COLBY COLLEGE
Department of Mathematics
and Computer Science

CS115 **Allen B. Downey**
Spring 1998

Glossary of Handy Words

Here are a few vocabulary words that might help you translate the craziness Allen says in lecture into something resembling English.

Variable: A container that holds a value. All variables have a name, a value, and a type.

Type: A type is a set of values. For example, the type `int` is the set of integers; the type `double` is the set of floating-point numbers. NOTE: the integers are not a subset of the floating-point numbers. The value 32.0 has type `double` even though it is integral (it's fraction part is zero).

Statement: A sentence of Java code, usually ending with a semi-colon. There are several kinds of statements; so far we have seen declarations, assignments and method invocations.

Declaration: A statement that creates a variable, specifying its name, type and sometimes its initial value. For example `int x;` declares a variable named x with type int; `int x = 5;` declares x and sets its initial value to 5.

Assignment: A statement that changes the value of a variable. For example, `x = 5;` assigns the value 5 to the variable x. NOTE: Assignment is an operation that has a one-time effect on one variable. It is not a statement of equality!

Expression: A collection of operators, variables and literals that can be evaluated. For example, `x+5`, `Math.sqrt(y)`, and `Math.ceil((length-9) * 3.0/2.0) + 15.0`. Expressions, all by themselves, are not statements; they are like nouns without verbs.

Operator: A symbol indicating a mathematical or logical calculation. For example `+` (addition), `*` (multiplication), `%` (modulus), and `&` (logical bitwise AND).

Literal: A simple value, like 7, 32.6, or "Hello!". Like variables, literals have types (the examples above are an int, a double, and a String), but unlike variables, they do not have names and they cannot be modified. So it makes no sense to write `7 = x + 2;`, because you cannot assign a new value to the literal 7.

Cast: Sometimes you can convert an expression from one type to another by casting. For example, the assignment statement `int i = (int) x + 0.5;` adds 0.5 to the value of the variable x and then casts the resulting value (which is a double) to be an integer.

Class: For now, a class is just a named collection of related methods. For example, the Math class contains all the methods that perform mathematical operations. Also, the programs we are writing consist of a single class definition that contains all the methods.

Later on, we will have another definition for a class: it is a blueprint for creating new objects.

Method: In English, a method is a way of doing something. In Java a method is a named set of statements that specify how to do something. Methods take zero or more inputs (called arguments) and return zero or one outputs (called the return value). `main` is a special method — it indicates where the execution of your program begins.

Prototype: A way of describing a method, specifying the types of its arguments and the type of the return value. For example, `double pow (double a, double b);` indicates that the method named `pow` takes two doubles as arguments and returns a double as a return value. NOTE: the arguments you pass to this method do *not* have to be named `a` and `b`.

Invocation: When you use a method, you provide arguments of the appropriate types, and then invoke the method. The method executes, does whatever it is supposed to do, and then returns a value of the promised type. For example, System.out.println ("Hello."); invokes the println method, giving it the String literal "Hello." as an argument. As another example, `c = Math.cos (pi/2.0);` evaluates the expression `pi/2.0` and sends it as an argument to the cos method. It then assigns the return value to the variable c.

NOTE: when you invoke a method, you do not have to specify the types of the arguments or the return value.