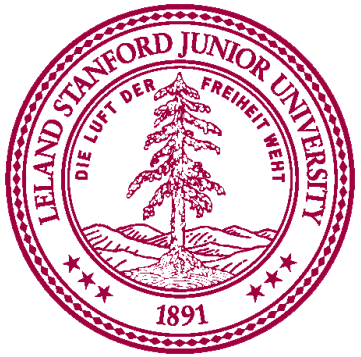
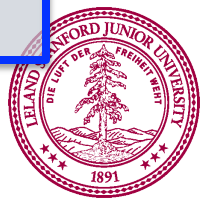


“Con un **método** y con **lógica** puedes alcanzar cualquier cosa”

(Agatha Christie, Pairot Investigates, 1924)

Métodos



Lo que necesitamos para hoy

Vamos a aprender un concepto difícil, prepárensen para la aventura

WhatsApp - Android

- ¿Cómo construyes algo tan grande como WhatsApp?

- Necesitas “algo” para partirlo en diferentes partes…

Los métodos son la solución

Civilization advances by extending the number of operations we can perform without thinking about them.

-Alfred North Whitehead

Objetivos de aprendizaje:

1. Escribir un método que toma un insumo de entrada
2. Escribir un método que devuelve un *output*
3. Practicar usando tus propios métodos

Llamando a los métodos

```
girarDerecha();

moverse();    leerEntero("Entero por favor!");

imprimir("hola mundo");

rect.cambiarRelleno(true);

dibujarCaraRobot();

agregar(rect);

int miDado = intAleatorio(1, 6);
```


Definiendo un método

```
private void girarDerecha(){
    girarIzquierda();
    girarIzquierda ();
    girarIzquierda ();
}
```

¡La GRAN diferencia con los métodos en Java:
Los métodos de Java pueden **recibir información** y **devolver información**!

Los métodos son como una tostadora

- Gracias Mehran

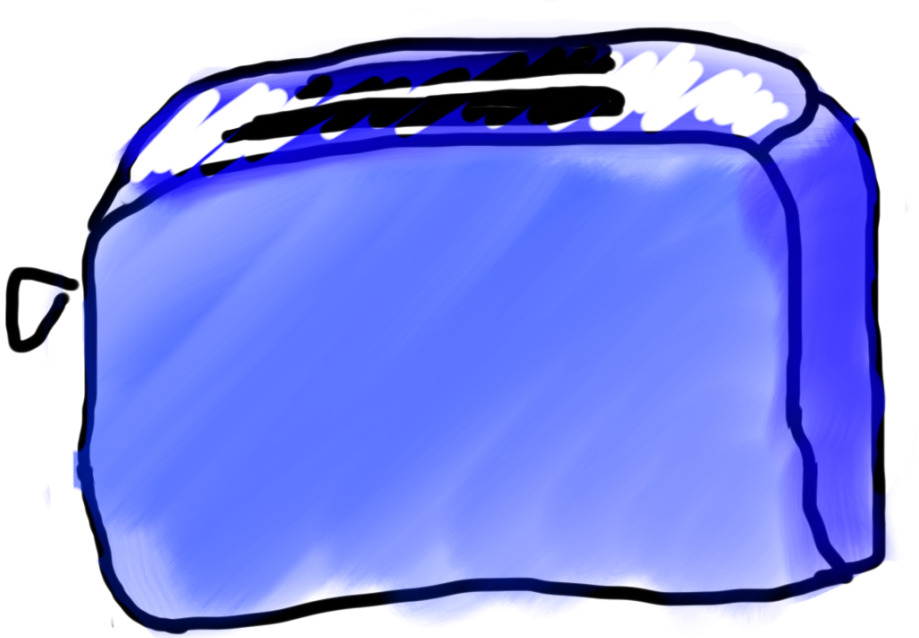
Los métodos son como una tostadora

parámetro

Los métodos son como una tostadora

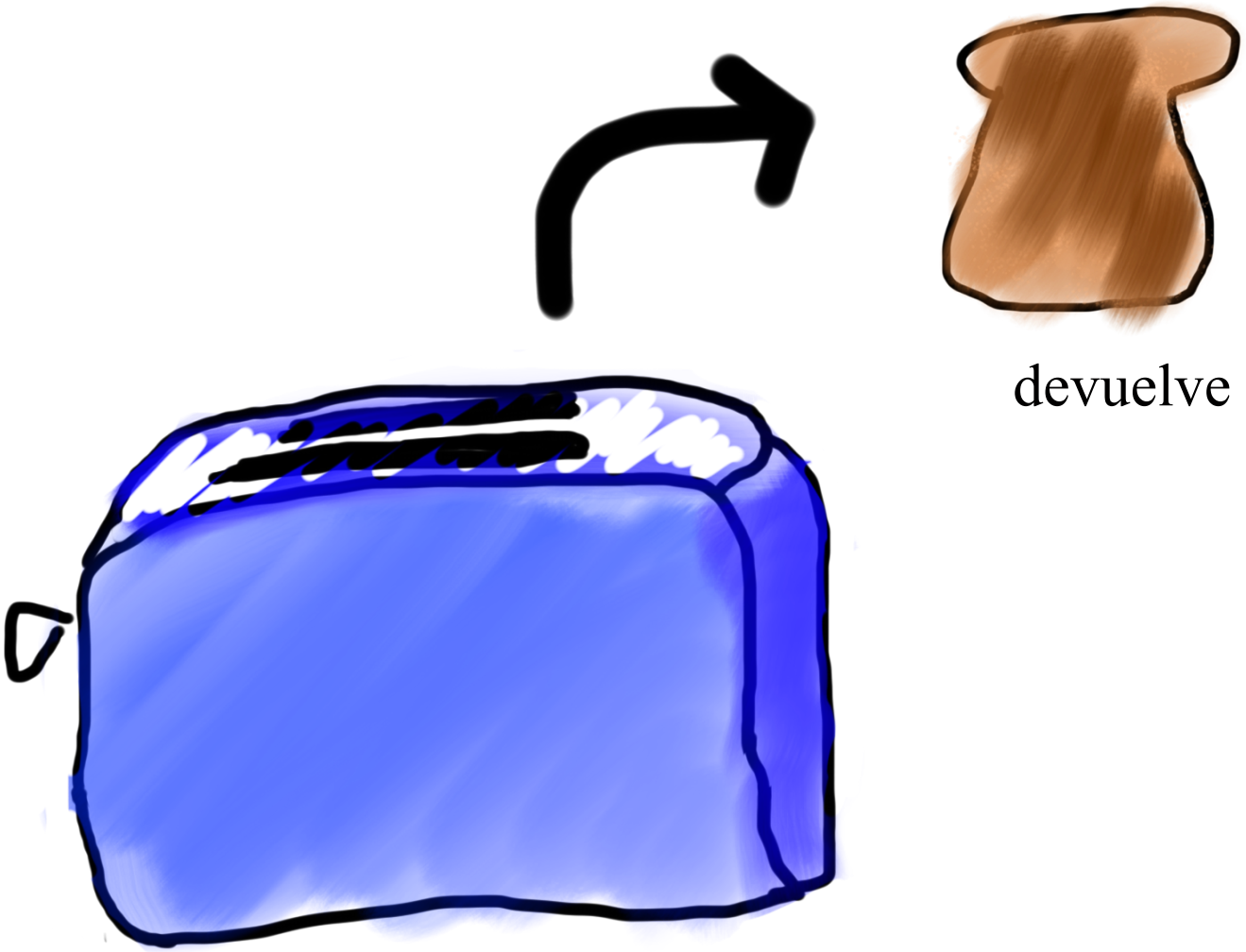
parámetro

Los métodos son como una tostadora

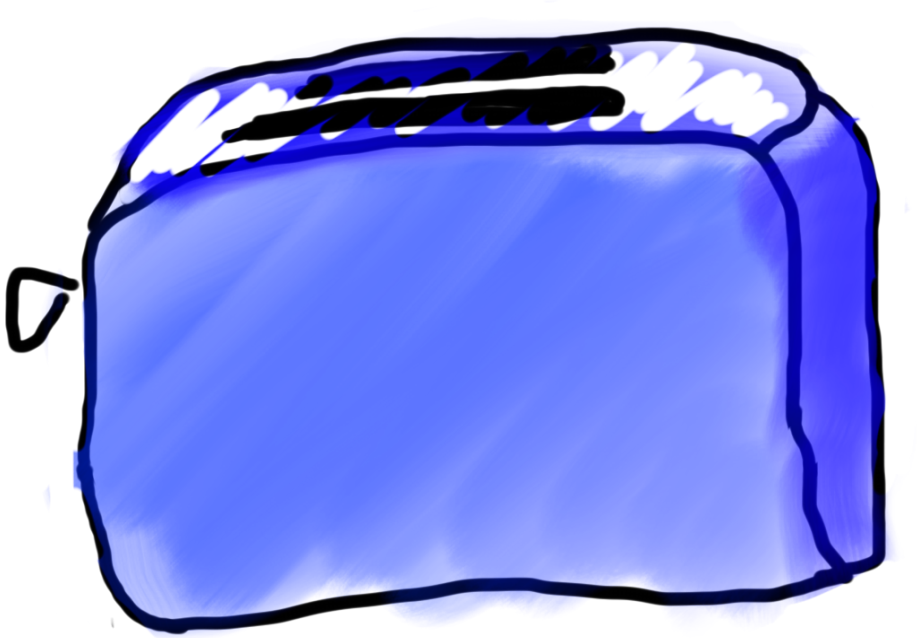


Los métodos son como una tostadora

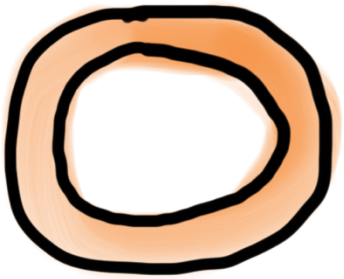
Los métodos son como una tostadora



Los métodos son como una tostadora

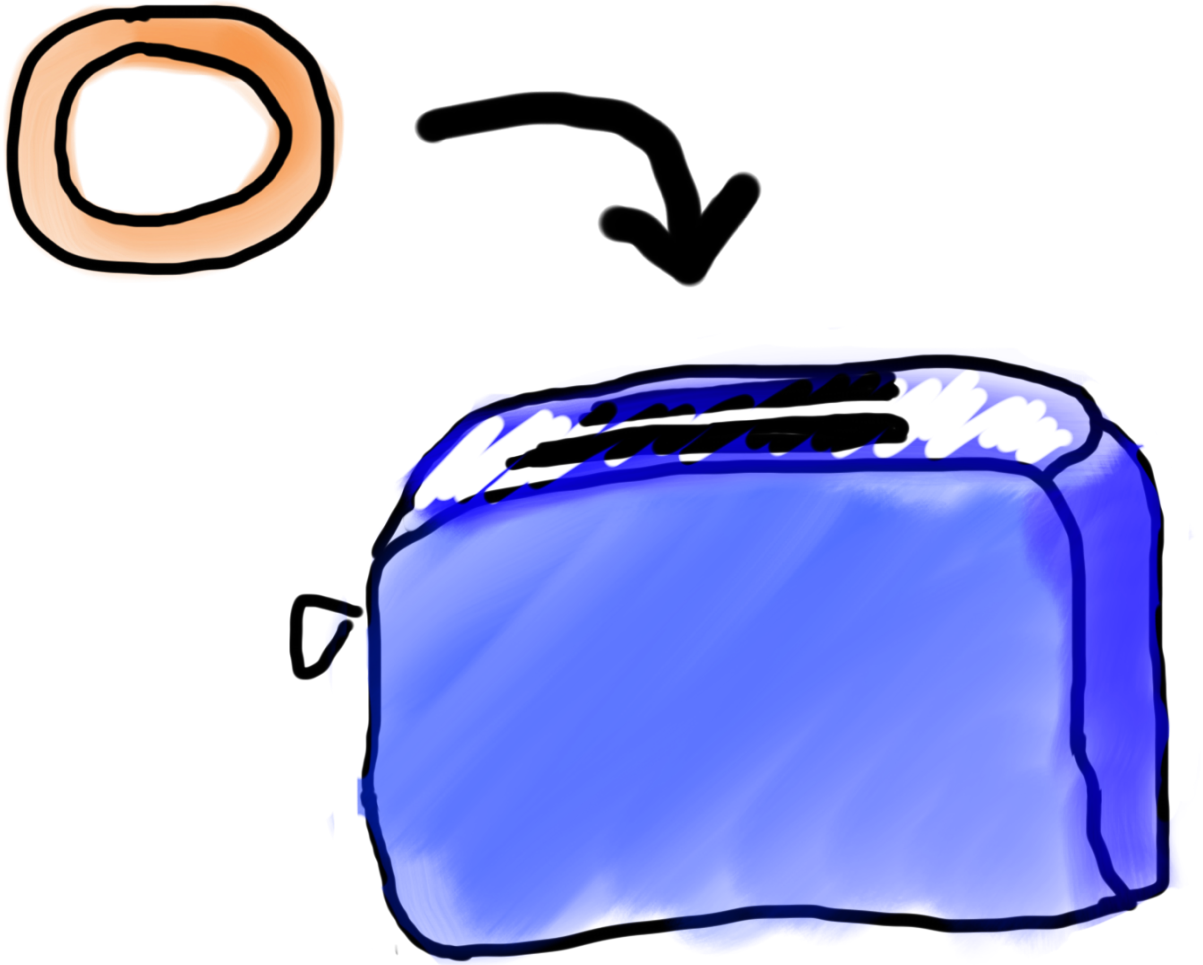


Los métodos son como una tostadora

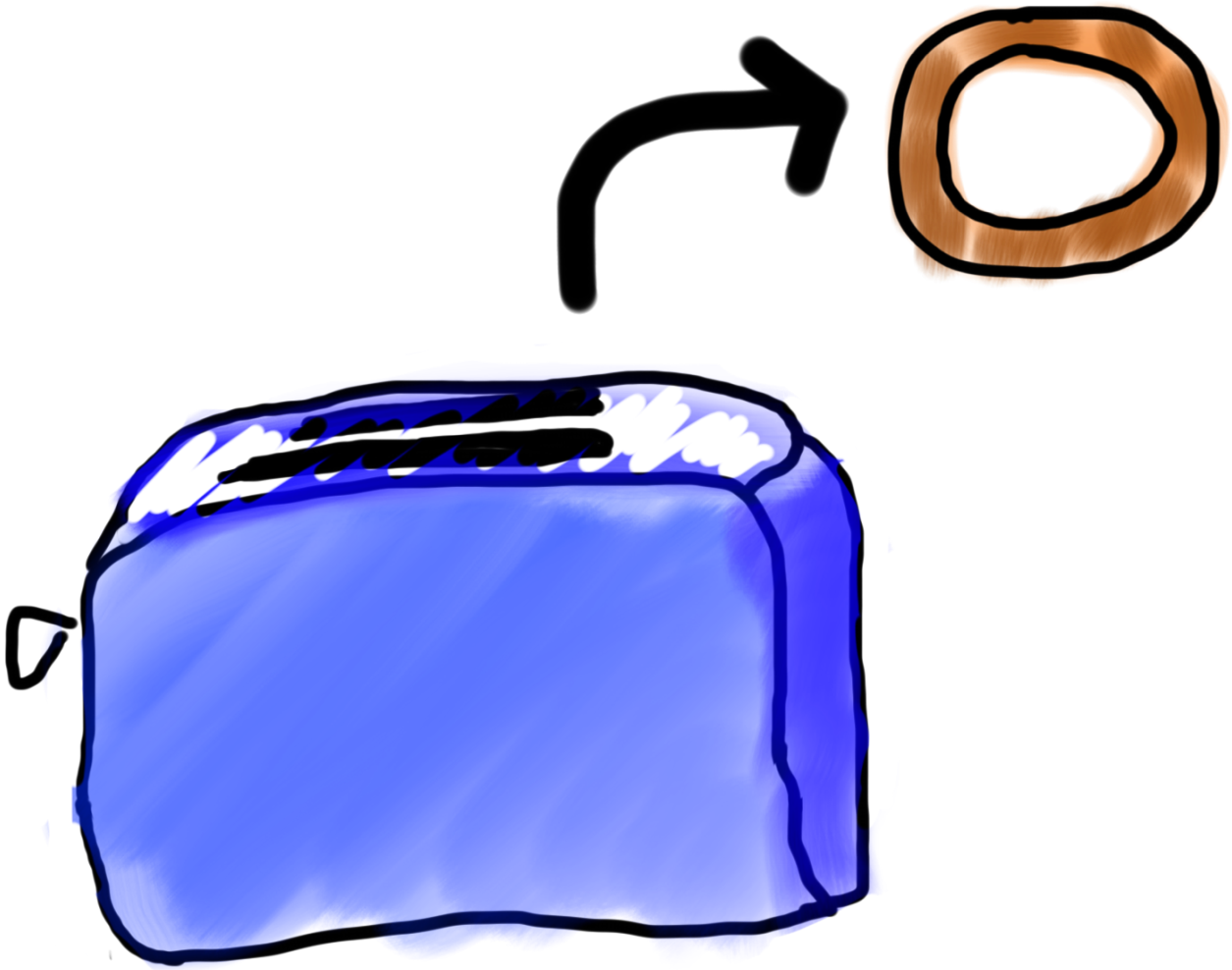


* No necesitas una segunda tostadora si quieres tostar pan blandito. ¡Puedes usar la misma!

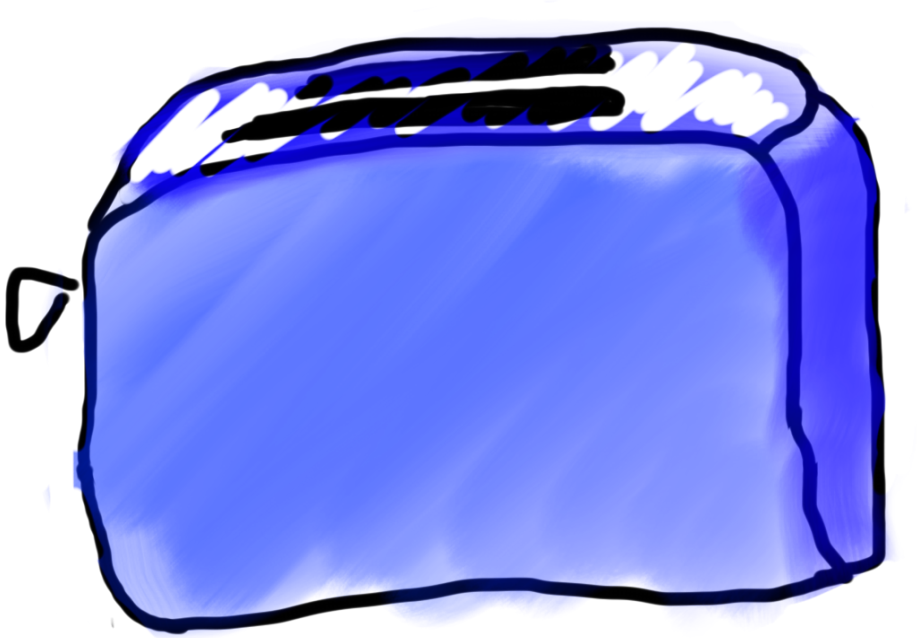
Los métodos son como una tostadora



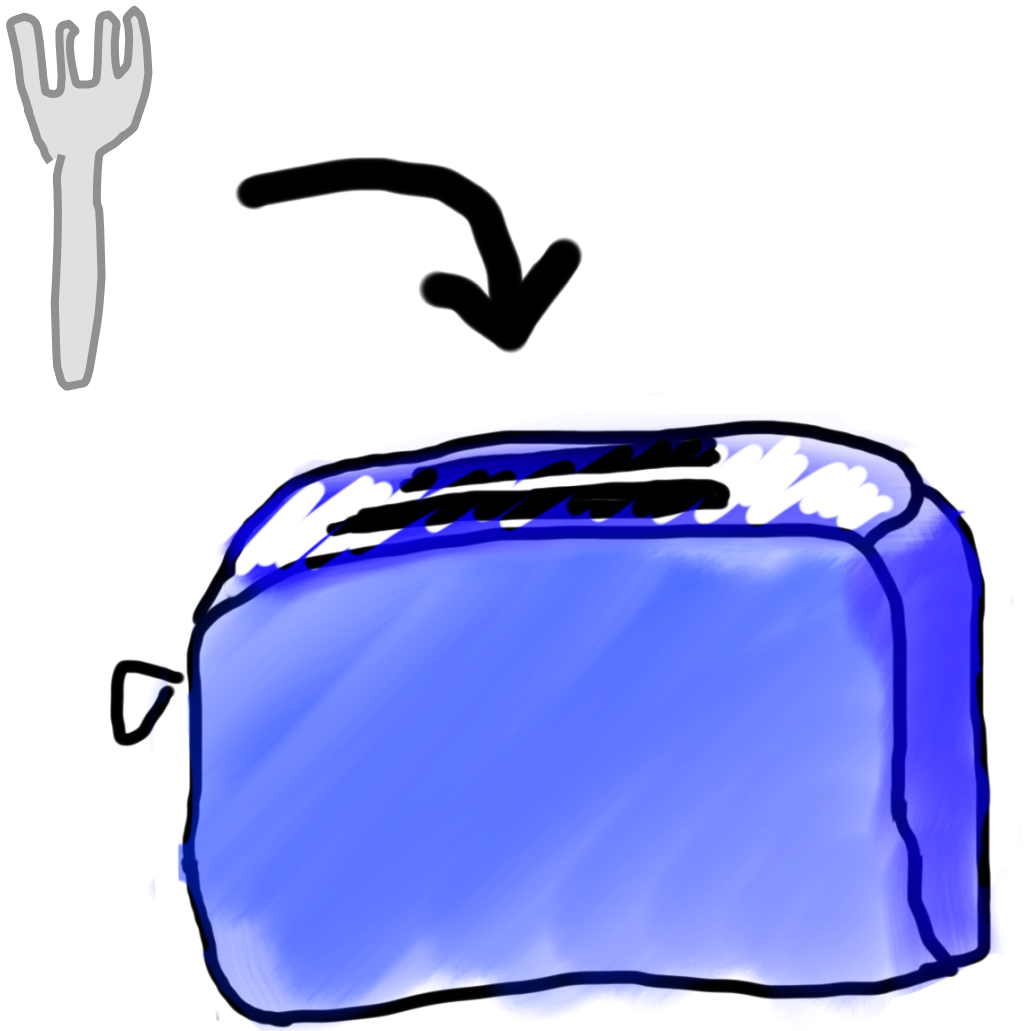
Los métodos son como una tostadora



Los métodos son como una tostadora

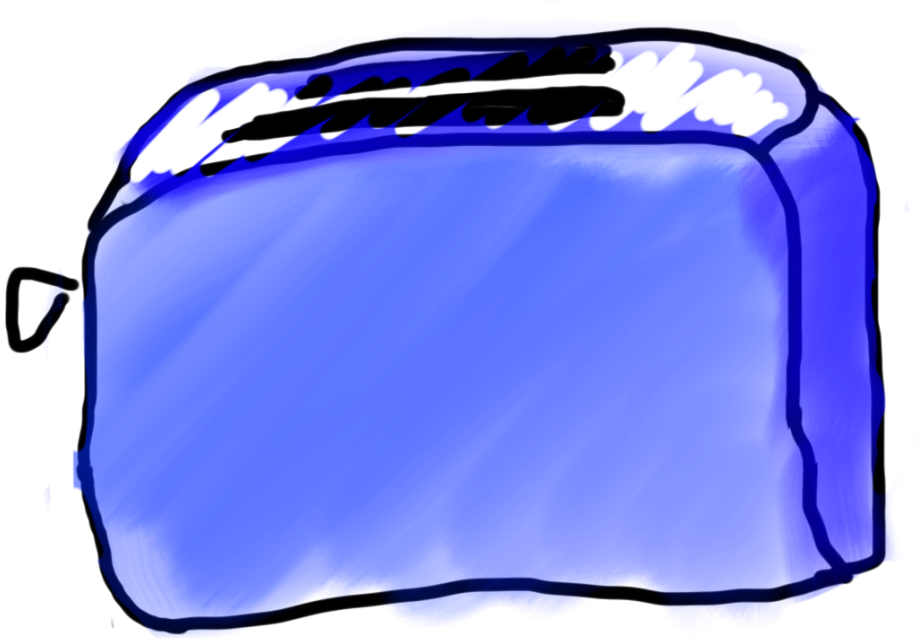


Los métodos son como una tostadora

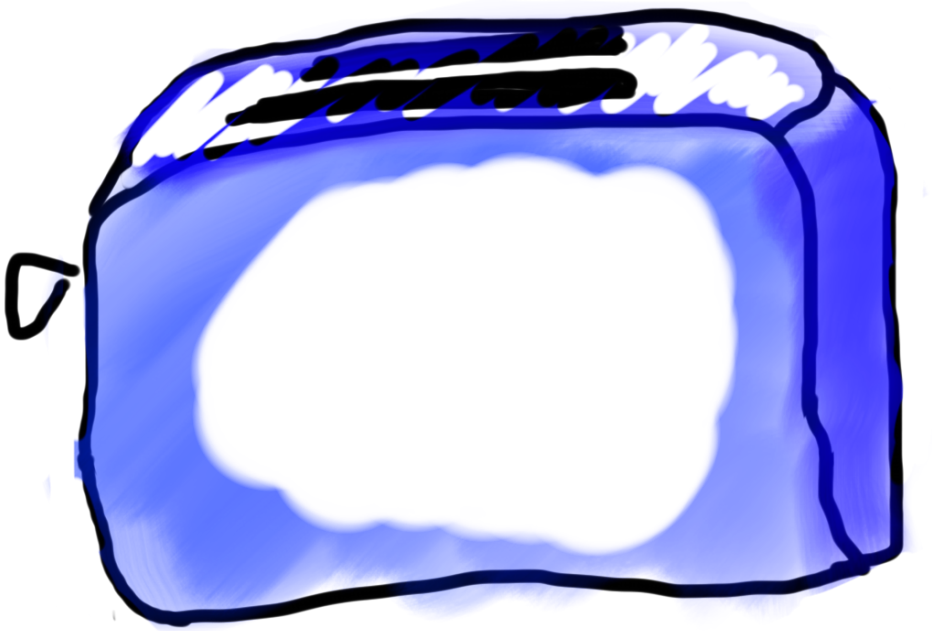


Los métodos son como una tostadora

Los métodos son como una tostadora

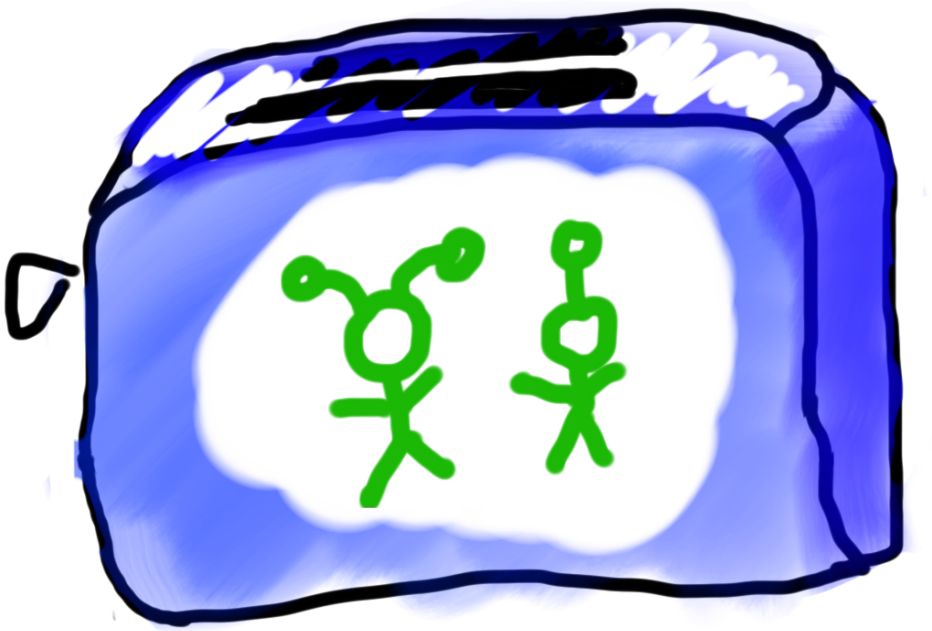


Los métodos son como una tostadora

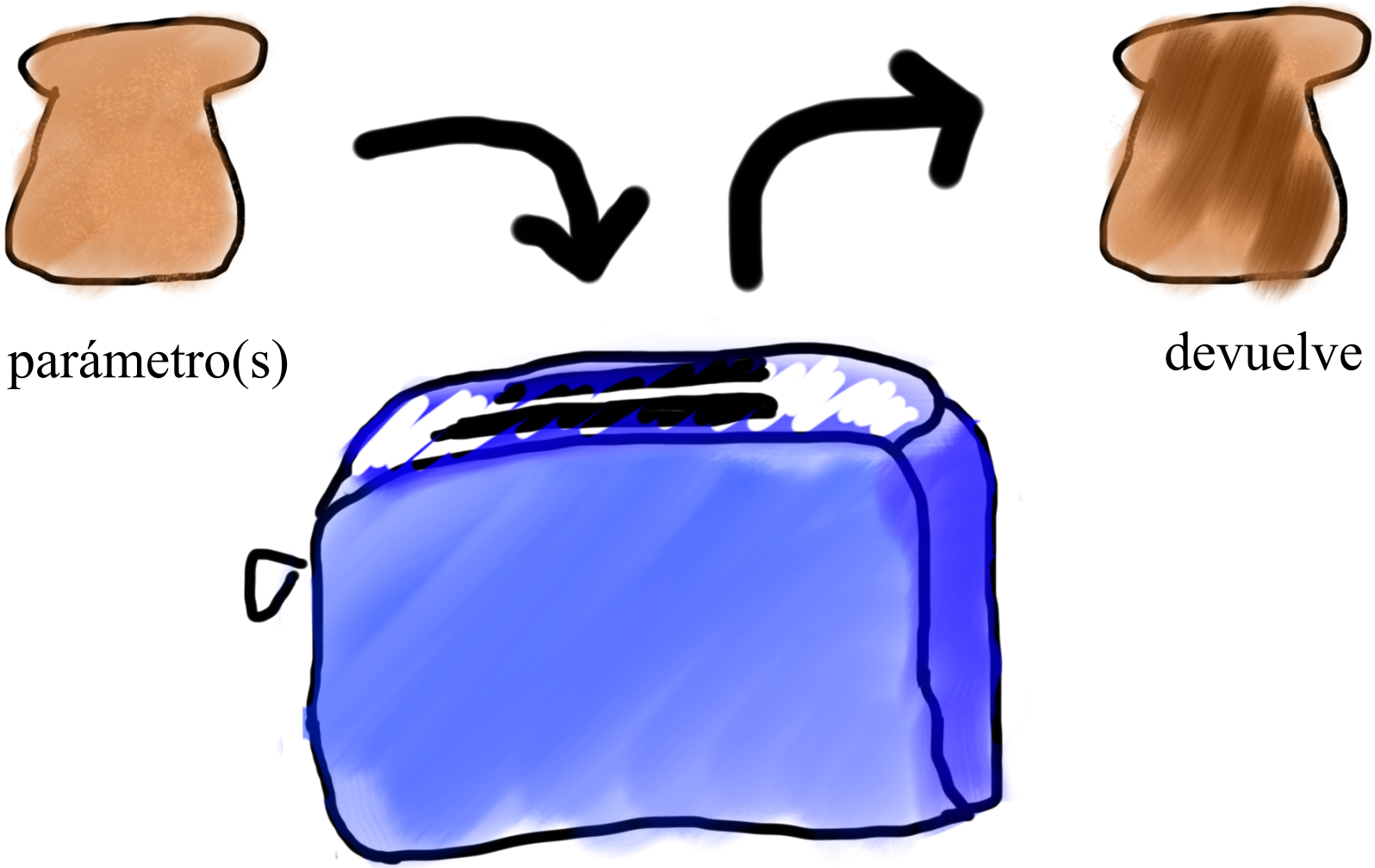


Los métodos son como una tostadora

Los métodos son como una tostadora



Los métodos son como una tostadora



Anatomía de un método

```
public void run() {
    double mitad = promedio(5.0, 10.2);
    imprimir(mitad);
}
```

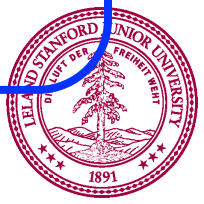
```
private double promedio(double a, double b) {
    double suma = a + b;
    return suma / 2;
}
```


Anatomía de un método

```
public void run() {
    double mitad = promedio(5.0, 10.2);
    imprimir(mitad);
}
```

"definición" del método

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



Anatomía de un método

```
public void run() {
    double mitad = promedio(5.0, 10.2);
    imprimir(mitad);
}
```

Se espera que devuelva

Espera información de entrada

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


Anatomía de un método

```
public void run() {
    double mitad = promedio(5.0, 10.2);
    imprimir(mitad);
}
```

Tipo que devuelve

Parámetros

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


Anatomía de un método

```
public void run() {
    double mid = average(5.0, 10.2);
    imprimir(mid);
}
```

nombre

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


Anatomía de un método

```
public void run() {
    double mitad = promedio(5.0, 10.2);
    imprimir(mitad);
}
```

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

cuerpo

Anatomía de un método

```
public void run() {
    double mid = average(5.0, 10.2);
    imprimir(mid);
}
```

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

Termina el método y regresa solo UN valor

Anatomía de un método

```
public void run() {
    double mitad = promedio(5.0, 10.2);
    imprimir(mitad);
}
```

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

Esta instrucción es necesaria porque **promedio** prometió devolver un double

Anatomía de un método

"llamada" del método

```
public void run() {
    double mitad = promedio(5.0, 10.2);
    imprimir(mitad);
}

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


Anatomía de un método

```
public void run() {
    double mitad = promedio(5.0, 10.2);
    imprimir(mitad);
}
```

argumentos

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


Anatomía de un método

Tipo que devuelve Parámetros

```
public void run() {
    double mitad = promedio(5.0, 10.2);
    imprimir(mitad);
}


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

Anatomía de un método

```
public void run() {
    double mitad = promedio(5.0, 10.2);
    imprimir(mitad);
}
```

Cuando un método termina "devuelve"

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

La manera formal

visibilidad *tipo* ***nombreDelMétodo*** **(***parámetros***)** **{**
    instrucciones
}

- *visibilidad:* normalmente **`private`** o **`public`**
- *tipo:* tipo devuelto por el el método (e.g., **`int`**, **`double`**, *etc*.)
  - Puede ser **`void`** para indicar que nada se devuelve
- *parámetros:* información que se le pasa al método

Parámetros

Los parámetros te permiten darle al método alguna información cuando lo estás llamando.

Aprender a través del ejemplo

Ejemplo Void

```
private void imprimirIntro() {
    imprimir("Bienvenidos a clase");
    imprimir("Es la mejor parte de mi día.");
}


public void run() {
    imprimirIntro();
}
```


Ejemplo Void

```
private void imprimirIntro() {
    imprimir("Bienvenidos a clase");
    imprimir("Es la mejor parte de mi día.");
}


public void run() {
    imprimirIntro();
}
```


Ejemplo Void

```
private void imprimirIntro() {
    imprimir("Bienvenidos a clase");
    imprimir("Es la mejor parte de mi día.");
}


public void run() {
    imprimirIntro();
}
```


Ejemplo Void

```
private void imprimirIntro() {
    imprimir("Bienvenidos a clase");
    imprimir("Es la mejor parte de mi día.");
}


public void run() {
    imprimirIntro();
}
```

Ejemplo Void

```
private void imprimirIntro() {
    imprimir("Bienvenidos a clase");
    imprimir("Es la mejor parte de mi día.");
}


public void run() {
    imprimirIntro();
}
```


Ejemplo Void

```
private void imprimirIntro() {
    imprimir("Bienvenidos a clase");
    imprimir("Es la mejor parte de mi día.");
}


public void run() {
    imprimirIntro();
}
```

Ejemplo Void

```
private void imprimirIntro() {
    imprimir("Bienvenidos a clase");
    imprimir("Es la mejor parte de mi día.");
}

public void run() {
    imprimirIntro();
}
```


Ejemplo Void

```
private void imprimirIntro() {
    imprimir("Bienvenidos a clase");
    imprimir("Es la mejor parte de mi día.");
}


public void run() {
    imprimirIntro();
}
```


Ejemplo Void

```
private void imprimirIntro() {
    imprimir("Bienvenidos a clase");
    imprimir("Es la mejor parte de mi día.");
}


public void run() {
    imprimirIntro();
}
```


Ejemplo Void

```
private void imprimirIntro() {
    imprimir("Bienvenidos a clase");
    imprimir("Es la mejor parte de mi día.");
}


public void run() {
    imprimirIntro();
}
```


Ejemplo - Parámetro

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```


Ejemplo - Parámetro

Run memoria

No hay variables

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```


Ejemplo - Parámetro

Run memoria

No hay variables

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```


Ejemplo - Parámetro

Run memoria

No hay variables

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```


Ejemplo - Parámetro

Run memoria

No hay variables

imprimirOpinion memoria

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```

Ejemplo - Parámetro

Run memoria

No hay variables

imprimirOpinion memoria

num

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```

Ejemplo - Parámetro

Run memoria

No hay variables

imprimirOpinion memoria

num **5**

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```


Ejemplo - Parámetro

Run memoria

No hay variables

imprimirOpinion memoria

num **5**

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```


Ejemplo - Parámetro

Run memoria

No hay variables

imprimirOpinion memoria

num **5**

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```


Ejemplo - Parámetro

Run memoria

No hay variables

imprimirOpinion memoria

num **5**

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```


Ejemplo - Parámetro

Run memoria

No hay variables

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```


Ejemplo - Parámetro

Run memoria

No hay variables

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```


Ejemplo - Parámetro

Run memoria

No hay variables

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```


Ejemplos - Parámetros

```
private void imprimirOpinion(int num) {
    if(num == 5) {
        imprimir("Yo amo 5!");
    } else {
        imprimir("Lo que sea");
    }
}

public void run() {
    imprimirOpinion(5);
}
```

```
private String darOpinion(int num) {
    if(num == 5) {
        return "Yo amo 5!";
    } else {
        return "Lo que sea";
    }
}

public void run() {
    String opinion = darOpinion(5);
    imprimir(opinion);
}
```

Entiendan el mecanismo

Devolver múltiples parámetros

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}

public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

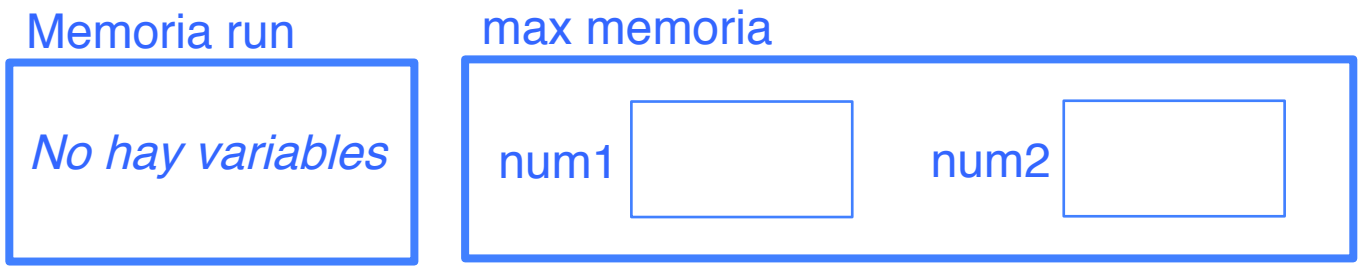
Devolver múltiples parámetros

Memoria run

No hay variables

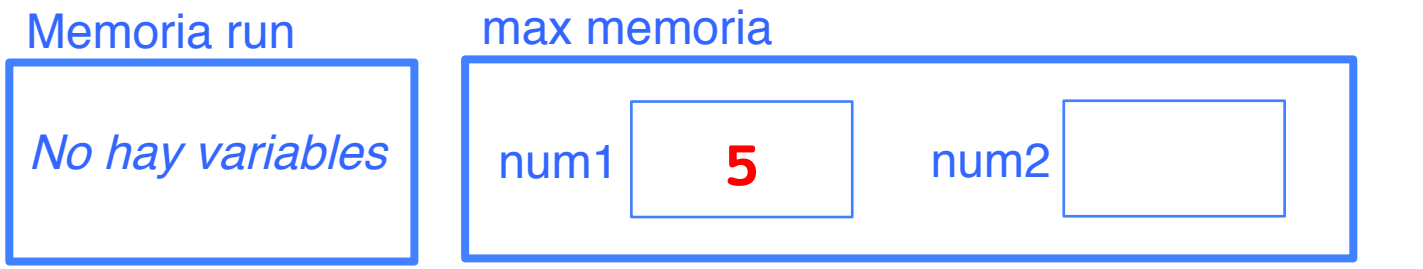
```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros



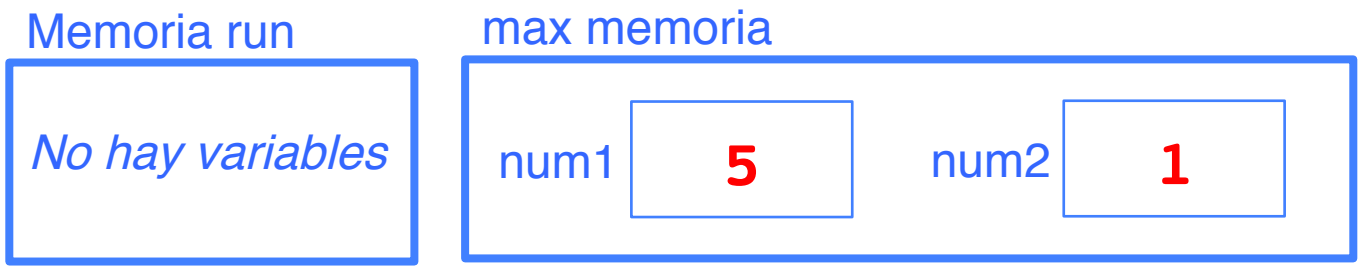

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros



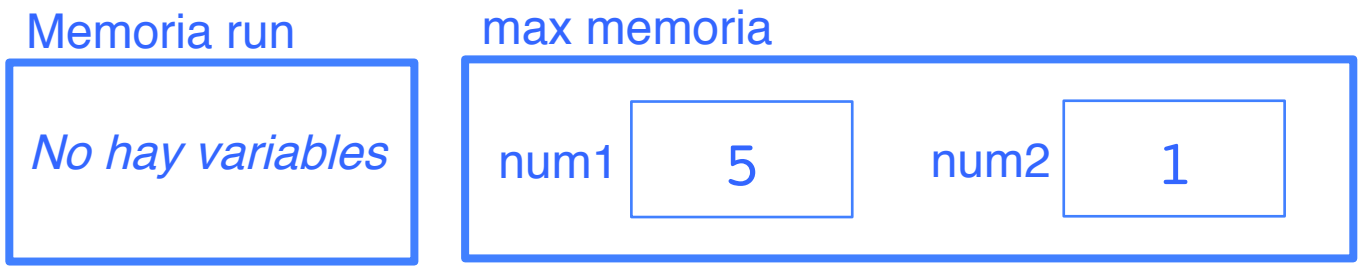

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros



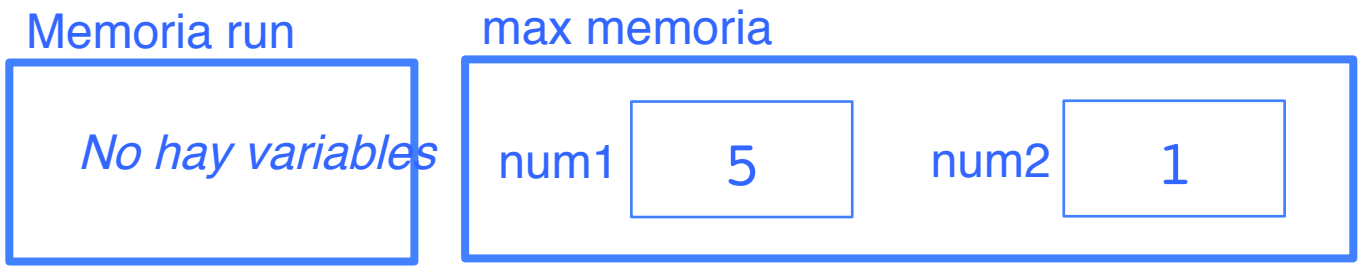

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros



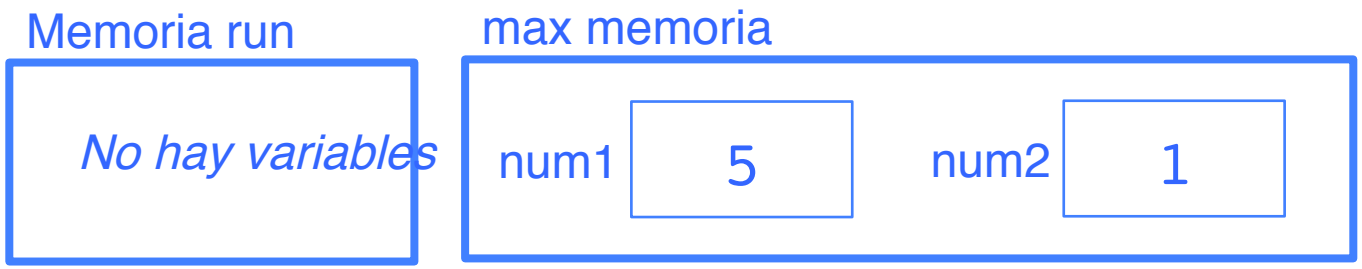

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros




```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

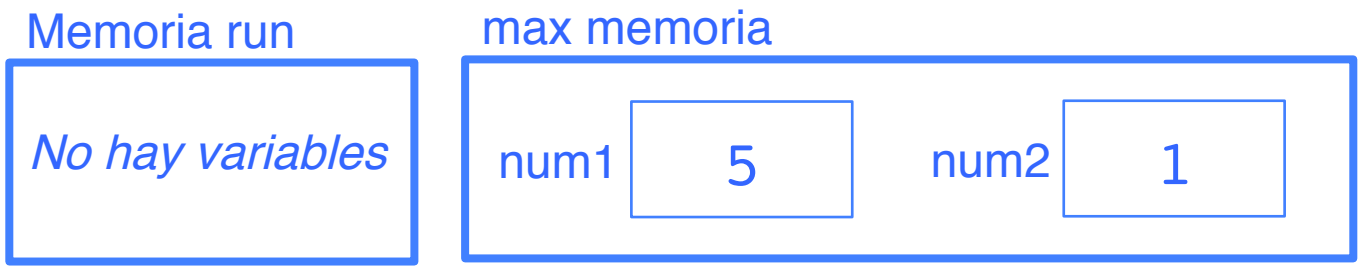
Devolver múltiples parámetros




```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

5

Devolver múltiples parámetros




```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() { 5
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() { 5
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));  8
}
```

Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

8

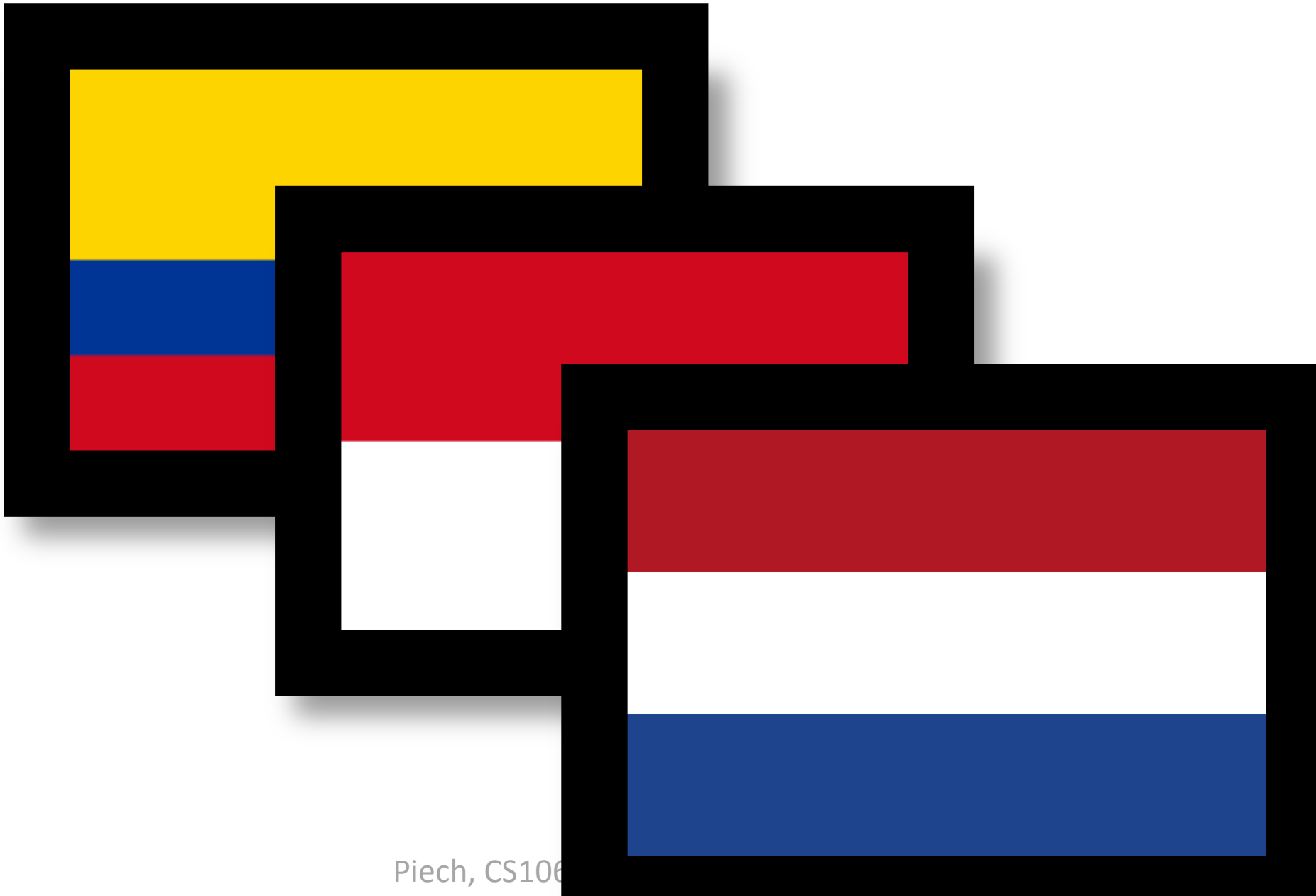
Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    imprimir(max(5, 1));
    imprimir(max(3, 8));
}
```

Pasando colores

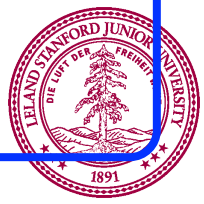


Banderas

```
private void pintarBanderaColombia() {
    // Llama pintarRaya tres veces
    // con diferentes entradas cada vez
    pintarRaya(Color.YELLOW, 0, 0.5);
    pintarRaya(Color.BLUE, 0.5, 0.75);
    pintarRaya(Color.RED, 0.75, 1.0);
}
```

```
// Definir pintarRaya
// pintarRaya necesita tres entradas (que vienen como variables)
// Primero un color, then a start (in screen percent) and end y.
private void pintarRaya(Color color, double yStart, double yEnd) {
    double yStartPx = darAlto() * yStart;
    double yEndPx = darAlto() * yEnd;
    SRect rect = new SRect(darAncho(), yEndPx - yStartPx);
    rect.cambiarColor(color);
    rect.cambiarRelleno(true);
    add(rect, 0, yStartPx);
}
```

"definición" del método



Banderas

```
private void drawColombiasFlag() {
    // Llama pintarRaya tres veces
    // con diferentes entradas cada vez
    pintarRaya(Color.YELLOW, 0, 0.5);
    pintarRaya(Color.BLUE, 0.5, 0.75);
    pintarRaya(Color.RED, 0.75, 1.0);
}
```

Toma tres entradas de información: (1) un color, (2) una fracción de inicio, (3) una fracción de cierre

```
// Define pintarRaya
// pintarRaya needs
// First a color, then a start (in screen percent) and end y.
private void pintarRaya(Color color, double yStart, double yEnd) {
    double yStartPx = darAlto() * yStart;
    double yEndPx = darAlto() * yEnd;
    SRect rect = new SRect(darAncho(), yEndPx - yStartPx);
    rect.cambiarColor(color);
    rect.cambiarRelleno(true);
    add(rect, 0, yStartPx);
}
```

Banderas

```
private void drawColombiasFlag() {
    // Llama pintarRaya tres veces
    // con diferentes entradas cada vez
    pintarRaya(Color.YELLOW, 0, 0.5);
    pintarRaya(Color.BLUE, 0.5, 0.75);
    pintarRaya(Color.RED, 0.75, 1.0);
}
```

Entonces, cada vez que el método se llama, tres entradas de información se deben dar!

```
// Define pintarRaya
// pintarRaya needs three inputs (which come as variables)
// First a color, then a start (in screen percent) and end y.
private void pintarRaya(Color color, double yStart, double yEnd) {
    double yStartPx = darAlto() * yStart;
    double yEndPx = darAlto() * yEnd;
    SRect rect = new SRect(darAncho(), yEndPx - yStartPx);
    rect.cambiarColor(color);
    rect.cambiarRelleno(true);
    add(rect, 0, yStartPx);
}
```


Banderas

```
private void drawColombiasFlag() {
    // Llama pintarRaya tres veces
    // con diferentes entradas cada vez
    pintarRaya(Color.YELLOW, 0, 0.5);
    pintarRaya(Color.BLUE, 0.5, 0.75);
    pintarRaya(Color.RED, 0.75, 1.0);
}
```

Entonces, cada vez que el método se llama, tres entradas de información se deben dar!

El método recibe la primera información de entrada en una caja (aka variable), con el nombre color

```
// Define
// pintarR
// Primero a color, then a start (in screen percent) and end y.
private void pintarRaya(Color color, double yStart, double yEnd) {
    double yStartPx = darAlto() * yStart;
    double yEndPx = darAlto() * yEnd;
    SRect rect = new SRect(darAncho(), yEndPx - yStartPx);
    rect.cambiarColor(color);
    rect.cambiarRelleno(true);
    add(rect, 0, yStartPx);
}
```


Parámetros

Cada vez que se llama un método, una nueva memoria es creada para la llamada.

Objetivos de aprendizaje:

1. Escribir un método que toma un insumo de entrada
2. Escribir un método que devuelve un *output*
3. Practicar usando tus propios métodos

Para los tipos primitivos: Las variables **NO** se pasan cuando utilizas parámtros. Son los valores los que se pasan.

Malos tiempos con métodos

```
// NOTA: este programa es pirata (buggy)!!

private void sumarCinco(int x) {
  x += 5;
}


public void run() {
  int x = 3;
  sumarCinco (x);
  imprimir("x = " + x);
}
```

Sigamos el “rastro” de este programa en el tablero

Buenos tiempos con métodos

```
// NOTA: Este programa se siente justo
  bien...

private int sumarCinco(int x) {
  x += 5;
  devolver x;
}

public void run() {
  int x = 3;
  x = sumarCinco(x);
  imprimir("x = " + x);
}
```

Pasar a través del “valor”

- Thanks Mehran

Boolean Variable

```
boolean karelIsAwesome = true;

boolean myBool = 1 < 2;
```


Boolean Operations

```
boolean a = true;

boolean b = false;


boolean and = a && b;

boolean or = a || b;

boolean not = !a;
```


Logical Operators

In order of precedence:

Operator	Description	Example	Result
!	not	`!(2 == 3)`	`true`
&&	and	`(2 == 3) && (-1 < 5)`	`false`
||	or	`(2 == 3) || (-1 < 5)`	`true`

Cannot "chain" tests as in algebra; use && or || instead

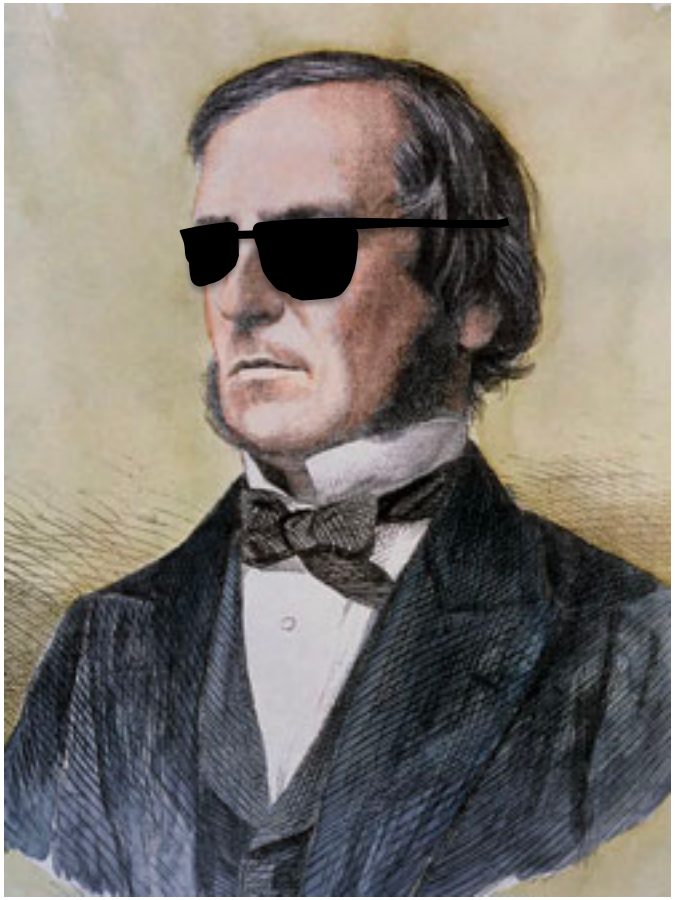
```
// assume x is 15
2 <= x <= 10
true   <= 10
Error!
```

```
// correct version
2 <= x && x <= 10
true   && false
false
```


```
boolean food = true;
boolean drinks = true;
boolean isAllowed = !food || drinks;
```

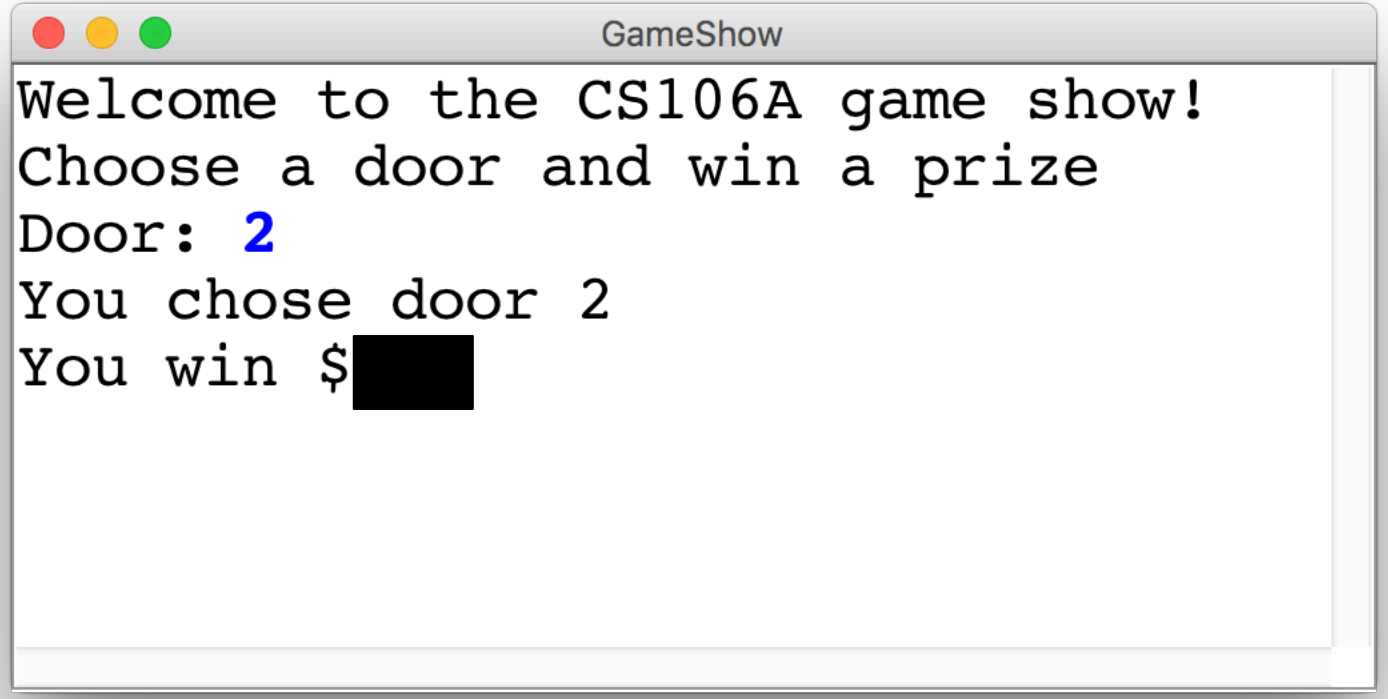
*know your logical precedence

George Boole



English Mathematician 1815 – 1864
Boole died of being too cool

Game Show


GameShow
Welcome to the CS106A game show!
Choose a door and win a prize
Door: 2
You chose door 2
You win $

Choose a Door

```
int door = readInt("Door: ");
// while the input is invalid
while(door < 1 || door > 3) {
   // tell the user the input was invalid
   imprimir("Invalid door!");
   // ask for a new input
   door = readInt("Door: ");
}
```

`||`	or
`&&`	and

The Door Logic

```
int prize = 4;
if(door == 1) {
   prize = 2 + 9 / 10 * 100;
} else if(door == 2) {
   boolean locked = prize % 2 != 0;
   if(!locked) {
      prize += 6;
   }
} else if(door == 3) {
   prize++;
}
```


The Door Logic

```
int prize = 4;
if(door == 1) {
   prize = 2 + 9 / 10 * 100;
} else if(door == 2) {
   boolean locked = prize % 2 != 0;
   if(!locked) {
      prize += 6;
   }
} else if(door == 3) {
   prize++;
}
```


The Door Logic

```
int prize = 4;
if(door == 1) {
   prize = 2 + 9 / 10 * 100;
} else if(door == 2) {
   boolean locked = prize % 2 != 0;
   if(!locked) {
      prize += 6;
   }
} else if(door == 3) {
   prize++;
}
```


The Door Logic

```
int prize = 4;
if(door == 1) {
   prize = 2 + 9 / 10 * 100;
} else if(door == 2) {
   boolean locked = prize % 2 != 0;
   if(!locked) {
      prize += 6;
   }
} else if(door == 3) {
   prize++;
}
```


The Door Logic

```
int prize = 4;
if(door == 1) {
   prize = 2 + 9 / 10 * 100;
} else if(door == 2) {
   boolean locked = prize % 2 != 0;
   if(!locked) {
      prize += 6;
   }
} else if(door == 3) {
   prize++;
}
```


Parameter and Return Example

```
private double metersToCm(double meters) {
    return 100 * meters;;
}
```

```
public void run() {
    double result = metersToCm(5.2);
    imprimir(result);
}
```


Parameter and Return Example

Memoria run

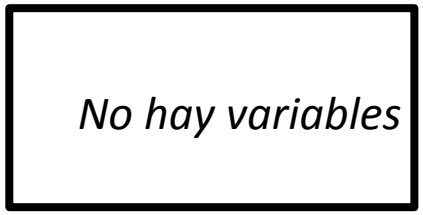
No hay variables

```
private double metersToCm(double meters) {
    return 100 * meters;;
}
```

```
public void run() {
    double result = metersToCm(5.2);
    imprimir(result);
}
```


Parameter and Return Example

Memoria run




```
private double metersToCm(double meters) {
    return 100 * meters;;
}
```

```
public void run() {
    double result = metersToCm(5.2);
    imprimir(result);
}
```


Parameter and Return Example

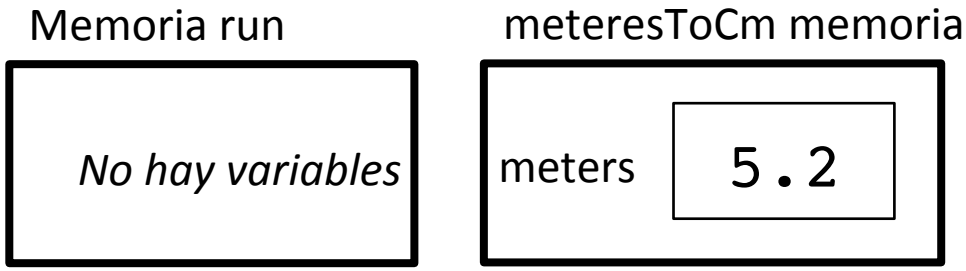
Memoria run

No hay variables

meteresToCm memoria

```
private double metersToCm(double meters) {
    return 100 * meters;;
}


public void run() {
    double result = metersToCm(5.2);
    imprimir(result);
}
```

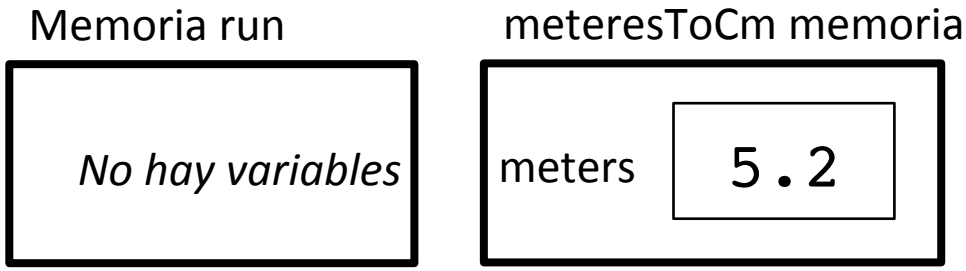

Parameter and Return Example




```
private double metersToCm(double meters) {
    return 100 * meters;;
}
```

```
public void run() {
    double result = metersToCm(5.2);
    imprimir(result);
}
```

Parameter and Return Example




```
private double metersToCm(double meters) {
    return 100 * meters;
}
```

520.0

```
public void run() {
    double result = metersToCm(5.2);
    imprimir(result);
}
```

Parameter and Return Example

Memoria run

No hay variables

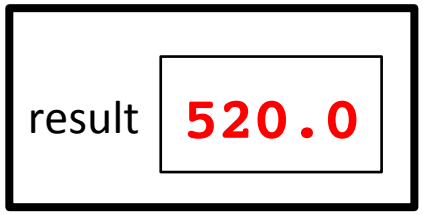
```
private double metersToCm(double meters) {
    return 100 * meters;;
}
```

```
public void run() {
    double result = metersToCm(5.2);
    imprimir(result);
}
```

520.0

Parameter and Return Example

Memoria run



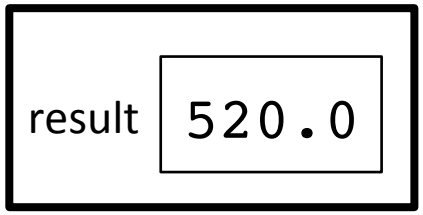

```
private double metersToCm(double meters) {
    return 100 * meters;;
}
```

520.0

```
public void run() {
    double result = metersToCm(5.2);
    imprimir(result);
}
```

Parameter and Return Example

Memoria run




```
private double metersToCm(double meters) {
    return 100 * meters;;
}
```

```
public void run() {
    double result = metersToCm(5.2);
    imprimir(result);
}
```


Parameter and Return Example

```
private double metersToCm(double meters) {
    return 100 * meters;;
}
```

```
public void run() {
    imprimir(metersToCm(5.2));
    imprimir(metersToCm(9.1));
}
```


Devolver múltiples parámetros

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
    int other = max(3, 8);
}
```


Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
    int other = max(3, 8);
}
```


Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
    int other = max(3, 8);
}
```


Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```

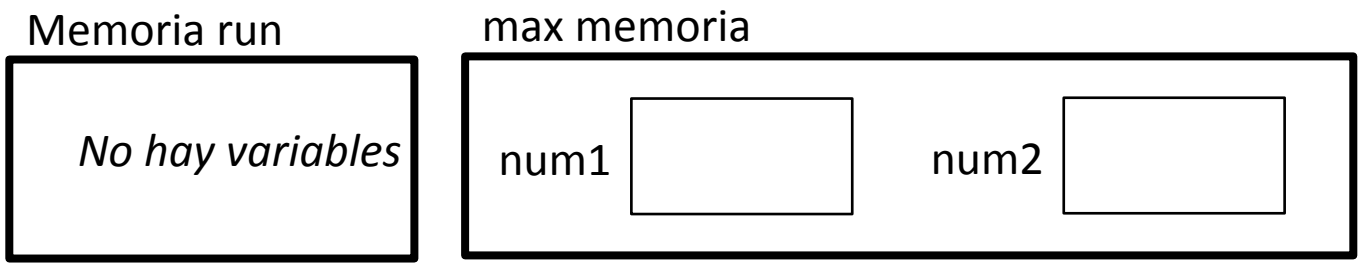

Devolver múltiples parámetros

Memoria run

No hay variables

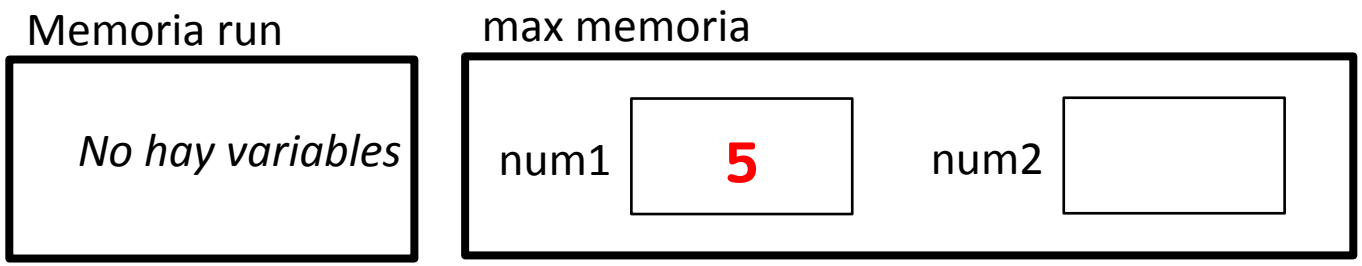
```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```


Devolver múltiples parámetros



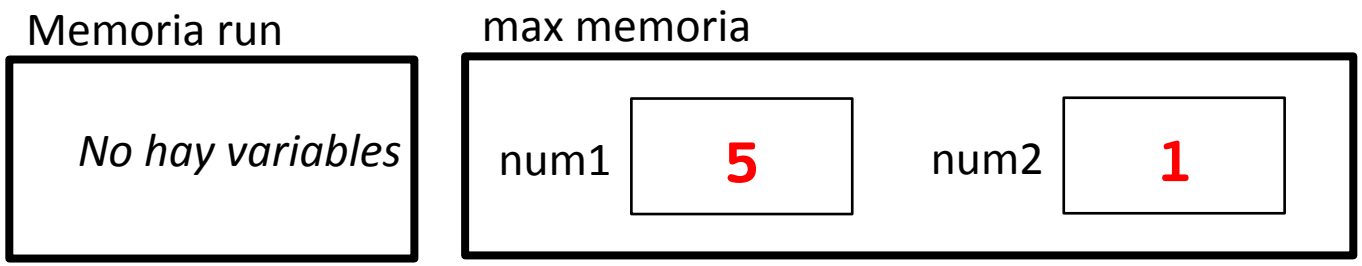

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```

Devolver múltiples parámetros



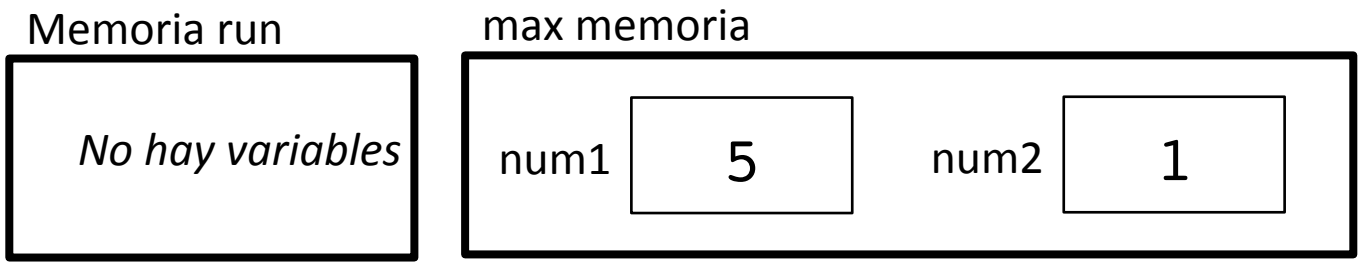

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```

Devolver múltiples parámetros



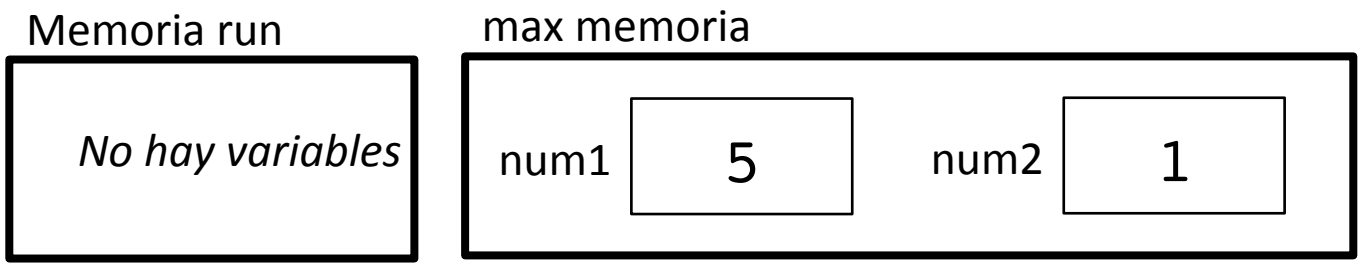

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```

Devolver múltiples parámetros



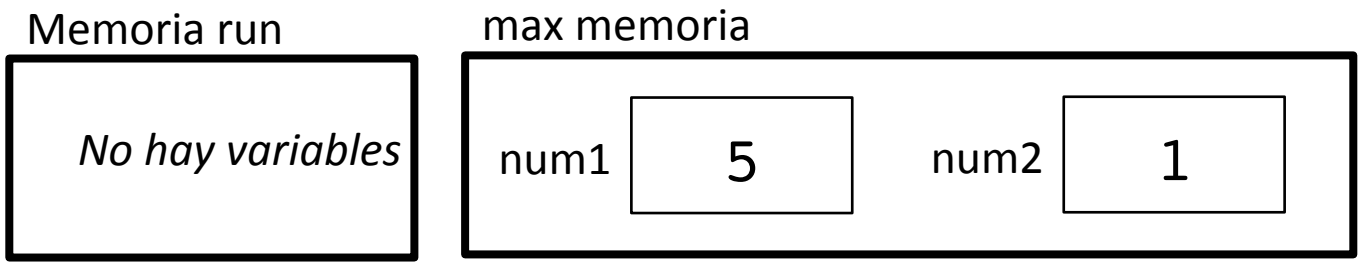

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```

Devolver múltiples parámetros




```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```

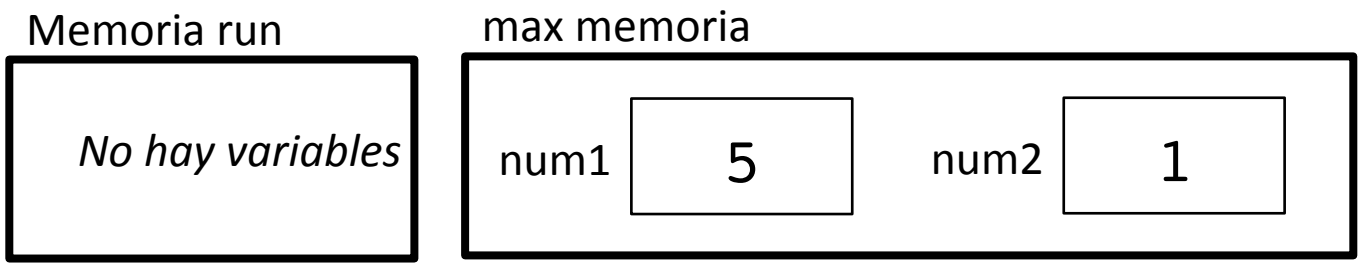
Devolver múltiples parámetros




```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```

5

Devolver múltiples parámetros




```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```

5

Devolver múltiples parámetros

Memoria run

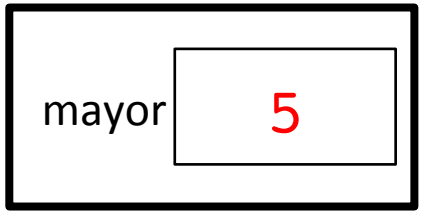
No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```

5

Devolver múltiples parámetros

Memoria run

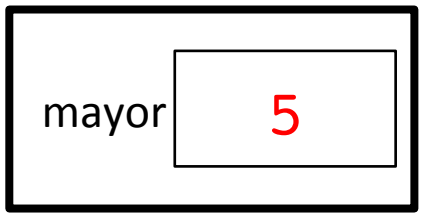



```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```

5

Devolver múltiples parámetros

Memoria run




```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(5, 1);
}
```

Devolver múltiples parámetros

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}

public void run() {
    int mayor = max(5, 1);
}
```


Devolver múltiples parámetros

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}

public void run() {
    int mayor = max(1, 5);
}
```

Devolver múltiples parámetros

Memoria run

No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(1, 5);
}
```

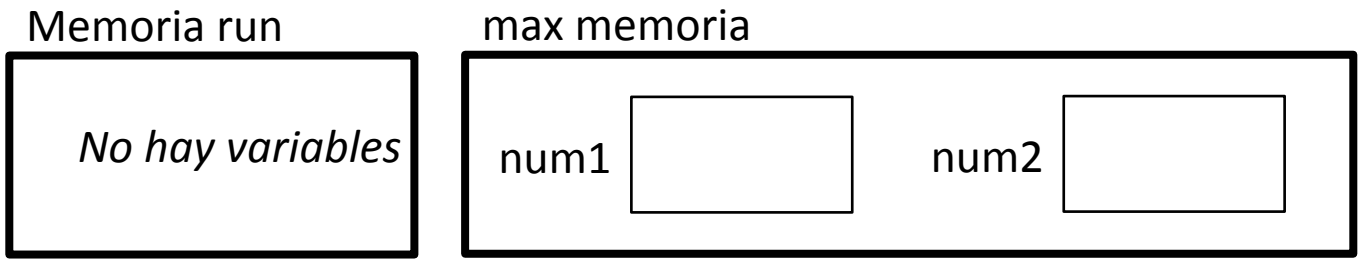

Devolver múltiples parámetros

Memoria run

No hay variables

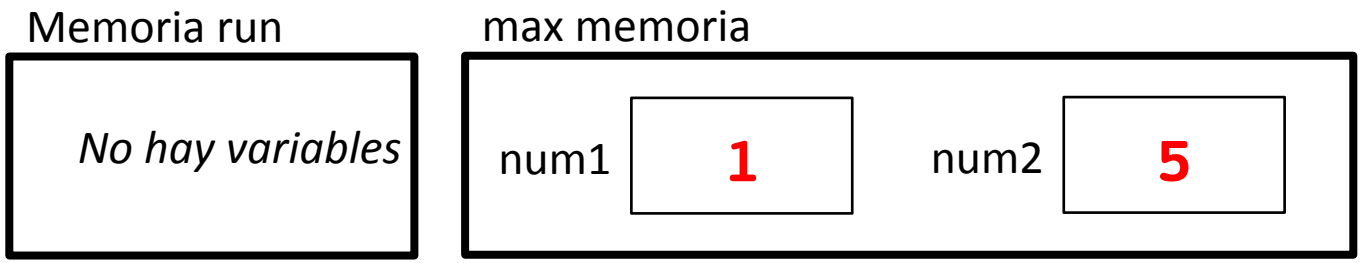
```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(1, 5);
}
```


Devolver múltiples parámetros



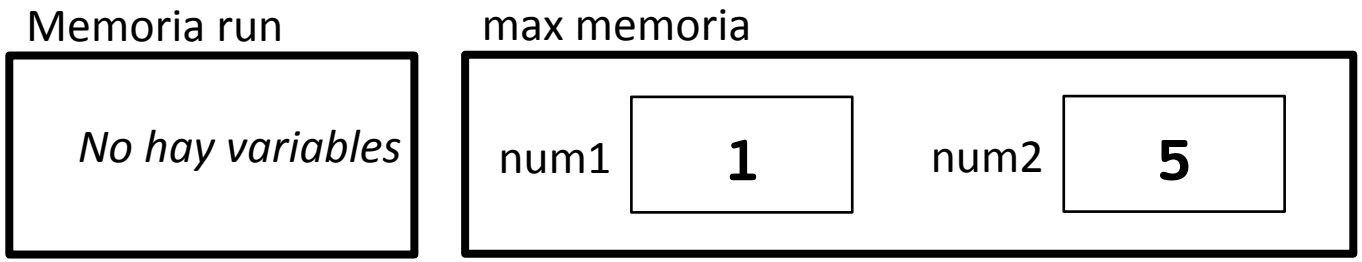

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(1, 5);
}
```

Devolver múltiples parámetros



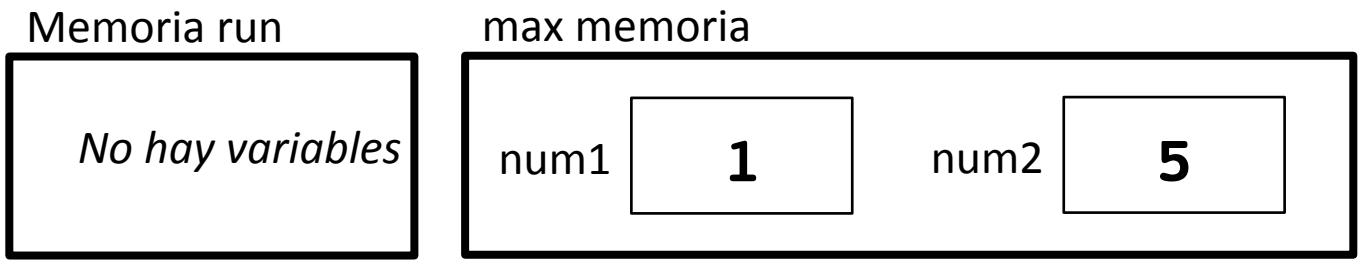

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(1, 5);
}
```

Devolver múltiples parámetros




```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(1, 5);
}
```

Devolver múltiples parámetros




```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}
```

5

```
public void run() {
    int mayor = max(1, 5);
}
```

Devolver múltiples parámetros

Memoria run

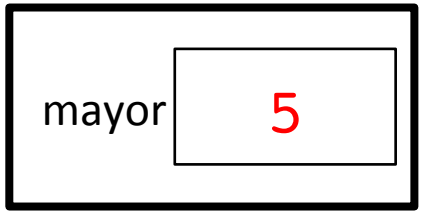
No hay variables

```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(1, 5);
}
```

5

Devolver múltiples parámetros

Memoria run

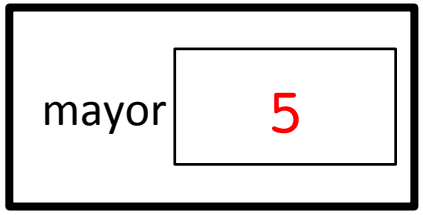



```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(1, 5);
}
```

5

Devolver múltiples parámetros

Memoria run




```
private int max(int num1, int num2) {
    if(num1 >= num2) {
        return num1;
    }
    return num2;
}


public void run() {
    int mayor = max(1, 5);
}
```

More Examples

A Full Program

```
public class FactorialExample extends ConsoleProgram {

   private static final int MAX_NUM = 4;

   public void run() {
       for(int i = 0; i < MAX_NUM; i++) {
             imprimir(i + "! = " + factorial(i));
       }
   }

   private int factorial(int n) {
       int result = 1;
       for (int i = 1; i <= n; i++) {
             result *= i;
       }
       return result;
   }
}
```

A Full Program

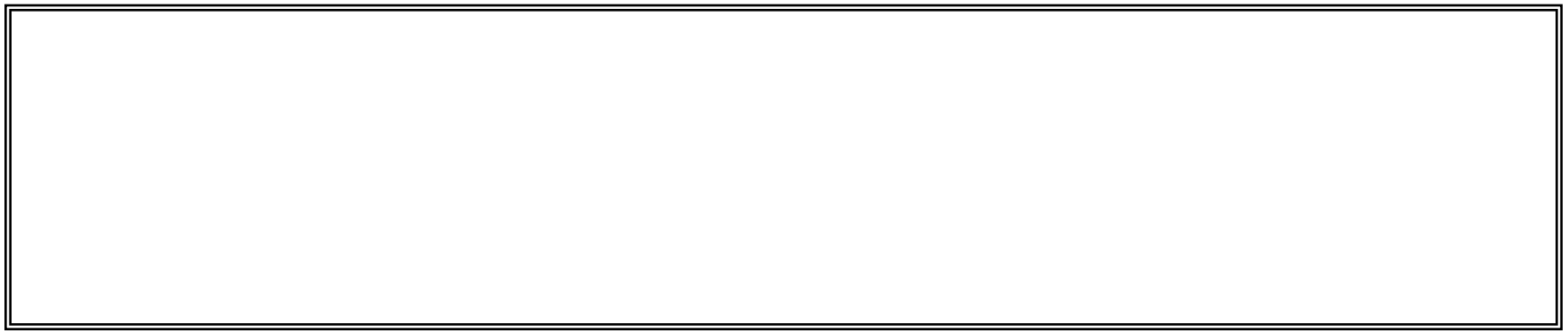
```
public class FactorialExample extends ConsoleProgram {

   private static final int MAX_NUM = 4;

   public void run() {
       for(int i = 0; i < MAX_NUM; i++) {
             imprimir(i + "! = " + factorial(i));
       }
   }

   private int factorial(int n) {
       int result = 1;
       for (int i = 1; i <= n; i++) {
             result *= i;
       }
       return result;
   }
}
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      imprimir(i + "! = " + factorial(i));
   }
}
```

i 

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```

i | 0



```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      imprimir(i + "! = " + factorial(i));
   }
}
```

i | 0

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```

i | 0

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      imprimir(i + "! = " + factorial(i));
   }
}
```

i | 0

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n 0 result i

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n 0 result 1 i

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	0	result	1	i	1

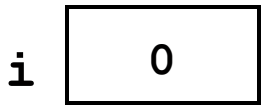
```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n 0 result 1 i 1

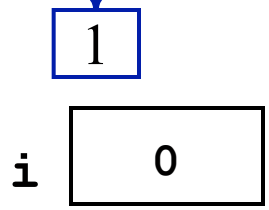
```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	0	result	1	i	1

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      imprimir(i + "! = " + factorial(i));
   }
}
```



```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```




```
0! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```

i | 1

```
0! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```

i | 1

```
0! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```

i | 1

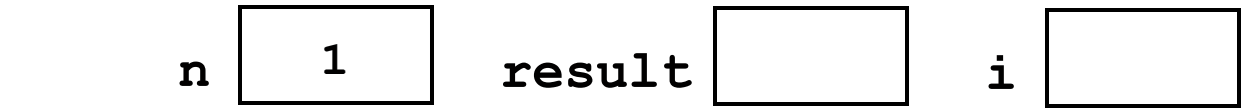
```
0! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```

i | 1

```
0! = 1
```

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```



0! = 1

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n 1 result 1 i

0! = 1

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n 1 result 1 i 1

0! = 1

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	1	result	1	i	1

0! = 1

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	1	result	1	i	1

```
0! = 1
```

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	1	result	1	i	2

```
0! = 1
```

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n 1 result 1 i 2

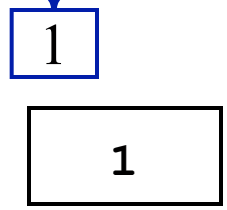
0! = 1

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	1	result	1	i	2

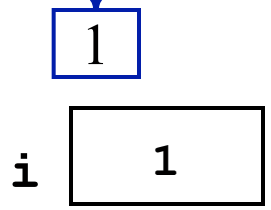
```
0! = 1
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      imprimir(i + "! = " + factorial(i));
   }
}
```



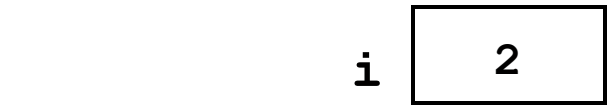

```
0! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```




```
0! = 1
1! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```




```
0! = 1
1! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```

i | 2

```
0! = 1
1! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```

i | 2

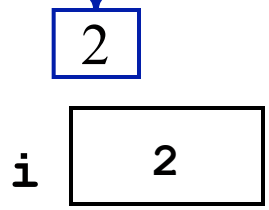
```
0! = 1
1! = 1
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      imprimir(i + "! = " + factorial(i));
   }
}
```

i | 2

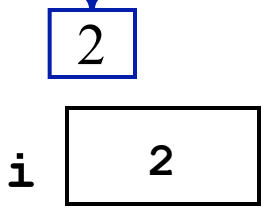
```
0! = 1
1! = 1
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      imprimir(i + "! = " + factorial(i));
   }
}
```




```
0! = 1
1! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```




```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```

i | 3

```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      imprimir(i + "! = " + factorial(i));
   }
}
```

i | 3

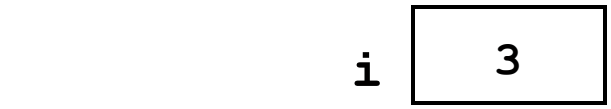
```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```

i | 3

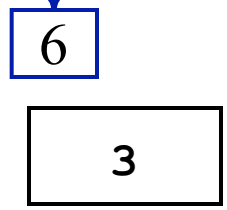
```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      imprimir(i + "! = " + factorial(i));
   }
}
```



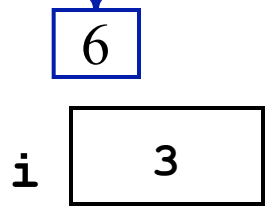

```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      imprimir(i + "! = " + factorial(i));
   }
}
```




```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```




```
0! = 1
1! = 1
2! = 2
3! = 6
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        imprimir(i + "! = " + factorial(i));
    }
}
```

i | 4

```
0! = 1
1! = 1
2! = 2
3! = 6
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      imprimir(i + "! = " + factorial(i));
   }
}
```

i | 4

```
0! = 1
1! = 1
2! = 2
3! = 6
```

Parameters

Every time a method is called, new memoria is created for the call.

Bad Times With Methods

```
// NOTE: This program is buggy!!

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


public void run() {
  int x = 3;
  addFive(x);
  imprimir("x = " + x);
}
```

Let's "trace" this program on the board

Good Times With Methods

```
// NOTE: This program is feeling just fine...

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

public void run() {
   int x = 3;
   x = addFive(x);
   imprimir("x = " + x);
}
```


For primitives: Variables are **not** passed when you use parameters. Values are passed

Pass by “Value”

- Thanks Mehran

More Examples

Changed Name

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

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


Same Variable Name

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

private void cow() {
    int num = 10;
    imprimir(num);
}
```


No Methods in Methods

```
private void run() {
    imprimir("hello world");
    private void sayGoodbye() {
        imprimir("goodbye!");
    }
}
```

Illegal modifier for parameter goodbye, only final is permitted

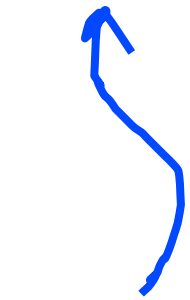
Huh?!?

No Methods in Methods

```
private void run() {
    imprimir("hello world");
    sayGoodbye();
}

private void sayGoodbye() {
    imprimir("goodbye!");
}
```


Methods Called on Objects

```
SRect rect = new SRect(20, 20);
rect.cambiarColor(Color.Blue);
```



receiver

* We will talk about how to define these later in the class

Learn How To:

1. Write a method that takes in input
2. Write a method that gives back output
3. Trace method calls using stacks

Remember Booleans?

Boolean Variable

```
boolean karelIsAwesome = true;

boolean myBool = 1 < 2;
```


Is Square

```
private void run() {
    for(int i = 1; i <= 100; i++) {
        if(isSquare(i)) {
            imprimir(i);
        }
    }
}
```


Boolean Return

```
public void run() {
    for(int i = 1; i <= 100; i++) {
        if(isSquare(i)) {
            imprimir(i);
        }
    }
}

private boolean isSquare(int x) {
    double root = Math.sqrt(x);
    if(root == Math.floor(root)) {
        return true;
    } else {
        return false;
    }
}
```

Boolean Return

```
public void run() {
    for(int i = 1; i <= 100; i++) {
        if(isSquare(i)) {
            imprimir(i);
        }
    }
}

private boolean isSquare(int x) {
    double root = Math.sqrt(x);
    return root == Math.floor(root);
}
```

Boolean Return

```
public void run() {
    for(int i = 1; i <= 100; i++) {
        if(isSquare(i)) {
            imprimir(i);
        }
    }
}

private boolean isSquare(int x) {
    double root = Math.sqrt(x);
    return root == (int)root;
}
```

Extra Exercise

- Greek mathematicians took a special interest in numbers that are equal to the sum of their proper divisors (a proper divisor of *n* is any divisor less than *n* itself). They called such numbers ***perfect numbers***. For example, 6 is a perfect number because it is the sum of 1, 2, and 3, which are the integers less than 6 that divide evenly into 6. Similarly, 28 is a perfect number because it is the sum of 1, 2, 4, 7, and 14.
- Design and implement a Java program that finds all the perfect numbers between two limits. For example, if the limits are 1 and 10000, the output should look like this: