Data and algorithms: variables, assignment, and interactive programs

CSC 1051 – Algorithms and Data Structures I

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
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 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>char</td>
<td>characters</td>
<td>&quot;A&quot;</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;
```

- 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();
```

Error: declaring variable `age` again

---

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;
```

Example

Computing the total number of seconds

```java
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:

```java
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):

  - \(14 \div 3\) \(14 \% 3\)
  - \(143 \div 60\) \(143 \% 60\)
  - \(20 \div 16\) \(20 \% 16\)
  - \(8 \div 12\) \(8 \% 12\)
Extracting hours, minutes seconds from total number of seconds

```java
int totalSeconds = 7222;
int hours = totalSeconds/3600;
int remainingSeconds = totalSeconds%3600;
int minutes = remainingSeconds/60;
int seconds = remainingSeconds%60;
```

**Operator Precedence**

```
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";
```
Trace: A table of variable values after each statement.

<table>
<thead>
<tr>
<th></th>
<th>age</th>
<th>x</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>18</td>
<td>undefined</td>
<td>&quot;Sherlock&quot;</td>
</tr>
<tr>
<td>int age = 18;</td>
<td>18</td>
<td>undefined</td>
<td>&quot;Sherlock&quot;</td>
</tr>
<tr>
<td>double x;</td>
<td>19</td>
<td>0.5</td>
<td>&quot;Sherlock&quot;</td>
</tr>
<tr>
<td>String name = &quot;Sherlock&quot;;</td>
<td>19</td>
<td>0.7</td>
<td>&quot;SherlockHolmes&quot;</td>
</tr>
<tr>
<td>age = 19;</td>
<td>19</td>
<td>undefined</td>
<td>&quot;Sherlock&quot;</td>
</tr>
<tr>
<td>x = 0.5;</td>
<td>19</td>
<td>0.7</td>
<td>&quot;SherlockHolmes&quot;</td>
</tr>
<tr>
<td>x = x + 2;</td>
<td>19</td>
<td>0.7</td>
<td>&quot;SherlockHolmes&quot;</td>
</tr>
<tr>
<td>name = name + &quot;Holmes&quot;;</td>
<td>19</td>
<td>0.7</td>
<td>&quot;SherlockHolmes&quot;</td>
</tr>
</tbody>
</table>

Final values: age = 19, x = 0.7, name = "SherlockHolmes"

---

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
```
Increment and Decrement

- The *increment operator* (++) adds one to its operand
- The *decrement operator* (--) subtracts one from its operand
- The statement
  
  ```java
  count++;
  ```
  
is functionally equivalent to
  
  ```java
  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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable</th>
<th>Declaration Statement</th>
<th>Assignment Statement</th>
<th>Literal</th>
<th>Combined Declaration and Assignment Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int</code></td>
<td><code>age</code></td>
<td><code>int age;</code></td>
<td><code>age = 18;</code></td>
<td><code>double</code></td>
<td><code>final int INCHES_PER FOOT = 12;</code></td>
</tr>
<tr>
<td><code>double</code></td>
<td><code>x</code>, <code>y</code></td>
<td><code>double x = 3.2, y = -0.80;</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:
- [Python.org](http://www.python.org)
- [Wikipedia](https://en.wikipedia.org/wiki/Muhammad_ibn_Musa_al-Khwarizmi)
Algorithms in everyday life

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. \( gpa = \frac{qp}{credits} \)
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. \( gpa = \frac{qp}{credits} \)
4. Print gpa

When the type is not obvious you can add a note.

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

Algorithm:
1. Input \( q_p \)
2. Input credits
3. \( gpa = \frac{q_p}{\text{credits}} \)
4. Print \( gpa \)

Java Program ➔

```java
// GPA.java Author: Joyce/Papalaskari
// Demonstrates the use of Scanner input and a simple computation.
import java.util.Scanner;
public class GPA {
    public static void main(String[] args) {
        // Inputs the quality points and credits and calculates GPA.
        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("\nGPA: "+ gpa);
    }
}
```

Interactive Programs – Input/Output

• Programs can use data obtained during runtime, eg:

```java
public class GPA {
    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();
    }
}
```

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();
```

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;
```

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)
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();
   ```

---

**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");
        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: " + age*10.5);
    }
}
```

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

---

More examples – see text: Echo.java  GasMileage.java