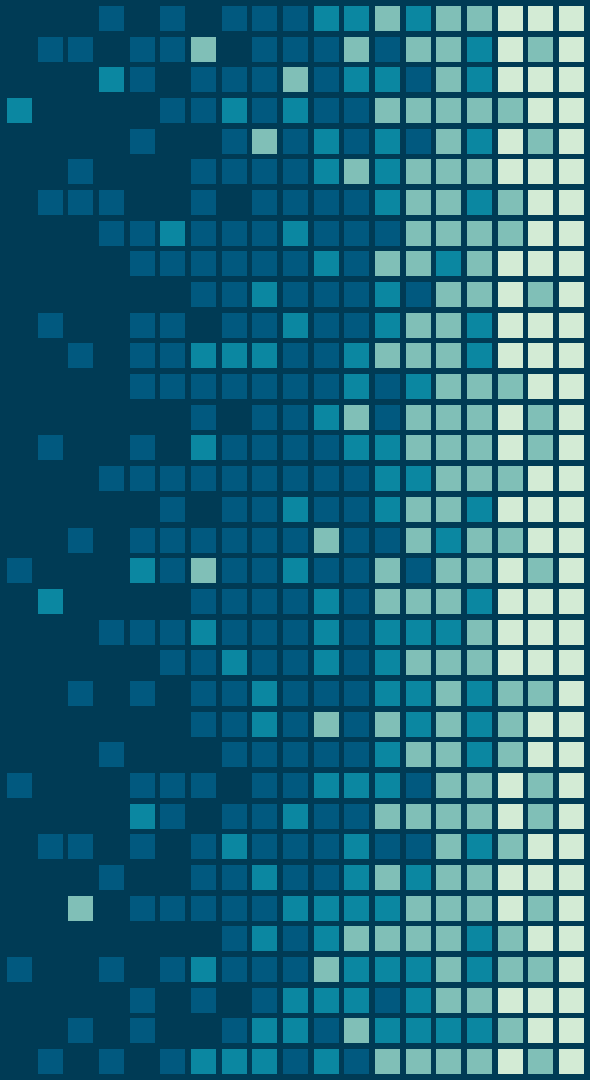
Lesson 2

More Java!

Logic



Logic

And (&&) → a && b

returns *true* if both *a* and *b* are true, false otherwise

Or (||) → a || b

returns *true* if either *a* or *b* are true, false otherwise

Not (!) → !a

returns *true* if *a* is false, *false* if *a* is true

Relational Operators

These help you compare two separate values

(<, >, <=, >=, ==, !=)

Respectively:

Less than, greater than, less than or equal to, greater than or equal to, equal to, not equal to

```
boolean StuyPulse = <number> <comparator> <number>;
```

Creates a boolean called StuyPulse that will be true if the comparison is correct. 6>5 would return true, but 6>7 would be false. Keep in mind that you can't use comparators on strings!

Applications of Logical Operators and Relational Operators

```
1 > 0 && 2 > 4
```

This would return false because only one parameter is true.

```
1 > 0 || 2 > 4
```

This would return true because at least one of the parameters need to be true for the entire statement to be true.

```
!(1 < 0)
```

This would return true because the parameter is false, but the “!” symbol negates it, therefore it is true.

CHALLENGE

What is each variable equal to?

```
1)int a = 102/4;

2)String b = “hi” - “i”;

3)double c = 9 * 3;
```

What would this code print out?

```
boolean coach = true;
boolean mentor = false;

4)System.out.println(coach && mentor);

5)System.out.println(coach || mentor);
```

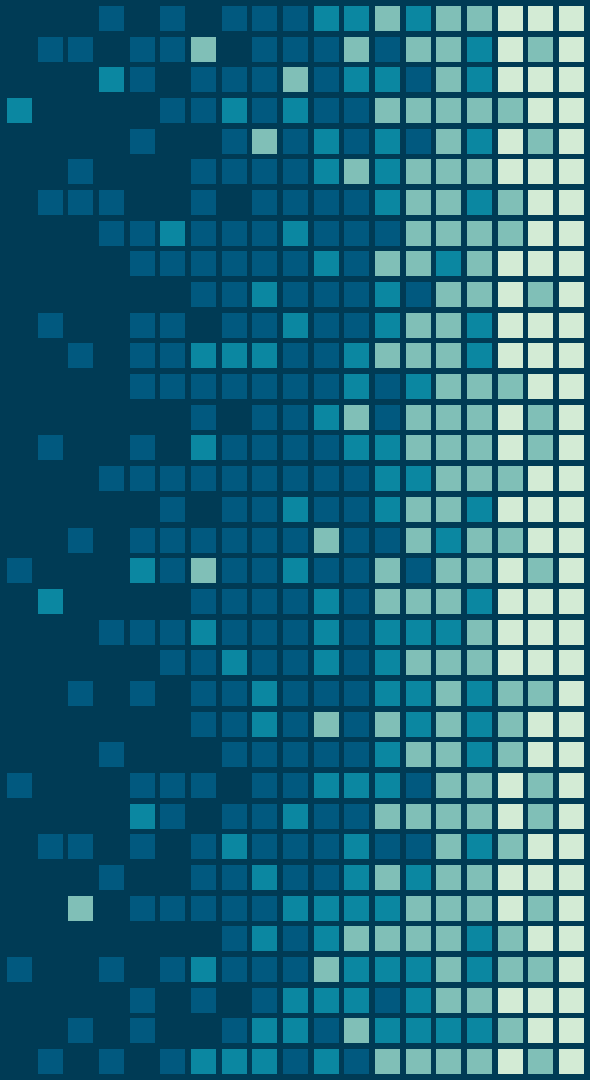
CHALLENGE (cont'd)

What value is the variable equivalent to?

```
6)int f = 5 * 3 - 2;

7)boolean beepboop == 5 >= 5;

8)boolean c = (5*3) >= 100;

9)boolean j = 39 > 8 && 100 <= 90+20;

10)boolean m = 420 == 90 || (89 != 57 + 32);

11)int j = 334 < 694;

12)int a = 320;
```

The “if” statement

AKA Conditionals



The “if” statement is a key component of code.

If statements make the decisions in code depending on certain circumstances.

“If you joined SE, then you’re cool”

Notice: the “you’re cool” part only applies if you are part of SE.

Syntax

```
if(boolean) {

    //code that runs when boolean is true

}
```

An example:

```
if(x > 3){

System.out.println(“x is greater than 3.”);

}
```

Else and Else if

Extensions of if statement. Runs if initial if statement is false.

```
if (boolean1){

    //runs if conditional is true

}else if (boolean2){

    /*runs if boolean1 is false and boolean2
is true*/

}else{

    //runs if all other statements are false

}
```

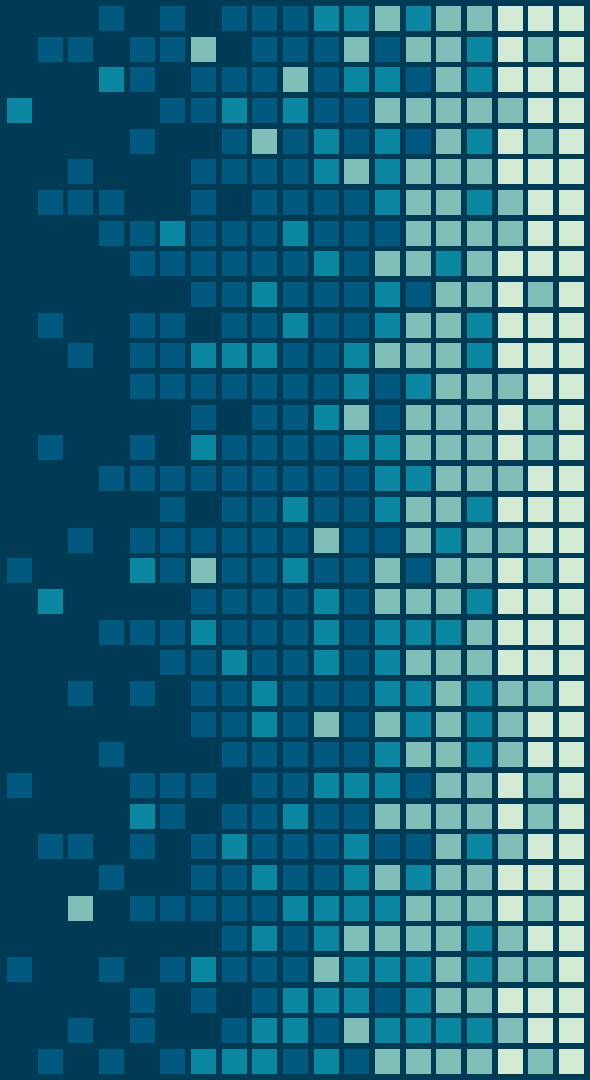
Exercise!

Make a variable called age that stores an integer.

Write code that takes this variable and …

- Prints out “Yay, you can drive!” if your age is at least 16 OR
- Prints out “Sorry, you’re too young to drive. ” if you’re younger than 16.

Loops in Java



While Loops

- While loops are controlled by booleans
- While loops contain statements that are repeated until the specified logical statement is no longer true

```
int num = 0;

while(num < 5){

        num++;

}
```

- In this case, num < 5 is the boolean we are testing, and as long as it's true, the code in the loop runs.

Assignments

Use a loop to display the numbers from 0 to 25, inclusive.

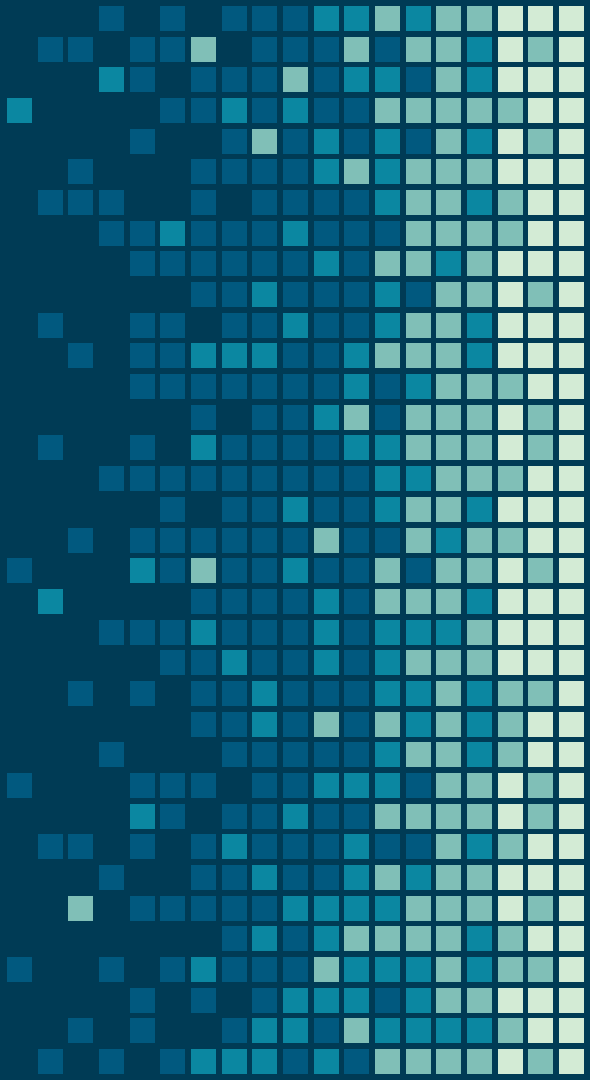
If one of the numbers being displayed is a multiple of 5, then multiply that number by 2, and print the result.

Then use a loop to display the numbers from 0 to 26, inclusive.If one of the numbers being displayed is a multiple of 7, then display it alongside text saying “I like java”’

NOTE: the % operator can be used to get the remainder.

Ex: 4 % 5 = 4, 16 % 5 = 1, 10 % 5 = 0.

Methods in Java



What are methods?

A Java method is a collection of statements that are grouped together to perform an operation. When you call the System.out.println() method, for example, the system actually executes several statements in order to display a message on the console.

For example, washing a plate, drying it off, and putting it in the dishwasher is equivalent to doing the dishes.

This is similar to functions in math eg. f(x) = x + 3.

An example of a method:

```
public class Add{

    private static int add(int a, int b) {

        return a + b;

    }

    public static void main(String [] args) {

        //What do the following function calls return?

        add(25,5);
        add(7,3);
}
```

How does it work?

In the example, we created a function called “add” by having 3 parts: the return type, the parameters/inputs, and the sequence of code inside the function.

An example of a method:

Return Type: Methods can return a value, similar to a function in math.

This is done with the **return** keyword

```
int getSum(int n, int k) {
    return n + k;
}
int sqr(int a) {
    return a * a;
}
```

What do the following function calls return?

```
getSum(2, 0);
getSum(5, -1);
sqr(7);
sqr(-3);
```

Deciphering a method:

This is the **return type**, or the data type the method will return. If there is no return type, use void.

This is the **name** of the method.

Inside this set of parentheses will go the **parameters**. Values are placed here as inputs that the method can utilize. Multiple variables are separated by a coma.

```
public int bottomtext(int x) {

    x += 3;

    return x;

}
```

The code inside the method is what the method does. For example, this method adds 3 to int x, the parameter, and inside the method goes “x += 3”. The int x will then be returned in the following line.

Calling a method

Calling the method executes the statements in the method. To call a method, you write the method name. It is basically pasting the statements of the method to where it was called. What would the following methods return?

```
bottomtext(5);
bottomtext(3 + 2);
bottomtext(true);
```

```
public int bottomtext(int x){
    x = x + 3;
    return x; }
```

Easier Challenge!

1. Create a method that has parameters : 2 integers and a boolean
2. If the boolean is true, return double the sum of the 2 integers
3. If boolean is false, return the sum of the 2 integers

- To see your result, use `System.out.println(yourFunctionCallHere);` in the main method

Challenge time!

Create a method takes in int *k* as a parameter

Add the 10 consecutive numbers after (and not including) *k*

If sum is greater than 200

Then return the value of *k*

Otherwise return *k* - 3.