Programmer Thinking = Variables, Separate Declarations, Box Analogy

Dr. William L Honig, Loyola University Chicago

A. Getting Started with Variables (Box Analogy)

Three things to be done with variables; think and do them separately:

1. Declare the Variable
    a. Give the variable a meaningful name (put a name on the outside of a box of memory).
    b. Select the correct type for the variable (label the outside of the box with the type of values that will *always* be inside it)
2. Initialize the variable if necessary (open up the box and put a value of the correct type inside). How do you know if it is necessary? If the first use of the variable will be on the left side of an assignment, you can skip initialization; however, it never is wrong to initialize *every* variable at the start of the function containing its declaration.
3. Use the variable (open up the box and either use the current value inside it, or replace the current value with a new one).
    a. In an expression, using the variable name yields its current value (and the value is unchanged).
    b. On the left side of the assignment statement, the value of the variable is replaced by value of the expression on the right hand side (which must match the original type declared for the variable in step 1). It does not matter what the original value was; it replaced and gone forever after the assignment statement.
    c. You *never, never* use the variable name and variable type together with each other again. As programs get more complex this can cause very big problems, ones that the compiler will not find due to “scope” rules.

B. Keeping Declarations Separate From Using Variables

See the three things to be done with variables above; think and do them separately. Be clear what one you are doing (use the order above)

At least when getting started (some programming style guides require always), I find it useful to separate the *declaration* of a variable and its first use (often an assignment). It helps to get the concepts clear in the mind!

A declaration gives a name and a type to a variable. It is a compile time operation, telling the language what name you use for a piece of memory and what type value will be allowed to go into the variable. Nothing is actually stored in the runtime program as code for a declaration. It is just busy work needed to write a program.

Using and initializing or changing the value of a variable is an action taken when the program runs. The computer on which your program is running actually accesses the piece of memory to get its current value or update that value. There is real physical "code" stored elsewhere in memory that will do these steps according to your high level language instructions. At runtime something really happens, the computer uses time, and memory is accessed or changed.

See the following examples(C# and Java).

```
// Bad C# style:
string xString = Console.ReadLine();
int x = int.Parse(xString);

//Proper C# style:
string firstInputString;
int firstNumber;

firstInputString = Console.ReadLine();
firstNumber = int.Parse(firstInputString);
```

```
// Bad Style = Java
Scanner console = new Scanner(System.in);
String firstInputString = console.nextLine();

// Proper Style = Java
Scanner console;
String firstInputString;
int firstInt;

console = new Scanner(System.in);
firstInputString = console.nextLine();
firstInt = console.nextInt();
```

Yes, I know it's longer. It is also easier to keep straight the pure declarations (define a variable and give it a type) from the executable action statements that do things (e.g. get a line from the Console and put it into the firstInputString variable).

C. How to be a Good Programmer with Variables

Some additional thoughts on good programmer thinking with variables:

1. Use long and meaningful names for variables. They are usually nouns.
2. Pick the type of each variable carefully. At the start concentrate on a few primitive types (int, double, boolean).

3. When you use a variable in an expression be aware of its type and the type of other variables and literals elsewhere in the expression.

4. (A bit further on in programming) Be careful not to confuse variables with function or method parameter names. They are a bit alike and a bit different. *Never* use the same name for a variable and a function / method parameter within the same program.

Programming Thinking, for Introduction to Object Oriented Programming, 2016