**Making Complicated Decisions**

Reference: Gaddis 3.5-6

Topics: nested if statements, complex boolean expressions, string comparison

For the (poorly formatted) code below, if temp == 60, what is the value of comfortLevel?

int comfortLevel = 0; Flowchart:

if (temp > 70)

if (temp > 90)

comfortLevel = -1;

else

comfortLevel = 1;

**What if I want to be able to make a more complicated decision?**

I want to output “yes” if both Bob and Mary are taking at least 16 credits, and “no” otherwise.

*Algorithm:*

1. Ask Bob how many credits he’s taking, store in bob\_credits
2. Ask Mary how many credits she’s taking, store in mary\_credits
3. If

*Code?*

//Declare variables

Scanner scan = new Scanner(System.in);

int bob\_credits, mary\_credits;

//read in values

bob\_credits =

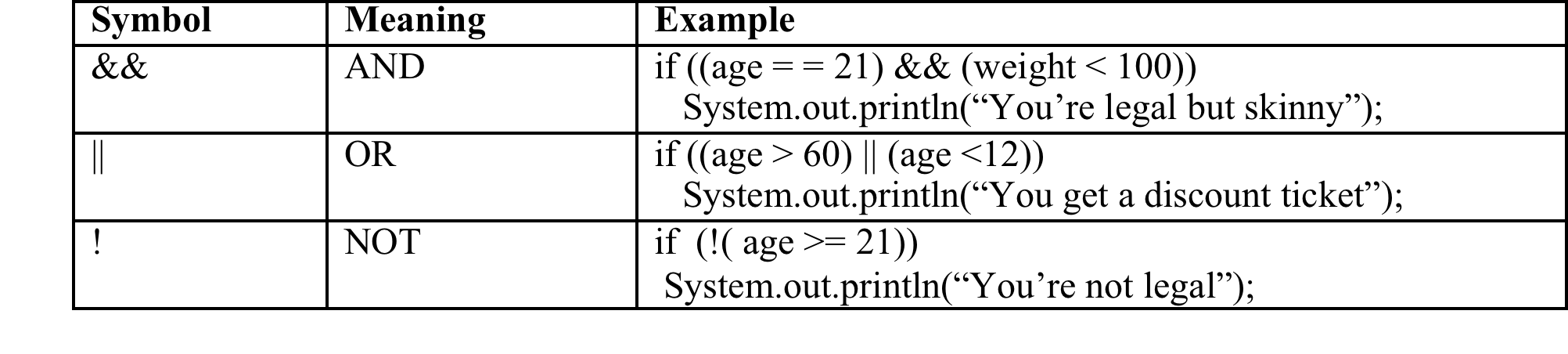
mary\_credits =

//output yes if both credits are at least 16, output no otherwise

if(

Can I do it all in one if-statement??

Connect boolean expressions using logical operators:



* Is 7 > 5 && 2 < 1 true or false??
* Is 7 > 5 || 2 < 1 true or false??
* Is ! ( 3 > 5) true or false??
* **Order of Operations** including the relational and logical operators: see **table 3-10** on p. 142

//Declare variables

Scanner scan = new Scanner(System.in);

int bob\_credits, mary\_credits;

//read in values

bob\_credits =

mary\_credits =

//output yes if both credits are at least 16, output no otherwise

if(

*Another Example:*

Let’s say I want to output “valid entry” only if my variable “age” is between 0 and 120:

Let’s say I want to output “Valid entry” only if my variable named “value” is less than 10 or between 90 and 100:

**Comparing Strings**

What if I want to know if two people have the same name?

*String name1 = “Nicky”;*

*String name2 = “Nicky”;*

*if (name1 == name2)*

*System.out.println (“they are equal”);*

*else*

*System.out.println(“they are not equal”);*

What is output?

Why?

**for reference variables (**i.e., objects**), = = compares**

There is an equals method for String objects:

*if (name1.equals(name2) )*

*System.out.println (“same strings”);*

* And there's another method that gives more info:

*name1.compareTo(name2)* returns:

* + 0 if s1 and s2 are the same string
  + a negative number if s1 precedes s2 in lexicographic order
  + a positive number if s1 comes after s2 in lexicographic order

*Lexicographic Order*

* lexicographic order uses the Unicode (same as ASCII but in 16 bits) codes to make decisions

Assume s1 and s2 are String objects that have been initialized to the listed value:

|  |  |  |
| --- | --- | --- |
| **s1** | **s2** | **s1.compareTo(s2)** |
| “cat” | “cat” |  |
| “bat” | “cat” |  |
| “cat” | “cast” |  |
| “cat” | “CAT” |  |
| “my cat” | “mycat” |  |

*Writing as an if-statement*

* To check if name1 and name2 are equal, write:

if(

* To check if name1 comes before name2, write:

if(

* To check if name 1 comes after name2, write:

if(

*Ignoring Case*

What if I want to compare and have “cat” be the same as “CAT”?

2 ways I can ignore case when comparing:

1. Convert to all one case:

2. Use the *equalsIgnoreCase(String)* method