Lecture 7: Evaluation Order

Last time:
1. More on if
2. Project assigned
3. Named constants in Java
4. Loops

Today:
1. Project
2. More assignment operators
3. Precedence and short-circuiting
Get Started on Project #1

- The assignment is on the CMSC 131 web-site (click “Projects” link).
- It is due Sunday, 2/11 at 11 pm
- The project is open
- Start now!
  - Read entire assignment from beginning to end before starting to code
  - Check out assignment now from CVS
  - Follow the instructions exactly, as much of grading is automated
Java Variable Names

- Last time: strategies for naming variables / classes / constants
- What are the legal names (= identifiers) in Java?
  - First character must be letter, _, $
  - Second and subsequent characters may be letter, _, $ or digit
  - Identifiers are case-sensitive: A != a
  - No keywords!
- Another convention: Avoid variable names that differ only on case (i.e. don’t use foo, fOO)
Java Keywords

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

- Java “phrases” that yield values
e.g.
  
  \( x \)
  
  \( x + 1 - y \)
  
  \( x == y \) && \( z == 0 \)
  
  \( \text{foo.equals ("cat") } \)

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

- `int x, y;
  x = y = 1;
  Yes. Result assigns 1 to x, y`

- `int x = 0, y = 1;
  boolean b;
  if (b = (x <= y)){
    x = y;
  }
  Yes. Result assigns true to b, 1 to x`
Other Expressions with Side Effects

- Java includes abbreviations for common forms of assignment
- Example: increment operations
  \[ ++x \quad \text{“Pre-increment”} \]
  \[ \text{Equivalent to} \quad x = x + 1 \]
  \[ x++ \quad \text{“Post-increment”} \]
  \[ \text{Increments } x, \text{ returns old value of } x \]
- Difference
  - \[ x == x++ \quad \text{always true} \]
  - \[ x == ++x \quad \text{never true} \]
Other Assignment Operators

- **Decrement**
  - `--x`  
    - “Pre-decrement”
    - Equivalent to `x = x - 1`
  - `x--`  
    - “Post-decrement”
    - Decrements `x`, returns old value

- **General modification by constant**
  - General form: `<var> <op> = <constant>`
  - Examples
    - `x += 2` equivalent to `x = x + 2`
    - `x *= 2` equivalent to `x = x * 2`
    - `x /= 2` equivalent to `x = x / 2`
Precedence

- Explains how to evaluate expressions
  - What is value of $1 - 2 + 3 \times 4$?
  - **Precedence rules** answer this question
    - Higher-precedence operators evaluated first
    - Example from math: “My Dear Aunt Sally”
      Multiple and divide (higher precedence) before you add and subtract (lower precedence)

- Java follows “My Dear Aunt Sally” … but what about other operators?
Java Precedence Rules

- parentheses:  (  )
- unary ops:  +x  -x  ++x  --x  x++  x--  !x
- multiply/divide:  *  /  %
- add/subtract:  +  -
- comparisons:  <  >  <=  >=
- equality:  ==  !=
- logical and:  &&
- logical or:  ||
- assignments:  =  +=  *=  /=  %=  etc.

increasing precedence
Examples

- `x++ + --x`
  Equivalent to `(x++) + (--x)`

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

- `x <= y && y <= z || w > z`
  Equivalent to `((x <= y) && (y <= z)) || (w > z)`

- What is value of `1 - 2 + 3 * 4`?
  
  
  ```
  1 - 2 + 3 * 4
  = (1-2) + (3*4)
  = -1 + 12
  = 11
  ```

You are not responsible for figuring out values of expressions where the same variable is incremented/decremented more than once on the same line. But you should understand the rules of precedence and also be able to figure values for expressions such as the following: `x++ + --y`. 
Should You Rely on Precedence?

- No!
- The only ones people can remember are “My Dear Aunt Sally” (and parentheses)
- Bad
  \[ 2 \times x ++ < 5 \times z + 3 \land \land -w \neq x / 2 \]
- Better
  \[ (2 \times (x ++) < (5 \times z + 3)) \land \land ((-w) \neq (x / 2)) \]
Short-circuiting

- What does Java print?
  ```java
  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?
  ```java
  int x = 0, y = 1;
  if ((y >= 1) && (++x == 0)) {
    --y;
    System.out.println (x);
  }
  ```
  1

- What does Java print?
  ```java
  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?
  ```java
  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”