Data and algorithms: variables, assignment, and input

CSC 1051 – Algorithms and Data Structures I

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

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

Some slides in this presentation are adapted from the slides accompanying:
• Java Software Solutions by Lewis & Loftus
• Introduction to Programming in Java: An Interdisciplinary Approach by Robert Sedgewick and Kevin Wayne
Variables

• A *variable* is a name for a location of data in memory

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

```
int sum;
double milesPerGallon;
String name, petName;
```
# Some types of data in Java

<table>
<thead>
<tr>
<th>type</th>
<th>set of values</th>
<th>literal values</th>
<th>operations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>char</strong></td>
<td>characters</td>
<td>'A', '@'</td>
<td>compare</td>
</tr>
<tr>
<td><strong>String</strong></td>
<td>sequences of characters</td>
<td>&quot;Hello World&quot;, &quot;jackie123&quot;</td>
<td>concatenate</td>
</tr>
<tr>
<td><strong>int</strong></td>
<td>integers</td>
<td>17, 12345</td>
<td>add, subtract, multiply, divide</td>
</tr>
<tr>
<td><strong>double</strong></td>
<td>floating-point numbers</td>
<td>3.1415, 6.022e23</td>
<td>add, subtract, multiply, divide</td>
</tr>
<tr>
<td><strong>boolean</strong></td>
<td>truth values</td>
<td>true, false</td>
<td>and, or, not</td>
</tr>
</tbody>
</table>
Variables from Lab 1:

```java
int x = 42, count = 100;
String name = "Kripke";

System.out.println ("Howdy " + name);
System.out.println ("The answer is " + x);
```
Variables from Lab 1:

```java
int x = 42, count = 100;
String name = "Kripke";

System.out.println ("Howdy " + name);
System.out.println ("The answer is " + x);

name = "Sheldon";
x = 33;

System.out.println ("Howdy " + name);
System.out.println ("The answer is " + x);
```

You can change their value
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
Combined declaration and assignment

A variable can be given an initial value in the declaration

```java
int age = 18;
double x = 3.2, y = -0.80;
String name = scan.nextLine();
```
Combined declaration and assignment

A variable can be given an initial value in the declaration - a new value can be assigned later:

```java
int age = 18;
double x = 3.2, y = -0.80;
String name = scan.nextLine();
age = 19;
x = x + 0.5;
name = scan.nextLine();
```
Combined declaration and assignment –
Note: CANNOT declare twice

A variable can be given an initial value in the declaration
- a new value can be assigned later:

```java
int age = 18;
double x = 3.2, y = -0.80;
String name = scan.nextLine();
```

```
X int age = 19;
```

**Error:** declaring variable `age` again
Computing the total number of seconds

```c
int hours = 1;
int minutes = 25;
int seconds = 31;

int totalMinutes = (hours * 60) + minutes;
int totalSeconds = (totalMinutes * 60) + seconds;
```
Example

Computing the total number of seconds

Another alternative:

```cpp
int hours = 1;
int minutes = 25;
int seconds = 31;

int totalSeconds =
    (hours * 3600) + (minutes * 60) + seconds;
```
Arithmetic Operators

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

If either or both operands used by an arithmetic operator are floating point (e.g., type `double`), then the result is a floating point.
Division and Remainder

- If both operands are integers (e.g., type `int`), the division result is an integer (the fractional part is discarded):

  \[ 14 \div 3 \]

  \[ 143 \div 60 \]
Division and Remainder

• If both operands are integers (e.g., type `int`), the division result is an integer (the fractional part is discarded):

% gives the remainder of the division:

\[
\begin{align*}
14 & \div 3 & 14 & \% 3 \\
143 & \div 60 & 143 & \% 60 \\
20 & \div 16 & 20 & \% 16 \\
8 & \div 12 & 8 & \% 12
\end{align*}
\]
Example

Extracting hours, minutes seconds from total number of seconds

```c
int totalSeconds = 7222;
int hours = totalSeconds/3600;
int remainingSeconds = totalSeconds%3600;
int minutes = remainingSeconds/60;
int seconds = remainingSeconds%60;
```
result  =  total + count / max - offset;

**Order of evaluation:**

1. Multiplication, division, remainder
2. addition, subtraction, string concatenation

- Operators with the same precedence: left → right
- Use parentheses to override default order
Examples

\[ a + b + c + d + e \]

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

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

\[ a / (b * (c + (d - e))) \]
Tracing the values of variables after each statement.

```java
int age = 18;
double x;
String name = "Sherlock";
age = 19;
x = 0.5;
x = x + 0.2;
name = name + "Holmes";
```

- **age**: 18
- **x**: 0.5
- **name**: "Sherlock"
Trace: A table of variable values after each statement.

```java
int age = 18;
double x;
String name = "Sherlock";
age = 19;
x = 0.5;
x = x + 0.2;
name = name + "Holmes";
```

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>x</td>
<td>name</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>&quot;undefined&quot;</td>
</tr>
<tr>
<td>18</td>
<td>undefined</td>
<td>&quot;Sherlock&quot;</td>
</tr>
<tr>
<td>19</td>
<td>undefined</td>
<td>&quot;Sherlock&quot;</td>
</tr>
<tr>
<td>19</td>
<td>0.5</td>
<td>&quot;Sherlock&quot;</td>
</tr>
<tr>
<td>19</td>
<td>0.7</td>
<td>&quot;Sherlock Holmes&quot;</td>
</tr>
</tbody>
</table>
int a, b;

a = 3;

b = 4;

a = b;

double pi = 3.14;

String word;
int a, b;

a = 3;

b = 4;

int c = a;

a = b;

b = 5;

b = c;
Assignment operator

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

```java
int davesAge = 21;
int susesAge = davesAge;

davesAge = 22;

System.out.println (davesAge);  // prints 22
System.out.println (suesAge);   // prints 21
```
Increment and Decrement

• The *increment operator* (**``++``**) adds one to its operand
• The *decrement operator* (**``--``**) subtracts one from its operand
• The statement

```
count++;           
```

is functionally equivalent to

```
count = count + 1; 
```
CONSTANTS: like variables, but value cannot change – declare using `final` modifier:

```java
final int INCHES_PER FOOT = 12;
final double LBS_PER KG = 2.2;
```

Convention: Use UPPER_CASE identifiers.
Variables & Assignment Summary

• Variable. A name that refers to a value of declared type.
• Literal. Programming language representation of a value.
• Assignment statement. Associates a value with a variable.

```java
int age;
age = 18;
double x = 3.2, y = -0.80;
String name = scan.nextLine();
final int INCHES_PER FOOT = 12;
```

- **type**
- **variable**
- **declaration statement**
- **assignment statement**
- **literal**
- **combined declaration and assignment statement**
- **constant declaration** (always initializes value)
Variables & Assignment Summary

- **Variable.** A name that refers to a value of declared type.
- **Literal.** Programming language representation of a value.
- **Assignment statement.** Associates a value with a variable.

```java
int age;
age = 18;
double x = 3.2, y = -0.80;
String name = scan.nextLine();
```

- **Declaration statement**
- **Assignment statement**
- **Literal**
- **Combined declaration and assignment statement**
- **Constant declaration (always initializes value)**

Next: Algorithms that use variables and values obtained while the program is running
Algorithms

An algorithm is a specific set of instructions for carrying out a procedure or solving a problem, usually with the requirement that the procedure terminate at some point. Specific algorithms sometimes also go by the name method, procedure, or technique. The word "algorithm" is a distortion of al-Khwārizmī, a Persian mathematician who wrote an influential treatise about algebraic methods.

Sources: [http://mathworld.wolfram.com/Algorithm.html](http://mathworld.wolfram.com/Algorithm.html) and Wikipedia ([http://en.wikipedia.org/wiki/Muhammad_ibn_Mos%C4%81_al-Khw%C4%81rizm%C4%AB](http://en.wikipedia.org/wiki/Muhammad_ibn_Mos%C4%81_al-Khw%C4%81rizm%C4%AB))
Dear various parents, grandparents, co-workers, and other "not computer people."

We don't magically know how to do everything in every program. When we help you, we're usually just doing this:

START

Find a menu item or button which looks related to what you want to do.

I can't find one

Pick one at random

I've tried them all

OK

Click it

Have you been trying this for over half an hour?

No

Did it work?

No

Ask someone for help or give up.

Yes

You're done!

Google the name of the program plus a few words related to what you want to do. Follow any instructions.

Please print this flowchart out and tape it near your screen. Congratulations; you're now the local computer expert!

Source: http://xkcd.com/627/
Algorithm Example: *Input-Compute-Output* pattern

**GPA problem:** Write a program that computes and outputs the GPA, given the credits and quality points earned.

**Variables:** qp, credits, gpa

**Algorithm:**

1. Input qp
2. Input credits
3. gpa = qp / credits
4. Print gpa

*CSC 1051 M.A. Papalaskari, Villanova University*
Writing an algorithm in pseudocode

- List the variables used.
- List the steps for solving the problem, in order.
- Try to be brief and unambiguous; use Java expressions only when it is simpler to specify a step in Java than in English.

**Variables:** qp, credits, gpa

**Algorithm:**
1. Input qp
2. Input credits
3. gpa = qp / credits
4. Print gpa
Writing an algorithm in pseudocode

• List the variables used.
• List the steps for solving the problem, in order.
• Try to be brief and unambiguous; use Java expressions only when it is simpler to specify a step in Java than in English.

Variables: qp, credits, gpa

Algorithm:
1. Input qp
2. Input credits
3. gpa = qp / credits
4. Print gpa

(Note: use floating point division)
Variables: qp, credits, gpa

Algorithm:
1. Input qp
2. Input credits
3. gpa = qp / credits
4. Print gpa

Next: A closer look at input in Java

Java Program ➔

```java
// GPA.java    Author: Joyce/Papalaskari
// Demonstrates the use of Scanner input and simple computation.
import java.util.Scanner;

public class GPA
{
    public static void main (String[] args)
    {
        double qp, credits, gpa;
        Scanner scan = new Scanner(System.in);

        // get input
        System.out.print ("Enter Quality Points > ");
        qp = scan.nextInt();
        System.out.print ("Enter Credits > ");
        credits = scan.nextInt();

        // output information entered
        System.out.println ("\nQuality Points: "+ qp);
        System.out.println ("Credits:        "+ credits);

        // calculate and output GPA
        gpa = qp / credits;
        System.out.println ("\ntGPA:  "+ gpa);
    }
}
```
Interactive Programs – Input/Output

- Programs can use data obtained during runtime, e.g:

```java
int age;

String name;

System.out.print("Enter your name");
name = scan.nextLine();

System.out.print("Enter your age");
age = scan.nextInt();
```
Interactive Programs – Input/Output

- In Java, you first need to create a Scanner object

```java
int age;

String name;

Scanner scan = new Scanner(System.in);

System.out.print("Enter your name");
name = scan.nextLine();

System.out.print("Enter your age");
age = scan.nextInt();
```
Reading Input

• The `Scanner` class is part of the `java.util` class library, and must be `imported` into a program in order to be used

• The import statement goes at beginning of your program (above class definition)

```java
import java.util.Scanner;
```
Using the Scanner class

1. import the class, i.e., add this before the class definition of your program:

```java
import java.util.Scanner;
```

2. In your main method, before doing any input, declare and initialize the Scanner object

```java
Scanner scan = new Scanner(System.in);
```

3. Input away!

```java
System.out.print("Enter your name");
name = scan.nextLine();

System.out.print("Enter your age");
age = scan.nextInt();
```
import java.util.Scanner;

public class TellMeAboutYou
{
    public static void main(String[] args)
    {
        int age;
        String name;

        Scanner scan = new Scanner(System.in);

        System.out.print("Enter your name");
        name = scan.nextLine();

        System.out.print("Enter your age");
        age = scan.nextInt();

        System.out.println("Pleased to meet you, " + name + "!");
        System.out.println("Your age in dog years is " + age*10.5);
    }
}

Inspired by: http://www.onlineconversion.com/dogyears.htm
Input methods

- `nextInt()` → input an int
- `nextDouble()` → input a double
- `nextLine()` → input a String (until end of line)
- `next()` → input a String `token` (one word or other delimited “chunk” of text)

- White space (space, tab, new line) are used to separate input tokens
Using the **Scanner** class

```java
import java.util.Scanner;

public class TellMeAboutYou
{
    public static void main(String[] args)
    {
        int age;
        String name;

        Scanner scan = new Scanner(System.in);

        System.out.print("Enter your name and age:");
        name = scan.nextLine();
        age = scan.nextInt();

        System.out.println("Pleased to meet you, " + name + "!");
        System.out.println("Your age in dog years: " + age*10.5);
    }
}
```

More examples – see text:  [Echo.java](#)  [GasMileage.java](#)