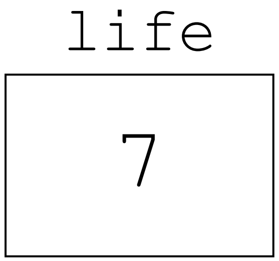
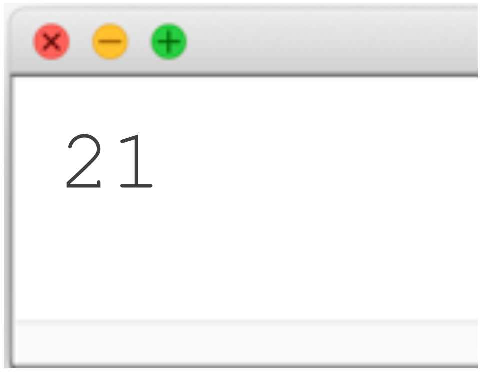
Methods

Updating Variables

Console Programs

```
int life = 42;
life = 42 - life;
life = 15;
life = life / 2;
println(life * 3);
```

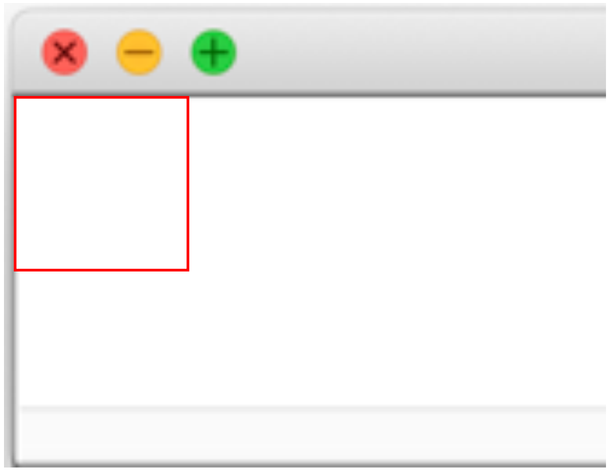


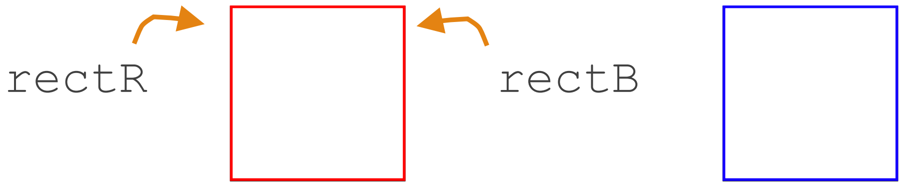


```
21
```

Graphics Programs

```
GRect rectR = new GRect(100, 100);
rectR.setColor(Color.RED);
GRect rectB = new GRect(100, 100);
rectB.setColor(Color.BLUE);
rectB = rectR;
add(rectB, 0, 0);
```





So Many Boxes

```
int life = 42;

double d = 14.0 / 2.5;

String s = "This is a string";

GRect rect = new GRect(width, height);

GRect rect = new GRect(x, y, width, height);
```

We can create many types of variables in Java!!

Animation loop

```java
int count = 0;
GLabel countDisplay = new GLabel("" + count);
add(countDisplay, 1,50);
while(true) {
    // updates text of label
    countDisplay.setLabel("" + count);
    count += 1;

    /* What happens when we insert
     * the code from cases 1, 2, and 3? */
}
```

(1)
```java
if (count > 10) {
    break;
}
pause(500);
```

(2)
```java
// nothing
```

(3)
```java
pause(500);
```

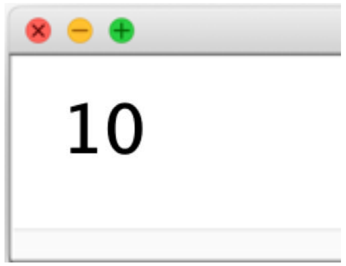
Animation loop

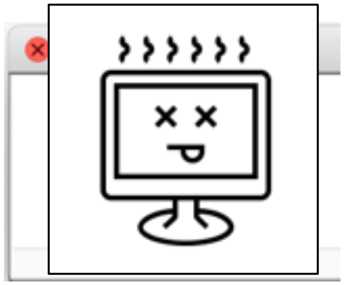
```
int count = 0;
GLabel countDisplay = new GLabel("" + count);
add(countDisplay, 1,50);
while(true) {
    // updates text of label
    countDisplay.setLabel("" + count);
    count += 1;

    /* What happens when we insert
     * the code from cases 1, 2, and 3? */
}
```

(1)
```
if (count > 10) {
    break;
}
pause(500);
```

(2)
```
// nothing
```

(3)
```
pause(500);
```





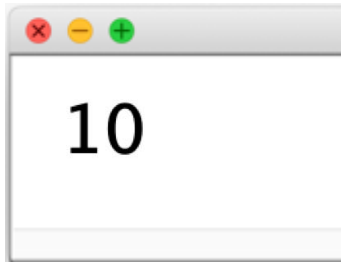
Animation loop

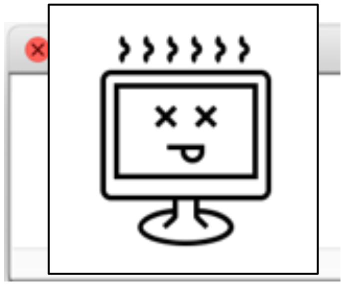
```
int count = 0;
GLabel countDisplay = new GLabel("" + count);
add(countDisplay, 1,50);
while(true) {
    // updates text of label
    countDisplay.setLabel("" + count);
    count += 1;

    /* What happens when we insert
     * the code from cases 1, 2, and 3? */
}
```

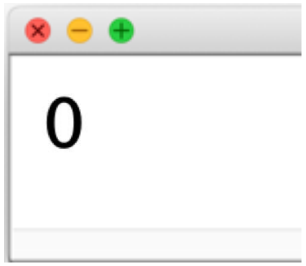
(1)
```
if (count > 10) {
    break;
}
pause(500);
```

(2)
```
// nothing
```

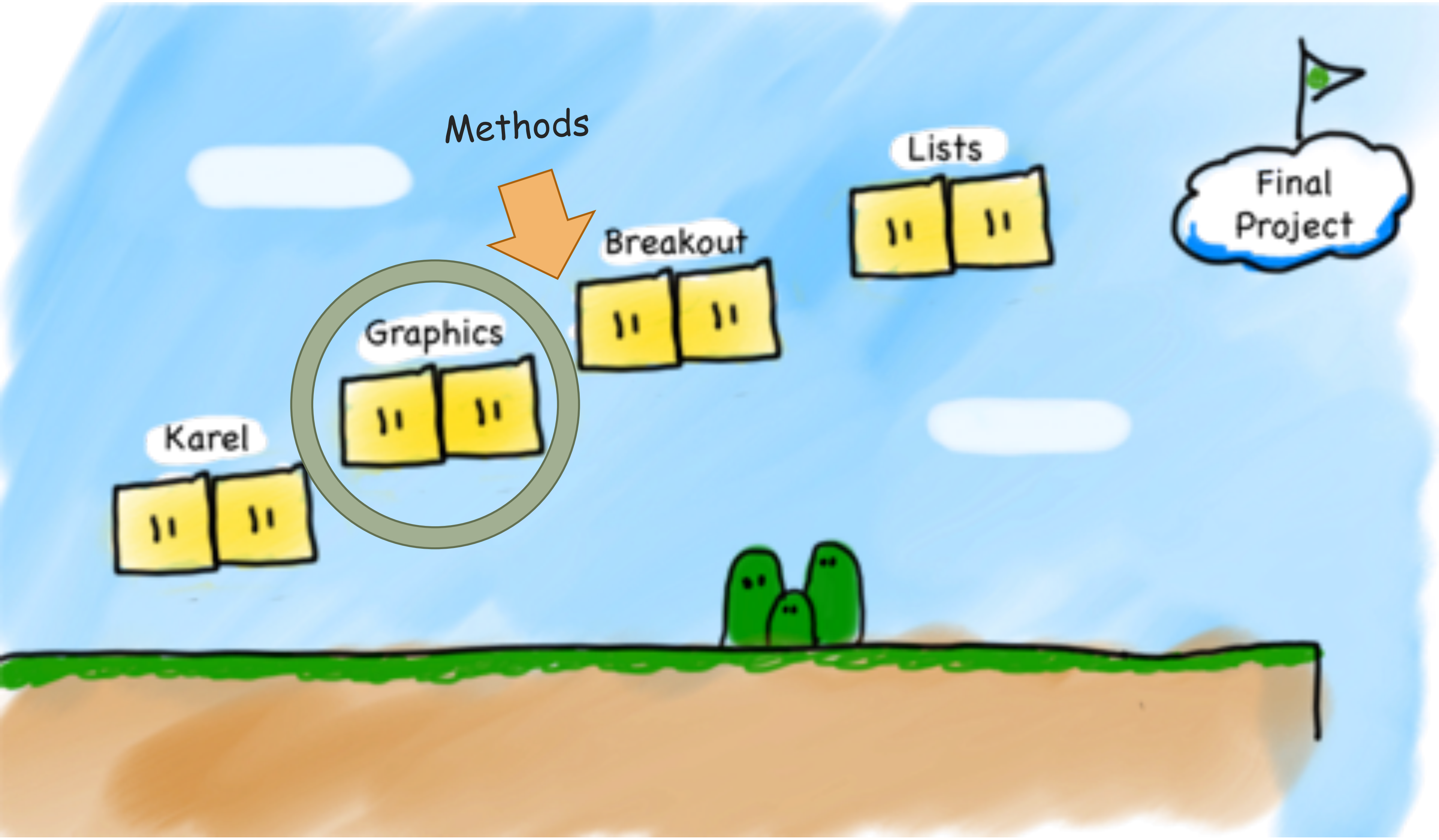
(3)
```
pause(500);
```





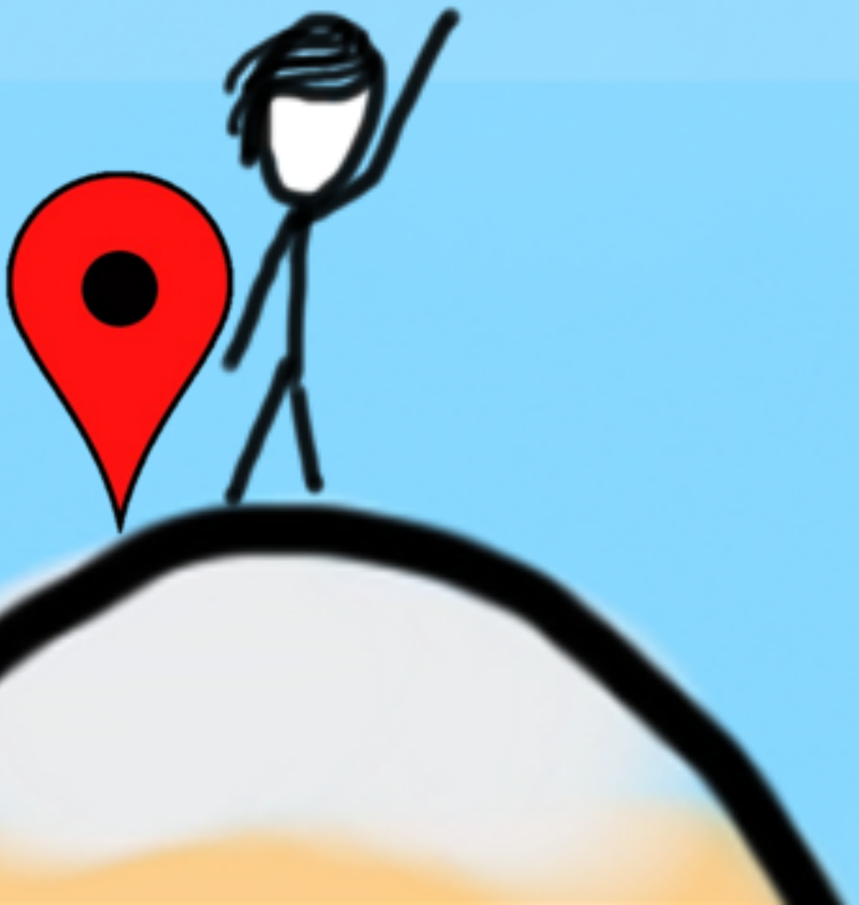


Our Second Step



Today's Goals

1. What is a method and how do we talk about it?
2. How do we define our own methods?
3. What is happening when we call a method?



Methods

```
turnRight();
```

```
move();
```

`readInt("Int please! ");`

`println("hello world");`

`rect.getX();`

```
drawRobotFace();
```

`rect.setLocation(10, 20);`

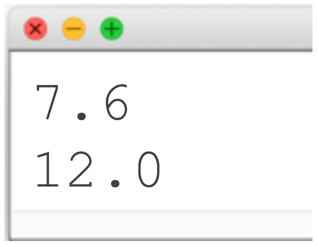
Today, we will learn exactly what these methods are doing!

Defining a Method

```
private void turnRight() {
    turnLeft();
    turnLeft();
    turnLeft();
}
```

Defining a Method

```
public void run() {
   printAverage1();
   printAverage2();
}

private void printAverage1() {
   double a = 5.0;
   double b = 10.2;
   double sum = a + b;
   double mid = sum / 2;
   println(mid);
}

private void printAverage2() {
   double a = 6; // int 6 → double 12.0
   double b = 18.0;
   double sum = a + b;
   double mid = sum / 2;
   println(mid);
}
```

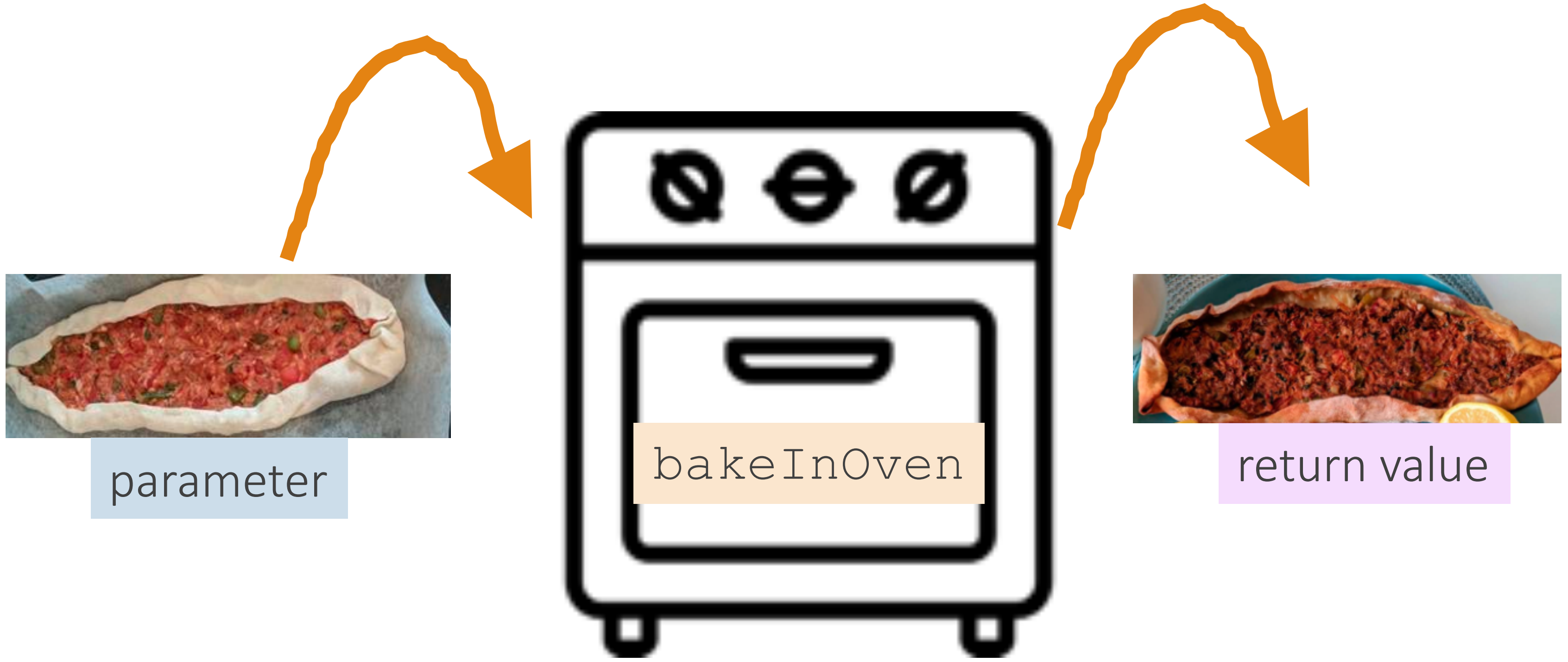


```
7.6
12.0
```

But wait…I thought methods help reuse code!

Methods are Ovens

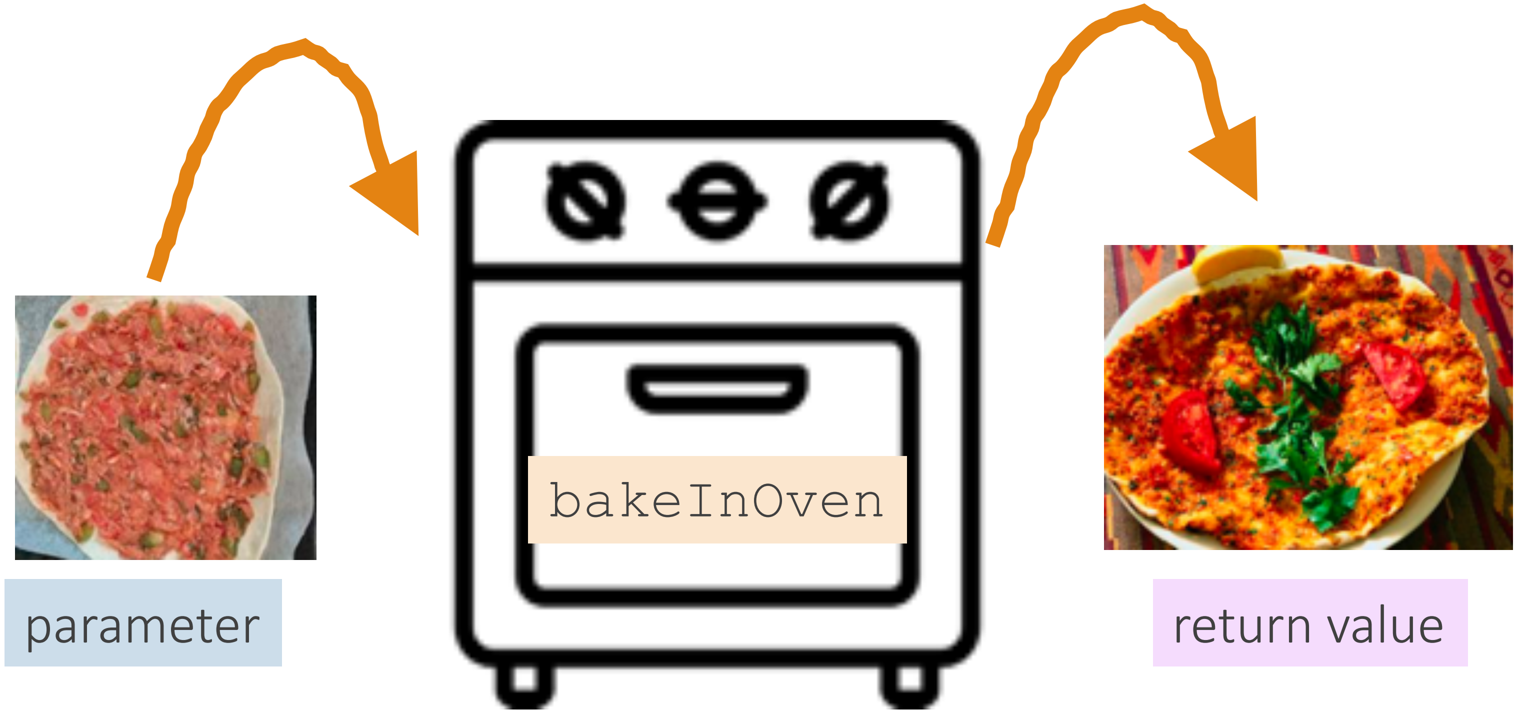
Java methods can take in data and return other data!!



Ovens are Methods

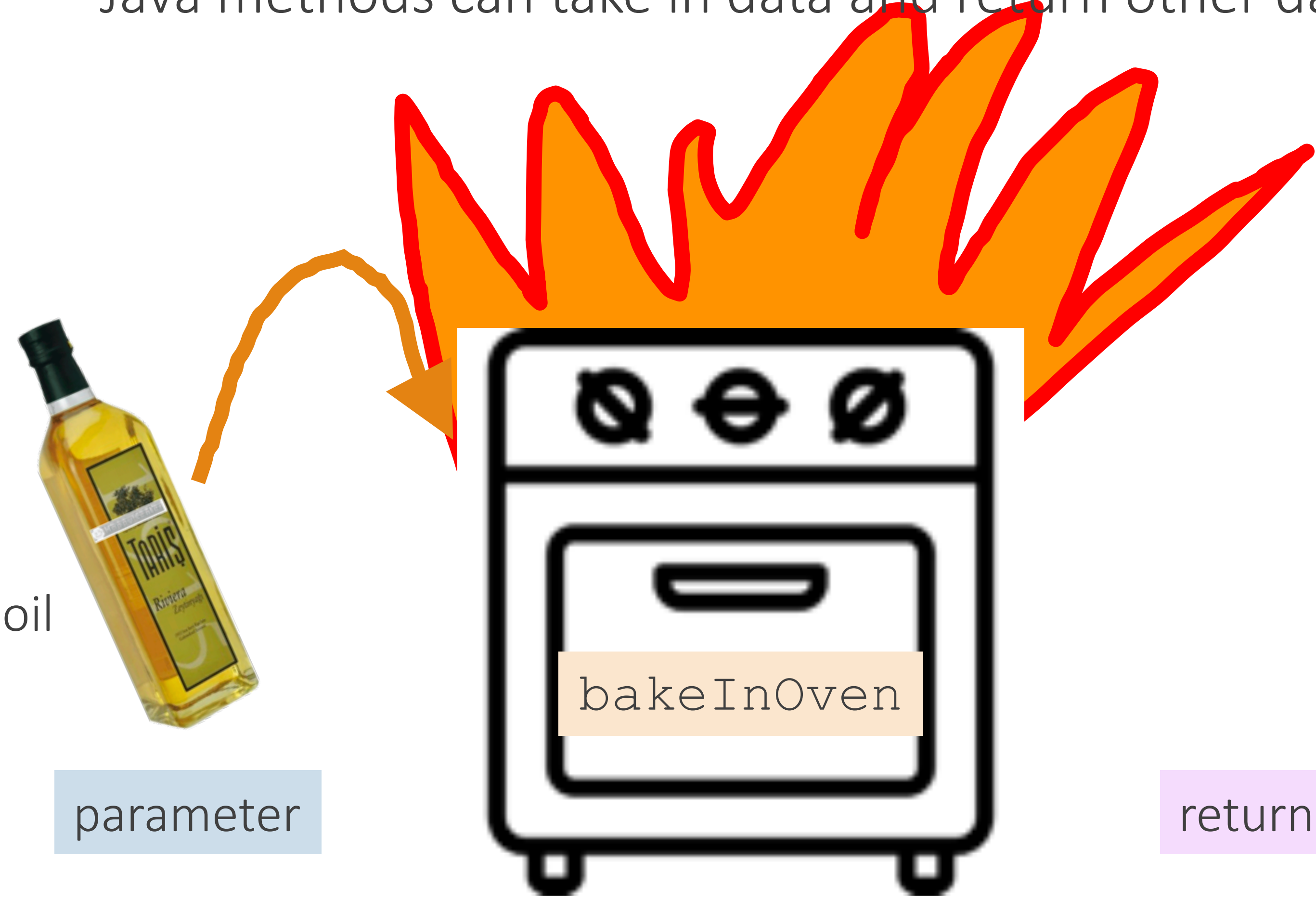
Java methods can take in data and return other data!!

You don’t need a different oven for lahmacun. Use the same one.



Ovens are Methods

Java methods can take in data and return other data!!



parameter

return value

Not all inputs work.

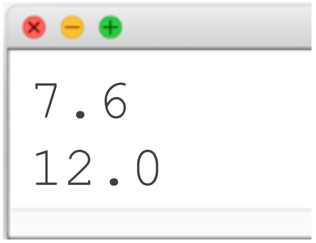
The Java method

```
public void run() {
    double mid1 = average(5.0, 10.2);
    println(mid1);
    double mid2 = average(6, 18);
    println(mid2);
}

private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```

method "call"

method name

method definition



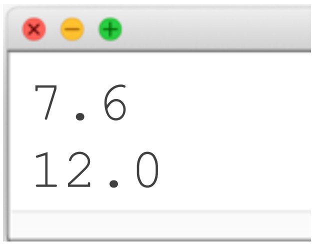
```
7.6
12.0
```

`average(double a, double b)` is a method that:

- Takes as input two **doubles** (a and b).
- Outputs a **double**
- Averages the two inputs.

The Algebra Version

```
public void run() {
    double mid1 = average(5.0, 10.2);
    println(mid1);
    double mid2 = average(6, 18);
    println(mid2);
}

private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```



```
7.6
12.0
```

Method definition:

$$f(a, b) = (a + b)/2$$

Method calls:

$$f(5.0, 10.2) = 7.6$$

$$f(6, 18) = 12.0$$

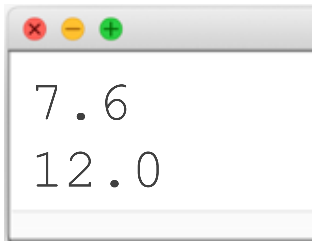
The Java method

```java
public void run() {
    double mid1 = average(5.0, 10.2);
    println(mid1);
    double mid2 = average(6, 18);
    println(mid2);
}
```

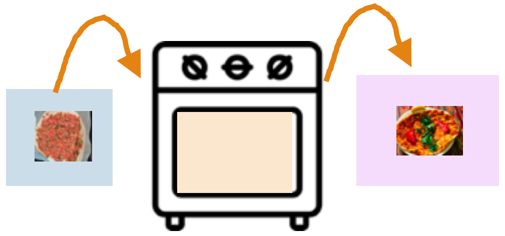
Return type

Parameters

```java
private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```



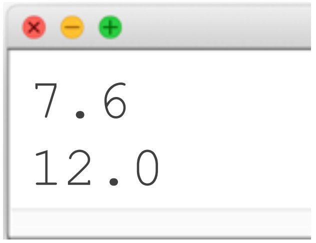
```
7.6
12.0
```



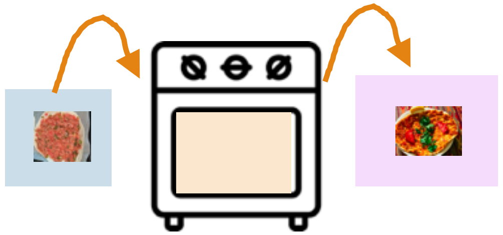
Anatomy of a method

```
public void run() {
    double mid1 = average(5.0, 10.2);
    println(mid1);
    double mid2 = average(6, 18);
    println(mid2);
}

private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```

$\left.\begin{array}{l}\texttt{double sum = a + b;}\\ \texttt{return }\underbrace{\texttt{sum / 2}}_{\text{return value}}\texttt{;}\end{array}\right\}$ method body



```
7.6
12.0
```



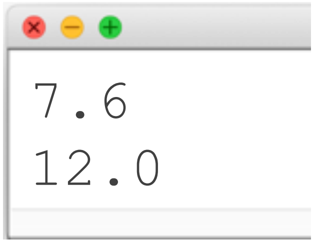
Calling and Defining Methods

arguments

method "call"

```
public void run() {
    double mid1 = average(5.0, 10.2);
    println(mid1);
    double mid2 = average(6, 18);
    println(mid2);
}

parameters

private double average(double a, double b) {
    double sum = a + b;
    return sum / 2;
}
```

method definition

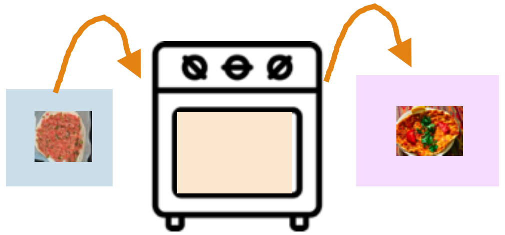


```
7.6
12.0
```

arguments: calling (with actual **int** values)

vs

parameters: defining method input (any **int**)



Explaining the `void` and the `()`

```
public void run() {

    printIntro();

}
```

return type name parameters

```
private void printIntro( ) {
    println("Welcome to class");
    println("It's the best part of my day.");
    // nothing here
}
```

⚠️ `void` methods don't need a `return`.

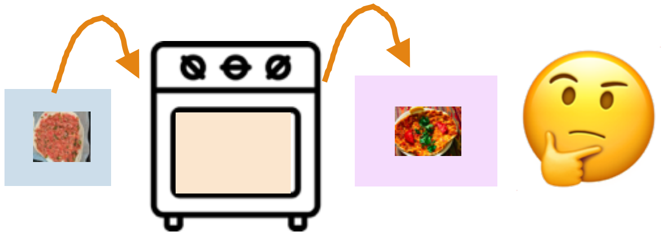
`printIntro()` is a method that:

- Takes no parameters.
- Returns nothing.
- It just always prints:

```
Welcome to class
It's the best part of my day.
```

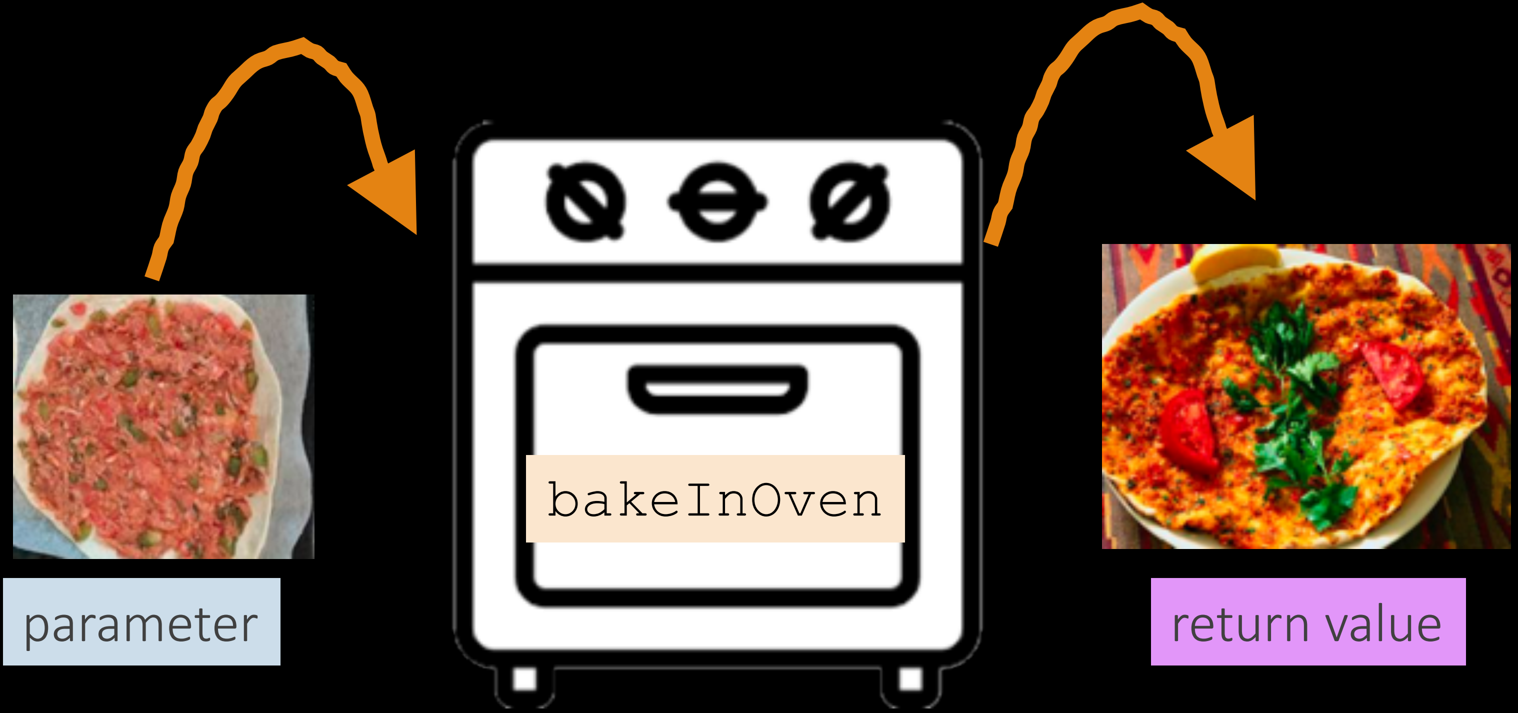
Methods Dear to Our Heart

Method call	Parameter Types?	Return Types?
`average(5.0, 10.2);`	double,double	double
`printIntro();`	(nothing)	void
`turnRight();`	(nothing)	void
`readInt("Enter age: ");`	String	int
`println("You're cool!");`	String	void
`getWidth();`	(nothing)	double
`rect.setLocation(10,20);`	double,double	void



Questions?

TL;DR: (too long; don't read)
(means the summary of what just happened)

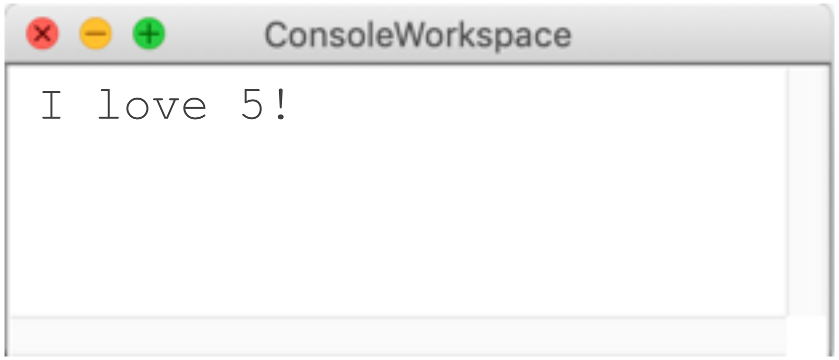


Today's Goals

✓ 1. What is a method and how do we talk about it?
2. How do we define our own methods?
3. What is happening when we call a method?

Parameter Example

```java
public void run() {
    printOpinion(5);
}

private void printOpinion(int num) {
   if(num == 5) {
      println("I love 5!");
   } else {
      println("Whatever");
   }
}
```



```
I love 5!
```

Multiple Returns are OK

```
private String getMonthName(int index) {
    if (index == 0) {
        return "January";
    }
    if (index == 1) {
        return "February";
    }
    ...
    return "Unknown";
}
```

`getMonthName(0)?`	`getMonthName(1)?`	`getMonthName(200)?`
returns	returns	returns
`"January"`	`"February"`	`"Unknown"`

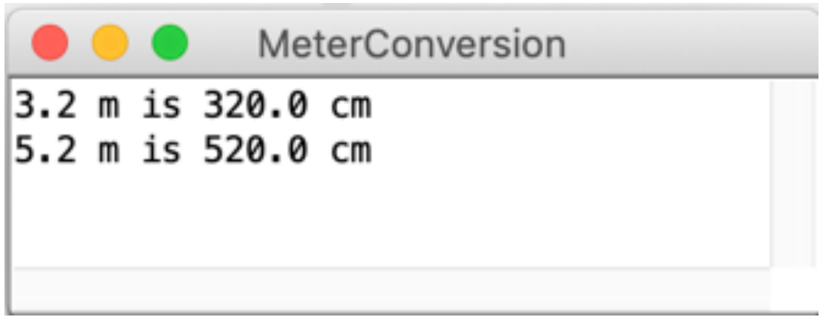
Multiple Returns are OK, but...

```
private String getMonthName(int index) {
    if (index == 0) {
        return "January";
    }
    if (index == 1) {
        return "February";
    }
    ...
    // return "Unknown";
}
```

This method must return a result of type String

For all possible arguments of this type, *something* must be returned!

Parameter + Returns



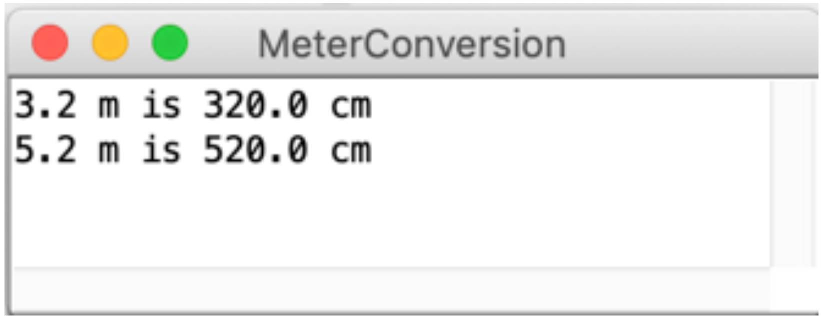
```
3.2 m is 320.0 cm
5.2 m is 520.0 cm
```

```
public void run() {
    double conversion = metersToCm(3.2);
    println("3.2 m is " + conversion + " cm");
    println("5.2 m is " + metersToCm(5.2) + " cm");
}

private ??????? metersToCm(???????) {

    /* Fill this in too */

}
```

Step (1)

Step (2)

Parameter + Returns

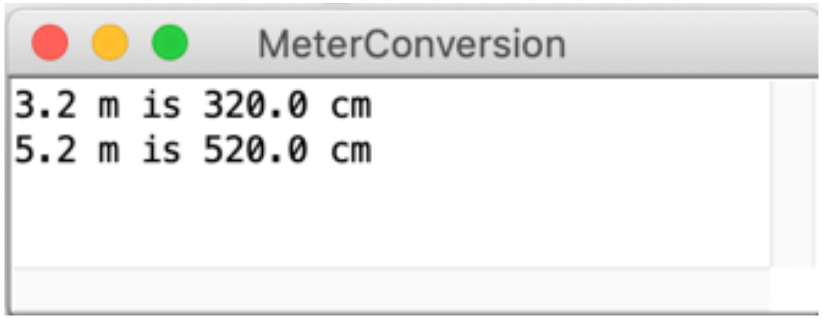


```
3.2 m is 320.0 cm
5.2 m is 520.0 cm
```

```
public void run() {
    double conversion = metersToCm(3.2);
    println(”3.2 m is " + conversion + " cm");
    println("5.2 m is " + metersToCm(5.2) + " cm");
}

private double metersToCm(double meters) {

    return meters * 100;

}
```

⚠️ You must name your input variables!

Parameter + Returns



```
3.2 m is 320.0 cm
5.2 m is 520.0 cm
```

```
public void run() {
    double conversion = metersToCm(3.2);
    println("3.2 m is " + conversion + " cm");
    println("5.2 m is " + metersToCm(5.2) + " cm");
}

private double metersToCm(double meters) {

    return meters*100;

}
```

⚠️ Any non-`void` method must `return` something!

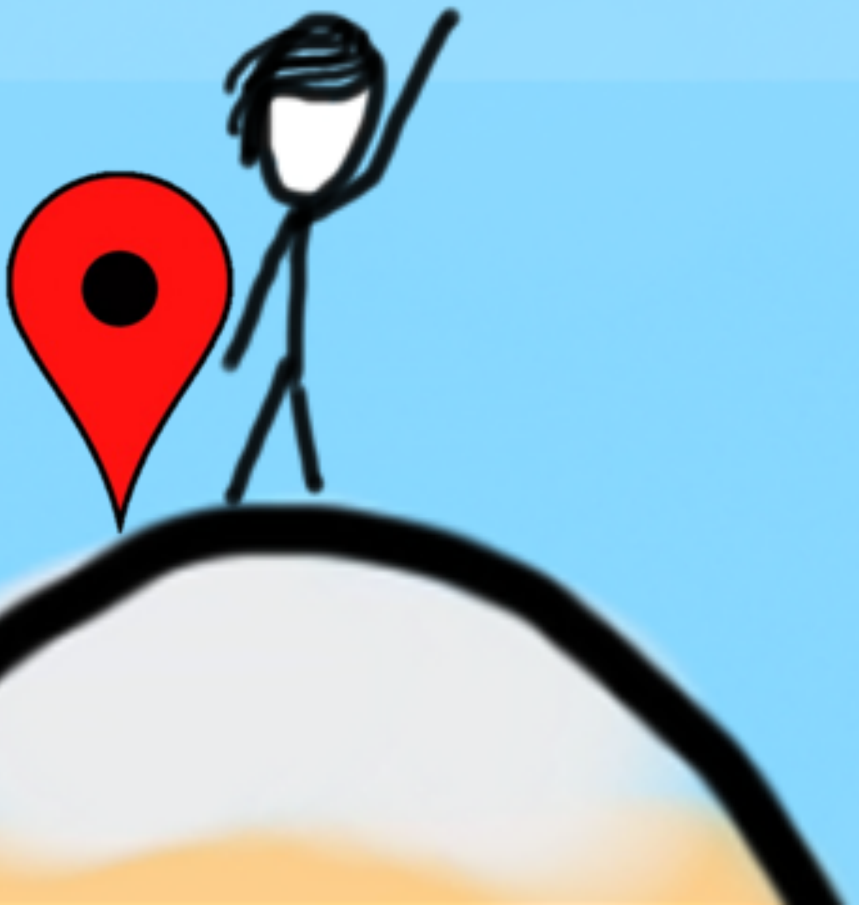
Summary: Defining a Method

```
visibility type nameOfMethod(parameter types and names) {
        statements
}
```

- *visibility:* usually **`private`** or **`public`**
- *type*: type returned by method
  - **`int`**, **`double`** , etc. must include a **`double`** value!
  - Can be **`void`** to indicate that nothing is returned
- Input ***parameters:*** information passed into method
  - Must declare variable type AND variable name! (like **`double`** `meter`)
  - Can be empty `()`

Today's Goals

✓ 1. What is a method and how do we talk about it?
✓ 2. How do we define our own methods?
3. What is happening when we call a method?

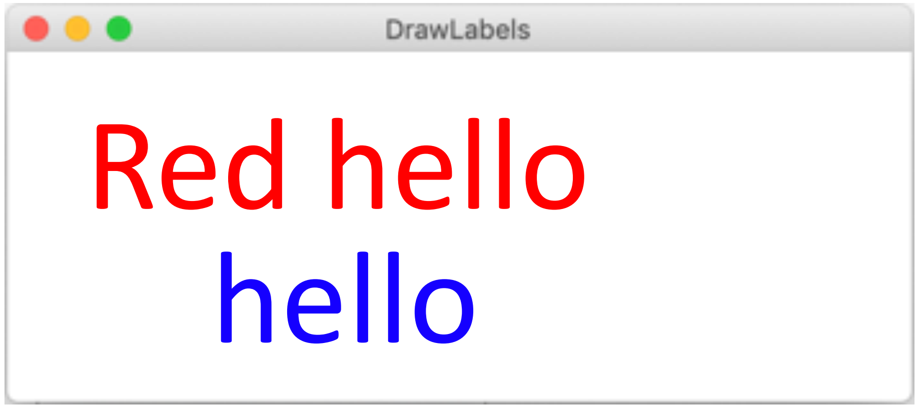


Java Execution of Methods

"equals" (1) Evaluate right hand side

= (2) Store result in variable on left hand side

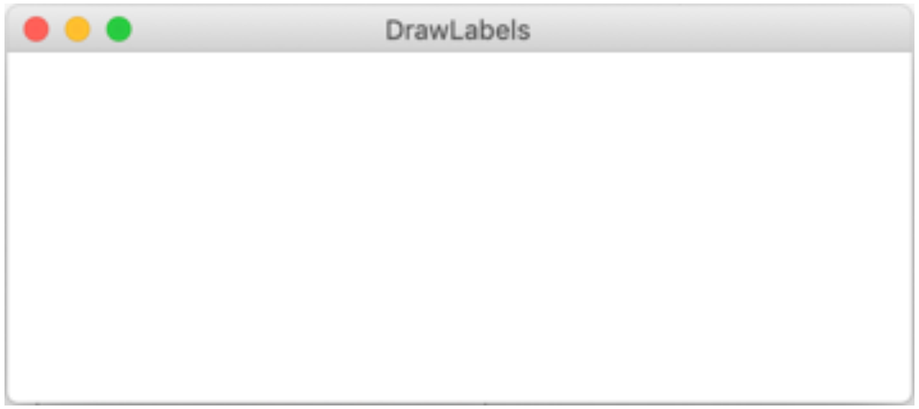
```java
public void run() {
    GLabel redLabel = coloredLabel("Red hello", Color.RED);
    add(redLabel, 50, 50);
    GLabel label = coloredLabel("hello", Color.BLUE);
    add(label, 100, 100);
}


private GLabel coloredLabel(String text, Color fill) {
    GLabel label = new GLabel(text);
    label.setFont("Calibri-50");
    label.setColor(fill);
    return label;
}
```



What happens when we run this program?

```
public void run() {
    GLabel redLabel = coloredLabel("Red hello", Color.RED);
    add(redLabel, 50, 50);
    GLabel label = coloredLabel(”hello", Color.BLUE);
    add(label, 100, 100);
}
```

```
private GLabel coloredLabel(String text, Color fill) {
    GLabel label = new GLabel(text);
    label.setFont("Calibri-50");
    label.setColor(fill);
    return label;
}
```



What happens when we run this program?

```
public void run() {
    GLabel redLabel = coloredLabel("Red hello", Color.RED);
    add(redLabel, 50, 50);
    GLabel label = coloredLabel(”hello", Color.BLUE);
    add(label, 100, 100);
}
```

```
private GLabel coloredLabel(String text, Color fill) {
    GLabel label = new GLabel(text);
    label.setFont("Calibri-50");
    label.setColor(fill);
    return label;
}
```

(1) Evaluate right hand side
(2) Store result in variable on left hand side

```
public void run() {
    GLabel redLabel = coloredLabel("Red hello", Color.RED);
    add(redLabel, 50, 50);
    GLabel label = coloredLabel("hello", Color.BLUE);
    add(label, 100, 100);
}
```

```
private GLabel coloredLabel(String text, Color fill) {
    GLabel label = new GLabel(text);
    label.setFont("Calibri-50");
    label.setColor(fill);
    return label;
}
```

text "Red hello"   fill Color.RED

(1) Evaluate right hand side
(2) Store result in variable on left hand side

```
public void run() {
     GLabel redLabel = coloredLabel("Red hello", Color.RED);
     add(redLabel, 50, 50);
     GLabel label = coloredLabel(”hello", Color.BLUE);
     add(label, 100, 100);
}
```

```
private GLabel coloredLabel(String text, Color fill) {
     GLabel label = new GLabel(text);
     label.setFont("Calibri-50");
     label.setColor(fill);
     return label;
}
```

text "Red hello"   fill Color.RED   label

Red hello

```
public void run() {
    GLabel redLabel = coloredLabel("Red hello", Color.RED);
    add(redLabel, 50, 50);
    GLabel label = coloredLabel("hello", Color.BLUE);
    add(label, 100, 100);
}
```

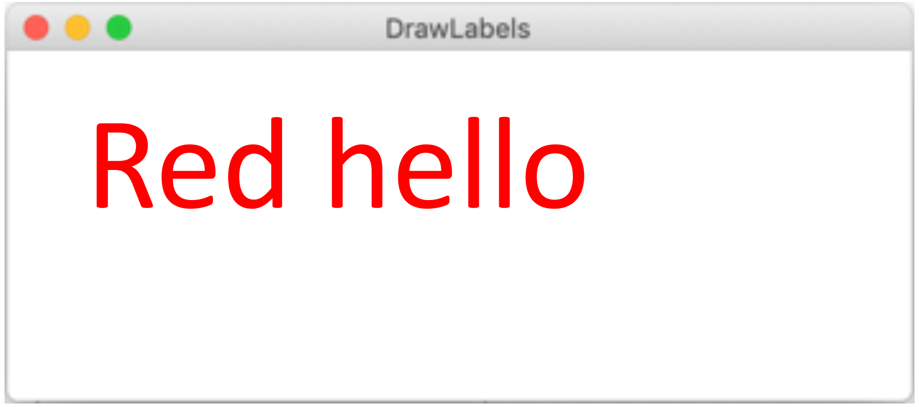
```
private GLabel coloredLabel(String text, Color fill) {
    GLabel label = new GLabel(text);
    label.setFont("Calibri-50");
    label.setColor(fill);
    return label;
}
```

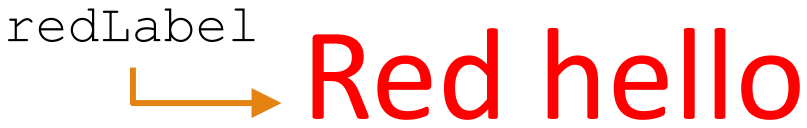
text `"Red hello"`   fill `Color.RED`   label

Red hello

```
public void run() {
     GLabel redLabel = coloredLabel("Red hello", Color.RED);
     add(redLabel, 50, 50);
     GLabel label = coloredLabel("hello", Color.BLUE);
     add(label, 100, 100);
}
```

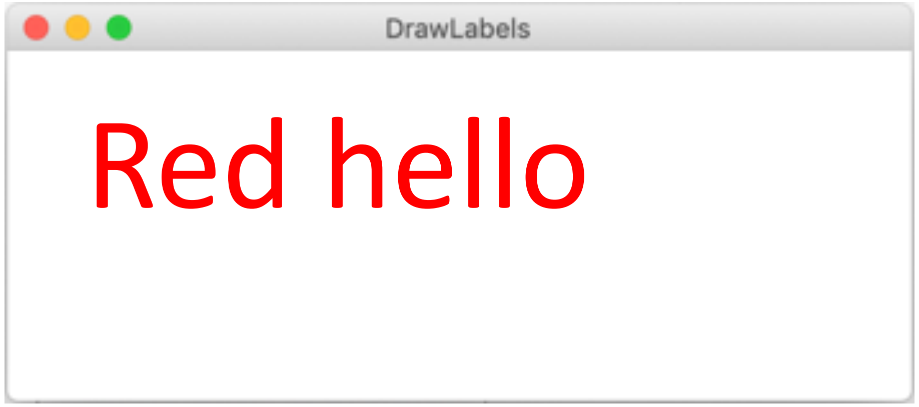
```
private GLabel coloredLabel(String text, Color fill) {
     GLabel label = new GLabel(text);
     label.setFont("Calibri-50");
     label.setColor(fill);
     return label;
}
```

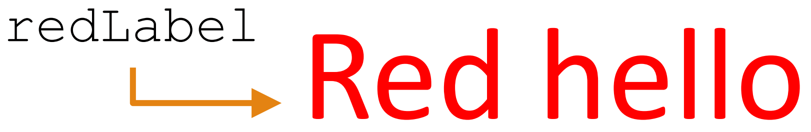




```
public void run() {
    GLabel redLabel = coloredLabel("Red hello", Color.RED);
    add(redLabel, 50, 50);
    GLabel label = coloredLabel(”hello", Color.BLUE);
    add(label, 100, 100);
}
```

```
private GLabel coloredLabel(String text, Color fill) {
    GLabel label = new GLabel(text);
    label.setFont("Calibri-50");
    label.setColor(fill);
    return label;
}
```

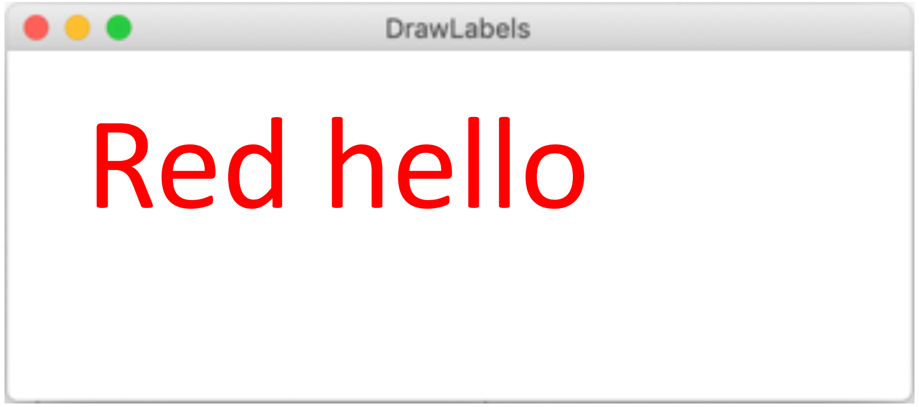




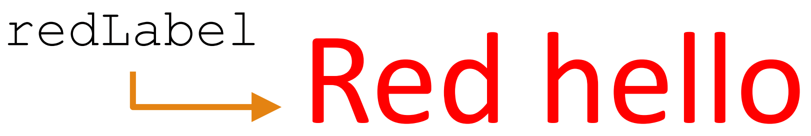
```
public void run() {
    GLabel redLabel = coloredLabel("Red hello", Color.RED);
    add(redLabel, 50, 50);
    GLabel label = coloredLabel("hello", Color.BLUE);
    add(label, 100, 100);
}
```

```
private GLabel coloredLabel(String text, Color fill) {
    GLabel label = new GLabel(text);
    label.setFont("Calibri-50");
    label.setColor(fill);
    return label;
}
```

text $\boxed{\texttt{"hello"}}$ fill $\boxed{\texttt{Color.BLUE}}$



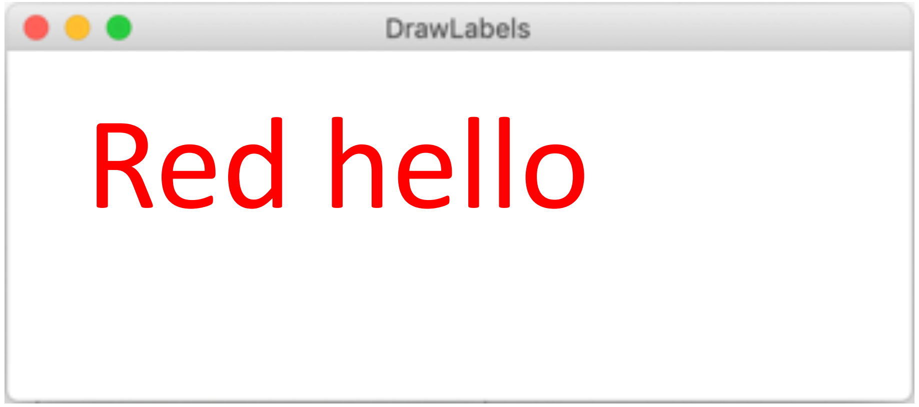
Red hello

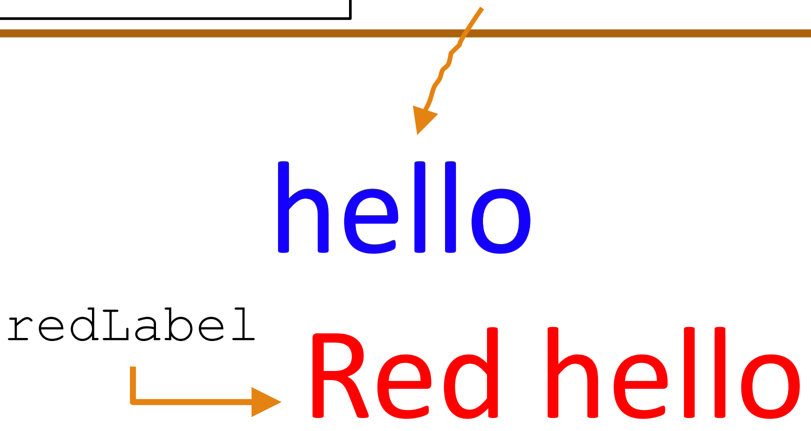


```
public void run() {
    GLabel redLabel = coloredLabel("Red hello", Color.RED);
    add(redLabel, 50, 50);
    GLabel label = coloredLabel("hello", Color.BLUE);
    add(label, 100, 100);
}
```

```
private GLabel coloredLabel(String text, Color fill) {
    GLabel label = new GLabel(text);
    label.setFont("Calibri-50");
    label.setColor(fill);
    return label;
}
```

text "hello"   fill Color.BLUE   label

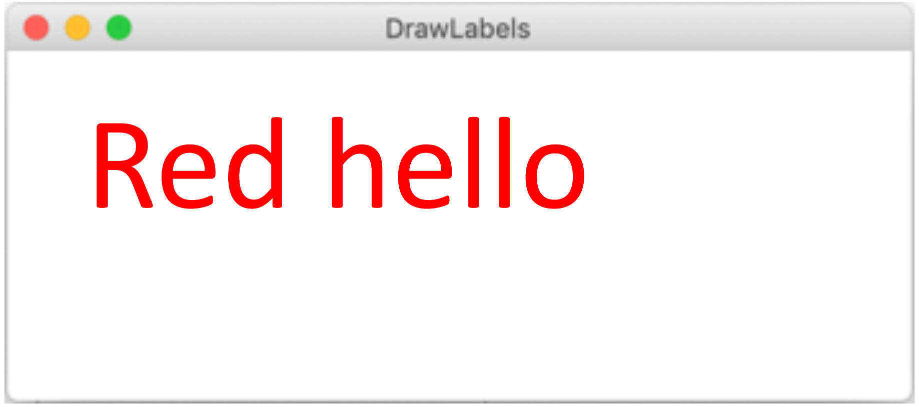




```
public void run() {
    GLabel redLabel = coloredLabel("Red hello", Color.RED);
    add(redLabel, 50, 50);
    GLabel label = coloredLabel("hello", Color.BLUE);
    add(label, 100, 100);
}
```

```
private GLabel coloredLabel(String text, Color fill) {
    GLabel label = new GLabel(text);
    label.setFont("Calibri-50");
    label.setColor(fill);
    return label;
}
```

text `"hello"` fill `Color.BLUE` label



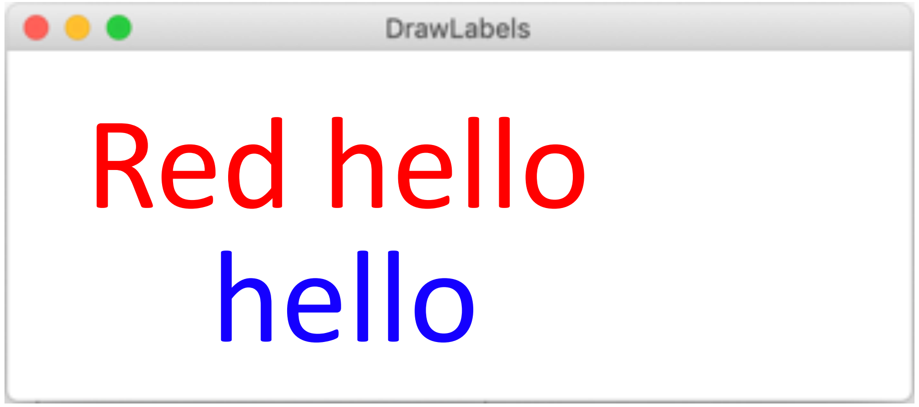
hello

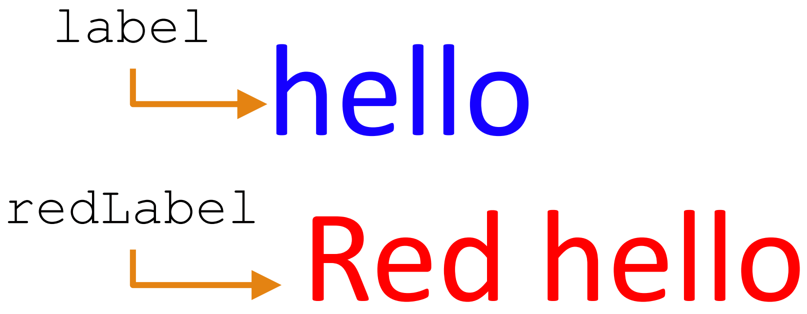
redLabel

Red hello

```
public void run() {
    GLabel redLabel = coloredLabel("Red hello", Color.RED);
    add(redLabel, 50, 50);
    GLabel label = coloredLabel("hello", Color.BLUE);
    add(label, 100, 100);
}
```

```
private GLabel coloredLabel(String text, Color fill) {
    GLabel label = new GLabel(text);
    label.setFont("Calibri-50");
    label.setColor(fill);
    return label;
}
```

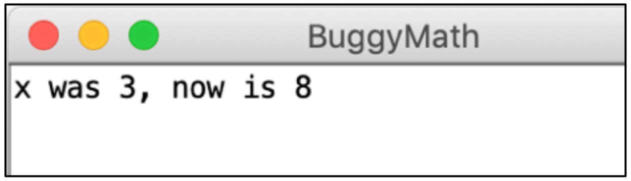




Questions?

More Examples

Bad Times with Methods



```
x was 3, now is 8
```

(intention)

```
public void run() {
    int x = 3;
    int prevX = x;
    addFive(x);
    println("x was " + prevX + ", now" + x);
}

private void addFive(int x) {
    x += 5;
    println(x);
}
```

There are <u>three</u> bugs in this program!

Good Times with Methods

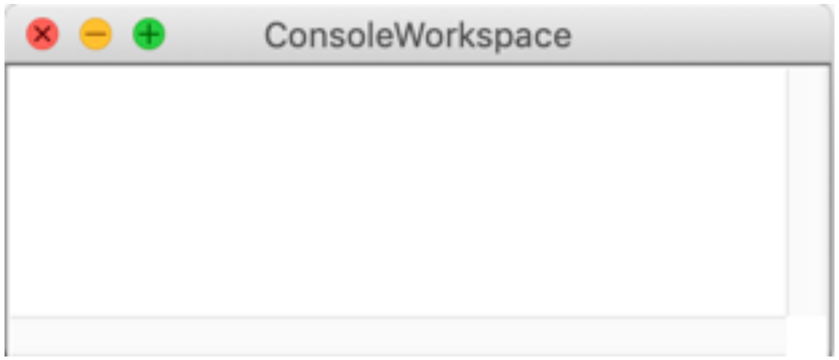
```
public void run() {
    int x = 3;
    int prevX = x;
    x = addFive(x);
    println("x was " + prevX + ", now" + x);
}

private int addFive(int x) {
    x += 5;
    return x;
}
```

At the end of these slides, there is a walkthrough of how Java runs this program.

That's more like it!

Changed Name

```
private void run() {
    int num = 5;
    cow(num);
}

private void cow(int grass) {
    println(grass);
}
```

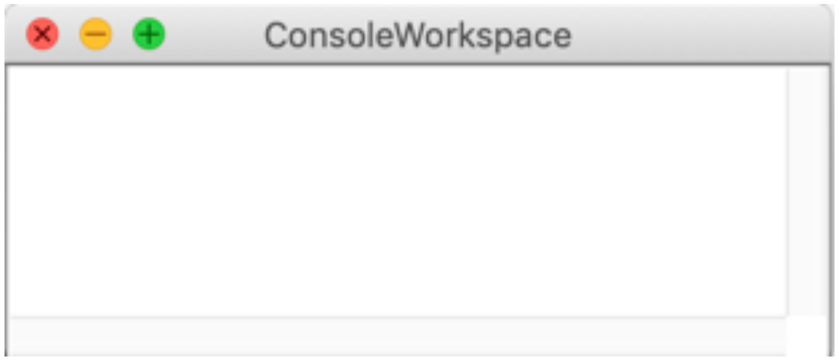


Changed Name

```
private void run() {
    int num = 5;
    cow(num);
}
```

num 5

```
private void cow(int grass) {
    println(grass);
}
```



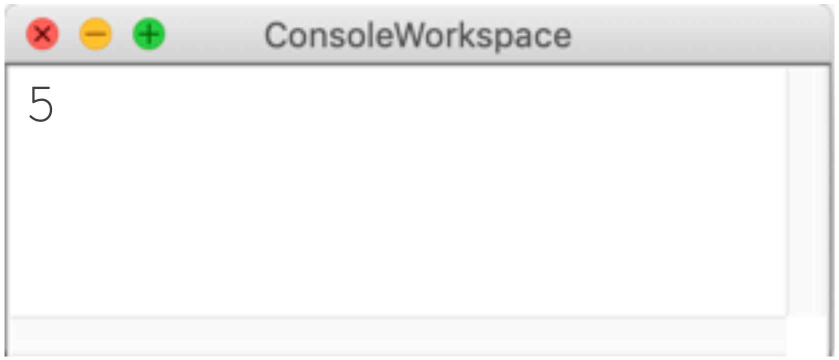
Changed Name

```
private void run() {
    int num = 5;
    cow(num);
}
```

num 5

```
private void cow(int grass) {
    println(grass);
}
```

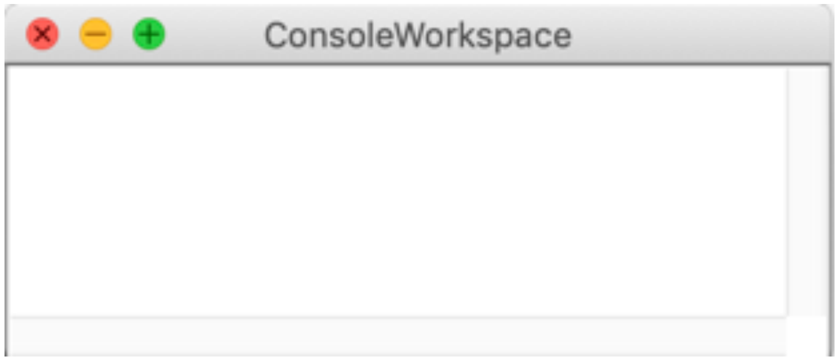
grass 5



```
5
```

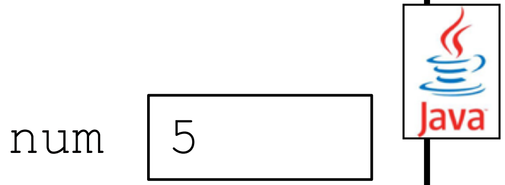
Same Variable

```
private void run() {
    int num = 5;
    cat();
}

private void cat() {
    int num = 10;
    println(num);
}
```

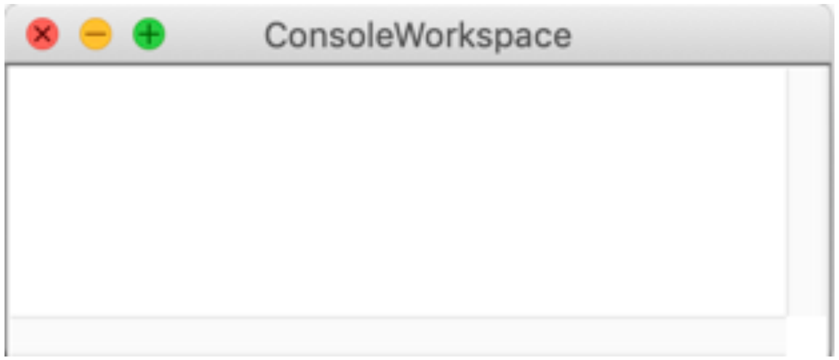


Same Variable

```
private void run() {
    int num = 5;
    cat();
}
```



```
private void cat() {
    int num = 10;
    println(num);
}
```



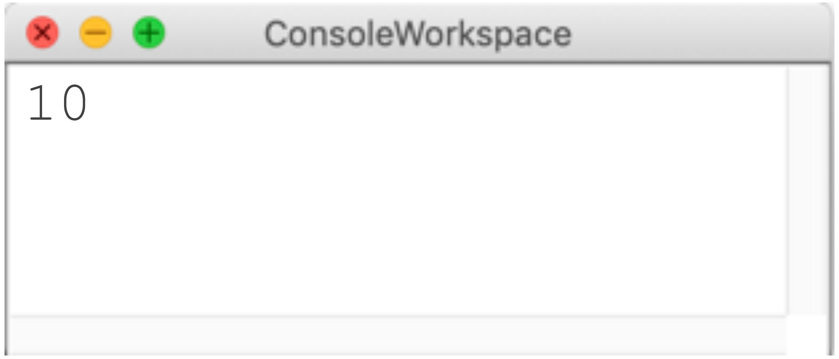
Same Variable

```
private void run() {
    int num = 5;
    cat();
}
```

num 5

```
private void cat() {
    int num = 10;
    println(num);
}
```

num 10


ConsoleWorkspace

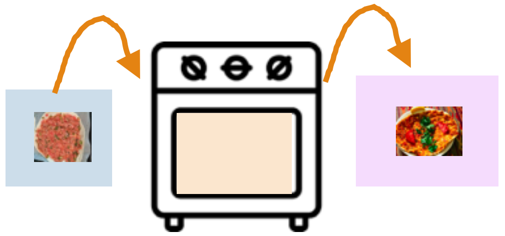
```
10
```

Today's Goals

✓ 1. What is a method and how do we talk about it?
✓ 2. How do we define our own methods?
✓ 3. What is happening when we call a method?

Review

A method:



```
private int addFive(int x) {
     x += 5;
    return x;
}
```

If you declare a return type,
you must return a value of that type.

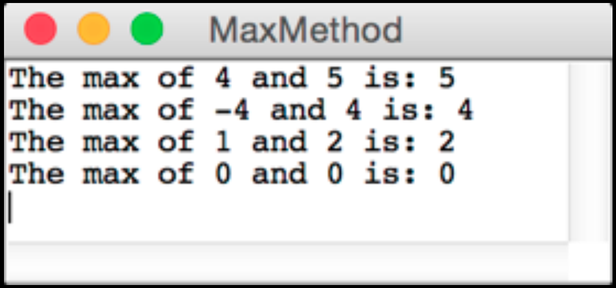
```
private void cat() {
    int num = 10;
    println(num);
}
```

`void`: no return values
`()`: no parameters.
`println()` is NOT **`return`**!!!!

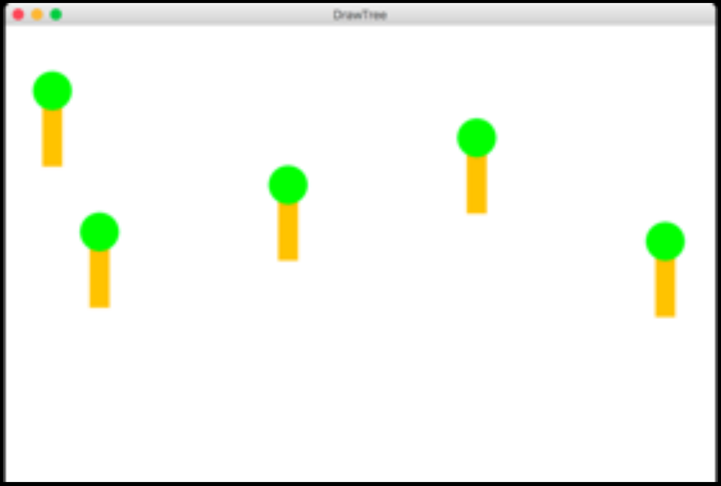
Today’s material is *difficult*.

👍 Good job surviving 👍

🙋‍♂️Bring your questions to section!🙋‍♀️



```
The max of 4 and 5 is: 5
The max of -4 and 4 is: 4
The max of 1 and 2 is: 2
The max of 0 and 0 is: 0
```

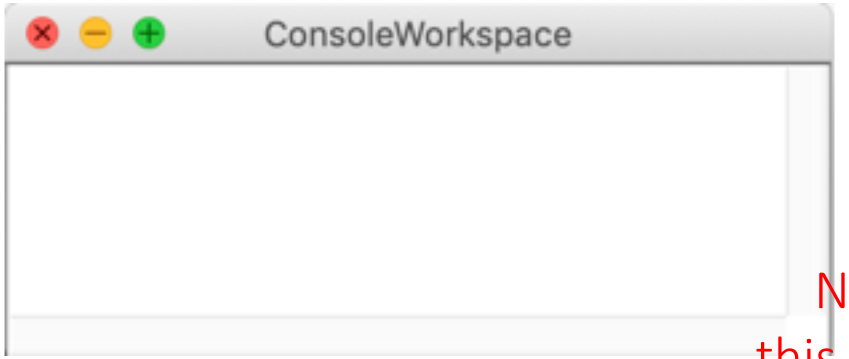


Mad Methods

Two different x's (for your viewing pleasure)

```
public void run() {
    int x = 3;
    int prevX = x;
    x = addFive(x);
    println("x was " + prevX + ", now" + x);
}

private int addFive(int x) {
    x += 5;
    return x;
}
```

These are not the same `x`!

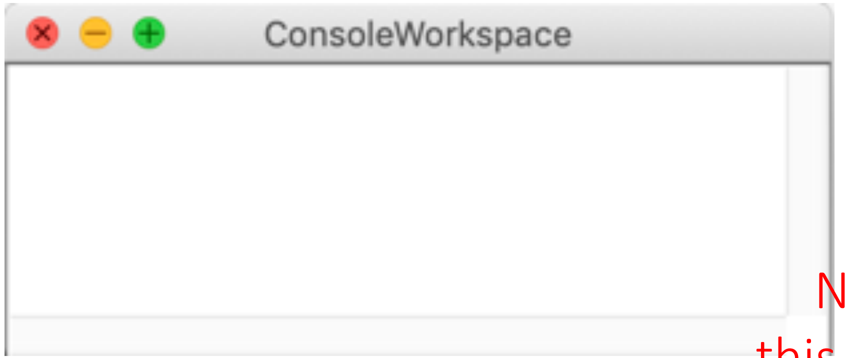


Not covered in lecture;
this is just for clarification

Two different x's (for your viewing pleasure)

```
public void run() {
    int x = 3;
    int prevX = x;
    x = addFive(x);
    println("x was " + prevX + ", now" + x);
}

private int addFive(int y) {
    y += 5;
    return y;
}
```

Let's rename this one.



Not covered in lecture;
this is just for clarification

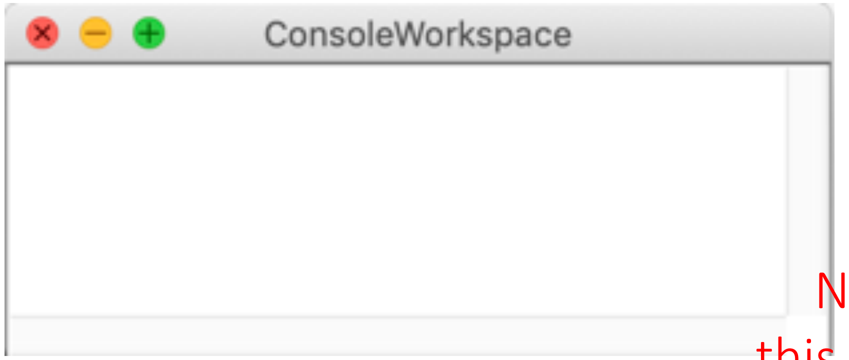
Two different x's (for your viewing pleasure)

```
public void run() {
    int x = 3;
    int prevX = x;
    x = addFive(x);
    println("x was " + prevX + ", now" + x);
}
```

x: 3

prevX: 3

```
private int addFive(int y) {
    y += 5;
    return y;
}
```



Not covered in lecture;
this is just for clarification

Two different x's (for your viewing pleasure)

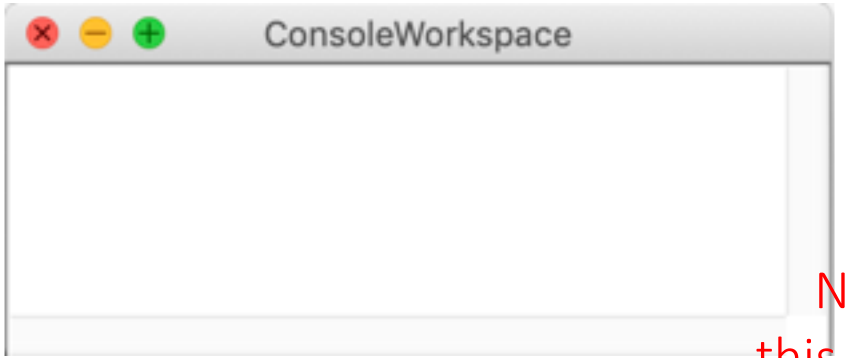
```
public void run() {
    int x = 3;
    int prevX = x;
    x = addFive(3);
    println("x was " + prevX + ", now" + x);
}
```

x 3

prevX 3

```
private int addFive(int y) {
    y += 5;
    return y;
}
```

y 3



Not covered in lecture; this is just for clarification

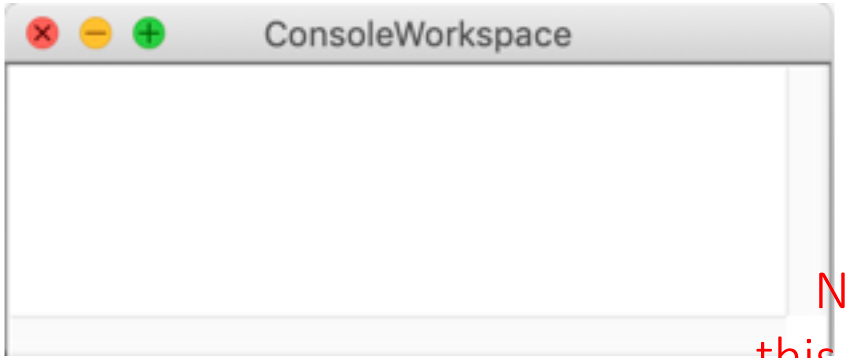
Two different x's (for your viewing pleasure)

```
public void run() {
    int x = 3;
    int prevX = x;
    x = addFive(x);
    println("x was " + prevX + ", now" + x);
}
```

x	
prevX	3

```
private int addFive(int y) {
    y += 5;
    return y;
}
```

y	8



Not covered in lecture;
this is just for clarification

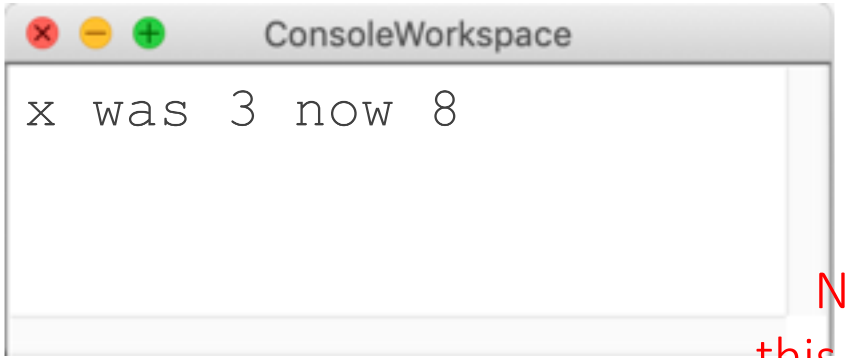
Two different x's (for your viewing pleasure)

```
public void run() {
     int x = 3;
     int prevX = x;
     x = addFive(x);
     println("x was " + prevX + ", now" + x);
}
```

x: 8

prevX: 3

```
private int addFive(int y) {
     y += 5;
     return y;
}
```



ConsoleWorkspace

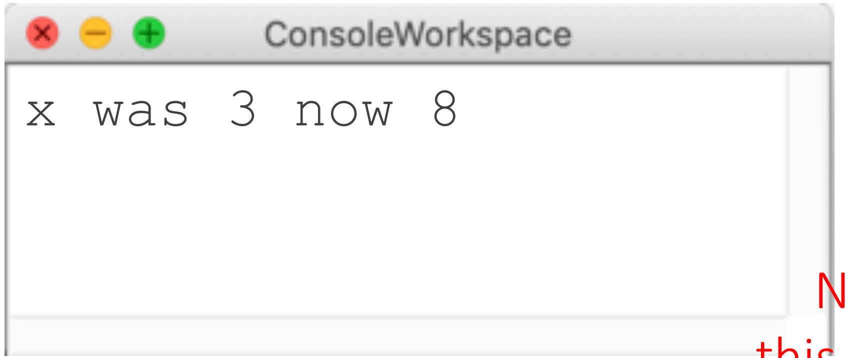
```
x was 3 now 8
```

Not covered in lecture; this is just for clarification

Two different x's (for your viewing pleasure)

```
public void run() {
    int x = 3;
    int prevX = x;
    x = addFive(x);
    println("x was " + prevX + ", now" + x);
}

private int addFive(int x) {
    x += 5;
    return x;
}
```

Renaming this back to x does not change the program behavior!



```
x was 3 now 8
```

Not covered in lecture; this is just for clarification