Algorithms and Java basics: pseudocode, variables, assignment, and interactive programs

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
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

I’VE TRIED THEM ALL

PICK ONE AT RANDOM

OK

NO

HAVE YOU BEEN TRYING THIS FOR OVER HALF AN HOUR?

NO

CLICK IT

YES

ASK SOMEONE FOR HELP OR GIVE UP.

NO

DID IT WORK?

NO

YOU'RE DONE!

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/
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 [named after Muhammad ibn al-Khwārizmī], a Persian mathematician who wrote an influential treatise about algebraic methods.

Sources: http://mathworld.wolfram.com/Algorithm.html and Wikipedia (http://en.wikipedia.org/wiki/Mu%E1%B8%A5ammad_ibn_M%C5%ABs%C4%81_al-Khw%C4%81rizm%C4%AB)
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. qp = input from user
2. credits = input from user
3. gpa = qp / credits
4. Print gpa
import java.util.Scanner;

public class GPA {
    public static void main (String[] args)
    {
        double qp, credits, gpa;
        Scanner scan = new Scanner(System.in);
        System.out.print ("Enter Quality Points > ");
        qp = scan.nextInt();
        System.out.print ("Enter Credits > ");
        credits = scan.nextInt();
        gpa = qp / credits;
        System.out.println ("\n\tGPA:  "+ gpa);
    }
}

Variables: qp, credits, gpa

Algorithm:
1. qp = input from user
2. credits = input from user
3. gpa = qp / credits
4. Print gpa

Next: A closer look at variables & input in Java

CSC 1051 M.A. Papalaskari, Villanova University
Interactive Programs – Input/Output

• Programs can use data obtained during runtime, eg:

```java
int age;

String name;

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

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

• 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;

public class GPA {
    
    public static void main(String[] args) {
        //-------------
    }
```
Interactive Programs – Input/Output

Summary:

1. import the Scanner 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.next();
   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.next();

        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: " + age*10.5);
    }
}
Scanner 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
Variables & Assignment

- **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();
```

- **type**
- **variable**
- **declaration statement**
- **assignment statement**
- **literal**
- **combined declaration and assignment statement**
- **constant declaration (always initializes value)**
- **input from user**

CSC 1051 M.A. Papalaskari, Villanova University
Variable Declaration

- 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.

```java
int age;
double x, y;
String name;
```
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>char</td>
<td>characters</td>
<td>'A'</td>
<td>compare</td>
</tr>
<tr>
<td>String</td>
<td>sequences of characters</td>
<td>&quot;Hello World&quot;</td>
<td>concatenate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;jackie123&quot;</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>integers</td>
<td>17</td>
<td>add, subtract, multiply, divide</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12345</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>floating-point numbers</td>
<td>3.1415</td>
<td>add, subtract, multiply, divide</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.022e23</td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td>truth values</td>
<td>true</td>
<td>and, or, not</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>

CSC 1051 M.A. Papalaskari, Villanova University
Assignment Statement

- *Changes the value of a variable*
- The assignment operator is the $=$ sign

```
total = 55 - 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();
```
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();
```

```java
// Error: declaring variable age again
int age = 19;
```
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 0.7

name

"Sherlock" "SherlockHolmes"

CSC 1051 M.A. Papalaskari, Villanova University
int a, b;

a = 3;

b = 4;

int c = a;

a = b;

b = 5;

b = c;

Final values:

a
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 suesAge = davesAge;

davesAge = 22;

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

- **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();
```

**SUMMARY**

- **Type** and **variable**
- **Declaration statement**
- **Assignment statement**
- **Literal**
- **Combined declaration and assignment statement**
- **Constant declaration (always initializes value)**
- **Input from user**
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
Example

```java
int feet = 25;
int inches = feet * INCHES_PER_FOOT;

int seconds = 143;
int minutes = seconds / 60;
int remainingSeconds = seconds % 60;
```
Division and Remainder

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

<table>
<thead>
<tr>
<th>Integer Division:</th>
<th>Remainder:</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 / 3</td>
<td>14 % 3</td>
</tr>
<tr>
<td>143 / 60</td>
<td>143 % 60</td>
</tr>
<tr>
<td>20 / 16</td>
<td>20 % 16</td>
</tr>
<tr>
<td>8 / 12</td>
<td>8 % 12</td>
</tr>
</tbody>
</table>
Operator Precedence

What is the order of evaluation of sub-expressions?

1. Multiplication, division, remainder
2. Addition, subtraction, string concatenation
   - Operators with the same precedence: **left → right**
   - Use **parentheses** to override default order

**more examples:**

```
result = total + count / max - offset;

a + b + c + d + e

a - b / c + d * e

a / (b + c) - d % e

a / (b * (c + (d - e))))
```
Increment and Decrement

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

\[
\text{count}++; \\
\text{is functionally equivalent to}
\]

\[
\text{count} = \text{count} + 1;
\]
import java.util.Scanner;

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

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

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

        // calculate GPA
        gpa = qp / credits;

        // print GPA
        System.out.println("\n\tGPA:  " + 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. qp = input from user
2. credits = input from user
3. gpa = qp / credits
4. Print gpa
Example

Write an algorithm to solve the following problem:

*Input values representing a time duration in hours, minutes, and seconds and then calculate and output the equivalent total number of seconds.*
Example

Write an algorithm to solve the following problem:

*Input a value representing a number of seconds, then calculate and output the equivalent amount of time as a combination of hours, minutes, and seconds.*