Conditionals

CSC 1051 – Data Structures and Algorithms I

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
www.csc.villanova.edu/~map/1051/
Last time: Lab2 - Algorithms

**Statement of GPA problem:**
Write a program that reads the credits and quality points earned and outputs the gpa.

**Algorithm:**
variables: qp, credits, gpa
1. Input qp
2. Input credits
3. gpa = qp / credits
4. Print gpa
Solution 5

**Algorithm:**
variables: qp, credits, gpa

1. Input qp
2. Input credits
3. gpa = qp / credits
4. Print gpa

```java
import java.util.Scanner;

public class GPA05 {
    public static void main(String[] args) {
        int qp;
        int credits;
        double 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();

        // calculate gpa
        gpa = (double) qp / credits;

        // output information
        System.out.println("Quality Points: " + qp);
        System.out.println("Credits: " + credits);
        System.out.println("GPA:  " + gpa);
    }
}
```
import java.util.Scanner;

public class GPA05
{
    public static void main (String[] args)
    {
        int qp;
        int credits;
        double 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();

        gpa = (double) qp / credits;

        // output information
        System.out.println("\nQuality Points: " + qp);
        System.out.println("Credits:        " + credits);
        System.out.println("GPA:          " + gpa);
    }
}

Algorithm:
variables: qp, credits, gpa
1. Input qp
2. Input credits
3. gpa = qp / credits
4. Print gpa

What if credits = 0????
Algorithm:
variables: qp, credits, gpa
1. Input qp
2. Input credits
3. if credits = 0
   a) Print “No gpa yet”
else
   a) gpa = qp / credits
   b) Print gpa
4. Print goodbye message
Algorithm:
variables: qp, credits, gpa
1. Input qp
2. Input credits
3. if credits = 0
   a) Print “No gpa yet”
else
   a) gpa = qp / credits
   b) Print gpa
4. Print goodbye message

```java
if (credits == 0)
    System.out.println("\n\tGPA:     None");
else
{
    gpa = (double) qp / credits;
    System.out.println("\n\tGPA:  " + gpa);
}
```
Order of statement execution

• Unless specified otherwise, the flow of control through a method is **linear**

• ie, **statements are executed in the order they appear**

• We can modify this using **conditional** and **repetition** statements
Conditional statements

- Conditional statements depart from linear flow of control:
- Example:

  ```java
  if (credits > 0)
  gpa = (double) qp / credits;
  ```
Topic Thread

- 2.1 Character Strings
- 2.2 Variables, Assignment
- 2.3 Data Types, in particular int, double
- 2.4 Expressions (simple)
- 2.5 Data Conversion
- 2.6 Interactive Programs
- 5.1 Boolean Expressions
- 5.2 The if Statement
- 5.5 The while Statement
5.1 Boolean Expressions

• A condition often uses one of Java's equality operators or relational operators, which all return 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 (=)
Boolean Expressions

• A boolean value represents a true or false condition

• The reserved words true and false are the only valid values for a boolean type

• Example boolean declaration:

  ```java
  boolean aboveAgeLimit = false;
  ```

• Another example:

  ```java
  boolean usePlural = quarters > 1;
  ```

A boolean expression using a relational operator
Boolean Expressions

• An if statement with its boolean condition:

        if (sum > MAX)
        delta = sum - MAX;

• First, the condition is evaluated: the value of sum is either greater than the value of MAX, or it is not

• If the condition is true, the assignment statement is executed; if it isn't, it is skipped

• See Age.java
import java.util.Scanner;

public class Age
{
    //---
    //  Reads the user's age and prints comments accordingly.
    //---
    public static void main (String[] args)
    {
        final int MINOR = 21;

        Scanner scan = new Scanner (System.in);

        System.out.print ("Enter your age: ");
        int age = scan.nextInt();

        continue
    }
}
continue

    System.out.println ("You entered: " + age);

    if (age < MINOR)
        System.out.println ("Youth is a wonderful thing. Enjoy.");

        System.out.println ("Age is a state of mind.");
    }
}
System.out.println("You entered: " + age);

if (age < MINOR)
    System.out.println("Youth is a wonderful thing. Enjoy.");

System.out.println("Age is a state of mind.");
}
}

Another Sample Run

Enter your age: 12
You entered: 12
Youth is a wonderful thing. Enjoy.
Age is a state of mind.
The if Statement

• Let's now look at the if statement in more detail
• The if statement has the following syntax:

```java
if ( condition )
statement;
```

The condition must be a boolean expression. It must evaluate to either true or false.

If the condition is true, the statement is executed. If it is false, the statement is skipped.

if is a Java reserved word
Logic of an if statement

condition evaluated

true

statement

false
Indentation

• The statement controlled by the `if` statement is indented to indicate that relationship

• The use of a consistent indentation style makes a program easier to read and understand

• The compiler ignores indentation
The if-else Statement

• An *else clause* can be added to an `if` statement to make an *if-else statement*

```java
if ( condition )
    statement1;
else
    statement2;
```

• If the *condition* is true, `statement1` is executed; if the condition is false, `statement2` is executed

• One or the other will be executed, but not both

• See `Wages.java`
import java.text.NumberFormat;
import java.util.Scanner;

public class Wages
{
    public static void main (String[] args)
    {
        final double RATE = 8.25;  // regular pay rate
        final int STANDARD = 40;   // standard hours in a work week

        Scanner scan = new Scanner (System.in);

        double pay = 0.0;
    }
}
System.out.print("Enter the number of hours worked: ");
int hours = scan.nextInt();

System.out.println();

// Pay overtime at "time and a half"
if (hours > STANDARD)
    pay = STANDARD * RATE + (hours-STANDARD) * (RATE * 1.5);
else
    pay = hours * RATE;

NumberFormat fmt = NumberFormat.getCurrencyInstance();
System.out.println("Gross earnings: " + fmt.format(pay));
System.out.print ("Enter the number of hours worked: ");
int hours = scan.nextInt();
System.out.println ();
// Pay overtime at "time and a half"
if (hours > STANDARD)
    pay = STANDARD * RATE + (hours-STANDARD) * (RATE * 1.5);
else
    pay = hours * RATE;

NumberFormat fmt = NumberFormat.getCurrencyInstance();
System.out.println ("Gross earnings: "+ fmt.format(pay));
Logic of an if-else statement

- The condition is evaluated.
- If the condition is true, execute statement1.
- If the condition is false, execute statement2.
Block Statements

• Several statements can be grouped together into a *block statement* delimited by braces

• A block statement can be used wherever a statement is called for in the Java syntax rules

```java
if (total > MAX)
{
    System.out.println("Error!!");
    errorCount++;  
}
```
Block Statements

• The **if** clause, or the **else** clause, or both, could govern block statements

```java
if (total > MAX)
{
    System.out.println ("Error!!");
    errorCount++;
}
else
{
    System.out.println ("Total: " + total);
    current = total*2;
}
```

• See **Guessing.java**
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
Nested if Statements

• The statement executed as a result of an if or else clause could be another if statement

• These are called nested if statements

• An else clause is matched to the last unmatched if (no matter what the indentation implies)

• Braces can be used to specify the if statement to which an else clause belongs

• See MinOfThree.java
import java.util.Scanner;

public class MinOfThree
{
    //-----------------------------------------------------------------
    // Reads three integers from the user and determines the smallest
    // value.
    //-----------------------------------------------------------------
    public static void main (String[] args)
    {
        int num1, num2, num3, min = 0;

        Scanner scan = new Scanner (System.in);

        System.out.println ("Enter three integers: ");
        num1 = scan.nextInt();
        num2 = scan.nextInt();
        num3 = scan.nextInt();

        continue
\begin{verbatim}
continue

    if (num1 < num2)
        if (num1 < num3)
            min = num1;
        else
            min = num3;
    else
        if (num2 < num3)
            min = num2;
        else
            min = num3;

    System.out.println ("Minimum value: " + min);
}
\end{verbatim}
continue

```java
if (num1 < num2)
    if (num1 < num3)
        min = num1;
    else
        min = num3;
else
    if (num2 < num3)
        min = num2;
    else
        min = num3;

System.out.println("Minimum value: " + min);
```

**Sample Run**

Enter three integers:  
84  69  90  
Minimum value: 69
Logical Operators

• Boolean expressions can also use the following \textit{logical operators}:

\begin{verbatim}
! Logical NOT
&& Logical AND
|| Logical OR
\end{verbatim}

• They all take boolean operands and produce boolean results

• Logical NOT is a unary operator (it operates on one operand)

• Logical AND and logical OR are binary operators (each operates on two operands)
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
Logical AND and Logical OR

• A truth table shows all possible true-false combinations of the terms

• Since && and || each have two operands, there are four possible combinations of conditions a and b

| a    | b    | a && b | a || b |
|------|------|--------|--------|
| true | true | true   | true   |
| true | false| false  | true   |
| false| true | false  | true   |
| false| false| false  | false  |
Logical Operators

• Expressions that use logical operators can form complex conditions

```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 `||`
Boolean Expressions

- Specific expressions can be evaluated using truth tables – let’s try this one:

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

What do the following statements do?

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

if (found || !done)
    System.out.println("Ok");
```
How much of a boolean expression do we need to be 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.");
```
Homework

• Read Sections 5.1, 5.2
  
  – **Always** do all self-review exercises when you review material

• Do Exercises EX 5.1 – 5.5