GET SMART: WITH JAVA PROGRAMMING

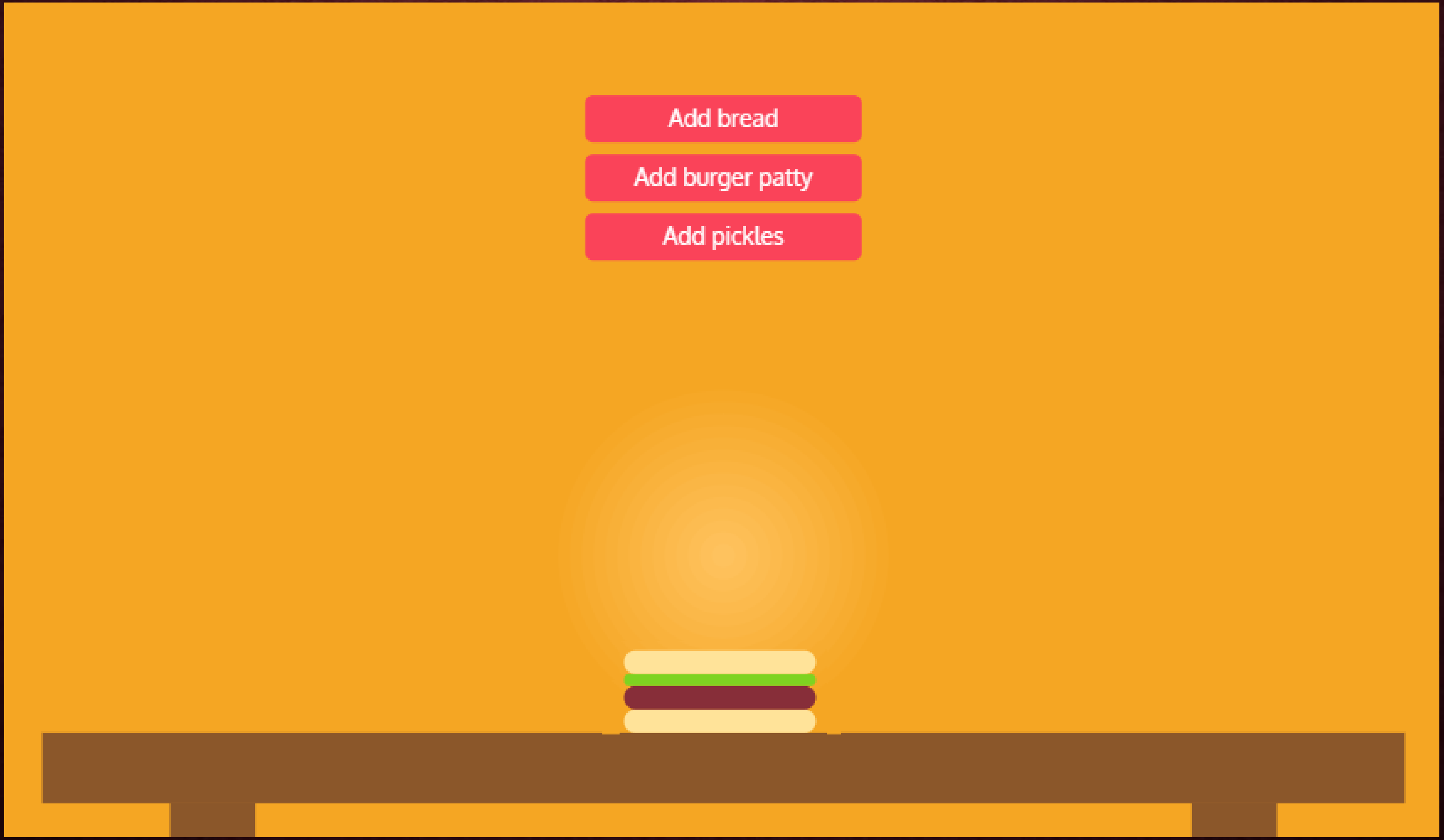
Yaman Omar Alashqar

```
System.out.println(“WELCOME TO THIS COURSE\n”);
```

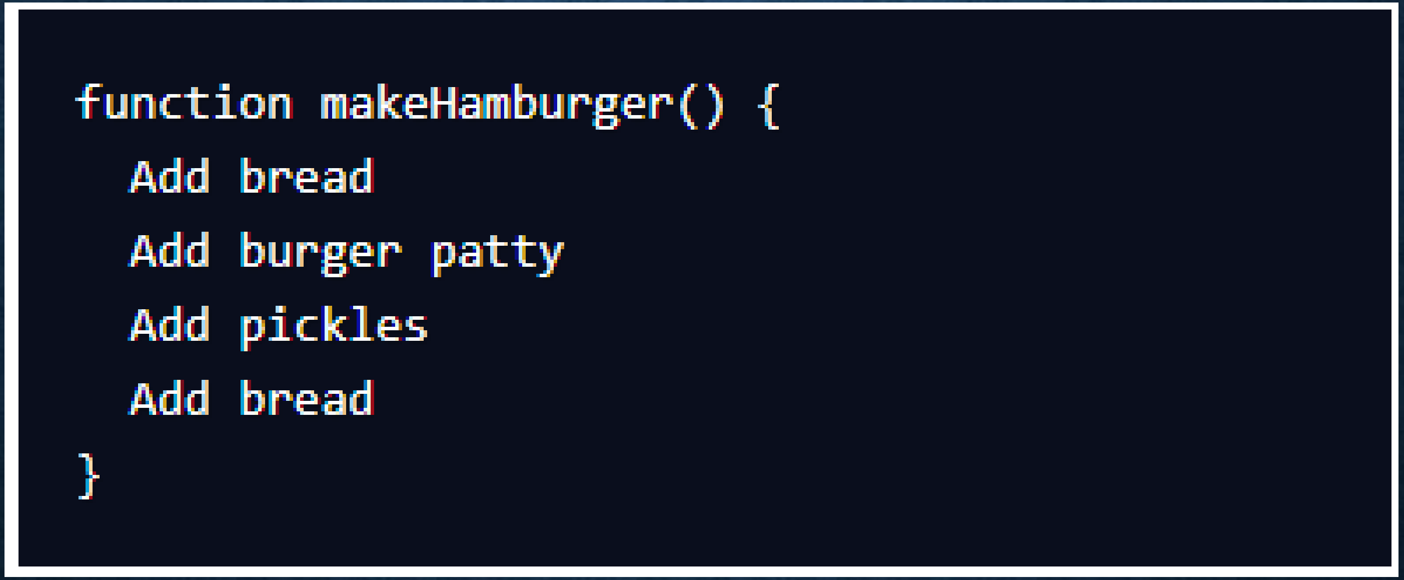
Methods

OPENING PROBLEM

Imagine writing a program that makes a sandwich



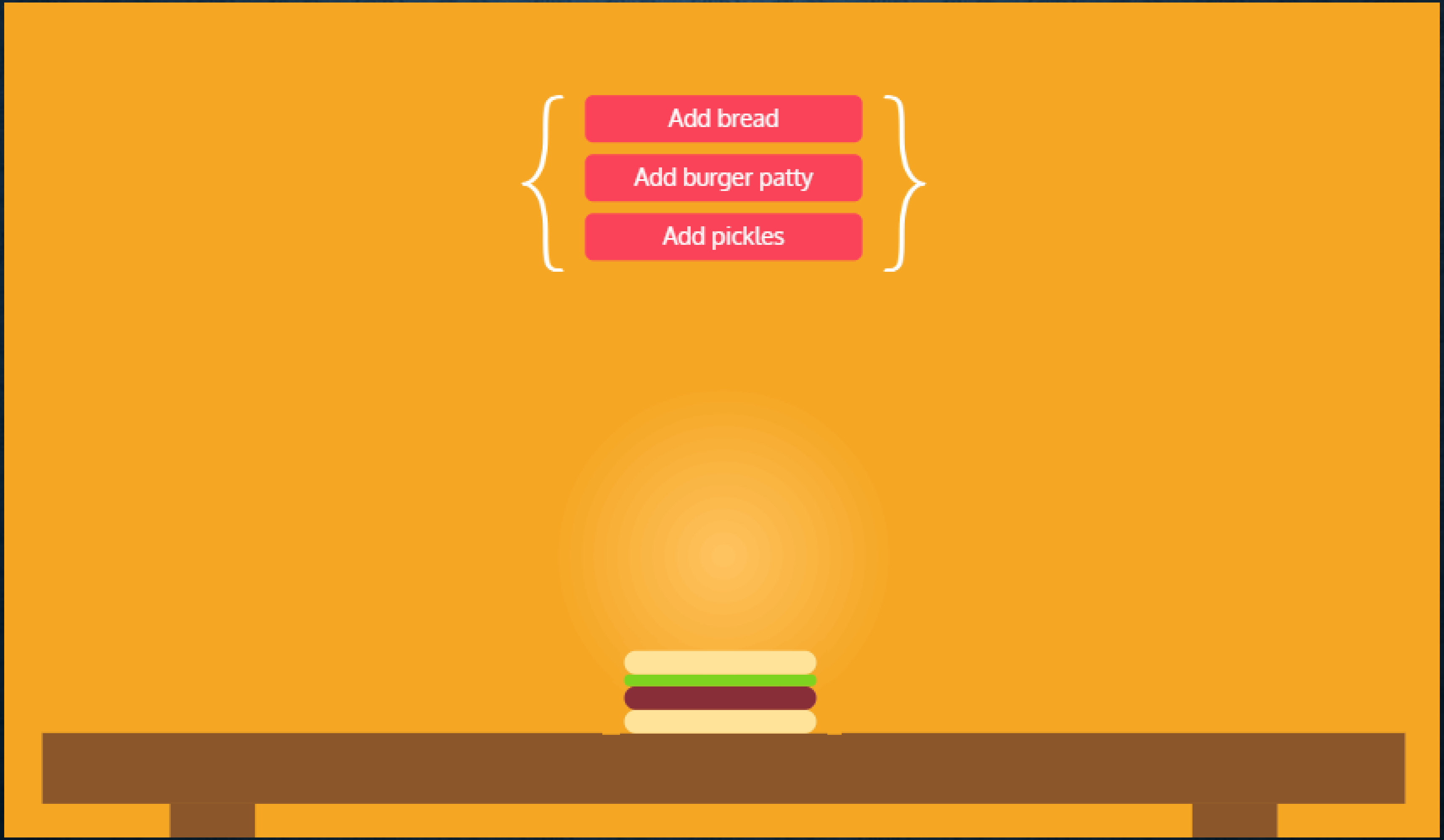
As a restaurant will you make one sandwich? Or many of them?



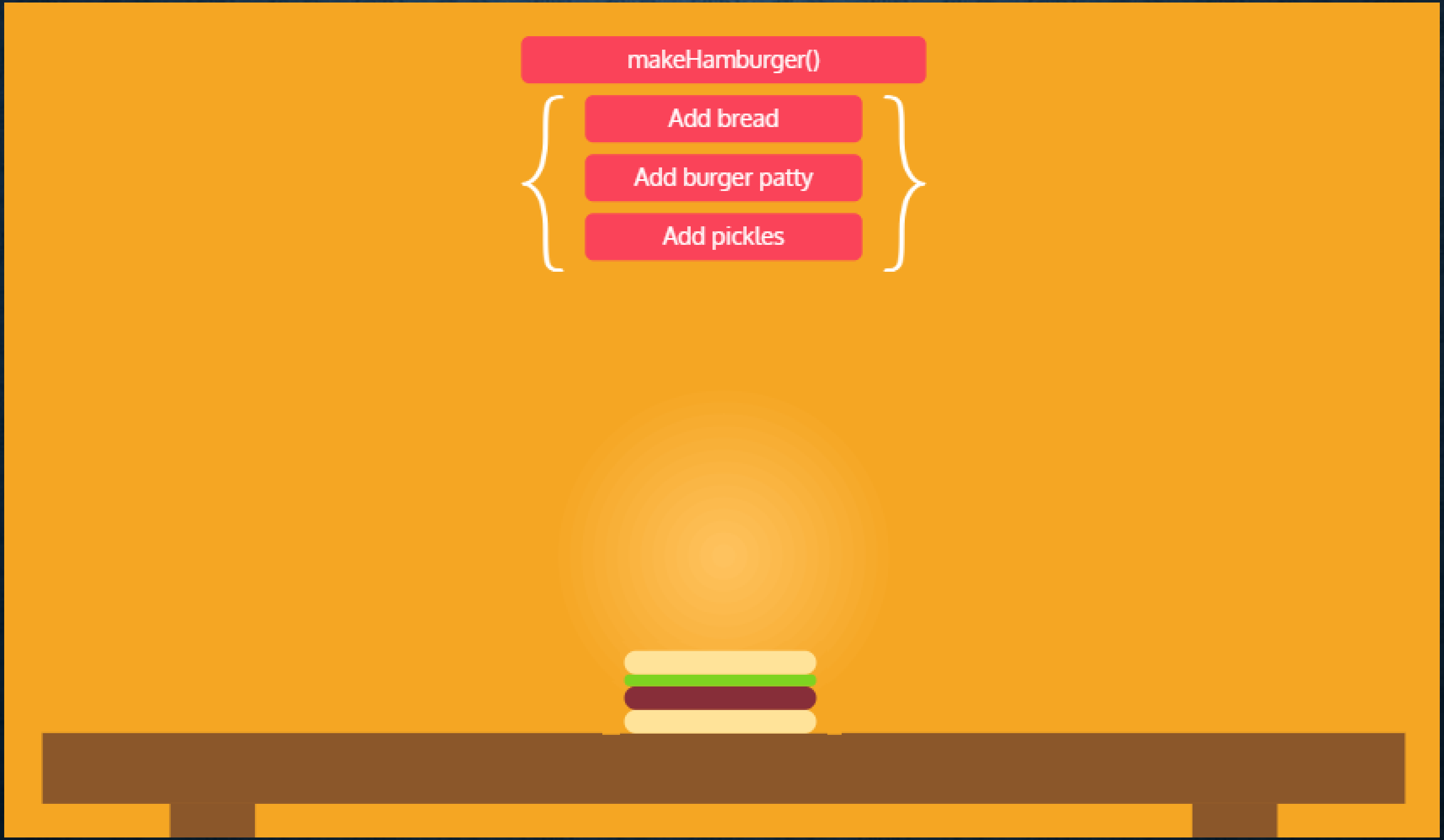
```
function makeHamburger() {
  Add bread
  Add burger patty
  Add pickles
  Add bread
}
```

So why would you keep repeating the code?

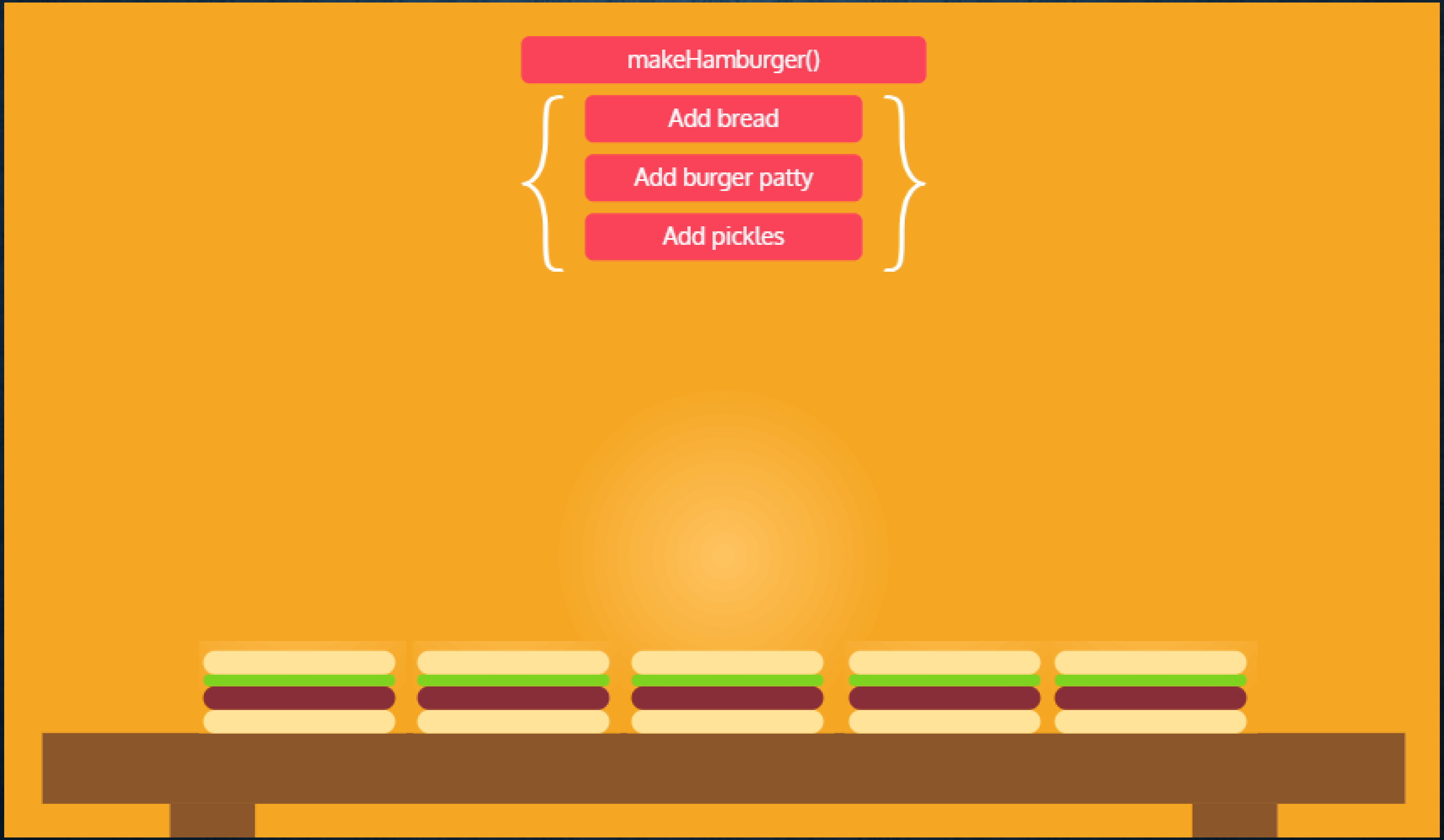
Combine these instructions in the right order



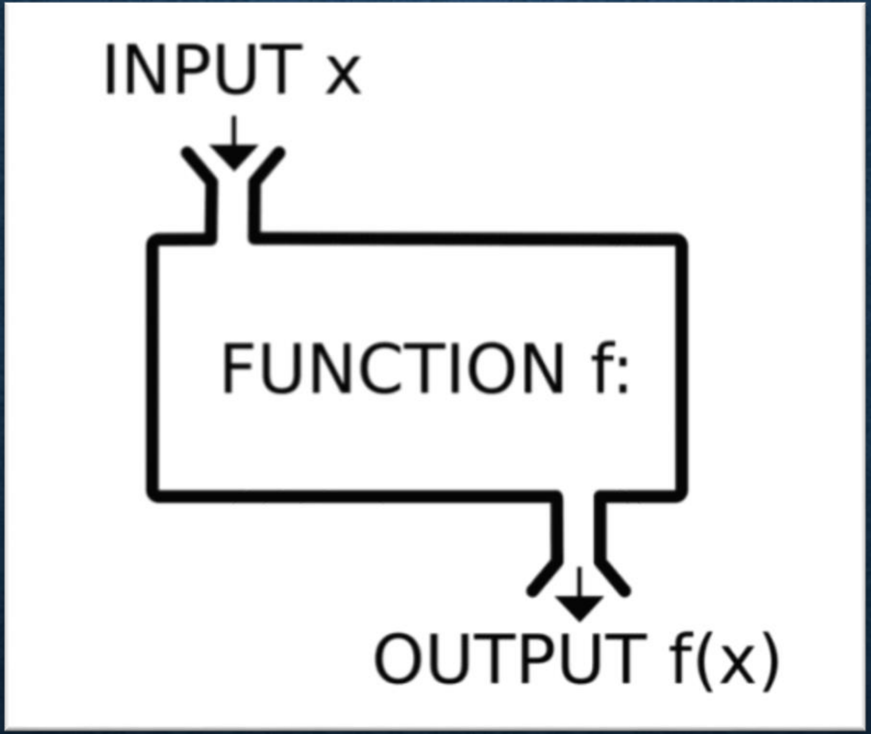
And wrap them inside a method



Every time you need a sandwich, just reuse the method



A function is a sequence of instructions that performs a specific task, packaged as a unit



In object-oriented programming, method is a jargon used for function. Methods are bound to a class and they define the behavior of a class.

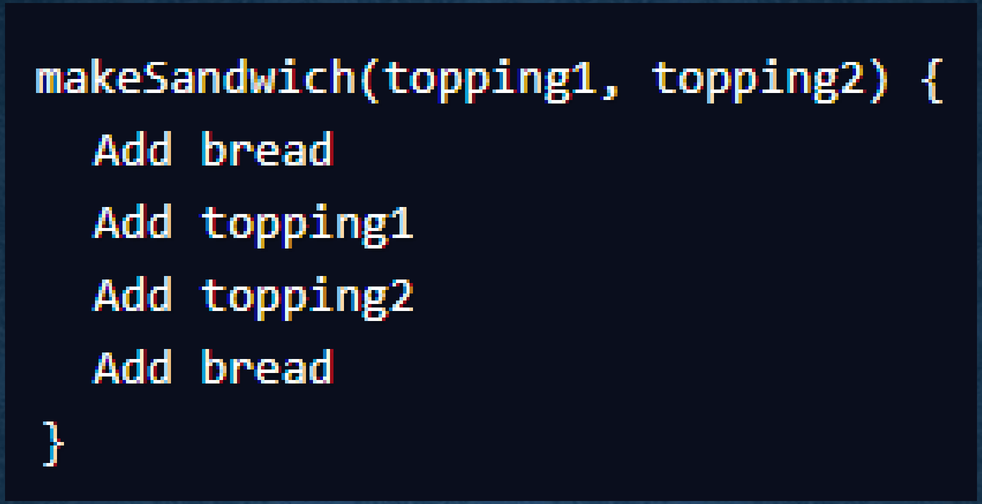
ANOTHER PROBLEM

Not everyone wants to eat hamburgers.

We could write a new method for each new sandwich type, but that takes a lot of work and risks making mistakes.

Instead we'll generalize the hamburger method to a sandwich method.

This new sandwich method will still make a bread-topping-topping-bread combination, but the toppings may change based on inputs to the method

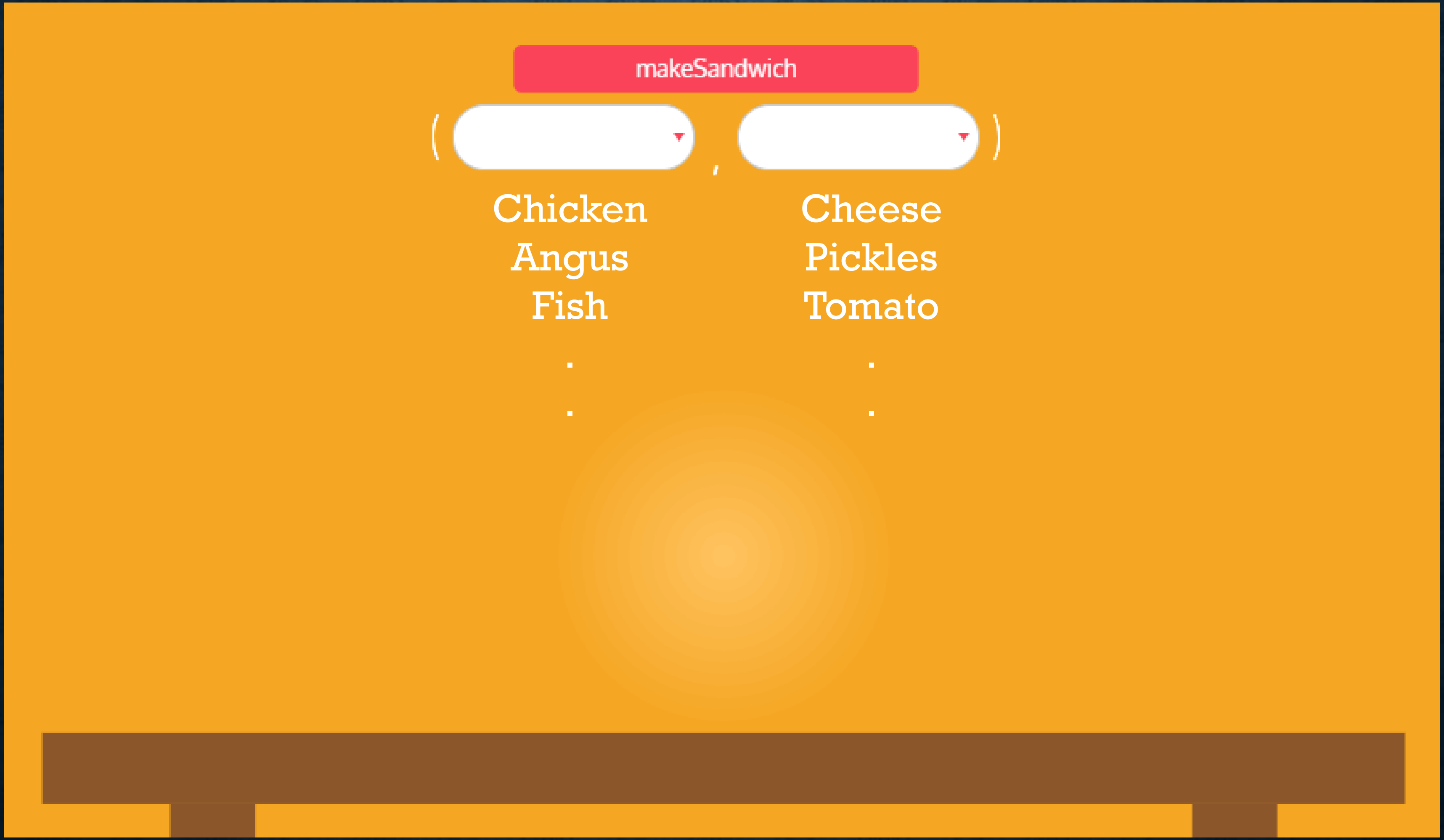


```
makeSandwich(topping1, topping2) {
   Add bread
   Add topping1
   Add topping2
   Add bread
}
```

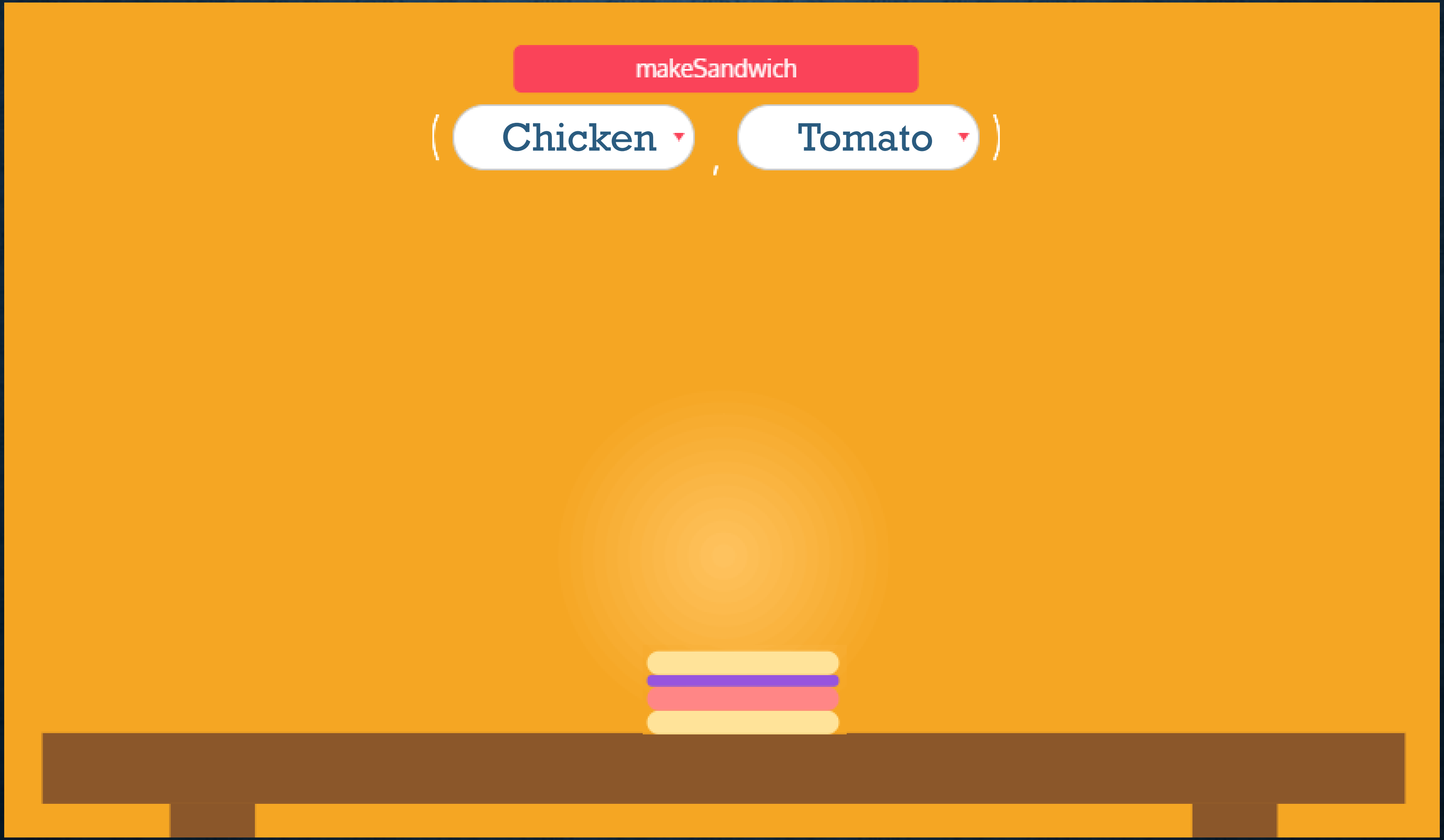
The method has two inputs, or parameters.
Each time we call the method "makeSandwich", we'll give actual values for each input, called arguments.

For example, we want a chicken-and-cheese sandwich using the method makeSandwich("chicken", "cheese").
We call the function with the arguments "chicken" and "cheese".
Those will be the values for the topping1 and topping2 parameters.

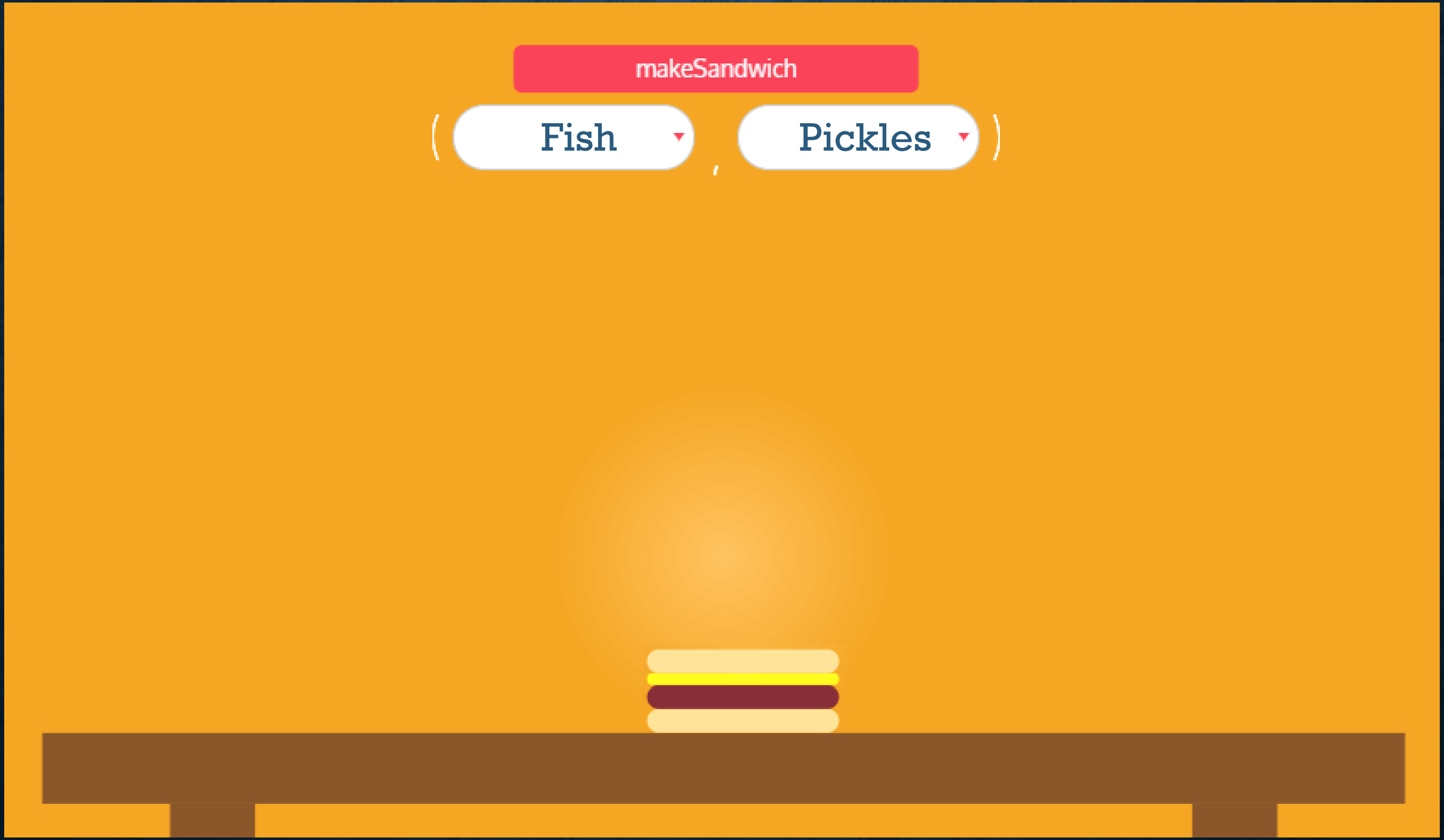
Instead of writing a different methods for each type of sandwich, we have one method that can make them all!



Chicken	Cheese
Angus	Pickles
Fish	Tomato
.	.
.	.



```
makeSandwich
( Chicken , Tomato )
```



makeSandwich

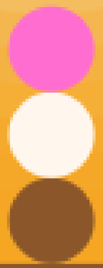
(Fish , Pickles)

makeSandwich()

```
makeSandwich()
{
    Add bread
    Add burger patty
    Add pickles
}
```

```
makeSomeFries()
{
    Add fried potatoes
    Add salt
    Add ketchup
}
```

```
makeIceCream()
{
    Add chocolate scoop
    Add vanilla scoop
    Add strawberry scoop
}
```



- A *function* is a:
  - sequence of instructions that performs a specific task, packaged as a unit.

- When we *define* a function, we specify:
  - the inputs, the outputs, the instructions, and name of the function
  - *functions can have parameters, which accept input values, making its instructions flexible*

- When we *call* a function, all of its instructions are executed.

- Functions can be executed many times, making its instructions *reusable*.

- Functions organize a program into distinct units, making interchanging and editing them easier. This makes your entire program organized and *modular*.

What are the advantages of using methods?

What are the advantages of using methods?

- The main advantage is code reusability.
  You can write a method once, and use it multiple times.
  You do not have to rewrite the entire code each time.

- Methods make code more readable and easier to debug.
  For example, getSalaryInformation() method is so readable,
  we can know what this does by reading the name of the method

Types of Java methods

Depending on whether a method is defined by the user, or available in standard library, there are two types of methods:

- Standard Library Methods
- User-defined Methods

Standard Library Methods
The standard library methods are built-in methods in Java that are readily available for use. These standard libraries come along with the Java Class Library (JCL) in a Java archive (*.jar) file with JVM and JRE.

For example,

print() is a method of java.io.PrintSteam. The print("...") prints the string inside quotation marks.
sqrt() is a method of Math class. It returns square root of a number.
toUpperCase()

e.g. System.out.print("Square root of 4 is: " + Math.sqrt(4));

User-defined Method

You can also define methods inside a class as per your wish.

How to create a user-defined method?

Before you can use (call a method), you need to define it.
Here is how you define methods in Java.

```
public static void myMethod() {
        //Instructions
                                    }
```

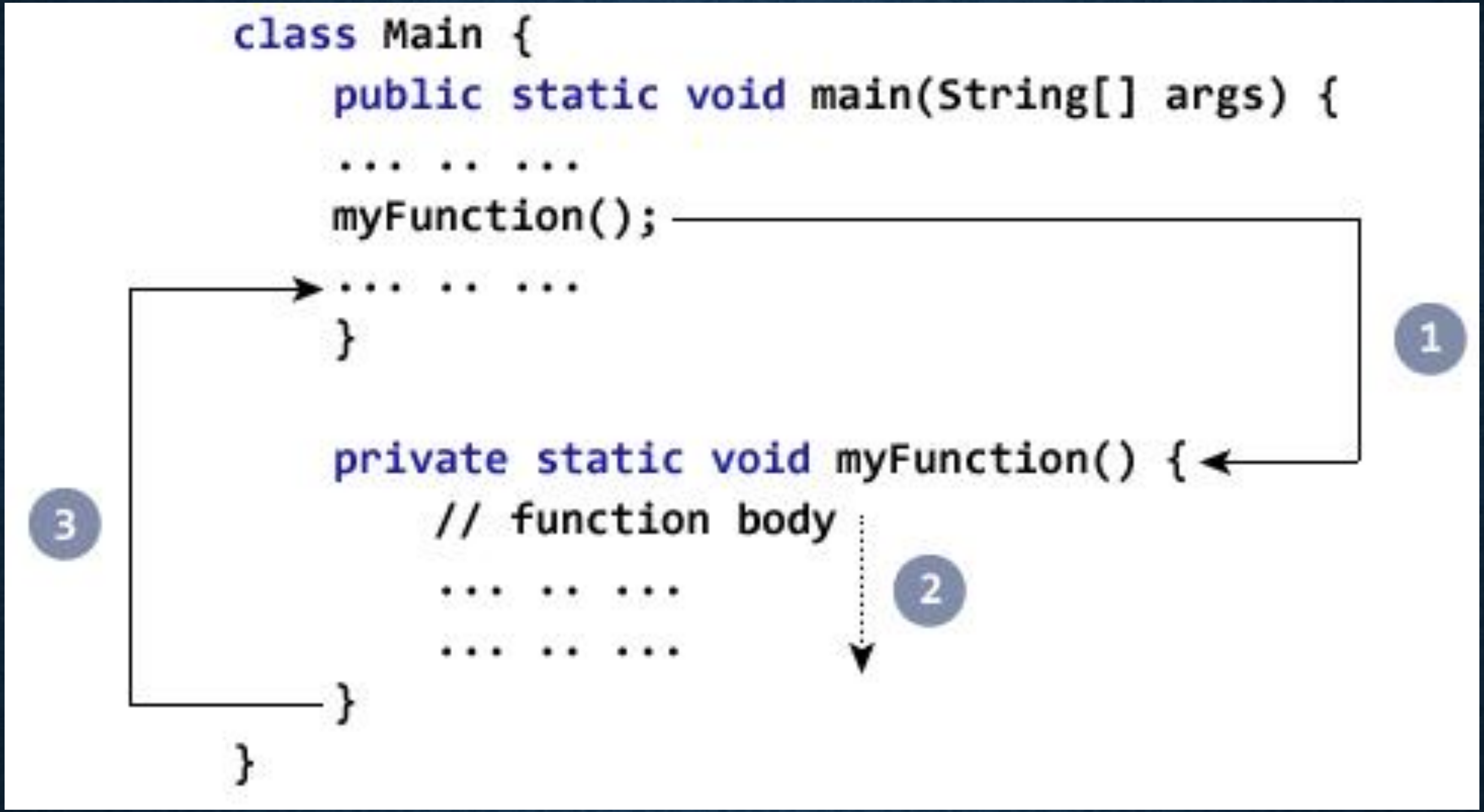
The public keyword makes myMethod() method public. Public members can be accessed from outside of the class.
The static keyword denotes that the method can be accessed without creating the object of the class.
The void keyword signifies that the method doesn’t return any value.

The complete syntax for defining a Java method is:

```
modifier static returnType nameOfMethod (Parameter List) {
// method body
return returnType;
}
```

Parameters (arguments) - Parameters are the values passed to a method. You can pass any number of arguments to a method.

Method body - It defines what the method actually does, how the parameters are manipulated with programming statements and what values are returned.
The codes inside curly braces { } is the body of the method.



```
class Main {
    public static void main(String[] args) {
    ... .. ...
    myFunction();
    ... .. ...
    }

    private static void myFunction() {
        // function body
        ... .. ...
        ... .. ...
    }
}
```

While Java is executing the program code, it encounters myMethod(); in the code.

The execution then branches to the myFunction() method,
and executes code inside the body of the method.

After the codes execution inside the method body is completed,
the program returns to the original state and executes the next statement.

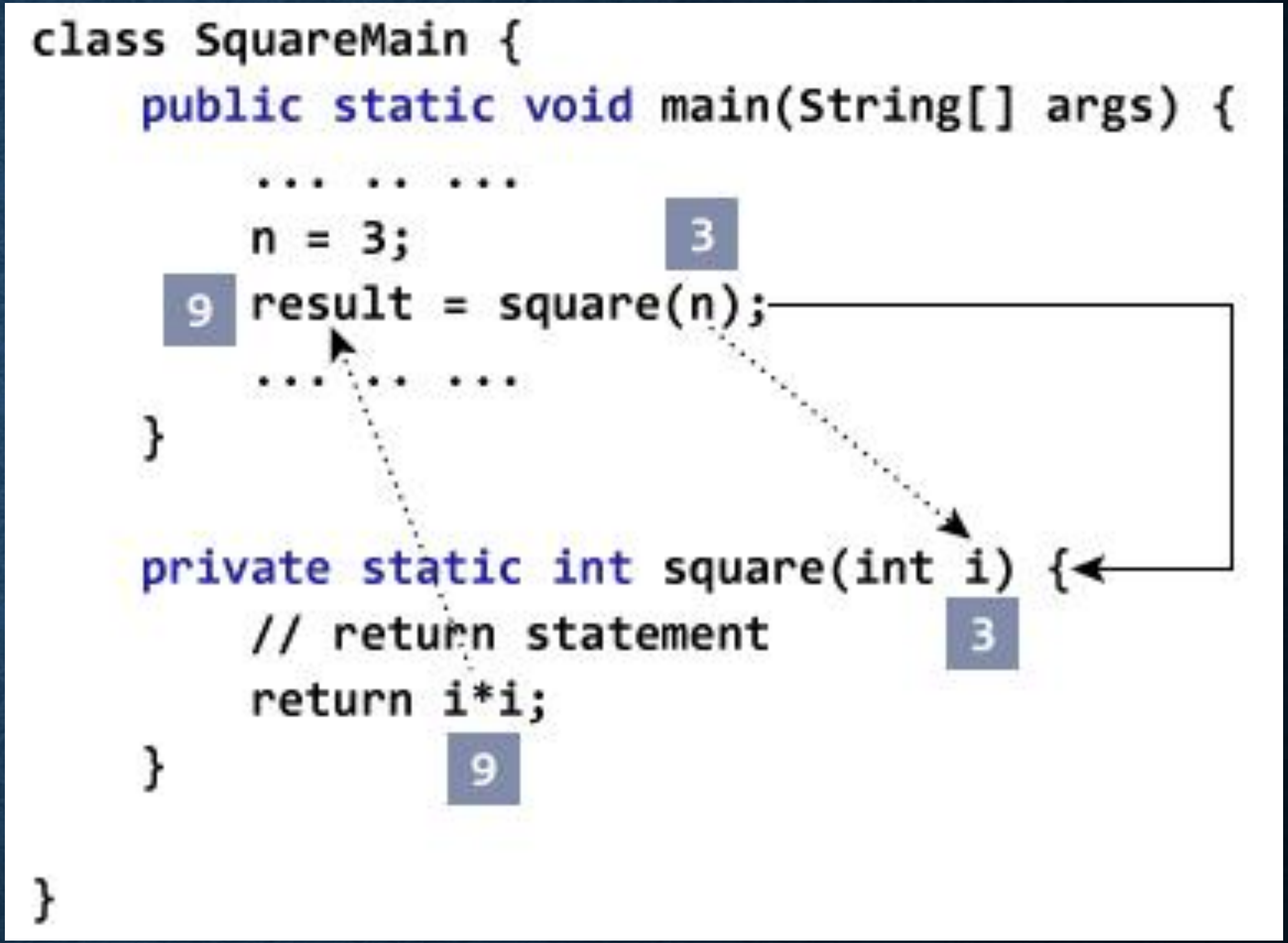
What does function calling mean?
And what is the return type?

Remember when using s.length( ) ? Or s.contains("a"); ?

The first one returns an integer (indicating how many characters the was in that string)
The second returned either true or false

The argument passed n to the getSquare() method during the method call is called actual argument.

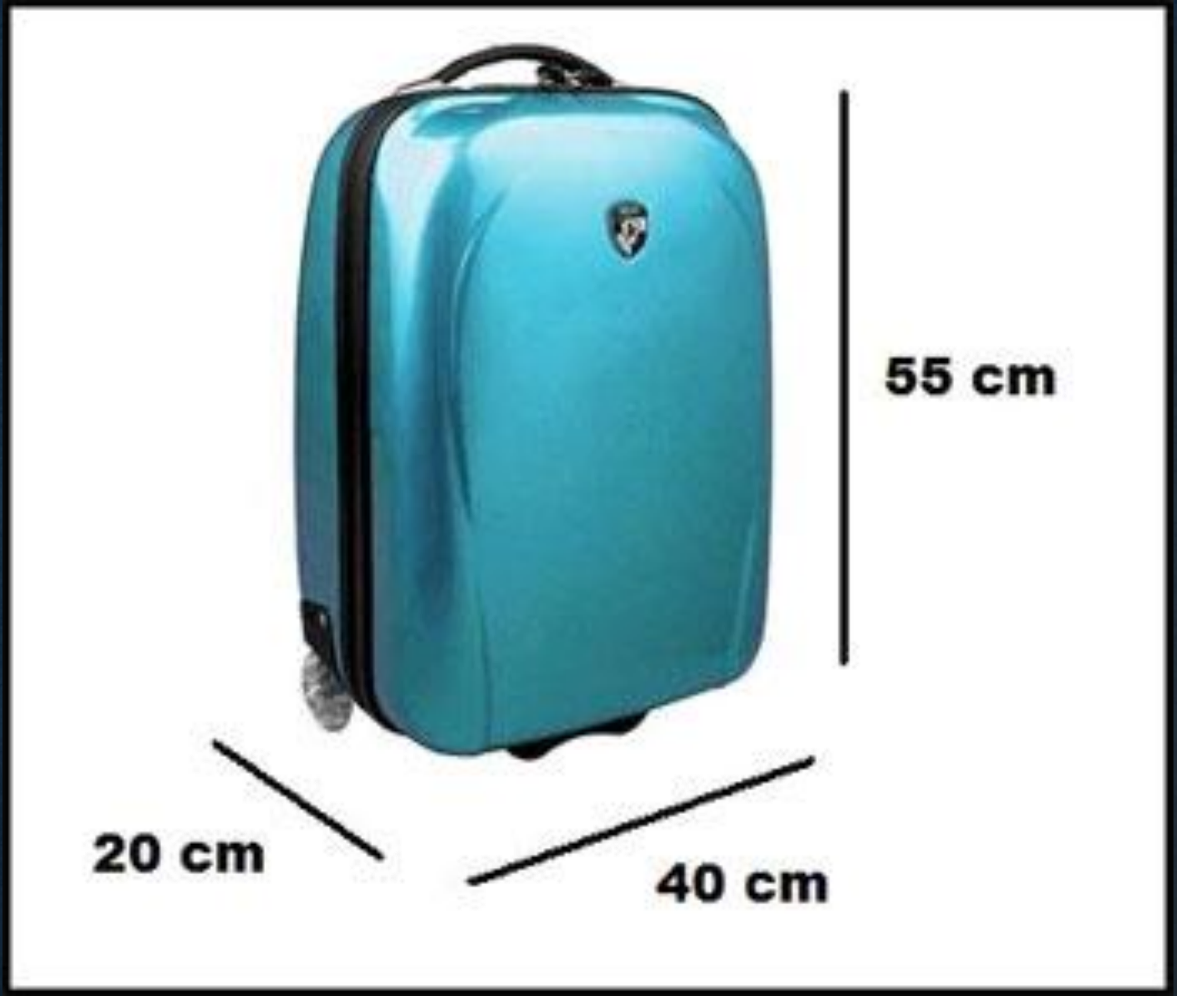
The parameter i accepts the passed arguments in the method definition getSquare(int i).
This is called formal argument (parameter)..



```
class SquareMain {
    public static void main(String[] args) {
        ... .. ...
        n = 3;
        result = square(n);
        ... .. ...
    }

    private static int square(int i) {
        // return statement
        return i*i;
    }

}
```



- Max weight: 23KG
- Max dimensions as mentioned in the image
- Write a method or more …

Write a Java function called absoluteValue()
The access modifier should be public,
it should have a return type of double,
and it should take one double parameter as input.
If the parameter is less than 0, it should return that number
negated. Otherwise, it should return the parameter unchanged.

fahrenheitToCelsius().
A public function with return type double that takes a double argument that represents a temperature in Fahrenheit degrees. It should return the equivalent temperature in Celsius degrees. (To convert from Fahrenheit to Celsius, use the formula:
$C = (F - 32) \times 5/9$.)