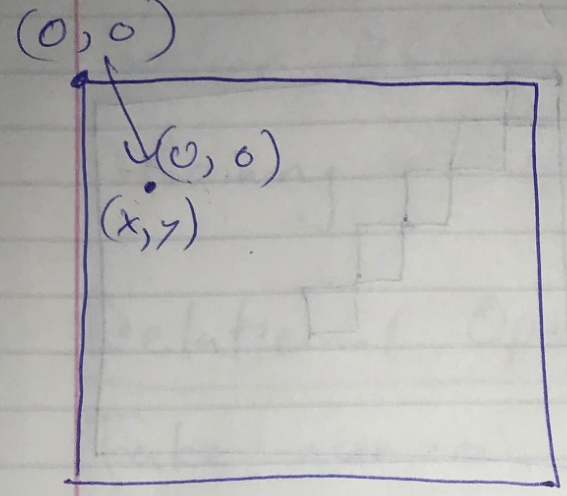
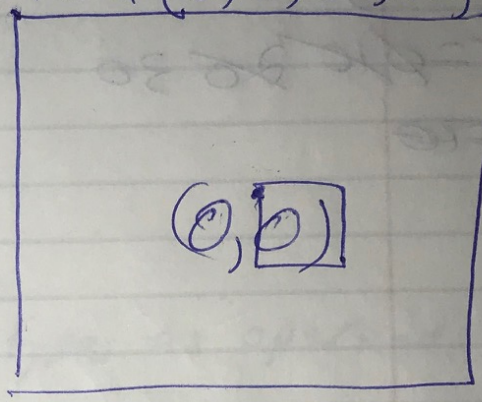
translate(x, y);

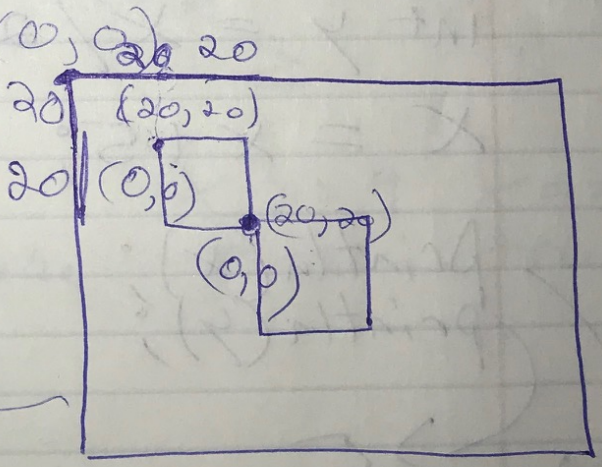


translate(50,50);
rect(0,0,20,20);

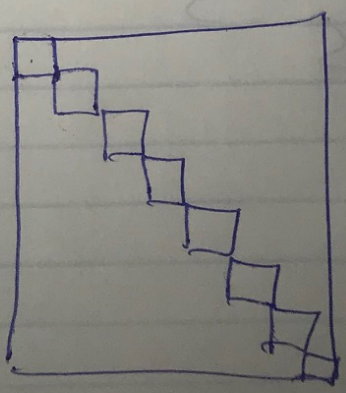


~~translate~~ translate is reset to default at the beginning of draw.

```
translate(20,20);
rect(0,0,20,20);
translate(20,20);
rect(0,0,20,20);
```

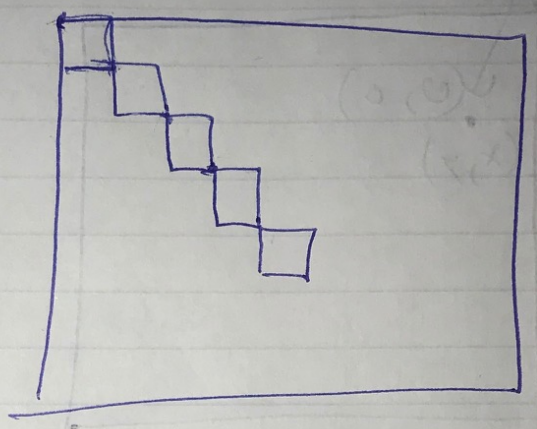



```
Void draw(){
    translate(10,10);
    rect(0,0,10,10);
}
```



```
int x = 0;
Void draw() {
    translate(x, x, 10, 10);
    rect(0, 0, 10, 10);
    x = x + 10;
}
```

x = ~~0~~ ~~10~~ ~~20~~ 30



```
int x = 10;
int y = x / 2;
x = x + 5;

println(x);
println(y);
```

x	y
~~10~~	5
15	

>5

>15

boolean data-type

true or false

```
boolean isRaining = true;
isRaining = false;
```

Relational Operators

Take numeric data-types as operands,
result in boolean data-type.

<, >, <=, >=, ==, !=

≥

```
int x = 9;
```

x < 8 → false

x > 9 → ~~true~~ false

x >= 9 → true

x == 9 → true

x != 10 → true

x != 9 → false

Relational Operators

>	greater than
<	less than
>=	greater than or equal
<=	less than or equal
==	equal
!=	not equal

```
int x = 5, y = 10;
```

true `x < y`

false `x < y/2`

true `x != y`

true `x == y/2`

false `x+y == y`

true `x+y != y`

true `x/2 <= y`

Boolean Operators

2 boolean operands, result is boolean

&& - AND
|| - OR

A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

A	B	A || B
true	true	true
true	false	true
false	true	true
false	false	false

| - shift + backslash

A	!A
true	false
false	true

&& AND
|| OR
! NOT

De Morgan's Law

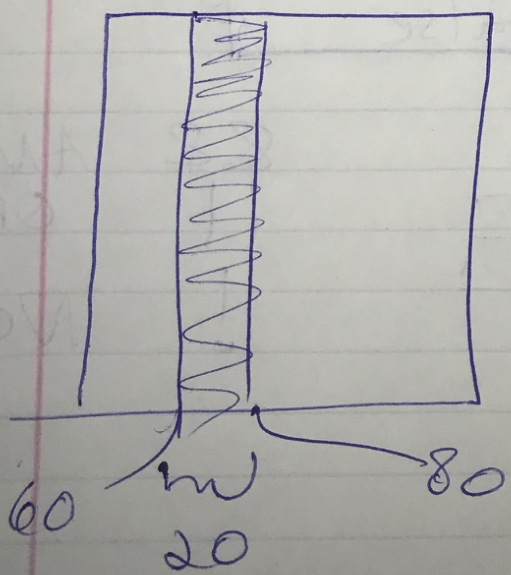
~~!(A && B) = !A || !B~~

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

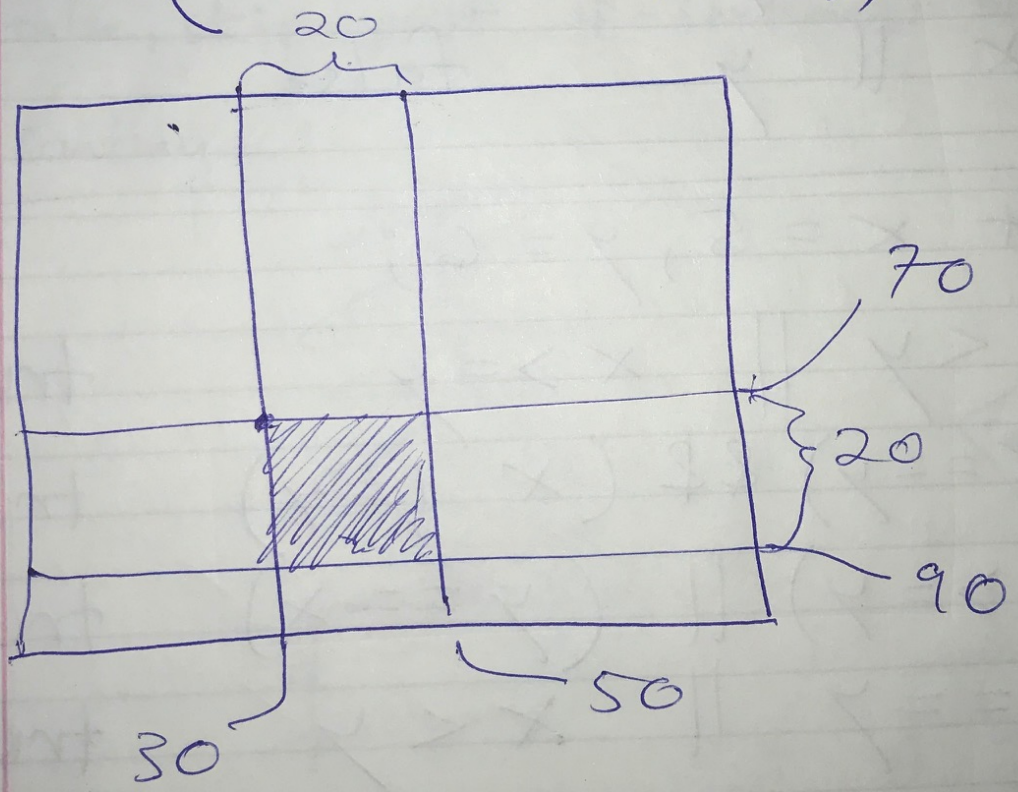
!(A || B) = !A && !B

A	B	!(A && B)	!A || !B
true	true	false	false
true	false	true	true
false	true	true	true
false	false	true	true

(mouseX < 80) && (mouseX > 60)



```
boolean inRect =
((mouseX < 50) && (mouseX > 30))
&&
((mouseY < 90) && (mouseY > 70))

rect(30, 70, 20, 20);
```



```
boolean x = true;
boolean y = false;
```

`!(x || !y)`	false
`x && !y`	true
`!x || !y`	true
`x && y`	false
`!x || y`	false

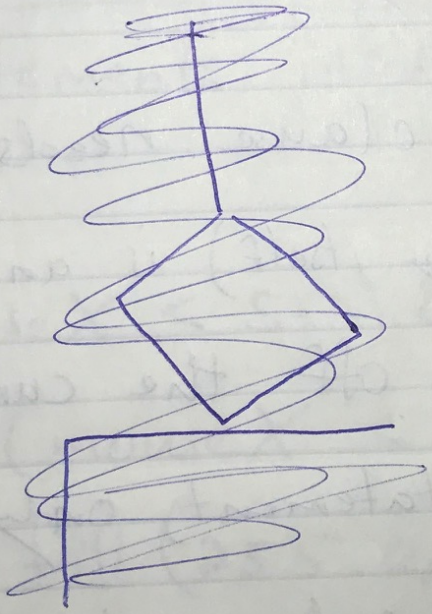
```
int x = 5, y = 6;
```

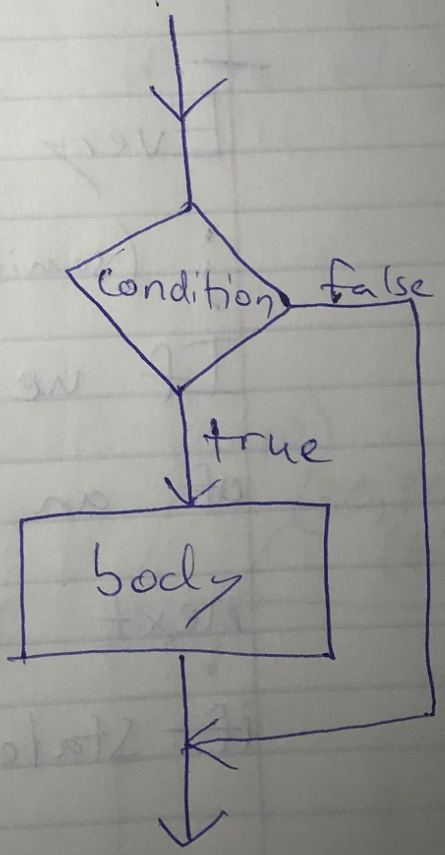
`x<y || x>=y`	true
`!(x==y) && (x!=y)`	true
`!(x>=y) || (y==x)`	true
`x==y || x<y`	true
`x!=y && x==y`	false

boolean value

```
if (condition) {
    // body
}
```

if condition is true, execute all statements in body. If condition is false, skip past if-statement block and continue.





```
if (condition) {
    // if body
}
else {
    // else body
}
```

*We will never execute both if and else body. One or the other.

—

Every else clause needs an if.

; (semicolon by itself) is an empty statement

If we leave off the curly braces of an if statement, only the very next line of code applies to the if if-statement.

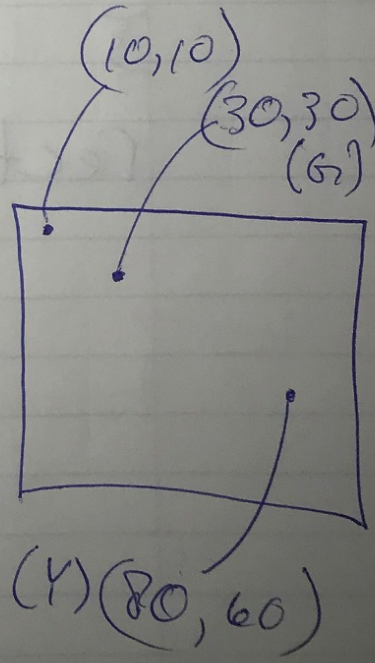
```
if (condition){

}else {
}
```

no lines of code can go b/w else and if

Nested if-statements

if statements inside other if (or else) blocks.

```
if(mouseX < 50) {
    if (mouseX < 20) {
        fill (255,0,0);
    }
    else {
        fill (0,255,0);
    }
}
else {
    fill (255, 255, 0);
}
```



(80,30)
Y

(50,50)
C

(20,70)
P

(90,90)
R

```
if(mouseY > 50){
    if(mouseX >50){
        fill(255,0,0);
    }
    else{
        fill(255,0,255);
    }
}
else {
    if(mouseX >50) {
        fill(255,255,0);
    }
    else{
        fill(0, 255,255);
    }
}
rect(50,50,50,50);
```

```
int grade = 75; 85
if (grade >= 90) {
    println("A");
}
else if (grade >= 80) {
    println("B");
}
else if (grade >= 70) {
    println("C");
}
else if (grade >= 60) {
    println("D");
}
else {
    println("F");
}

if(grade >= 90) {

}
else {
    if(grade >= 80) {

    }
    else {
        if (grade >= 70) {
```

```
if(mouseX < 50)
    if(mouseY < 50)
        println("upper left");
else
    println("what is happening?");
```

(60,10)

(30,10)

(30,90)

```
int x=10, y=10, z=5;
if(x == y)
    print("A");
    if(x < z)
        print("B");
else
    print("C");
```

AC

C

B

Flags - 5.6