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/

---

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

---

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:** q, credits, gpa

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

---

Algorithms in everyday life

**CSC 1051 M.A. Papalaskari, Villanova University**
Java Program

```java
import java.util.Scanner;
public class GPA {
    public static void main (String[] args) {
        // Inputs quality points and credits and calculate GPA.
        double qp, credits, gpa;
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter Quality Points > ");
        qp = scan.nextDouble();
        System.out.print("Enter Credits > ");
        credits = scan.nextInt();
        gpa = qp / credits;
        System.out.println("GPA: "+ gpa);
    }
}
```

Algorithm:

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

Interactive Programs – Input/Output

- Programs can use data obtained during runtime, eg:
  ```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();
  ```

- 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();
  ```

- 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) {
      // Demonstrates the use of Scanner.
      System.out.println("GPA: "+ gpa);
    }
  }
  ```
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();
```

```

Interactive Programs – Input/Output

Summary:

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);
    }
}
```

```java
name = scan.nextLine();
```

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

Enter your name: Fiona
Enter your age: 17
Pleased to meet you, Fiona!
Your age in dog years: 178.5

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();
```
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>int</td>
<td>integers</td>
<td>17, 12345</td>
<td>add, subtract, multiply, divide</td>
</tr>
<tr>
<td>double</td>
<td>floating-point numbers</td>
<td>3.1415, 6.022e23</td>
<td>add, subtract, multiply, divide</td>
</tr>
<tr>
<td>boolean</td>
<td>truth values</td>
<td>true, false</td>
<td>and, or, not</td>
</tr>
</tbody>
</table>

Assignment Statement

- Changes the value of a variable.
- The assignment operator is the `=` sign.

```java
total = 55 - discount;
```

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();
int age = 19;
```

Tracing the values of variables after each statement.

Trace: A table of variable values after each statement.

<table>
<thead>
<tr>
<th>age</th>
<th>x</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>?</td>
<td>undefined</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;SherlockHolmes&quot;</td>
</tr>
</tbody>
</table>

Final values:
### Trace: TRY THIS:

```java
int a, b;
a = 3;
b = 4;
int c = a;
a = b;
b = 5;
b = c;
```

<table>
<thead>
<tr>
<th>Final values:</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>b</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>c</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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;
final int INCHES_PER_FOOT = 12;
String name = scan.nextLine();
```

- *combined declaration and assignment statement*
- constant declaration (always initializes value)
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

```c
int minutes = 25;
int secondsInMinutes = minutes * 60;
int seconds = 100;
int minutesFromSeconds = seconds / 60;
int remainingSeconds = seconds % 60;
```

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:

- `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 `count++;` is functionally equivalent to `count = count + 1;`