CMSC131

More on expression and operations.
Expressions

• We have seen several examples of expressions in Java.
• Some of these returned numbers, other strings.
• The result of some expressions were assigned to variables, others were passed to methods (such as when printing) and others were themselves used as part of larger expressions.
• Is "x=1" an expression? If so, what does it return?
"Side Effects"

• Consider the following code...

```java
public static void main(String[] args) {
    int x,y;
    x=y=1;
    System.out.println(x+" "+y);
}
```

• Will it compile?
• If so, what will it output?
public static void main(String[] args) {
    int x, y;
    x = y = 1;
    System.out.println(x + " " + y);
}

1. Will not compile.
2. Output x y
3. Output 1 1
4. Output y 1
5. Output true 1
Increment and Decrement

• There are a few more math operators available in Java (and also some other languages).

• We need to be VERY careful with how we use these.

• Some new examples to consider:
  
  ```java
  x++;  //post increment
  ++x;  //pre increment
  x+=val; //increment by val
  ```

• There are also x--; --x; x-=val; x*=val; x/=val;
int x, y;

x = 2; y = 5;
System.out.println(x++ * y++);

x = 2; y = 5;
System.out.println(++x * ++y);

x = 2; y = 5;
System.out.println(++x * y++);

x = 2; y = 5;
System.out.println(x++ * ++y);
Precedence / Order of Operations

In Java they are (top being higher precedence)

- parentheses
- unary operations like
  - \(-x\), \(!x\), \(++x\), \(--x\), \(x++\), \(x--\)
- multiplication and division and modulus
- addition and subtraction
- inequality comparisons (greater than, less than, etc)
- equality comparisons (equal to, not equal to)
- logical and
- logical or
- assignment operations like
  - \(=\), \(+=\), \(*=\), \(/=\), \(%=\)
In the case of a tie...

- If two operators have the same precedence, then they are generally evaluated from left to right on the line.

- HOWEVER, assignments are actually done from right to left!

  \[ x = y = z = 4; \]
What does x end up holding in
\[ \text{int } x = 8/4 \times 2/2; \]

1. 0
2. 1
3. $\frac{1}{2}$
4. 2
5. 4
Readability

• While the following two expressions produce the same result, which is easier to read?
  \[(x \leq y \land y \leq z \lor w > z)\]
  \[( (x \leq y) \land (y \leq z) ) \lor (w > z) \]

• Consider breaking things down into smaller parts if there are several logical sub-tests.
  
  ```java
  if ((temp>98&&temp<=100)|| (systolic<=120&&diastolic<80))
  ```
  -versus-
  ```java
  boolean safeTemperature = temp>98 && temp<=100;
  boolean safeBloodPressure = systolic<140 && diastolic<90;
  if (safeTemperature || safeBloodPressure)
  ```
Short-circuiting

- We briefly discussed how once the left-hand operand of an "and" is false, there's no logical need to consider the right-hand operand.
- We also briefly discussed how once the left-hand operand of an "or" is true, there's no logical need to consider the right-hand operand.
- What is the output of the following?

```java
int x=1, y=1, z;
if (x++ > 5 && y-- < 5) {
    z = 10;
}
else {
    z = 20;
}
System.out.println(x + "   " + y + "   " + z);
```
MAKE IT READABLE

• While it is tempting to write "clever code" to make things appear short and sweet and fancy, it often makes it difficult to read and debug later.

• Conditional expressions really should be free of side effects.
Copyright © 2012 : Evan Golub