Java Basics

13 Jan 2011
CMPT166
Dr. Sean Ho
Trinity Western University

What's on for today

- Info on our programming labs
- Java syntax: expressions and statements
  - Types, operators
  - Choosing names: coding style
- Console I/O and String
- If statements and booleans
- While loops and for loops
- Switch
- Labeled blocks

CMPT166 programming labs

- CMPT166 is weighted heavily on programming labs (6 total)
- These are sizeable programming projects – allocate plenty of time to work on them!
- Individual work – you may discuss with your classmates, but your code should be your own
  - I'm open to team projects if you want, but the scope should expand accordingly
- Write-ups (see sample): design, libraries, variables, pseudocode(s), sample IO, test cases

Expressions and statements

- Legal identifiers: essentially same as in Python
  - Only letters, numbers, or underscore (_)
    - Also '$', but that's special
  - Must not start with number
- Expressions: composed of operators and type-compatible operands
- Statements: declare objects, call methods, or assign expressions to names

Java primitive types

- boolean (1 byte): true, false
- char (2 bytes): Unicode, '\u0000' to '\uFFFF'
- byte (1 byte): -128 to +127
- short (2 bytes): -32768 to +32767
- int (4 bytes): -2^{31} to $+2^{31}-1$
- long (8 bytes): -2^{65} to $+2^{65}-1$
- float (4 bytes): +/- 1.40129846432481707e–45 to 3.40282346638528860e+38
- double (8 bytes): +/- 4.94065645841246544e–324 to 1.79769313486231570e+308

Operator precedence

- In order from most tightly bound first:
  - Parentheses: ()
  - Unary postfix (r to l): x++, x--
  - Unary prefix (r to l): ++x, --x, +x, -x, (type) x
  - Multiplicative: *, /, %
  - Additive: +, -
  - Relational: <, >, <=, >=
  - Equality: ==, !=,
  - Conditional (r to l): ?:
  - Assignment (r to l): =, +=, -=, *=, /=, %=, etc.

Expression compatibility

- Statically typed: declare and initialize variables
  - int numApples = 5;
- Cannot assign mismatched types:
  - numApples = 3.4;   // won't work!
- But values can be promoted to higher precision:
  - float appleSize;
  - appleSize = 3;   // promoted from int to float
  - byte → short → int → long → float → double
    - note that "int / int → int": 14 / 5 → 2
- Type casting forces a type conversion:
  - numApples = (int) 3.99;   // truncated to 3

Coding style

```
public class HelloWorld {
    public static void main( String args[] ) {
        System.out.println( "Hello, World!" );
    }
}
```

- Class names are nouns in CamelCase
- Method names are usually verbs in lowercase:
  - useLowerCamelCase() or use_underscores()
- Local variable names are also lowercase
- Constants: ALL_UPPERCASE

Text output: System.out

- System is a class in the java.lang library
- java.lang is automatically imported
  - Can import other libraries with import
- System.out is the standard output file object
- Its methods include print() and println():
  - System.out.println("Hello!");
  - System.out.print("Hello!\n");
- Other escape characters:
  - Tab (\t), backslash (\\), quote (\")

Console input: Scanner

- System.in is the standard input channel
  - Yields raw text (strings) like Python's input()
- Parse the input using a Scanner object:
  - **import java.util.Scanner;**
  - **Scanner kbd = new Scanner(System.in);**
- Now we can read integers, floats, or words:

```
kbd.nextInt()          // returns an int
kbd.nextDouble()  // returns a double
kbd.next()              // returns next word (string)
```

.nextLine: handling newlines

- The Scanner's .nextLine() method reads from the current file postion to the next newline
  - Returns a string
- Remember to swallow newlines at end of input!
- Say our code does .nextInt(), then .nextLine()
- If the user's keyboard input is “12 apples”,
  - Then the .nextInt() gets 12,
    and .nextLine() gets “apples\n”
- If the user inputs just “12”, then
  - The .nextLine() gets just the newline!

Standard Java class String

- Not a primitive type in Java (unlike Python)
- String class, instantiate with literal strings:
  - String motto = “We aim to please”;
- Concatenation: the “+” operator is overloaded
  - System.out.println(motto + “ you!”);
- Other string operators:
  - motto.length()
  - motto.equals(“We aim to wheeze”)
  - motto.equalsIgnoreCase(“we aim to PLEASE”)
  - motto.toLowerCase()
  - more! See book p.38-41.

If and Booleans

- if (*condition*) *statement*;

- Condition is of type boolean
  - Literals: true, false
  - Binary operators: ==, !=, <, >, <=, >=,
  - Boolean operators (shortcut): &&, ||
- Compound statement using { }:

```
if (condition) {
    statement1;
    statement2;
}
```

Selection: if … else …

```
if (condition)
    statement1;
else
    statement2;
```

- How to do elif?

```
if (condition)
    statement1;
else if (condition2)
    statement2;
```

The “dangling else” problem

```
if (cond1)
    if (cond2)
        statement1;
else
    statement2;
```

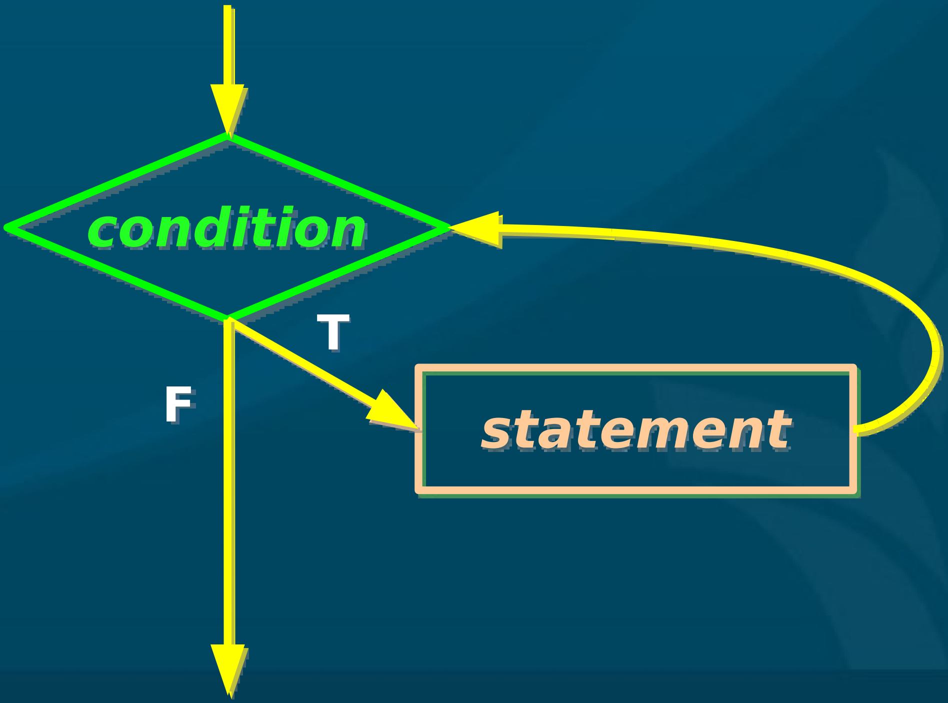
- Which if is the else attached to?
- Solution: always use braces

```
if (cond1) {
    if (cond2) {
        statement1;
    }
} else {
    statement2;
}
```

While loops

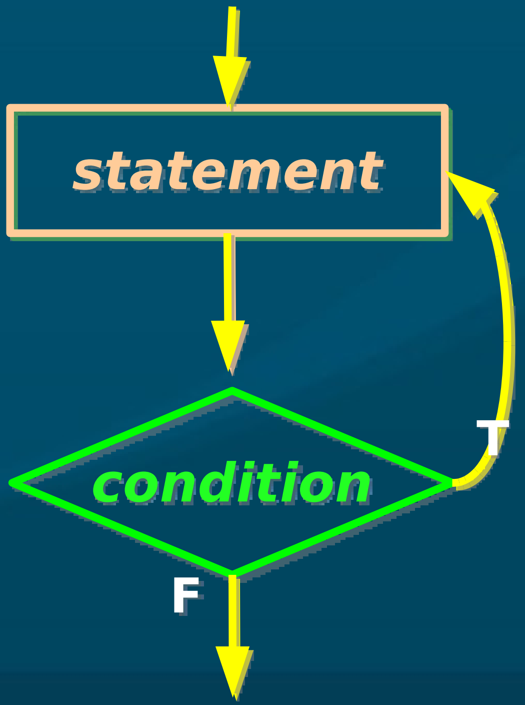
- while (*condition*) *statement*;

- As usual, statement can be a { } block
- *condition* evaluates to a boolean
- Top-of-loop testing
- break and continue as in Python



do/while loops

  - do *statement* while (*condition*);
- As usual, statement can be a { } block
- *condition* evaluates to a boolean
- Bottom-of-loop testing



For loops as while loops

- Any given for loop …

```
for (init; condition; increment) statement;
```

- … can be expressed as an equivalent while loop:

```
init;
while (condition) {
   statement;
   increment;
}
```

Switch statement

```
switch (expression) {
   case val1: statement; ...; break;
   case val2: statement; ...; break;
   ...
   default: statement; ...;
}
```

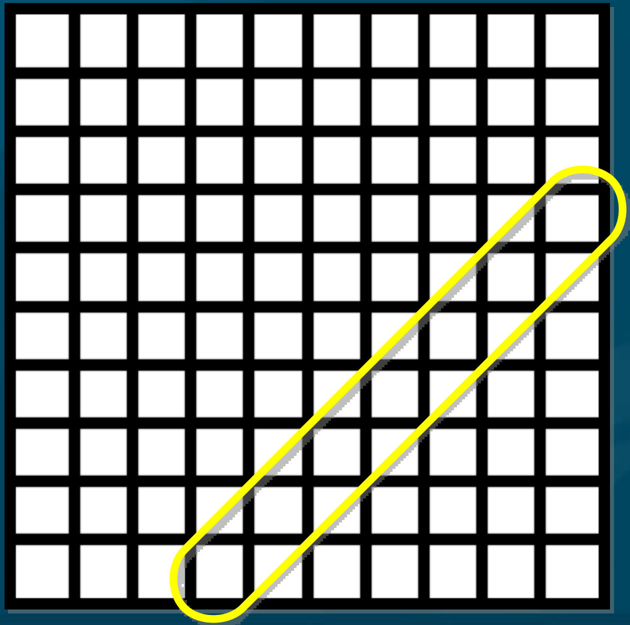
- Similar to a nested if/else structure
  - But *expression* is only evaluated once
- If omit a break, execution continues next case:

```
   case val1:
   case val2: statement; ...; break;
```

Labeled blocks

- Blocks can be named
- break/continue can specify a name:
  - Go to start/end of named block

```
main: {
  for (row=0; row<n_rows; row++) {
    for (col=0; col<n_cols; col++) {
      if (row+col == 12) break main;
    }
  }
}
```



TODO

- Lab0 (due Tue): Eclipse tutorial
  - Get familiar with a Java development environment: Eclipse, NetBeans, or other
  - Write a simple “Hello, World!” program
  - Nothing to turn in
- Lab1 (due Thu 20 Jan): Control/Flow
  - Savitch text, pp.162-164. Choose one of:
  - #2: game of craps
  - #5: loan calculator
  - #8: cryptarithmetic puzzles