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:

• Learn about jGrasp - the programming environment that we will be using in this class
  – Compile and run a java program

• Understand the relationship between a Java class name and the name of the .java file where the class is defined

• Practice using basic Java output statements and adding documentation (comments) to your source code.

• Learn about variables, string literals, concatenation. E.g.,
  – int x = 42, count = 100;
  – "Many wise words"
  – "Many " + " many wise words"
  – "Many \"wise\" words"

• Explore Java syntax
• Experience some errors!
Character Strings

• A *string literal* is represented by putting double quotes around the text

• Examples:
  
  "This is a string literal."
  "123 Main Street"
  "X"

• Every character string is an object in Java, defined by the *String class*
The println Method

• In the Lincoln program we invoked the println method to print a character string

• The System.out object represents a destination (the monitor screen) to which we can send output

```
System.out.println ("Whatever you are, be a good one.");
```

CSC 1051 M.A. Papalaskari, Villanova University
The print Method

- The `System.out` object has another method:
  - `print` is similar to the `println` except that it does not advance to the next line
- Example: [Countdown.java](#)
public class Countdown
{
    public static void main (String[] args)
    {
        System.out.print ("Three... ");
        System.out.print ("Two... ");
        System.out.print ("One... ");
        System.out.print ("Zero... ");
        System.out.println ("Liftoff!"); // appears on first output line
        System.out.println ("Houston, we have a problem.");
    }
}
public class Countdown
{
    // Prints two lines of output representing a rocket countdown.
    public static void main (String[] args)
    {
        System.out.print ("Three... ");
        System.out.print ("Two... ");
        System.out.print ("One... ");
        System.out.print ("Zero... ");
        System.out.println ("Liftoff!");  // appears on first output line
        System.out.println ("Houston, we have a problem.");
    }
}
String Concatenation

• The *string concatenation operator* (+) is used to append one string to the end of another
  
  "And one more " + "thing"

• It can also be used to append a number to a string

• A string literal cannot be broken across two lines in a program

• See [Facts.java](https://example.com/Facts.java)
public class Facts {
    public static void main (String[] args) {
        System.out.println("We present the following facts for your " + "extracurricular edification:");
        System.out.println ("Letters in the Hawaiian alphabet: 12");
        continue
    }
}
// A numeric value can be concatenated to a string
System.out.println("Dialing code for Antarctica: " + 672);

System.out.println("Year in which Leonardo da Vinci invented " + "the parachute: " + 1515);

System.out.println("Speed of ketchup: " + 40 + " km per year");
}
Output

We present the following facts for your extracurricular edification:

Letters in the Hawaiian alphabet: 12
Dialing code for Antarctica: 672
Year in which Leonardo da Vinci invented the parachute: 1515
Speed of ketchup: 40 km per year

   System.out.println ("Speed of ketchup: " + 40 + " km per year");
   }
}
Escape Sequences

• What if we wanted to print the quote character?
• Let’s try something like this…

    System.out.println ("I said "Hello" to you.");

• An escape sequence is a series of characters that represents a special character

• An escape sequence begins with a backslash character (\)

    System.out.println ("I said \"Hello\" to you.");
Escape Sequences

• Some Java escape sequences:

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\b</td>
<td>backspace</td>
</tr>
<tr>
<td>\t</td>
<td>tab</td>
</tr>
<tr>
<td>\n</td>
<td>newline</td>
</tr>
<tr>
<td>\r</td>
<td>carriage return</td>
</tr>
<tr>
<td>&quot;</td>
<td>double quote</td>
</tr>
<tr>
<td>'</td>
<td>single quote</td>
</tr>
<tr>
<td>\</td>
<td>backslash</td>
</tr>
</tbody>
</table>

• See [Roses.java](Roses.java)
/**
 * Roses.java        Author: Lewis/Loftus
 * /
 * Demonstrates the use of escape sequences.
 * */

public class Roses
{
    //---
    // Prints a poem (of sorts) on multiple lines.
    //---
    public static void main (String[] args)
    {
        System.out.println ("Roses are red,
            Violets are blue,
            Sugar is sweet,
            But I have " + "commitment issues",
            "So I'd rather just be friends
            At this point in our relationship.");
    }
}
public class Roses {
  public static void main (String[] args) {
    System.out.println("Roses are red,
    Violets are blue,
    Sugar is sweet,
    But I have "commitment issues",
    So I'd rather just be friends
    At this point in our " +
    "relationship.");
  }
}

Output
Roses are red,
Violets are blue,
Sugar is sweet,
But I have "commitment issues",
So I'd rather just be friends
At this point in our relationship.
Quick Check

Write a single `println` statement that produces the following output:

"Thank you all for coming to my home tonight," he said mysteriously.
Quick Check

Write a single `println` statement that produces the following output:

"Thank you all for coming to my home tonight," he said mysteriously.

```java
System.out.println ("\"Thank you all for " +
    "coming to my home\ntonight,\" he said " +
    "mysteriously.\")
```
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

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

• 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>

GPA = 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)
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

```c
int credits;
int count, temp, result;
```

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 \text{total} is overwritten

• See \textit{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 suesAge;
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 x 10^{18}</td>
<td>&gt; 9 x 10^{18}</td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td>+/- 3.4 x 10^{38}  with 7 significant digits</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>+/- 1.7 x 10^{308} with 15 significant digits</td>
<td></td>
</tr>
</tbody>
</table>
A variable is **only declared once!**

- A variable can be given an initial value in the declaration:
  
  ```
  int sum = 0;
  ```

- Later in the program the value of a variable can change:
  
  ```
  sum = 1;
  ```
Strings are *not* primitive data

- We can still declare and use String variables:

  ```java
  String message = "Roses are red";
  ```

  ... And use assignment to change their values:

  ```java
  message = message + "\nViolets are blue";
  ```
Solution 2

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

CSC 1051 M.A. Papalaskari, Villanova University
Division and Remainder

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

\[
14 \div 3 = 4 \\
8 \div 12 = 0
\]

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

\[
14 \mod 3 = 2 \\
8 \mod 12 = 8
\]
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?

\[ a + b + c + d + e \quad a - b / c + d * e \]

\[ a / (b + c) - d \% e \]

\[ a / (b * (c + (d - e))) \]
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.
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
  \[
  \text{count} = \text{count} + 1; \\
  \]
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

- Print/println
- String concatenation
- Escape sequences
- Variables
- Data types
- Assignment operator
- Arithmetic operators
- Operator precedence
Homework

• Review Sections 2.1-2.4
  
  – **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/PRIMITIV20CAVEMAN%20WORDS_25547#