Lecture 3: Variables and assignment

CSC 1051 – Data Structures and Algorithms I

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
www.csc.villanova.edu/~map/1051/
Last time:

- Lab 1:
  - Get familiar with jGrasp programming environment
  - Try some example programs
  - Modify programs to create your own
  - Learn about string literals, concatenation, and escape sequences
  - Explore Java syntax
  - Experience some errors!
Today: Problem Solving

• Create a program that will help us calculate a grade point average (GPA) given the number of quality points (QP) and the number of credits.

• The appropriate formula is

\[ GPA = \frac{QP}{\text{credits}} \]

• We assume A, B, C, D, F grading system.
### For Example

<table>
<thead>
<tr>
<th>Course</th>
<th>Credits</th>
<th>Grade</th>
<th>QPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underwater Basket Weaving</td>
<td>3</td>
<td>A = 4</td>
<td>12</td>
</tr>
<tr>
<td>Main Line Boutiques</td>
<td>3</td>
<td>B = 3</td>
<td>9</td>
</tr>
<tr>
<td>Winning the Hoops Lottery</td>
<td>3</td>
<td>C = 2</td>
<td>6</td>
</tr>
<tr>
<td>Web Surfing</td>
<td>3</td>
<td>B = 3</td>
<td>9</td>
</tr>
<tr>
<td>Alg and Data Structures</td>
<td>4</td>
<td>A = 4</td>
<td>16</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td><strong>16</strong></td>
<td></td>
<td><strong>52</strong></td>
</tr>
</tbody>
</table>

\[
\text{GPA} = \frac{52}{16} = 3.25
\]
Topic Thread

• 2.1 Character Strings
• 2.2 Variables, Assignment
• 2.3 Data Types, in particular int, double
• 2.4 Expressions (simple)
• 2.5 Data Conversion
• 2.6 Interactive Programs
• 5.1 Boolean Expressions
• 5.2 The if Statement
• 5.4 The while Statement
The GPA Problem

Solution 1

- Not very exciting, is it?
- Let’s add some storage (remember our model of computing)

```java
//**********************************************
// GPA01.java
//
// Prints out a QPA
//**********************************************

class GPA01 {
    public static void main (String[] args) {
        System.out.println("Quality Points: 52");
        System.out.println("Credits: 16");
        System.out.println("\n\tGPA: 3.25");
    }
}
```
Variables

• A variable is a name for a location in memory

• A variable must be declared by specifying the variable's name and the type of information that it will hold

\[
\begin{align*}
\text{data type} & \quad \text{variable name} \\
\text{int credits;} & \quad \text{int count, temp, result;} \\
\end{align*}
\]

Multiple variables can be created in one declaration
Variable Initialization

• A variable can be given an initial value in the declaration

```
int sum = 0;
int base = 32, max = 149;
```

• When a variable is referenced in a program, its current value is used.
Assignment Statement

• *Changes the value of a variable*
  
• The assignment operator is the \( = \) sign

\[
\text{total} = 55 - \text{discount};
\]

• The expression on the right is evaluated and the result is stored in the variable on the left

• The old value that was in total is overwritten

• See Geometry.java (page 68)
Assignment operator

- Assignment ( = ) copies the value of the right side into the memory location associated with the left side.
- It does not (for primitive types) set up an ongoing equivalence.

```java
int davesAge = 21;
int susAge;
suesAge = davesAge;
davesAge = 22;
System.out.println (davesAge); // prints 22
System.out.println (suesAge);  // prints 21
```

Tracing program code is an important skill!!
Primitive Data

• There are eight primitive data types

• Four of them represent integers:
  – byte, short, int, long

• Two of them represent floating point numbers:
  – float, double

• One of them represents characters:
  – char

• And one of them represents boolean values:
  – boolean
## Numeric Primitive Data

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8 bits</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>16 bits</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>int</td>
<td>32 bits</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>long</td>
<td>64 bits</td>
<td>$&lt; -9 \times 10^{18}$</td>
<td>$&gt; 9 \times 10^{18}$</td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td>$\pm 3.4 \times 10^{38}$ with 7 significant digits</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>$\pm 1.7 \times 10^{308}$ with 15 significant digits</td>
<td></td>
</tr>
</tbody>
</table>
Solution 2

- Still not very exciting, is it?
- Let’s add some processing

//********************************************
// GPA02.java
//
// Prints out a GPA
//********************************************
public class GPA02
{
    public static void main (String[ ] args)
    {
        int qp = 52;
        int credits = 16;
        double gpa = 3.25;

        System.out.println ("Quality Points: " + qp);
        System.out.println ("Credits: " + credits);
        System.out.println ();
        System.out.println ("\tGPA: " + gpa);
    }
}
Expressions

• An *expression* is a combination of one or more operators and operands

• *Arithmetic expressions* compute numeric results and make use of the arithmetic operators:

  - Addition: +
  - Subtraction: -
  - Multiplication: *
  - Division: /
  - Remainder: %

• If either or both operands used by an arithmetic operator are floating point, then the result is a floating point
Division and Remainder

• If both operands to the division operator (/) are integers, the result is an integer (the fractional part is discarded)

  $14 \div 3$ equals $4$

  $8 \div 12$ equals $0$

• The remainder operator (%) returns the remainder after dividing the second operand into the first

  $14 \% 3$ equals $2$

  $8 \% 12$ equals $8$
Operator Precedence

- Operators can be combined into complex expressions
  \[ \text{result} = \text{total} + \frac{\text{count}}{\text{max}} - \text{offset}; \]

- Operators have a well-defined precedence which determines the order in which they are evaluated

- Multiplication, division, and remainder are evaluated prior to addition, subtraction, and string concatenation

- Arithmetic operators with the same precedence are evaluated from left to right, but parentheses can be used to force the evaluation order
Operator Precedence

• What is the order of evaluation in the following expressions?

\[ \begin{align*}
& a + b + c + d + e \\
& a + b \times c - d / e \\
& a / (b + c) - d \% e \\
& a / (b \times (c + (d - e)))
\end{align*} \]
Assignment Revisited

• The assignment operator has a lower precedence than the arithmetic operators

First the expression on the right hand side of the = operator is evaluated

\[ \text{answer} = \frac{\text{sum}}{4} + \text{MAX} \times \text{lowest}; \]

Then the result is stored in the variable on the left hand side

4 1 3 2
Assignment Revisited

• The right and left hand sides of an assignment statement can contain the same variable

  First, one is added to the original value of count

  \[ \text{count} = \text{count} + 1; \]

  Then the result is stored back into count (overwriting the original value)
Increment and Decrement

- The increment and decrement operators use only one operand
- The *increment operator* (++) adds one to its operand
- The *decrement operator* (--) subtracts one from its operand
- The statement
  
  \[ \text{count}++; \]
  
  is functionally equivalent to
  
  \[ \text{count} = \text{count} + \text{1}; \]
Increment and Decrement

- The increment and decrement operators can be applied in *postfix* form:

  \[ \text{count}++ \]

- or *prefix form*:

  \[ ++\text{count} \]

- When used as part of a larger expression, the two forms can have different effects

- Because of their subtleties, the increment and decrement operators should be used with care
Solution 3

- A little more interesting but ...
- What happened to the output?
Summary

- Variables
- Data types
- Assignment operator
- Arithmetic operators
- Operator precedence
Homework

• Review Sections 2.2-2.4
  – *Always* do all self-review exercises when you review material
• Do Exercises EX 2.6, 2.7, 2.8, 2.11
• Read Sections 2.5, 2.6 to prepare for next class

Some slides adapted from a presentation by Daniel Joyce and from the slides accompanying Java Software Solutions by Lewis & Loftus
Caveman image is from http://www.toonpool.com/cartoons/PRIMITIVE%20CAVEMAN%20WORDS_25547#

CSC 1051 M.A. Papalaskari, Villanova University