bool; cont_expr) statement;</pre>	<pre>for (x= 20; x > 0; x--) System.out.println("x=" + x);</pre>

for loops

<pre>for (start_expr; end_bool; cont_expr) statement;</pre>	<pre>for (j= 0; j < 20; j++) z += j;</pre>
<p>is equivalent to:</p>	
<pre>start_expr; while (end_bool) { statement; cont_expr; }</pre>	<pre>j= 0; while (j < 20) { z += j; j++; }</pre>

Example Method-Computing ln(x)

The natural logarithm of any number x
in the interval (0,2) is approximated
by the formula

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 \\ - (x-1)^4/4 + (x-1)^5/5 + \dots$$

Iteration Example 1: ln (x)

```
import javax.swing.*;

public class Iteration {
    public static void main(String[] args) {
        String input= JOptionPane.showInputDialog("Ent x (0-2)");
        double x= Double.parseDouble(input);
        // Compute 20 terms of
        // ln x= (x-1) - (x-1)^2/2 + (x-1)^3/3 - ...
        final int ITERATIONS= 20;          // Fixed no of iterations
        double logx= 0.0;
        double x1= x-1;
        for (int i= 1; i <= ITERATIONS; i++) {
            if (i % 2 == 0)                // i even
                logx -= Math.pow(x1, i)/i;
            else
                logx += Math.pow(x1, i)/i; }
        System.out.println("ln x= " + logx); } }
```

Iteration Example 2: Ln x

```
import javax.swing.*; // Same series as example 1
public class Iteration2 {
    public static void main(String[] args) {
        String input= JOptionPane.showInputDialog("Ent x (0-2)");
        double x= Double.parseDouble(input);
        final double TOLERANCE= 0.00001; // Tol sets no of terms
        double logx= 0.0;
        double x1= x-1;
        int i= 1;
        double term= 0.0; // Define outside do {}
        do {
            term= Math.pow(x1, i)/i;
            if (i % 2 == 0) // i even
                logx -= term;
            else
                logx += term;
            i++;
        } while (Math.abs(term) > TOLERANCE);
        System.out.println("Ln x= " + logx);
        System.out.println("Found in " + i + " iterations"); } }
```