Flow Control

Boolean expressions
& the `if` statement

Boolean expressions

- Boolean expressions are made up of relational and logical operators.
- The result of the evaluation of a Boolean expression is true or false.
- We can use this Boolean result to tell the computer whether we want to execute a set of instructions or not.
- This process alters the flow of the program during runtime.

Relational Operators

- Relational operations are binary operations that require two primitive data values and evaluate to a **`boolean`** result.
  - < is less than
  - > is greater than
  - <= is less than or equal to
  - >= is greater than or equal to
  - == is equal to
  - != is not equal to
- NOTE: == is not the same as = in Java!

Relational Operators

- Examples:

```
int x = 15;
int y = 100;
System.out.println(x >= y);
System.out.println(x == y);
System.out.println(x != y);
boolean z = (x - y < 0);
```

Logical Operators

- Logical operations are operations that require **`boolean`** values and evaluate to a **`boolean`** result.
  - `&&` logical AND
  - `||` logical OR
  - Example: Let `a` and `b` be `boolean` variables.

a	b	a && b	a || b
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Logical Operators

- Examples:

```
int x = 15;
int y = 100;
boolean a = (x > y);
boolean b = (x != y);
System.out.println(a && b);
System.out.println(a || b);
```

Logical Operators

- Examples:

  Assume the following variables are initialized:

```
int snowFall;              // in inches
boolean belowZero;         // 0 in Fahrenheit
```

  School is closed if at least one of these conditions occurs:
  - at least 12 inches of snow fall
  - temperatures are below 0 degrees (F)

```
boolean closed =

  ________________________________________;
```

Logical Operators

- Another logical operator is the **!** (NOT)

a	!a
false	true
true	false

- Example (assume **iq** is an **int** variable):

```
System.out.println("GENIUS: " + iq > 135));
```

  is the same as

```
System.out.println("GENIUS: " + !(__________));
```

DeMorgan's Law

!(A && B) == !A || !B

a	b	!(a && b)	!a || !b
false	false		
false	true		
true	false		
true	true		

A && B == !(!A || !B)

DeMorgan's Law

!(A || B) == !A && !B

a	b	!(a || b)	!a && !b
false	false		
false	true		
true	false		
true	true		

A || B == !(!A && !B)

The `if` statement

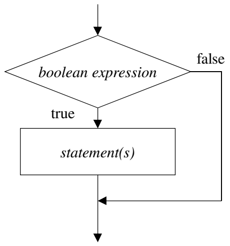
- The if statement is used to alter the flow of control in a program between two choices.
- Forms:

if (*boolean_expression*)
    statement **;**

if (*boolean_expression*)
{
    statement_list
}

If the given boolean expression is true, then the statement or statement list is executed. Otherwise, the statement or statement list is skipped.

The `if` flowchart



The `if` statement

- Example: Let **`age`** be an **`int`** variable and let **`male`** and **`citizenOfUSA`** be **`boolean`** variables.

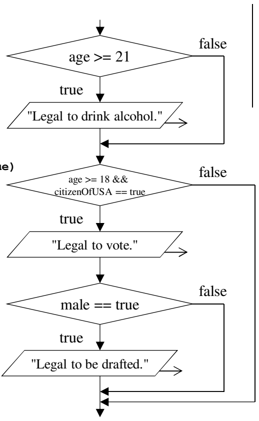
```
if ( age >= 21 )
    System.out.println("Legal to drink alcohol.");

if ( age >= 18 && citizenOfUSA == true)
{
    System.out.println("Legal to vote.");
    if (male == true)
          System.out.println("Legal to be drafted.");
}
```

We could also write: `if (male)`

Flowchart Example

```
if ( age >= 21 )
   System.out.println(
         "Legal to drink alcohol.");
if ( age >= 18 && citizenOfUSA == true)
{
   System.out.println(
                  "Legal to vote.");
   if (male == true)
        System.out.println(
             "Legal to be drafted.");
}
```



Exercise

age	citizenOfUSA	male	Legal to...
			(no output)
			drink
			vote
			be drafted

Exercise

age	citizenOfUSA	male	legal to...
			drink, vote
			drink, be drafted
			vote, be drafted
			drink, vote, be drafted

The `if` statement

- Is this code fragment the same logically as the previous example?

```
if ( age >= 18 && citizenOfUSA == true )
{
    System.out.println("Legal to vote.");
    if ( age >= 21 )
    {
       System.out.println("Legal to drink alcohol.");
       if (male == true)
          System.out.println("Legal to be drafted.");
    }
}
```

Using DeMorgan's Law

Can we write the 2nd `if` condition as:

```
if (!(age < 18 || citizenOfUSA == false))
```

A = (age < 18)

B = (citizenOfUSA == false)

!(A || B) == !A && !B

```
if (age >= 18 && citizenOfUSA == true)
```

The `if-else` statement

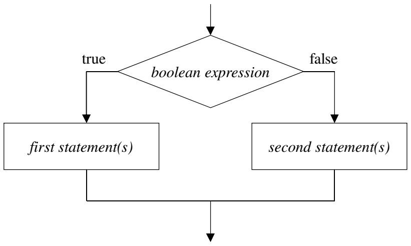
- Forms:

```
if ( boolean_expression )
    statement1 ;
else
    statement2 ;

if ( boolean_expression ) {
    statement_list1
}
else {
    statement_list2
}
```

If the given boolean expression is true, then the first statement or first statement list is executed. Otherwise, the second statement or second statement list is executed.

The `if-else` flowchart



The `if-else` statement

- Example: Let **`name`** be an **`String`** variable. We wish to print out at most the first ten characters of the name.

```
if ( name.length() > 10 )
    System.out.println(name.substring(0,10));
else
    System.out.println(name);
```

The `if-else` statement

- What is logically wrong with the following code fragment?

```
if ( age >= 65 )
    System.out.println("SENIOR CITIZEN");
else
    System.out.println("NOT A SENIOR CITIZEN");
    System.out.println("PAY THE REGULAR RATE");
```

Caution #1

- If you want to test if a variable has one of several values:
- WRONG:

```
int total = dieValue1 + dieValue2;
if ( total == 7 || 11 )
    System.out.println("YOU WIN");
```

- RIGHT:

```
int total = dieValue1 + dieValue2;
if ( total == 7 || total == 11 )
    System.out.println("YOU WIN");
```

Caution #1 (cont'd)

- If you want to test if a variable has a value within a certain range:
- WRONG:

```
int total = dieValue1 + dieValue2;
if ( 5 <= total <= 9 )
    System.out.println("YOU WIN");
```

- RIGHT:

```
int total = dieValue1 + dieValue2;
if ( total >= 5 && total <= 9 )
    System.out.println("YOU WIN");
```

Caution #2

- If you have only one **else** and two **if** statements, the **else** is paired with the nearest **if** unless you enclose the second **if** in brackets.

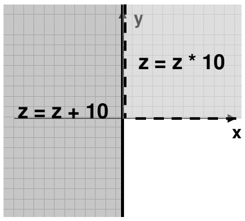
```
z = 10;
if (x > 0)
    if (y > 0)
          z = z * 10;
else
    z = z + 10;
```

This else is paired with this if, even though the indentation seems to indicate otherwise!

Caution #2 (cont'd)

- One way to write it:

```
z = 10;
if (x > 0) {
      if (y > 0)
            z = z * 10;
} else
      z = z + 10;
```

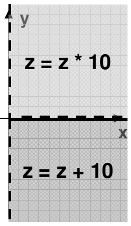


- Draw a flowchart for this code fragment.

Caution #2 (cont'd)

- Another way to write it:

```
z = 10;
if (x > 0)
      if (y > 0)
            z = z * 10;
      else
            z = z + 10;
```



- Draw a flowchart for this code fragment.

What's wrong?

- A company add a shipping charge based on the price of an item. If the item is less than $1000, the shipping charge is $50. If the item is greater than or equal to $1000, the shipping charge is $100.
  - Assume **price** is an **int** variable initialized to the price of the item (positive).

```
if (price < 1000)
   price += 50;
if (price >= 1000)
   price += 100;
```

What's wrong?

- Output a message based on the temperature reading.
  - Assume **temp** is an **int** variable initialized to the temperature.

```
if (temp <= 30)
    System.out.println("COLD");
if (temp > 30 || temp <= 50)
    System.out.println("COOL");
if (temp > 50 || temp <= 70)
    System.out.println("MILD");
if (temp > 70 || temp <= 90)
    System.out.println("WARM");
else
    System.out.println("HOT!");
```

Exercise

final indicates a variable that cannot change value (a constant)

```
final double PAY_RATE = 7.50;
Scanner scan = new Scanner(System.in);
System.out.println("Please input hours worked: ");
int hoursWorked = scan.nextInt();
double totalPay;
```

Compute `totalPay` here.

An employee receives the pay rate for each hour worked up to 40 hours, and "time-and-a-half" (1.5 × the pay rate) for all hours worked beyond 40 hours.

```
System.out.println("Your total pay is $" + totalPay);
```

Exercise Answer

Testing for equality

- For primitive values, use `==` to test for equality.
- For objects, use the **`equals`** method to test for equality.
  - The **`equals`** method requires an object of the same type as a parameter and returns true or false depending if the object running the **`equals`** method "is equal to" the object given as the parameter.
- Example: Let **`courseNum`** be an **`int`** variable and let **`instructor`** be a **`String`** variable.

```
if (courseNum == 15100 && instructor.equals("Cortina"))
    System.out.println("Major fun in large doses!");
```

For the `String` class, two strings are equal using the `equals` method if they have the same character sequence <u>exactly</u>.

Short-circuit evaluation

- In a logical expression involving **`&&`**, if the first argument is false, the second argument is not evaluated in Java.
  - This is called short-circuit evaluation (or lazy evaluation)
- Example:

```
if (numScores != 0 && sum/numScores > 70.0)
    System.out.println("The class did very well!");
```

- Short-circuit evaluation can be used with **`||`** also.
  - How is it different here?

More about short-circuit evaluation
later in the semester...

`switch` statement

- If you are specifying what to execute based on one of many values in an **`int`** or **`char`** variable or expression, you can use a **`switch`** statement instead of a long sequence of if-else statements.
- Example:

```
switch (rating)
    case 'A': System.out.println("Superb"); break;
    case 'B': System.out.println("Average"); break;
    case 'C': System.out.println("Fair"); break;
    default: System.out.println("Poor");
}
```