Control flow and conditionals

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: qp, credits, gpa

Algorithm:
1. Input qp
2. Input credits
3. Output values entered
4. gpa = qp / credits
5. Print gpa
6. Print goodbye message

Java Program

Algorithm:
variables: qp, credits, gpa

Algorithm:
1. Input qp
2. Input credits
3. Output values entered
4. gpa = qp / credits
5. Print gpa
6. Print goodbye message

if (credits == 0)
    System.out.println("No GPA yet");
else
    gpa = qp / credits;
    System.out.println("GPA: " + gpa);

if (credits == 0)
    System.out.println("No GPA yet");
else
    gpa = qp / credits;
    System.out.println("GPA: " + gpa);

Improved algorithm
Control flow and conditionals

**Updated program**

```java
import java.util.Scanner;

class GPA_Updated {
    public static void main(String[] args) {
        // Read the 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();

        System.out.println("Quality Points: "+qp);
        System.out.println("Credits: "+credits);

        if (credits == 0)
            System.out.println("No GPA yet");
        else {
            gpa = qp / credits;
            System.out.println("GPA: "+gpa);
        }

        System.out.println("Thanks for using my program.");
    }
}
```

**Algorithm:**

1. Input `qp`
2. Input `credits`
   - If `credits` equals 0
     - Print "No gpa yet"
   - Else
     - `gpa = qp / credits`
     - Print `gpa`
3. Print goodbye message

**Control flow**

- Sequence of statements that are actually executed in a program

```
statement 1
↓
statement 2
↓
statement 3
↓
statement 4
```

**Conditional and Repetition statements:** enable us to alter control flow

- **Condition 1:**
  - `true`
  - `statement 1`
  - `statement 2`
  - `statement 4`

- **Condition 2:**
  - `true`
  - `statement 4`
  - `statement 3`

- **Condition 1:**
  - `false`
  - `statement 2`

**Control flow**

- Sequence of statements that are actually executed in a program

- **Example:**
  - Input `qp`
  - Input `credits`
  - `gpa = qp / credits`
  - Print `gpa`
  - Print goodbye message
Control flow and conditionals

Control flow
- Sequence of statements that are actually executed in a program
- Example:

Java Conditional statements alter the linear flow of control. They use boolean expressions to determine what to do next. Example:

```java
if (credits == 0)
    System.out.println("No GPA yet");
else
    {
        gpa = qp / credits;
        System.out.println("GPA: " + gpa);
    }
```

Java relational operators
- relational operators can be used with numeric types and produce boolean results:
  - == equal to
  - != not equal to
  - < less than
  - > greater than
  - <= less than or equal to
  - >= greater than or equal to
- Note the difference between the equality operator (==) and the assignment operator (=)

Conditional statements
- if (condition) statement;
- if (condition) statement1;
- else statement2;
Control flow and conditionals

Example:
How do we fix output to use singular/plural as appropriate?
For example:
Enter the total amount to be given as change: 18
That amount can be given as:
0 quarters
1 dime
1 nickels
3 pennies

Create an application called Vacation that prompts for and inputs an integer representing someone’s age and then suggests an appropriate vacation destination. One of two destinations should be suggested depending on whether person is over 30.

```java
input age
if (age > 30)
    print “Florida”
else
    print “Grand Canyon”
print “goodbye”
```

Another example:
Create an application called Vacation that prompts for and inputs an integer representing someone’s age and then suggests an appropriate vacation destination. One of three destinations should be suggested depending on whether the answer is less than 20, between 20 and 50, or over 50.

```java
input age
if (age < 20)
    print “Florida”
else if (age > 50)
    print “Grand Canyon”
else
    print “Florida”
print “goodbye”
```

This slide adapted from Wayne E. Dennis. Princeton course: http://www.cs.princeton.edu/courses/archive/spring13/cos126/lectures.php

Prisoner's dilemma

Dr Papalaskari
Java Logical Operators

- Logical operators can be used with boolean operands to express more complex conditions:

  - Logical NOT: `!`
  - Logical AND: `&&`
  - Logical OR: `||`

Boolean Expressions

- The reserved words `true` and `false` are the only valid values for a boolean type
- Example: boolean variables:

  ```java
  boolean aboveAgeLimit = false;
  boolean usePlural = hours > 1;
  ```

Vacation example revisited:

Create an application called Vacation that prompts for and inputs an integer representing someone’s age and then suggests an appropriate vacation destination. One of three destinations should be suggested depending on whether the answer is less than 20, between 20 and 50, or over 50.

Sample output:

```
How old is the traveler?: 59
Suggestion: Florida.
```

Logical Operators – Another Example

```java
if (total < MAX + 5 && (!found))
    System.out.println("Processing...");
```

- All logical operators have lower precedence than the relational operators
- The `!` operator has higher precedence than `&&` and `||`
Logical NOT

- The logical NOT operation is also called *logical negation* or *logical complement*.
- If some boolean condition \( a \) is true, then \( \neg a \) is false; if \( a \) is false, then \( \neg a \) is true.
- Logical expressions can be shown using a *truth table*:

<table>
<thead>
<tr>
<th>( a )</th>
<th>( \neg a )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Logical AND and Logical OR

- The logical AND expression \( a \land b \) is true if both \( a \) and \( b \) are true, and false otherwise.
- The logical OR expression \( a \lor b \) is true if \( a \) or \( b \) or both are true, and false otherwise.

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( a \land b )</th>
<th>( a \lor b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

Quick Check 1

What does this statement do?

```java
if (found || !done)
    System.out.println("Ok");
```

`found` false, `done` false

Quick Check 2

What does this statement do?

```java
if (total != stock + warehouse)
    inventoryError = true;
```

<table>
<thead>
<tr>
<th>total</th>
<th>stock</th>
<th>warehouse</th>
<th>inventoryError</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>7</td>
<td>12</td>
<td>false</td>
</tr>
</tbody>
</table>
Boolean Expressions

• using truth tables – let’s try this one:

| found | done | !done | found || !done |
|-------|------|-------|-------|-------|
| false | false |       | false | false |
| false | true  |       | true  | false |
| true  | false |       | true  | true  |
| true  | true  |       | true  | true  |

Boolean Expressions

• using truth tables – another example:

<table>
<thead>
<tr>
<th>total &gt; MAX</th>
<th>found</th>
<th>!found</th>
<th>total &gt; MAX &amp;&amp; !found</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td></td>
<td>true</td>
</tr>
</tbody>
</table>

How much of a boolean expression do we need to evaluate before determining its value?

*** Short-Circuited Operators

• The processing of && and || is “short-circuited” in cases where the left operand is sufficient to determine the result (the right operand is not evaluated at all)

• This can be both useful and dangerous!

```
if (count != 0 && total/count > MAX)
    System.out.println("Testing.");
```

Indentation Revisited

• Remember that indentation is for the human reader, and is ignored by the computer

```
if (total > MAX)
    System.out.println("Error!!");
    errorCount = errorcount + 1;;
```

Despite what is implied by the indentation, the increment will occur whether the condition is true or not