LEAGUE OF AMAZING PROGRAMMERS

LEVEL 0: CHEAT GUIDE

CREATED BY STUDENTS:

ATHENA HERNANDEZ
CINDY FELDMAN
ELLA DEMAREST
RYLAND BIRCHMEIER
SHIVA KANSAGARA
SOFIA VELARDE

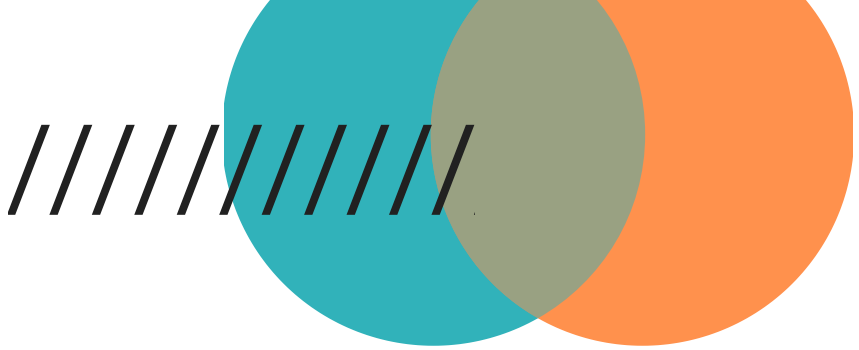
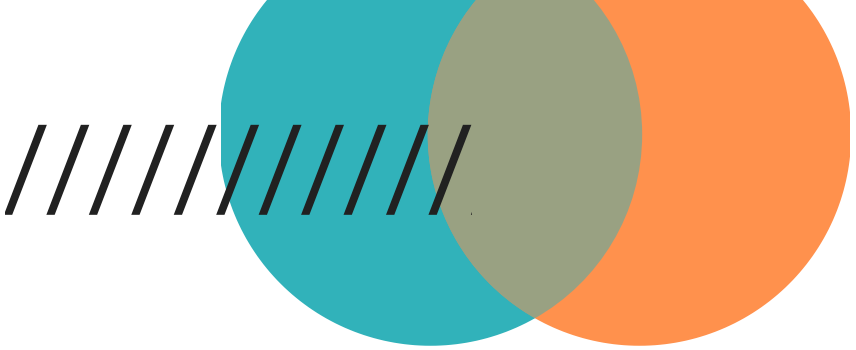


TABLE OF CONTENTS

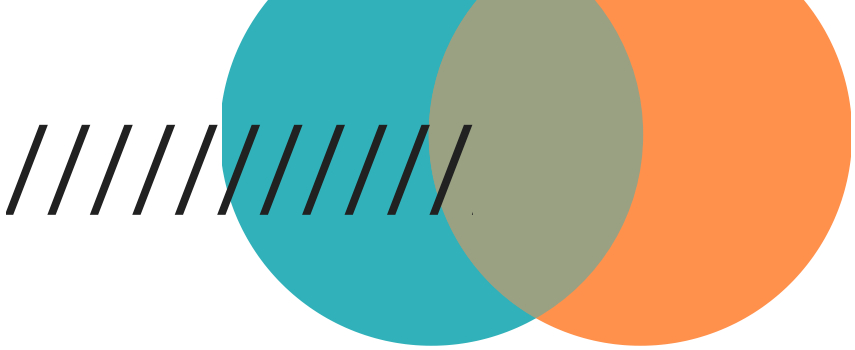
ORGANIZED BY MODULE



Module 0: Drawing, code flow, Strings, games

JOptionPane

- String input = JOptionPane.showInputDialog("What is your name?");
  - You saw this example in your practice. Java has a built in API which allows for popup windows like this one using JOptionPane....
  - You must save the output of the showInputDialog method to a String unless you change it to another data type manually.



If Else Statements

- They are written like this:

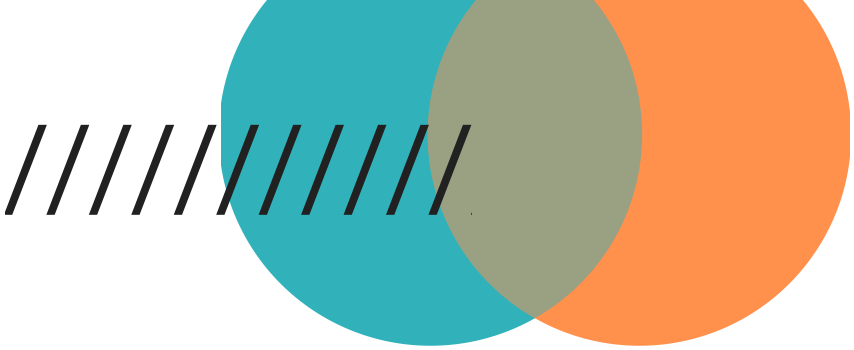
```
if(boolean expression here) {
    //execute this code if boolean expression is true
} else if(other boolean expression here) {
    //execute this code if the first boolean expression is not
    true and this one is true
} else {
    //if both of the other boolean expressions are false, this
    will run
}
```

int

- Integer numbers are stored like this...
  int variableName = 10;

String

- Strings are stored like this...
  String variableName2 = “This is a String!”;



Module 1: int variables, if/else, for loops, ++, --

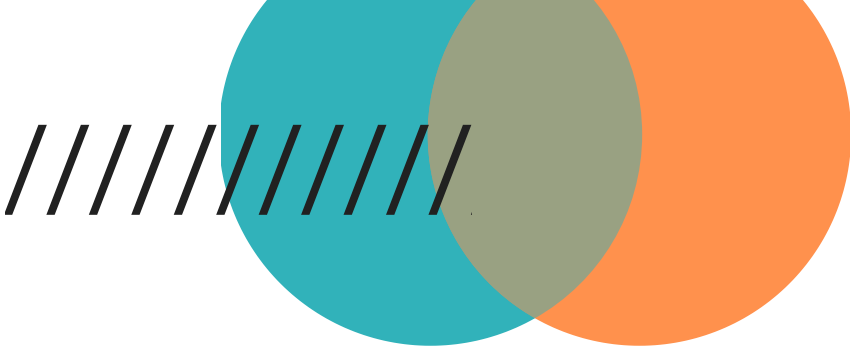
- An **int variable** can be used to hold a value of any integer (any whole number, positive or negative).
- In order to create an int variable, start with the type (int)

  Example: int num = 5;

- **if/else** - used to check if a statement is true. else is used to tell the program what to do if the statement is false.
- Example:

```
int myAge = 17;
if(myAge == 17) {
    System.out.println(myAge);
} else {
    System.out.println("we are not the same age");
}
```

*This will print 17 because myAge is 17, making a true statement.



- A **for loop** is a way to execute or run code repeatedly until the given condition is met.
- Example:

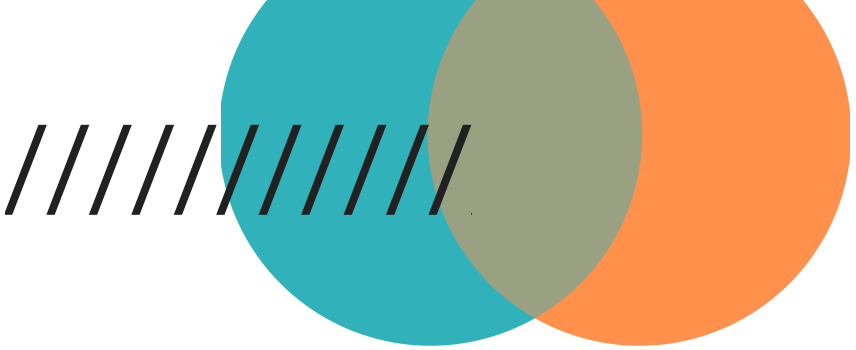
```
for(int i = 0; i< 5; i++) {
      System.out.println("hi");
}
```

*This will print out "hi" 5 times because 0-4 is the highest variable i can get until it meets the given condition, i<5.

- If you see **++**, this means you are incrementing, or adding 1 to a variable.
- Example:

```
int i = 0;
i++;
System.out.println(i);
```

*This will print out 1, as 0+1 = 1.



- If you see **--**, this means you are decrementing, or subtracting 1 from a variable.
- Example:

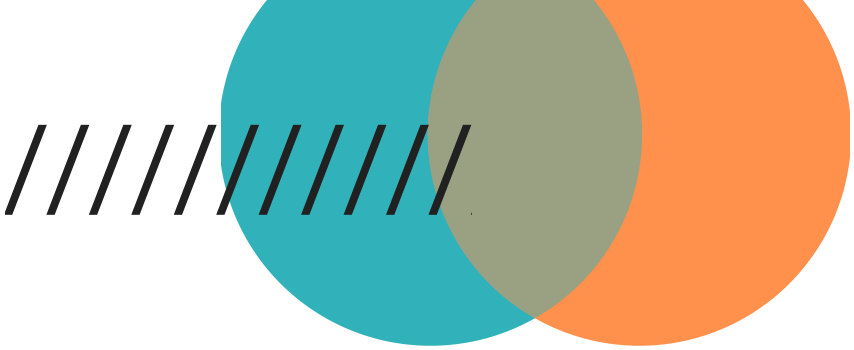
  ```
  int i = 5;
  i--;
  System.out.println(i);
  ```

  *When this runs, it will take 5 and subtract 1 from it, printing out 4.

Module 2: Random, methods, String-to-int

Random

- In Java, there is a class called “Random” which you can make objects of.
- You MUST import this method which Eclipse will prompt you to do every time you use it.
  - If you are getting a red line just hover over it and select import Random.



Random continued:

- Example:

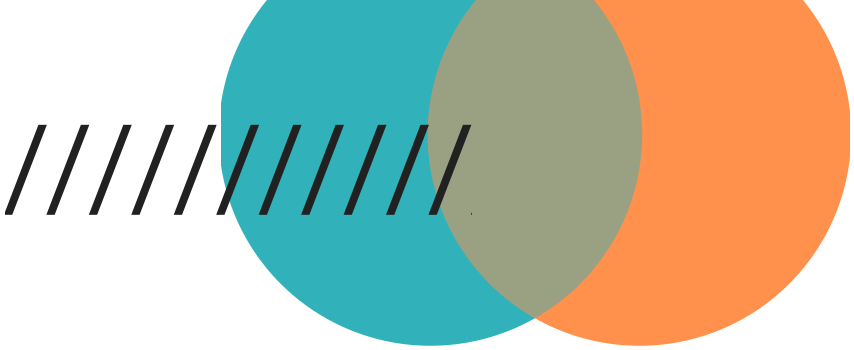
```
import java.util.Random;
public class Runner {
	public static void main(String[] args) {

		Random randy = new Random();

		System.out.println(randy.nextInt(5));
		//prints a random number between 0 and 4
		//inclusive

		System.out.println(randy.nextBoolean());
		//prints either true or false randomly

	}
}
```



Methods

- Here we have a main method inside the class “Runner."
- When we do “Runner r = new Runner();”, we are making a new object of that class which enables us to use all of the methods of that class.
    - When we do r.add(1, 3) for example, we are simply telling Java that we want that Runner object, r, to execute the code in the method, add.

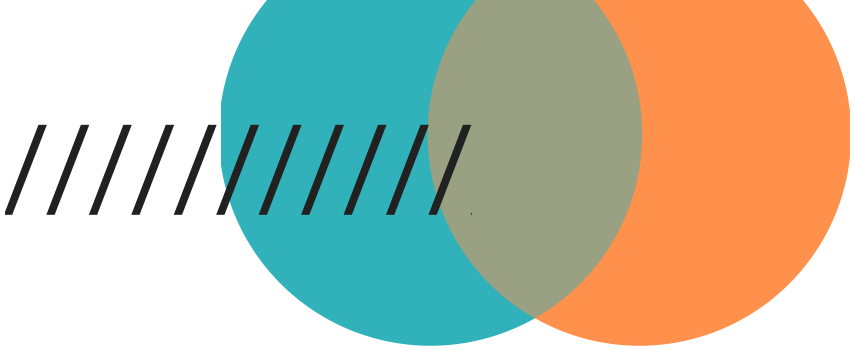
Example:

```
public class Runner {

    public static void main(String[] args) {
        Runner r = new Runner();
        System.out.println(r.add(1, 3));
    }

    public int add(int x, int y) {
        int z = x+y;
        return z;
    }

}
```



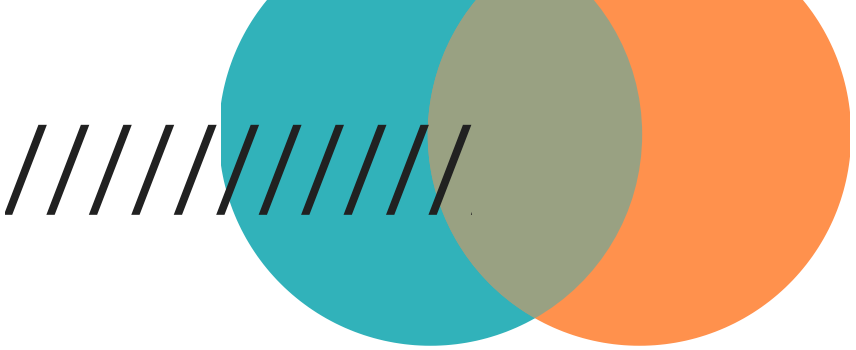
String to Int

- This one is rather straightforward actually, but can be confusing if you do it wrong.
- The “Integer” class (note the capital “I”) is what we call a wrapper class of the primitive type “int”.
- What this means is that since the “int” type doesn’t have any methods of its own which we can use, we must utalize the wrapper class to perform certain operations on it. One of these is changing it to a string.

- We can therefore do:

```
int i = Integer.parseInt(“304”);
int y = i+6;
System.out.println(y); //prints 310
```

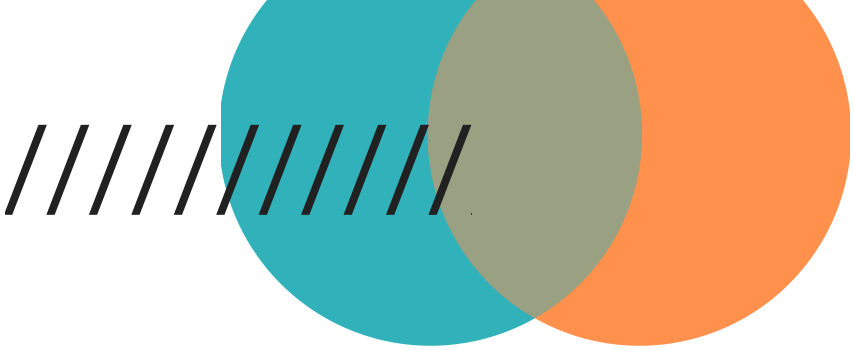
- As we cannot do addition to Strings, we must instead first convert to an int with Integer.parseInt();, then do the addition.



Module 3 : Else if, loop variables, modulo

Else-if

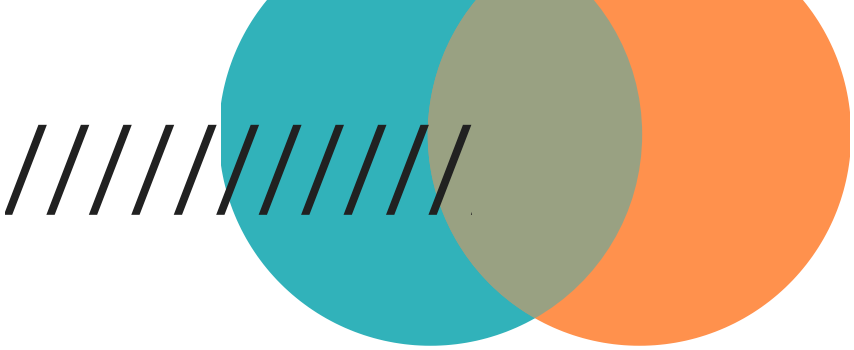
- An **else-if statement** is used when there are multiple conditions that must be tested.
- Else-if statements only come after if-statements and are not always needed when using an if-statement.
- If an else-if statement is not needed, the if-statement would just end with an else statement.
- Full example on next page.



- Example:

```
int myAge = 16;
if(myAge >= 18) {
      System.out.println("I am old enough to vote.");
} else if(myAge >= 16) {
      System.out.println("I am old enough to drive.");
} else {
      System.out.println("I am not old enough to vote or drive.");
}
```

*The output will be "I am old enough to drive" because myAge is equal to 16. Since myAge is not at least 18, the if-statement is skipped because the conditions were not met and moved on to the else-if statement. If myAge was equal to 15 or younger, the output would be "I am not old enough to vote or drive" because the first two conditions are not met.



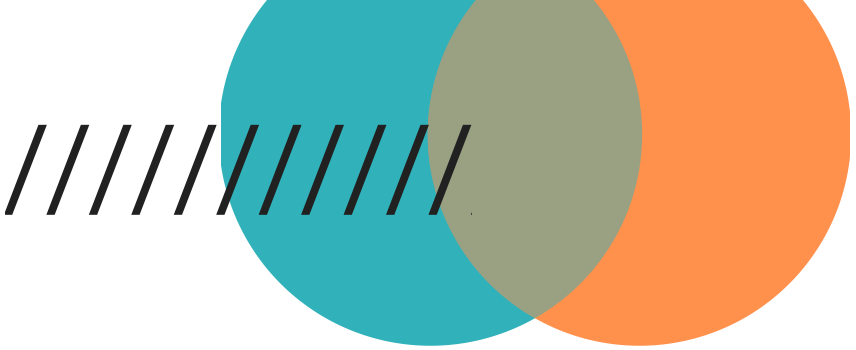
Loop Variables

- Example:

```
for(int i = 0; i < 10; i++){
    System.out.println("count:" + i);
}
```

- The bolded portion represents the loop variable.
- A loop variable is a variable that defines the index of where the loop is at during its iteration.

- Explaining the example:
  - The loop variable is called int i and is set to start at 0 (int i = 0)
  - The second part of the for loop (separated using a semicolon) indicates a condition that says:
    - if i is less than 10 (i < 10)
  - The last part of the for loop (separated using another semicolon) increments the loop variable, i, by one using two plus signs (++) if the condition from the previous step is true: i++
    - Since i is equal to 0, the condition that says i < 10 would be true, allowing the loop to increase by one
  - Note: the loop variable can be named anything

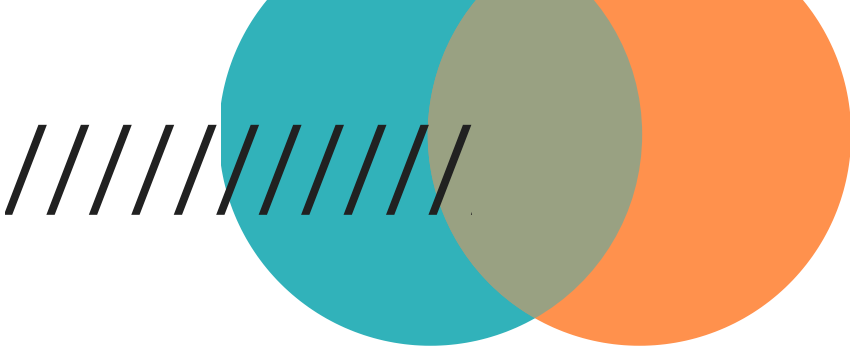


- Loop variables continued…
- Example 2:

```
for(int year = 2020; year < 2030; year++) {
        System.out.println("years in this decade:" + year);
}
```

Modulo

- A modulo operator is used to find the remainder of a division operation.
- Modulos are represented using a percent sign: '%'.
- Examples:
    - 7%2 = 1
        - Because: if you try to divide 7 by 2, the answer would be 3, remainder 1. Since modulos only find the remainder, the answer would just be
    - 30%4 = 2
        - Because: 30 divided by 4 equals 7, remainder 2.
    - 6%3 = 0
        - Because: 6 divided by 3 equals 2, remainder 0.



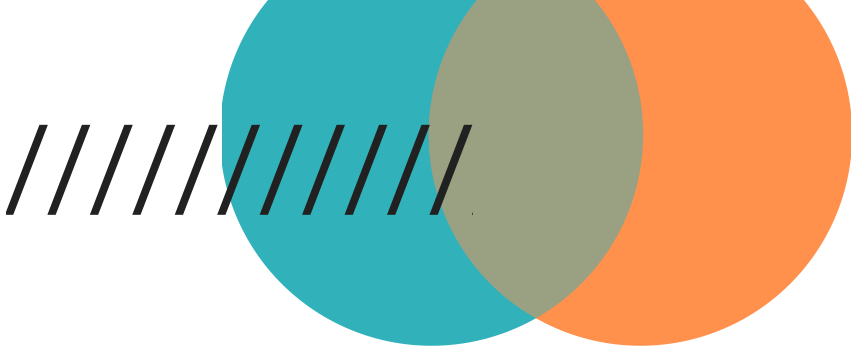
- Implementing modulo into code: print out all the even numbers between 1 and 10.

```
for(int i = 1; i < 10; i++) {
    if(i % 2 == 0) {
        System.out.println(i);
    }
}
```

*The output will be “2, 4, 6, 8”. Since even numbers do not have a remainder when they are divided by 2, a modulo is used to check that condition. Also, 10 is not printed out (even though it is an even number) because the loop stops running if ‘i’ is less than 10.

Module 4 - double, boolean, charAt, option dialog

- What is a **double**? A double is an 8 byte (64 bits) primitive data type that holds decimal numbers. It’s default value is 0.0d.
- Example on next page.



- Example:

  double first_number = 10.5;
  double second_number = 20.8;

  first_number and second_number are doubles!

- What is a **boolean**? A boolean is another primitive data type that can only store two values: true and false.

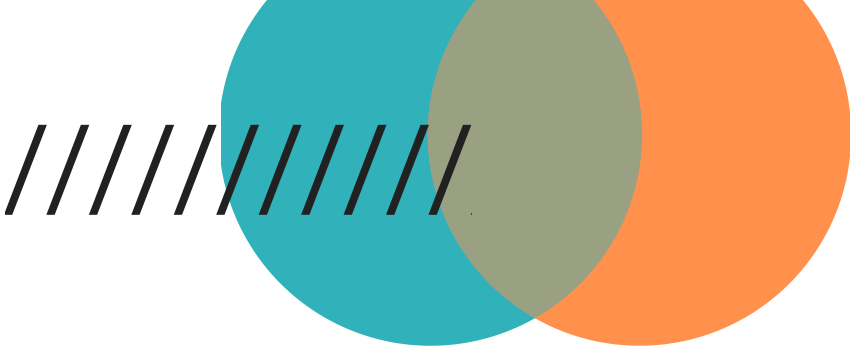
  boolean athena_is_hungry = true;

- Booleans are also outputted when comparing two things.

  int x = 10;
  int y = 9;
  System.out.println(x > y); // returns true because 10 > 9

- What is **charAt()**? charAt() is a method that returns the character of the index in the parentheses.

  String myStr = "Hello";
  char result = myStr.charAt(0);
  System.out.println(result); // outputs ‘H’



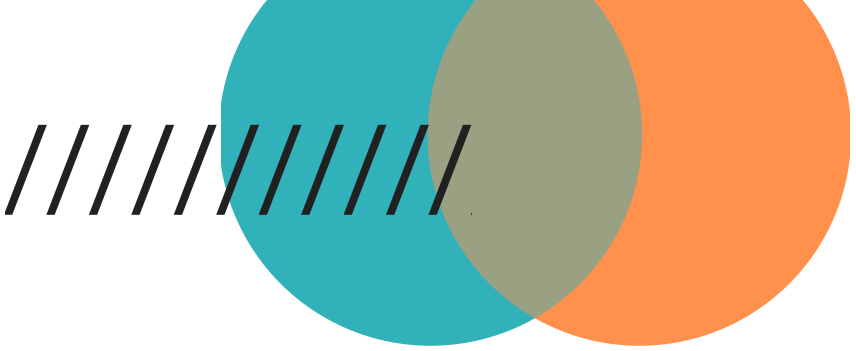
- What is **JOptionPane**? JOptionPane makes it easy to pop up a standard dialog box that prompts users for a value or informs them of something.
- Make sure you include import javax.swing.JOptionPane; at the top of your file.
- showMessageDialog() and showInputDialog() are the most frequently used methods of JOptionPane. Reference below to see how they work.

  JOptionPane.showMessageDialog(null, "Hi there!"); // make sure to save the data you asked for in a variable

  String name = JOptionPane.showInputDialog(null, "Enter your name.");

Module 5: algorithms, nested loops, practice

- A **nested loop** is a loop inside of a loop.
- This is used when access to rows and columns is needed. The inner loop will iterate until it reaches it's given condition, then the outer loop tells the inner loop how many times to run.



- **Rows** go from left to right ↔
- **Columns** go up and down ↕

- Example:

```
(outer loop) for (int i = 1; i < 6; i += 2) {
(inner loop)      for (int j = i; j < i + 2; j++) {
                        System.out.print(j + " ");
                  }
                  System.out.println();
            }
```

*This will print out all integers 1- 6, with 2 columns and 2 rows.

1 2
3 4
5 6