Control flow, conditionals, boolean expressions, block statements, nested statements

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
/*************************************************************/
// 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) {
        // 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 values entered
        System.out.println("\nQuality Points: "+qp);
        System.out.println("Credits: " + credits);
        // calculate & output GPA
        gpa = qp / credits;
        System.out.println("\nGPA: " + gpa);
        // print goodbye message
        System.out.println ("Thanks for using my program.");
    }
}
variables: qp, credits, gpa

Algorithm:
1. Input qp
2. Input credits
3. Output values entered
   
   4. if credits equals 0
      • Print “No gpa yet”
   else
      • gpa = qp / credits
      • Print gpa

5. Print goodbye message

Java code

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

Algorithm:
1. Input qp
2. Input credits
3. Output values entered
   if credits equals 0
     • Print “No gpa yet”
   else
     • gpa = qp / credits
     • Print gpa
4. 
5. Print goodbye message

Updated program

```java
import java.util.Scanner;

public class GPA_Updated {
    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 values entered
        System.out.println ("\nQuality Points: " + qp);
        System.out.println ("Credits: " + credits);

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

        // Print goodbye message
        System.out.println ("Thanks for using my program.");
    }
}
```

CSC 1051 M.A. Papalaskari, Villanova University
Control flow

- Sequence of statements that are actually executed in a program

Conditional and Repetition statements: enable us to alter control flow

This slide adapted from Wayne & Sedgewick Princeton course: [http://www.cs.princeton.edu/courses/archive/spring13/cos126/lectures.php](http://www.cs.princeton.edu/courses/archive/spring13/cos126/lectures.php)

CSC 1051 M.A. Papalaskari, Villanova University
Control flow

- Sequence of statements that are actually executed in a program

Example:

```
input qp
input credits
if credits is zero?
  true
  gpa = qp / credits
  print gpa
  print goodbye message
false
  gpa = qp / credits
  print gpa
  print goodbye message
```

• 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"); // if true, do this
} else {
    gpa = qp / credits;
    System.out.println("GPA: " + gpa); // if false, do these
}
```

A boolean expression

if more than one statement, enclose in braces

CSC 1051 M.A. Papalaskari, Villanova University
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

\[
\text{if ( condition )}
\]
\[
\text{statement ;}
\]

// no else clause

\[
\text{if ( condition )}
\]
\[
\text{statement1 ;}
\]
\[
\text{else}
\]
\[
\text{statement2 ;}
\]
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 dimes
1 nickels
3 pennies

get rid of this!
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.

input age

true

age > 30?

false

print “Florida”

print “Grand Canyon”

print “goodbye”

How old is the traveler?: 15
Suggestion: Grand Canyon.
Nested conditionals

statement 1;
if (condition 1)
    statement 2;
else
    if (condition 2)
        statement 4;
    else
        statement 3;
statement 5;

This slide adapted from Wayne&Sedgewick Princeton course:  

CSC 1051 M.A. Papalaskari, Villanova University
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.

Sample output

How old is the traveler?: 59
Suggestion: Florida.
Java Logical Operators

- Used with boolean operands to express more complex boolean conditions

  The logical NOT operation is also called logical negation or logical complement

<table>
<thead>
<tr>
<th>a</th>
<th>!a</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

  The logical AND operation is also called conjunction. Expression is true when both operands are true.

  The logical OR operation is also called disjunction. Expression is true one or both operands are true.

| a   | b   | a && b | a || b |
|-----|-----|--------|--------|
| true| true| true   | true   |
| true| false| false  | true   |
| false| true| false  | true   |
| false| false| false  | false  |
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.
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;
```

*boolean expression using a relational operator*
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 `||`.
if (found || !done)
    System.out.println("Ok");

What gets printed?

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

What changes?
Boolean Expressions

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

| found | done | !done | found || !done |
|-------|------|-------|--------|------|
| false | false |      |        |      |
| false | true  |      |        |      |
| true  | false |      |        |      |
| 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></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td></td>
<td></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!

```java
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

```java
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.