ICS 111
Variables and Arithmetic

- Review: Variable Declaration and Initialization, assignments
- Java naming conventions, constants, comments
- much more Java arithmetic

Review: Variable Declarations

- When declaring a variable, the type comes first, then the variable name, then the initial assignment
  - the initial assignment is not required
  - but you should make a habit of assigning an initial value whenever you declare a variable
- The variable declaration is only valid from this point in the code onwards!

Variable Declaration Validity: Scope

- The variable declaration is valid in the code only beginning from the point where it is declared:
  - correct:

```
int x = 3;
int y = 7;
x = x + y;    // x is set to 10
```

  - not correct:

```
int x = 3;
x = x + y;    // compile-time error: y undefined
int y = 7;    // too late
```

- The portion of the code where a variable declaration is valid is called the **scope** of the variable

Variable Declaration Validity: More about Scope

- A variable is in scope from its declaration to the end of the enclosing block
- We will see blocks later, but basically, code blocks are delimited by curly braces { and }:

```
if (condition) {
  int x = 3;  // scope of x begins here
  ...
}             // scope of x ends here
x = 7; // error: x cannot be used here
```

Review: Assignments

- Variable initialization is a kind of assignment
- In an assignment, the variable taking a new value is to the left of the = sign, the value is to the right

```
String hello = “Hello, world!”;
…
hello = “Hello, class!”;
System.out.println (hello);
```

- Later we will see different kinds of assignments

Variable Names

- Reminder: we try to write clear code so programmers can understand what they are reading and writing
- This clarity is improved by selecting appropriate variable names
  - This class uses examples like x and y, but all real programs, including your assignments, must use meaningful names
  - for example, `int lectureNumber = 3;`
- Can you have a variable named *%$(#?

Variable Name Constraints

- Can’t have a variable named *%$(#!
- Names only have alphabetic letters, digits, and _ (underscore), and cannot start with a number
  - $ is legal, but don’t use it
- CASE MATTERS!
- Words such as `int` are reserved and cannot be used as variable names

Java Variable Name Convention

- A multi-word variable begins with a lowercase letter, then every new word in the variable begins with uppercase
  - this is known as the camel convention or camel case, because the outline of the name goes up and down like a camel

```
boolean javaUsesCamelConvention = true;
```

Java Constants

- A constant is almost like a variable, but its value never changes
- So we can (and must!) initialize constants, but cannot assign to them
- ```
  final double PI = 3.1415926535;

    PI = 3;   // compile-time error
  ```
- Java constants usually have names in ALL_UPPERCASE, using _ instead of the camel convention
  ```

Java Comments

- Sometimes the code is self-explanatory, sometimes it is not
- When it is not, comments are required
- Comments should explain what the code is doing
- Beginning programmers need more comments, experts understand more and sometimes need fewer comments

Java Comments in Practice

- Each file should begin with a comment describing the purpose of a file
- in-line comments go from `//` to the end of the line
- longer comments are bracketed by

  ```
  /* ... */
  ```

  possibly on multiple lines
- an unterminated comment will cause a compile time error

Java Arithmetic Review

- We have seen the basic operators

  +, -, *, /, % (this last pronounced “modulo”)
- the acronym PEMDAS can remind us that
  - Parentheses have the highest precedence
  - then Exponential (which is not a basic operator in Java, but is in other languages)
  - then Multiplication, Division, and modulo
  - finally, Addition and Subtraction

Details of Precedence

- When equal-precedence operators are used, precedence is left-to-right

  `x / y / z` is `(x / y) / z`

  `10 / 4 * 3` is 6 (because 10/4 is 2!!)

- Exercise: what is the value of

  `10 + 2 * 3 / 7 + 5`

- The negation sign is the same as the minus sign, but has highest precedence

  `3 * -2` is -6

Increment and Decrement

- In programming, we often add one or subtract one from a variable
- This is written ++ or - -

```
int x = 3;
x++;     // x is now 4
x--;     // x is 3 again
x--;     // x is 2
```

- Increment and decrement combine a computation and an assignment

Increment and Decrement: be careful

- Increment and decrement can be used in the middle of an expression

```
int x = 2;
if (x++ > 2) ...
```

- With x++, the value of x changes *after* its value is used in the expression
  - so this condition evaluates to false
- We will look at this more later

Java Math Library

- Computers are good at math, we should be able to use them for more than +-*/%
- The Java Math library provides the most common math functions:
  - `double square = Math.pow(PI, 2);`
  - `double root = Math.sqrt(square);`
- and many more: sin, cos, tan, exp (e^x), log, and so on
- See here for a full definition:

  https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Math.html

Converting Between Integral and Floating Point Types

- Assigning an integer to a float is easy:

```
int three = 3;
double doubleThree = three; // doubleThree is 3.0
```

- Assigning a float to an integer always truncates:

```
double fourEight = 4.8;
int four = (int)fourEight; //four is 4
```

- The type in parentheses is a **typecast**, or simply a **cast**
- We can also round:

```
long five = Math.round(fourEight);
```

Strongly Recommended

- Do the self-check exercises at the end of Section 2.2 in the textbook
- Actually write the code. You can print a variable with

```
System.out.println (variable);
```

- For example, using a variable from the last slide, we can write

```
System.out.print (“five is “);
System.out.println (five);
```

Summary

- Computers are good at math
- Variable declarations must include:
  - initialization
  - standard naming convention
    - camel case for variables
    - all uppercase for constants
- A variable is in scope from its declaration to the end of the enclosing block