

# Lecture Set 4: Evaluation Order

Today:

- More assignment operators
- Precedence and short-circuiting



CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

## Expressions

- Java “phrases” that yield values

e.g.

`x`

`x + 1 - y`

`x == y && z == 0`

`foo.equals ("cat")`

- Expressions have values (int, boolean, etc.)
- Expressions can be assigned to variables, appear inside other expressions, etc.



CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

1



## Expressions and Side Effects

- Some expressions can also alter the values of variables  
e.g. `x=1`
- `x=1` is an expression?
  - Yes!
  - Value is result of evaluation right-hand side of `=`
  - It also alters the value of `x`
- Such alterations are called **side effects**

CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

2



## Are the Following Legal?

- ```
int x, y;  
x = y = 1;
```

**Yes.** Result assigns 1 to `x` and to `y`
- ```
int x = 0, y = 1;  
boolean b = false;  
if (b = (x <= y)){  
    x = y;  
}
```

**Yes.** Result assigns `true` to `b` and 1 to `x`

CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

3

## Other Expressions with Side Effects



- Java includes abbreviations for common forms of assignment
- Example: **increment** operations (Basically equivalent to  $x = x + 1$ )
  - $++x$  "Pre-increment"
    - Increments  $x$ , returns the new value of  $x$
  - $x++$  "Post-increment"
    - Increments  $x$ , returns the old value of  $x$
- Same or Different
  - $x == x++$  always true
  - $x == ++x$  never true
- Compare
  - $x++ * y++$
  - $++x * ++y$
  - $++x * y++$
  - $x++ * ++y$

CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

4

## Other Assignment Operators



- Example: **decrement** operations (Basically equivalent to  $x = x - 1$ )
  - $--x$  "Pre-decrement"
    - Decrements  $x$ , returns the new value of  $x$
  - $x--$  "Post-decrement"
    - Decrements  $x$ , returns the old value of  $x$
- General modification by constant
  - General form: **<var> <op with=> <constant>**
  - Examples
 

|          |               |             |
|----------|---------------|-------------|
| $x += 2$ | equivalent to | $x = x + 2$ |
| $x -= 2$ | equivalent to | $x = x - 2$ |
| $x *= 2$ | equivalent to | $x = x * 2$ |
| $x /= 2$ | equivalent to | $x = x / 2$ |

CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

5



## Precedence

- Explains how to evaluate expressions
  - What is value of  $1 - 2 + 3 * 4$ ?
  - Precedence rules** answer this question
    - Higher-precedence operators evaluated first
    - Example from math: “Please, Excuse my Dear Aunt Sally” or PEMDAS
 

Multiple and divide (higher precedence) before you add and subtract (lower precedence)
- Java follows “Aunt Sally’s Rules” ... but what about other operators?

CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

6



## Java Precedence Rules

- parentheses: ( )
- unary ops:  $+x$   $-x$   $++x$   $--x$   $x++$   $x--$   $!x$
- multiply/divide:  $*$   $/$   $\%$
- add/subtract:  $+$   $-$
- comparisons:  $<$   $>$   $<=$   $>=$
- equality:  $==$   $!=$
- logical and:  $\&\&$
- logical or:  $||$
- assignments:  $=$   $+=$   $*=$   $/=$   $\%=$  (only these are right to left associative)

increasing precedence

CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

7

## Examples



- $x * y + -z$

Equivalent to  $(x * y) + (-z)$

- $(x \leq y \ \&\& \ y \leq z \ || \ w > z)$

Equivalent to  $((x \leq y) \ \&\& \ (y \leq z)) \ || \ (w > z)$

- What is value of  $1 - 2 + 3 * 4$ ?

$1 - 2 + 3 * 4$   
 $= (1 - 2) + (3 * 4)$   
 $= (1 - 2) + 12$   
 $= -1 + 12$   
 $= 11$

CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

8

## Should You Rely on Precedence?



- No!
- The only ones people can remember are
  - "Please Excuse My Dear Aunt Sally"
  - PEMDAS
- Bad  
if  $(2 * x++ < 5 * z + 3 \ \&\& \ -w != x / 2)$
- Better  
if  $(2 * (x++) < ((5 * z) + 3)) \ \&\& \ ((-w) != (x / 2))$

CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

9



## Short-circuiting

- As soon as Java knows an answer – it quits evaluating the expression.
- What does Java print?

```
int x = 0, y = 1;
if ((y > 1) && (++x == 0)){
    --y;
}
System.out.println (x);
```

- 0
- Why?
  - $y > 1$  is false
  - The result of  $\&\&$  will be false, regardless of second expression
  - Java therefore does not evaluate second expression of  $\&\&$
- This treatment of  $\&\&$ ,  $\|\|$  is called **short-circuiting**
  - Subexpressions evaluated from left to right
  - Evaluation stops when value of over-all expression is determined



## Examples

- What does Java print?

```
int x = 0, y = 1;
if ((y >= 1) && (++x == 0)) {
    --y;
}
```

```
System.out.println (x);
```

- 1
- What does Java print?

```
int x = 0, y = 1;
if ( ((y > 1) && (++x == 0))
    ||
    ((y == 1) && (x++ == 0)) ) {
    --y;
}
System.out.println (y);
System.out.println (x);
```

- 0
- 1



## Examples (cont.)

- What does Java print?

```
int x = 0, y = 0;
while (x++ <= 4)
    y += x;
System.out.println (y);
```

- 15



## Programming with Side-Effects

Generally:

- Side effects in conditions are hard to understand
- Good programming practice
  - Conditions should be side-effect-free
  - Side effects should be in “stand-alone statements”
- Major Goal: Strive to create the most readable and maintainable code.

# Primitive Types and their Hierarchy



- double
- float
- long
- int
- short
- byte

```
int x = 7.2;  
double y = 6;
```

- Changing to something else Further Up this list is acceptable
  - called “Widening Conversion”
- Changing to Something else Further Down this list is not acceptable
  - called “Narrowing Conversion”
- Explicit casting needed for when you want a downcast

CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

14

# Type Casting



Which of the following are legal?

- `int x = 3.5;`  
**Illegal:** 3.5 is not an int
- `float x = 3;`  
**Legal:** 3 is an int, which is also a float
- `long i = 3;`  
**Legal:** 3 is an int, which is also a long
- `byte x = 155;`  
**Illegal:** 155 is too big to be a byte (> 127)
- `double d = 3.14159F;`  
**Legal:** 3.14159F is a float, which is also a double

CMSC 131 Spring 2007  
Jan Plane (adapted from Bonnie Dorri)

15





## Mixed Expressions

- What is result of  

```
float x = 3 / 4;
```

  - x assigned value 0.0F
  - Why?
    - 3, 4 are ints
    - So integer / operation is used, yielding 0, before upcasting is performed
- To get floating point result, use explicit casting  

```
float x = (float) 3 / (float) 4;
```

  - Assigns x the value 0.75F
- Can also do following  

```
float x = (float) 3 / 4;
```

  - Why?
  - (float) 3 returns a value type float (3.0F)
  - 4 is an int
  - In this case, Java compiler uses widening conversion on "lower" type (here, int) to obtain values in same type before computing operation