CSCI 136
Data Structures & Advanced Programming

Fall 2017

Instructors

Bill Jannen & Bill Lenhart

Administrative Details

- Lab 1 handout is now online
- Prelab (should be completed before lab):
  - Lab 1 design doc
    - Use Dice Design Doc as model - no pseudo-code needed this time!
- TA hours start on Wednesday
  - Wed/Thurs : 7:00-11:00pm (in TCL 216)
  - Saturday: 1:00-8:00pm
  - Sunday: 1:00-6:00pm & 7:00-11:00pm

Last Time

Basic Java elements so far

- Primitive and array types
- Variable declaration and assignment

Some basic Unix commands

- Compile (javac), run (java) cycle
- Navigating files: cd (change directory), ls (list)

Today

- Further examples
- Discussion: Lab 1
- Operators & operator precedence
- Expressions
- Control structures
    - Branching: if – else, switch, break, continue
    - Looping: while, do – while, for, for – each
- Object-Oriented Program (OOP) Design
    - Basic concepts and Java-specific features

Sample Programs

- Sum0-5.java
  - Programs that adds two integers
- Of Note:
  - System.in is of type ReadStream
  - Scanner class provides parsing of text streams (terminal input, files, Strings, etc)
  - args[] is passed to `main` from the OS environment
    - args[] contains command-line arguments held as Strings
  - Integer.valueOf(...) converts String to int
  - Static values/methods: in, out, valueOf, main

Lab 1

- Purpose
- Coinstrip Game
  - Demo of solution
- Dice Design Doc
  - Nouns: member variables
  - Verbs: methods

Operators

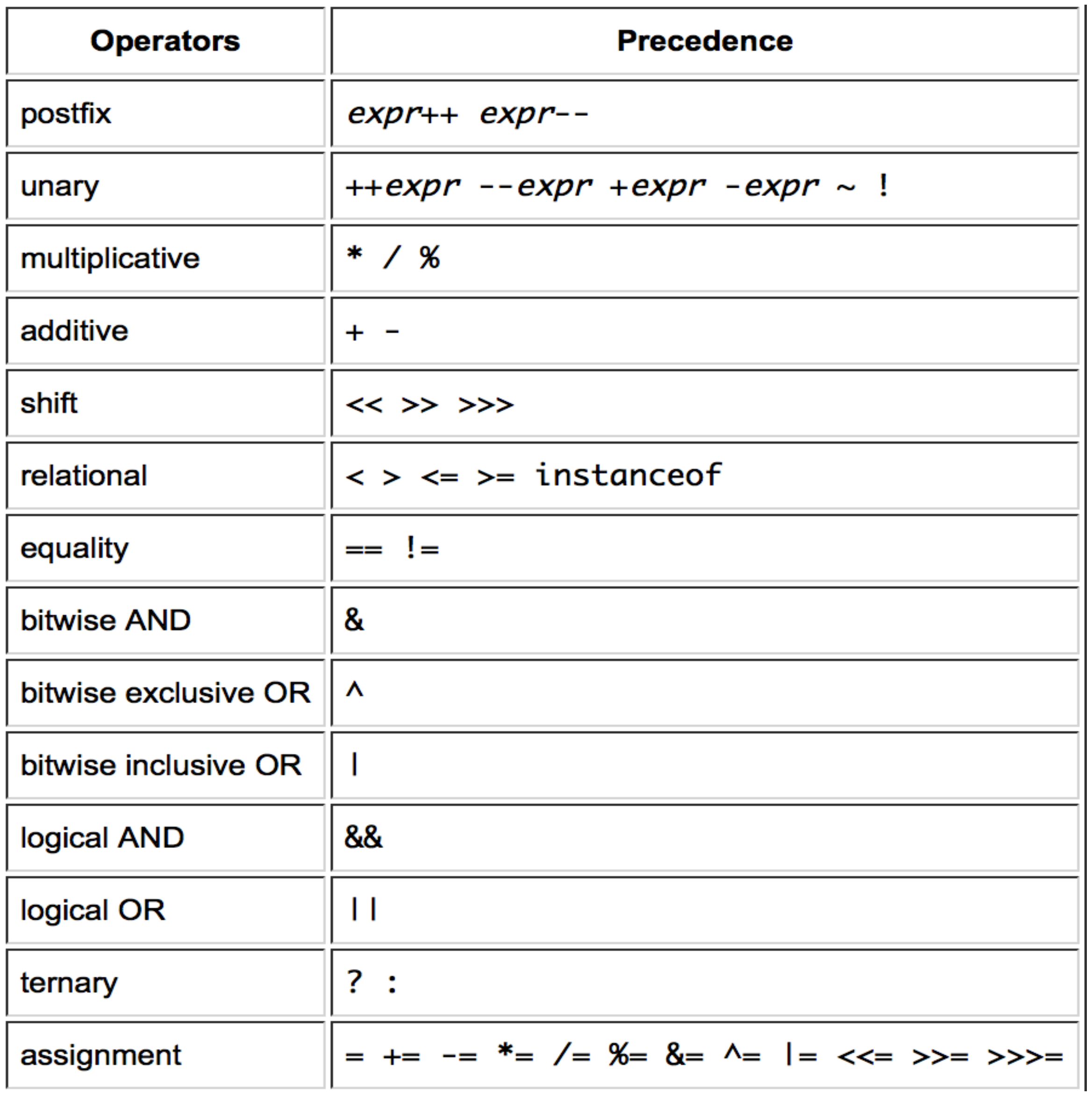
Java provides a number of built-in *operators* including

- Arithmetic operators: +, -, *, /, %
- Relational operators: ==, !=, <, ≤, >, ≥
- Logical operators &&, || (don't use &, |)
- Assignment operators =, +=, -=, *=, /=, ...

Common unary operators include

- Arithmetic: - (prefix); ++, -- (prefix and postfix)
- Logical: ! (not)

Operator Precedence in Java



Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Operator Gotchas!

- There is no exponentiation operator in Java.
  - The symbol ^ is the *bitwise or* operator in Java.
- The *remainder* operator % is the same as the mathematical 'mod' function for *positive* arguments,
  - For **negative** arguments **it is not**: -8 % 3 = -2
- The logical operators && and || use *short-circuit evaluation*:
  - Once the value of the logical expression can be determined, no further evaluation takes place.
  - E.g.: If n = 0, then (n != 0 && (k/n > 3), will yield false without evaluating k/n. Very useful!

Expressions

Expressions are either:

- literals, variables, invocations of non-void methods, or
- statements formed by applying operators to them

An expression returns a value

- `3+2*5 - 7/4      // returns 12`
- `x + y*z – q/w`
- `(- b + Math.sqrt(b*b – 4 * a * c) )/( 2* a)`
- `(n > 0) && (k/n > 2) // computes a boolean`

Expressions

Assignment operator also forms an expression

- x = 3; // assigns x the value 3 and returns 3
- What does this do? y = 4 * (x = 3);
  - sets x = 3, sets y = 12, and returns 12

Boolean expressions let us control program *flow of execution* when combined with *control structures*

Example:

```
– if ( (x < 5) && (y !=0 ) ) {...}
– while (! loggedIn) { ... }
```