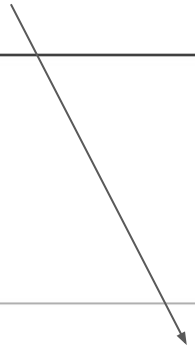
10/3/2022

g) y is greater in magnitude than z.	



Numbers [edit]

Main article: Absolute value

The magnitude of any number x is usually called its ***absolute value*** or *modulus*, denoted by $|x|$.[3]

Real numbers [edit]

The absolute value of a real number *r* is defined by:[4]

$$|r| = r, \text{ if } r \geq 0$$
$$|r| = -r, \text{ if } r < 0.$$

Absolute value may also be thought of as the number's distance from zero on the real number line. For example, the absolute value of both 70 and −70 is 70.

3.1

Boolean Expressions

Can you identify all of the

- Expressions
  - Which are Boolean Expressions?
- Statements
- Blocks
- Formal parameters
- Actual parameters

```java
import java.util.Scanner;
class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int value = scanner.nextInt();
    if (value < 0) {
      System.out.println("The number is negative.");
    } else if (value > 0) {
      System.out.println("The number is positive.");
    } else {
      System.out.println("The number is zero.");
    }
  }
}
```

Equality Operators

x == y	true if x is equal to y, false otherwise
x != y	true if x is not equal to y, false otherwise

Keep in mind that you must use "==", not "=", when testing if two values are equal.

= is the assignment operator and will assign the value of y to x!

Relational Operators

x > y	true if x is greater than y, false otherwise
x < y	true if x is less than y, false otherwise
x >= y	true if x is greater than or equal to y, false otherwise
x <= y	true if x is less than or equal to y, false otherwise

Tip: To remember >= and <=, think of it as the order in which you say it. Greater than (>) or equal to (=)

Relational Operators

x > y	true if x is greater than y, false otherwise
x < y	true if x is less than y, false otherwise
x >= y	true if x is greater than or equal to y, false otherwise
x <= y	true if x is less than or equal to y, false otherwise

These don't work on objects, such as Strings! This is where you use the String.compareTo method.
(compareTo isn't part of Object like equals - it comes from the Comparable interface.)

Are ==, != relational operators?

CS Awesome talks about == and != as relational operators. Are they?

Yes and no. Wikipedia does define equality operators as being relational operators.

But Java does define equality operators to have lower precedence than the other relational operators, so they behave slightly differently.

x < 2 == y > 5 is (x < 2) == (y > 5)

Operator Precedence

Operators	Precedence
postfix	*expr++ expr--*
unary	*++expr --expr +expr -expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Modulo Operator (%)

x % y	returns the remainder of dividing x by y

This is NOT a boolean operator, but it's a handy one frequently used with operators like ==. % works on integers but also on floats/doubles in Java.

One use is checking whether a number is even or odd.
  x % 2 == 0 means x is even
  x % 2 == 1 means x is odd

You can use it to check if a number is an even multiple of another:
  x % y == 0 means x is a multiple of y
  (can be divided by y with a remainder of 0)

3.2, 3.3
if-else statements

if syntax

`if` (*boolean expression*)
    then-statement

Example:

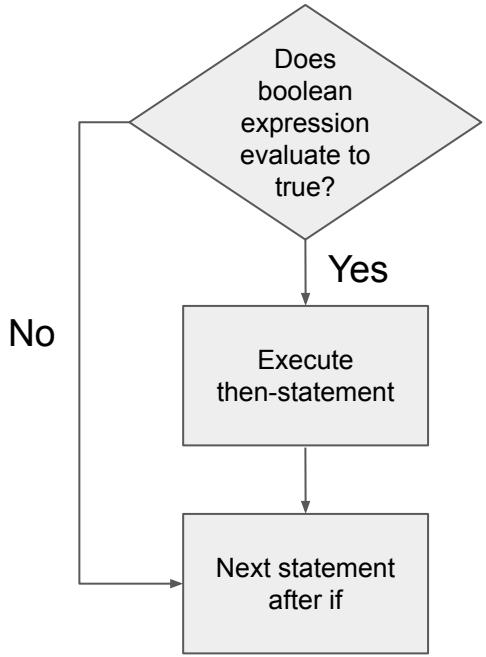
```
if (age >= 18) {
    System.out.println("You are eligible to vote!");
}
```

Legal, but not recommended:

```
if (age >= 18)
    System.out.println("You are eligible to vote!");
```

then-statement can be any statement, and a { block } is a statement.
It's recommended to always use blocks with if.

Flowchart of if



if-else syntax

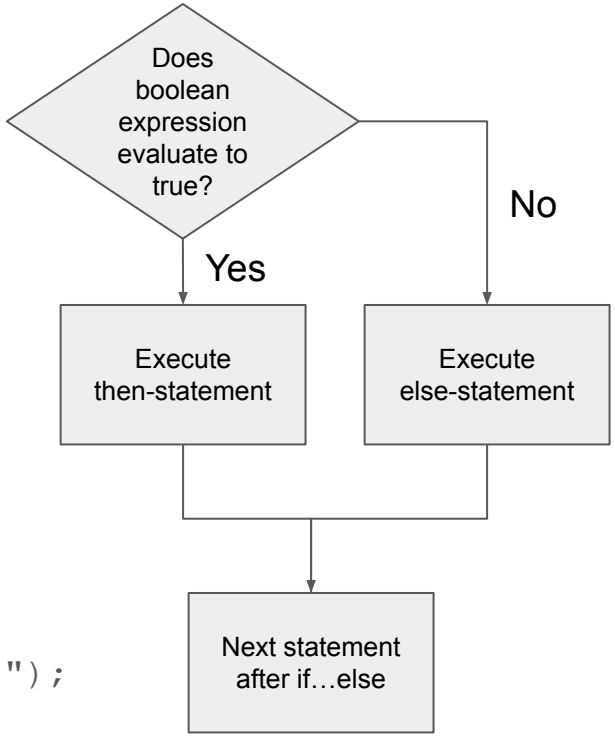
```
if (boolean expression)
    then-statement
else
    else-statement
```

The if statement has an optional else clause.

Example:

```
if (age >= 18) {
  System.out.println("You are eligible to vote!");
} else {
  System.out.println("You are too young to vote!");
}
```

Flowchart of if…else



Nested If-Statements

```
if (boolean expression) {
    if (boolean expression) {
        if (boolean expression) {
            <statement>;
            <statement>;
            …
        } else {
            <statement>;
            <statement>;
            …
        }
    }
} else {
    <statement>;
    <statement>;
    …
}
```

```
private void doRoomSpecificActions() {
  if (player.getLocation() == missionRoad) {
    if (Math.random() < 0.1) {
      // 10% probability of a car almost hitting you
      System.out.println();
      System.out.println("Careful! A speeding car almost hit you!");
      System.out.println("Maybe it's best to get out of the middle of the street!");
    }
  }
}
```

Dangling Else

```
int x = 0;
if (x >= 0)
    if (x > 0)
        System.out.println("x is positive");
else
    System.out.println("x is negative");
```

is paired with this **if**

this **else**

Prints "x is negative"!

The else clause will always be a part of the closest if statement if in the same block of code regardless of indentation…

Unless you use {}!

3.4
Multi-Selection : else-if

else-if syntax

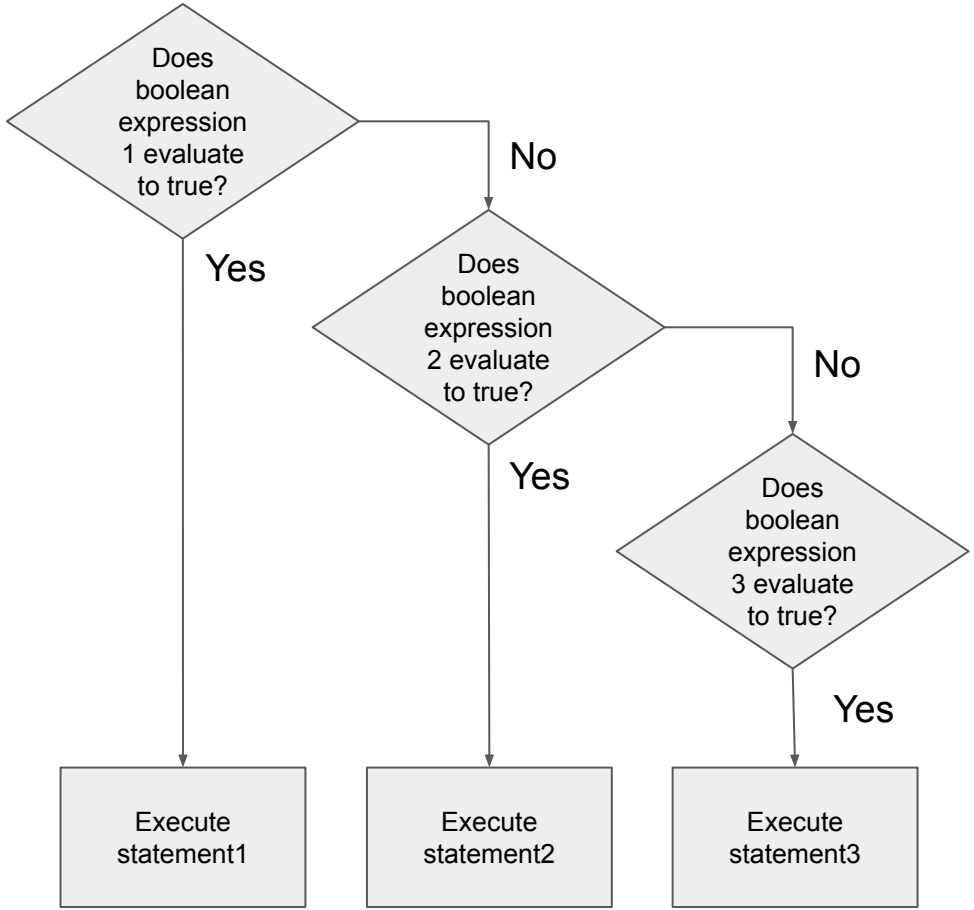
```
if (boolean expression 1)
    statement1
else if (boolean expression 2)
    statement2
else if (boolean expression 3)
    statement3
```

```java
public static String weekdayName(int weekDay) {
  if (weekDay == 0) {
    return "Sunday";
  } else if (weekDay == 1) {
    return "Monday";
  } else if (weekDay == 2) {
    return "Tuesday";
  } else if (weekDay == 3) {
    return "Wednesday";
  } else if (weekDay == 4) {
    return "Thursday";
  } else if (weekDay == 5) {
    return "Friday";
  } else if (weekDay == 6) {
    return "Saturday";
  } else {
    return "INVALID";
  }
}
```

else-if syntax

```
if (boolean expression 1)
    statement1
else if (boolean expression 2)
    statement2
else if (boolean expression 3)
    statement3
```

Flowchart of else-if



3.5

Compound Boolean Expressions

Logical Operators

Logical And

p `&&` ***q***

Evaluates boolean expressions ***p*** and ***q***.

Evaluates to `true` if ***p*** and ***q*** are both `true`, `false` otherwise.

```
if (sunny && warm) {
     …
}
```

Logical Or

p `||` ***q***

Evaluates boolean expressions ***x*** and ***y***.

Evaluates to `true` if ***p*** or ***q*** are `true`, `false` otherwise.

```
if (christmas || halloween)
{
     …
}
```

Logical Not

`!`***p***

Evaluates boolean expression ***p***.

Evaluates to `true` if ***p*** is `false`.

Evaluates to `false` if ***p*** is `true`.

```
if (!day.equals("Sunday"))
{
     …
}
```

Why p and q? In logic textbooks, the "default" names for logical propositions are p and q.

Truth Table - &&

p	q	p && q
true	true	true
true	false	false
false	true	false
false	false	false

Truth Table - ||

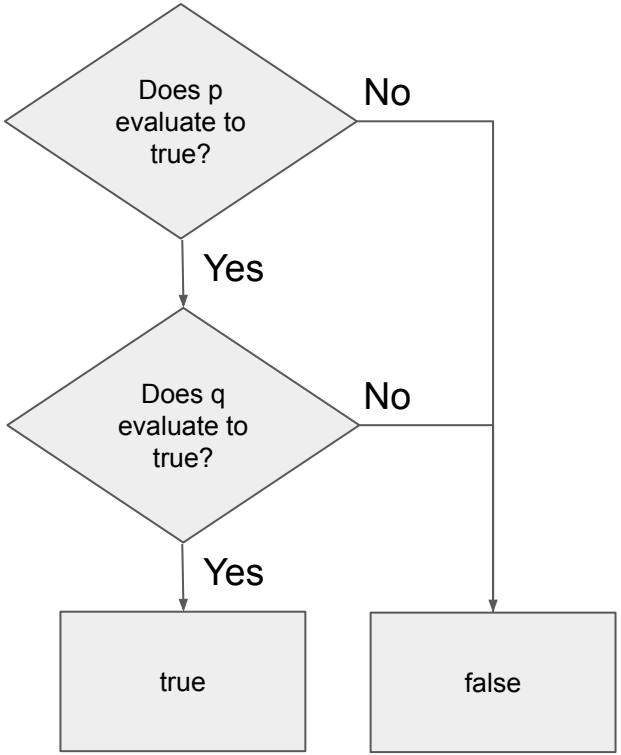
p	q	p || q
true	true	true
true	false	true
false	true	true
false	false	false

Truth Table - !

p	!p
true	false
false	true

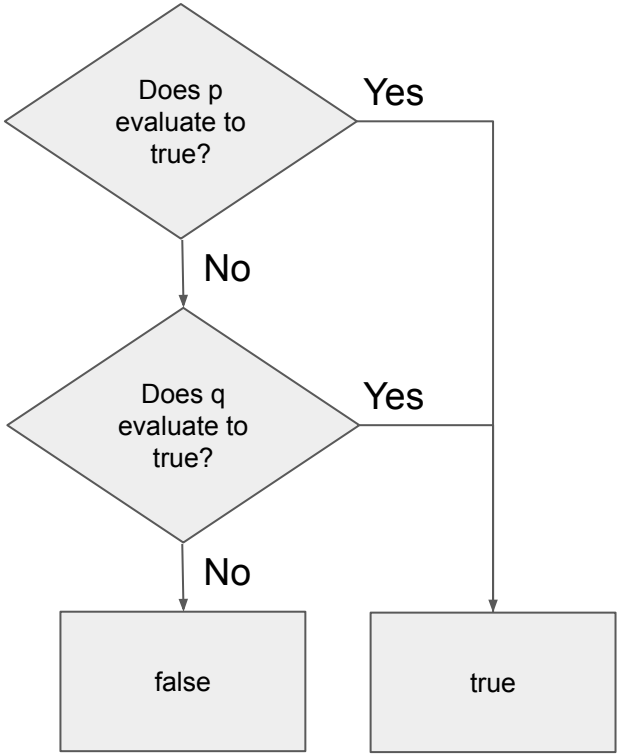
Short-Circuit Evaluation with &&

```
p && q
```



Short-Circuit Evaluation with ||

p || q



3.6
Equivalent Boolean Expressions

DeMorgan's Laws

These rules can simplify boolean expressions to make them easier to read or interpret.

- `!(a && b)` is equivalent to `!a || !b`
- `!(a || b)` is equivalent to `!a && !b`

De Morgan's Laws: Example

YOU: I'm leaving for school!

MOM: Let's make sure you have everything.

MOM: Do you have both your phone and your lunch?

(is phone && lunch true?)

YOU: No. !(phone && lunch) is true

MOM: What are you missing, your phone? Your lunch? Both?

!phone || !lunch is true

De Morgan's Laws: Truth Tables

```
!(phone && lunch) = !phone || !lunch
```

`phone`	`lunch`	`!(phone && lunch)`	`!phone || !lunch`
F	F	!(F && F) → !F → **T**	!F || !F → T || T → **T**
F	T	!(F && T) → !F → **T**	!F || !T → T || F → **T**
T	F	!(T && F) → !F → **T**	!T || !F → F || T → **T**
T	T	!(T && T) → !T → **F**	!T || !T → F || F → **F**

De Morgan's Laws: Example

`!(a && b) = !a || !b`

You must do pull-ups AND run a mile to complete the fitness test.

```
    if (!(completedPullUps && completedOneMileRun)) {
        System.out.println("You are not yet done with your
fitness test!");
    }
```

is the same as

```
    if (!completedPullUps || !completedOneMileRun) {
        System.out.println("You are not yet done with your
fitness test!");
    }
```

De Morgan's Laws: Example

`!(a && b) = !a || !b`

You must sign a waiver AND show your ID at the front desk to come in to the gym.

```
    if (!(signedWaiver && shownID)) {
        System.out.println("You must sign a waiver and show ID
to enter gym.");
    }
```

is the same as

```
    if (!signedWaiver || !shownID) {
        System.out.println("You must sign a waiver and show ID
to enter gym.");
    }
```

De Morgan's Laws: Example

!(a || b) = !a && !b

You must show a paper vaccine card or vaccination proof on your phone, to enter.

```
    if (!(hasVaccinePaperCard || hasVaccineProofOnPhone)) {
        System.out.println("ERROR: Proof of vaccination
required.");
    }
```

is the same as

```
    if (!hasVaccinePaperCard && !hasVaccineProofOnPhone) {
        System.out.println("ERROR: Proof of vaccination
required.");
    }
```

De Morgan's Laws: Example

`!(a || b) = !a && !b`

You can come to our concerts if you haven't jumped on stage or yelled really loud.

```
if (!(hasYelledLoud || hasJumpedOnStage)) {
    System.out.println("You may enter the concert venue");
}
```

is the same as

```
if (!hasYelledLoud && !hasJumpedOnStage) {
    System.out.println("You may enter the concert venue");
}
```

De Morgan's Laws: Truth Tables

```
!(a && b) = !a || !b
```

`a`	`b`	`!(a && b)`	`!a || !b`
F	F	!(F && F) → !F → **T**	!F || !F → T || T → **T**
F	T	!(F && T) → !F → **T**	!F || !T → T || F → **T**
T	F	!(T && F) → !F → **T**	!T || !F → F || T → **T**
T	T	!(T && T) → !T → **F**	!T || !T → F || F → **F**

De Morgan's Laws: Truth Tables

`!(a || b) = !a && !b`

`a`	`b`	`!(a || b)`	`!a && !b`
F	F	!(F || F) → !F → **T**	!F && !F → T && T → **T**
F	T	!(F || T) → !T → **F**	!F && !T → T && F → **F**
T	F	!(T || F) → !T → **F**	!T && !F → F && T → **F**
T	T	!(T || T) → !T → **F**	!T && !T → F && F → **F**

Negated Relational Expressions

For negated relational expressions, **you can flip the operator and remove the !**

- !(c == d) is equivalent to (c != d)
- !(c != d) is equivalent to (c == d)
- !(c < d) is equivalent to (c >= d)
- !(c > d) is equivalent to (c <= d)
- !(c <= d) is equivalent to (c > d)
- !(c >= d) is equivalent to (c < d)

Example:
Simplify **!(x > 2 && y < 4)**

Apply !(a && b) == !a || !b
!(x > 2) || !(y < 4)

Apply !(c > d) == (c <= d)
(x <= 2) || !(y < 4)

Apply !(c < d) == (c >= d)
x <= 2 || y >= 4

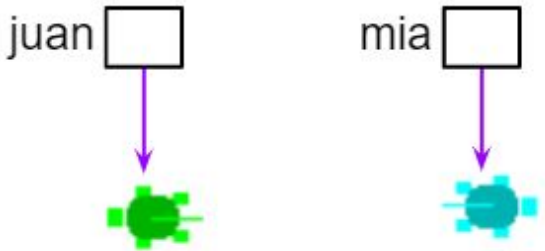
3.7
Object Equality

Object Equality

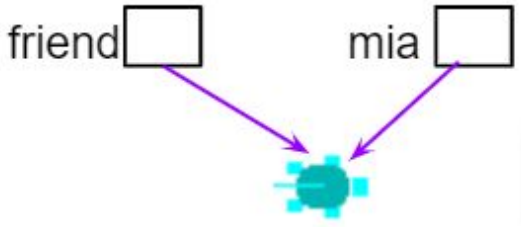
For object types, == and != compare whether the two sides are references to the same object… not whether anything else about the objects are equal, such as the characters in two Strings.

For Strings, remember to use equals() and not the == or != operators.

```
Turtle juan = new Turtle(world);
Turtle mia= new Turtle(world);
```




```
Turtle friend = mia;
```

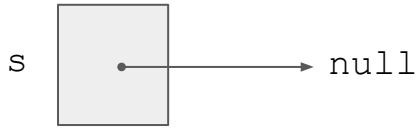


null means nothing is pointed to

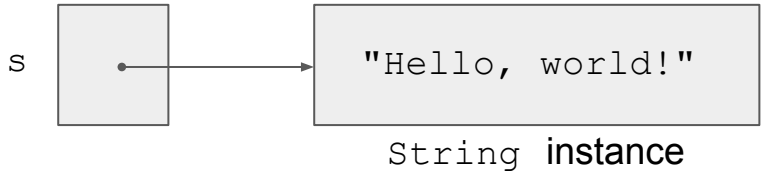
`String s;` is a reference variable of type `String`.

A `String` reference, like all object references, either points to a `String` object instance, or is `null`

```
String s;
```



`s = "Hello, world!";` ← `s` started out `null`, but now points to a legit string



NullPointerException

If you try usings methods or attributes of a `null` object reference, Java throws `NullPointerException`:

```
String s;

if (s.indexOf("a") >= 0) {

    System.out.println(s + " contains an a"); ←throws NullPointerException

}
```

Some exceptions are not bugs in your program, like an exception thrown if the network is down.

But a `NullPointerException` usually means you need to fix your code.

Practice!

Repl.it: deMorgan

```
> sh -c javac -classpath .:target/dependency/* -d . $(find . -type f -name '*.java')
> java -classpath .:target/dependency/* Main
Generated this HTML:
<style type="text/css">
  table { width: 80%; }
  table, th, td { padding: 8px; border: 1px solid; }
</style>
<h2>DeMorgan's Laws</h2>
<h3>!(p && q) <=> !p || !q</h3>
<table>
  <tr>
    <th>p</th>
    <th>q</th>
  </tr>
  <tr>
    <td>false</td>
    <td>false</td>
  </tr>
</table>
<h3>!(p || q) <=> !p && !q</h3>
<table>
  <tr>
    <th>p</th>
    <th>q</th>
  </tr>
  <tr>
    <td>false</td>
    <td>false</td>
  </tr>
</table>
```