Java Programming 2 – Lecture #3 – Jeremy.Singer@glasgow.ac.uk

Identifier Scope

In C-like languages, a block of statements is enclosed in curly braces `{}`. Generally all statements in the block have the same indentation level, but this is not mandatory (unlike Python). A local variable declared in a block is in scope (i.e. accessible) from the point of its declaration to the end of the enclosing block. For method parameters, the parameter is in scope until the end of the method. For iteration variables declared in a `for` loop initializer, the variables are in scope until the end of the loop body.

No two local variables or method parameters that are in scope can share the same identifier name. However a variable may have the same name as a package, type, method, field or statement label. Below is a pathological example, from the Java Language Specification (2nd ed, p113).

```
class a {
  a a(a a) {
    a:
    for(;;) {
      if (a.a(a) == a)
        break a;
    }
    return a;
  }
}
```

Switch statements

Whereas an `if` statement evaluates a `boolean` expression and conditionally executes one of two statements/blocks, on the other hand a `switch` statement evaluates an integer expression (`byte`, `short`, `int`, `long`, `char`, or `enum`) or a `String` (Java 7+) and conditionally executes one or more of the multiple `case` blocks.

In the example source code below, note the use of a catch-all `default` case as the last case block in the `switch` construct. This is mandatory unless all the possible values are covered by explicitly labeled cases. Case labels must be constant expressions. Note the use of `break` statements to prevent fall-through from one case to another (except where the cases share a code block).

```
/**
 * lookup scrabble points for a single letter
 * @arg c the letter to lookup
 * @return the scrabble score for this letter
 */
public static int letterScore(char c) {
  int score;
  switch (c) {
    case 'z':
      score = 10;
      break;
    case 'x':
      score = 8;
      break;
    // …
    case 'g':
    case 'd':
      score = 2;
      break;
    default:
      score = 1;
      break;
  }
  return score;
}
```

Advanced Loop Control Flow

We have already looked at `for` and `while` loops. The `break` and `continue` statements[1] can be used inside these loop bodies to direct control flow explicitly. A `break` is used to exit a loop entirely. A `continue` is used to skip to the end of the current iteration and commence the next iteration (if any).

```
  for (int i=0; i<args.length; i++) {
    if (args[i].equals("needle") {
      System.out.println("found needle in haystack");
      break;
    }
  }
```

[1] See the helpful official documentation at http://docs.oracle.com/javase/tutorial/java/nutsandbolts/branch.html for more details.