

Programación Orientada a Objetos

Centro de Educación y Formación Académica (CEDUK)

[Inserte frase intelectual abajo]

“”

Alguien importante

Recapitulemos...

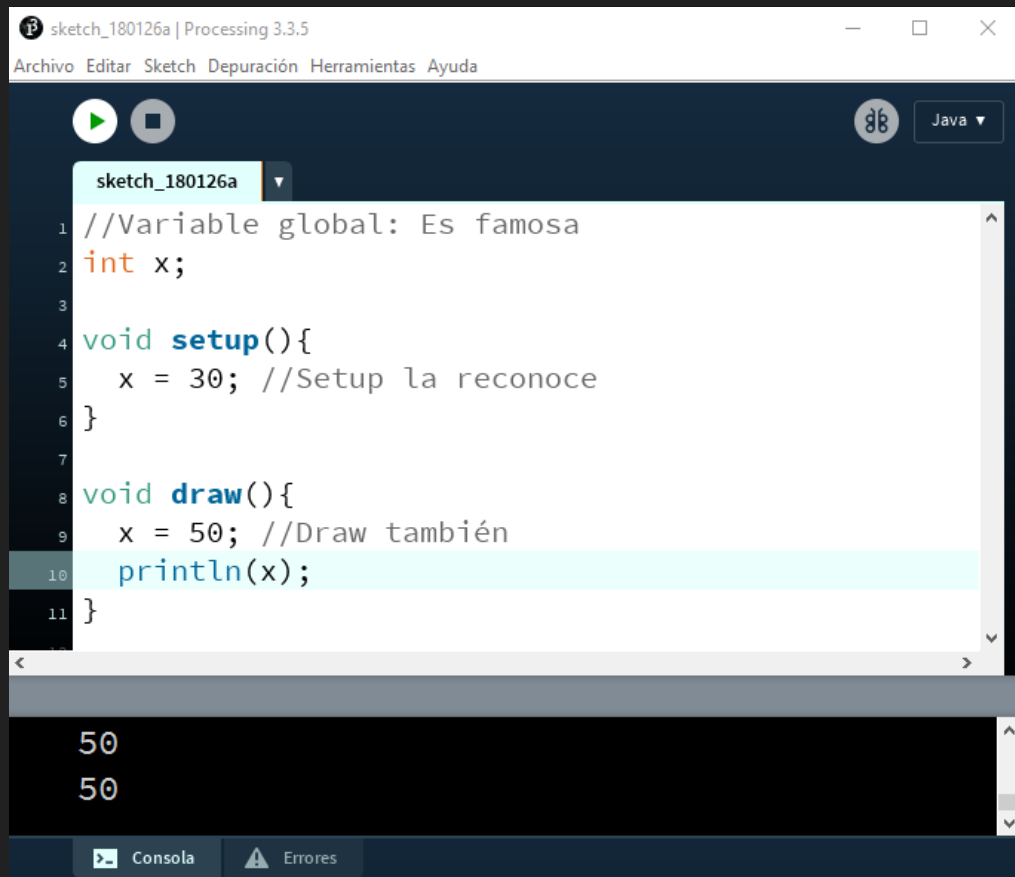
- ¿Qué es una clase?
- ¿Qué es un objeto?
- Función para sacar el promedio de 100 números

Variables locales y globales

- Una **variable local** es aquella que sólo es utilizable dentro del código de la función.
- Una **variable global** es aquella cuyo ámbito es todo el programa, incluso dentro de una función.



Variables locales y globales

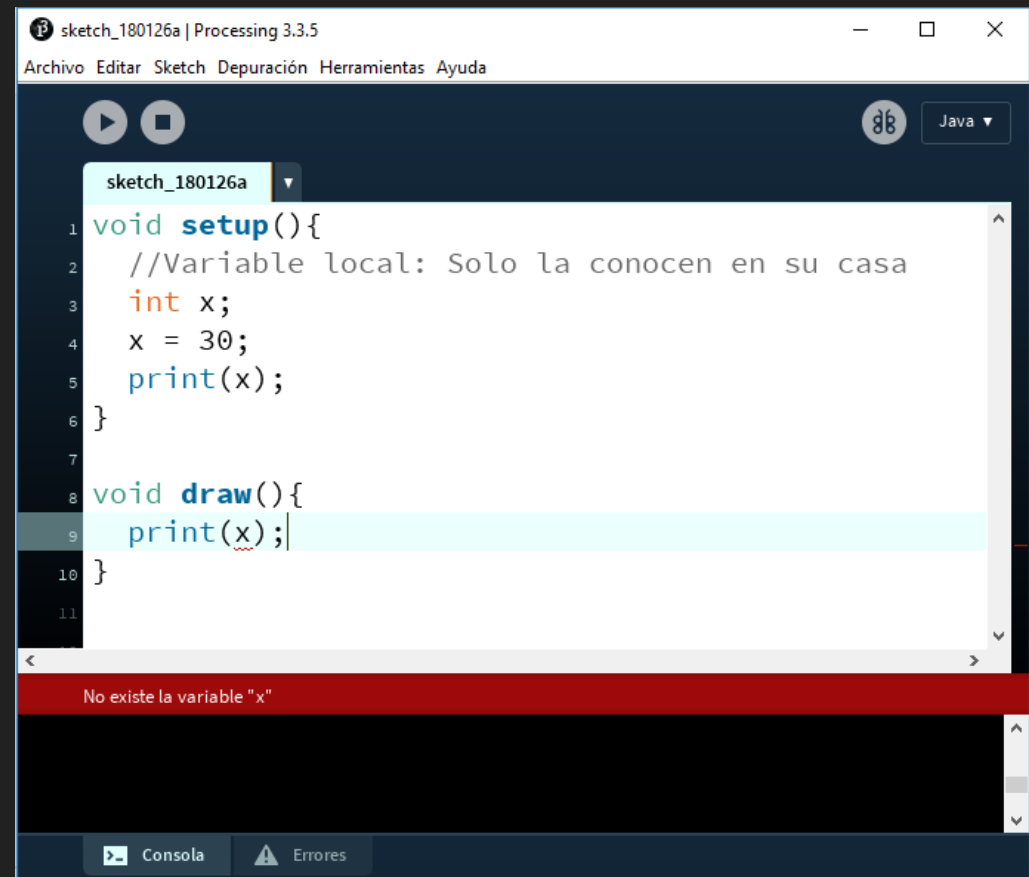


The screenshot shows the Processing IDE interface for a sketch named 'sketch_180126a'. The code is as follows:

```
1 //Variable global: Es famosa
2 int x;
3
4 void setup(){
5   x = 30; //Setup la reconoce
6 }
7
8 void draw(){
9   x = 50; //Draw también
10  println(x);
11 }
```

The console at the bottom shows the output of the sketch:

```
50
50
```



The screenshot shows the Processing IDE interface for a sketch named 'sketch_180126a'. The code is as follows:

```
1 void setup(){
2   //Variable local: Solo la conocen en su casa
3   int x;
4   x = 30;
5   print(x);
6 }
7
8 void draw(){
9   print(x);
10 }
11 }
```

The console at the bottom shows a red error message:

```
No existe la variable "x"
```

Usando un objeto



- Antes de ver como escribir una clase analicemos como usar objetos en nuestro programa principal hace del mundo un lugar mejor. Abajo se puede observar un programa para hacer la representación de un “limón” sin usar programación orientada a objetos.

Datos (Variables globales):

Color del limón,
Peso del limón,
Tamaño del limón,
Posición del limón,

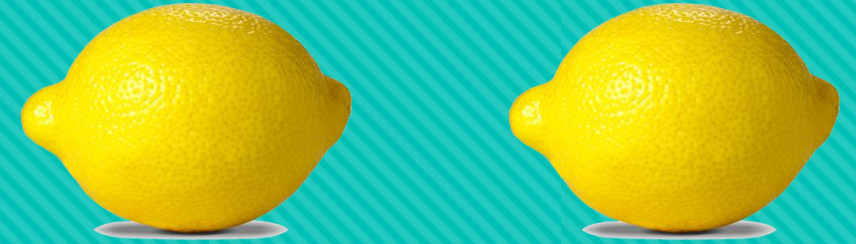
Setup:

Inicializar color del limón,
Inicializar el peso del limón,
Inicializar tamaño del limón,
Inicializar posición del limón,

Draw:

Pintar fondo,
Mostrar el limón en la posición y del tamaño indicado,
Incrementar la posición del limón (caer).

Usando un objeto



- Representación de dos “limones” sin usar programación orientada a objetos

Datos (Variables globales):

Color del limón 1,
Peso del limón 1,
Tamaño del limón 1,
Posición del limón 1,
Color del limón 2,
Peso del limón 2,
Tamaño del limón 2,
Posición del limón 2,

Setup:

Inicializar color del limón 1,
Inicializar el peso del limón 1,
Inicializar tamaño del limón 1,
Inicializar posición del limón 1,
Inicializar color del limón 2,
Inicializar el peso del limón 2,
Inicializar tamaño del limón 2,
Inicializar posición del limón 2,

Draw:

Pintar fondo,
Mostrar el limón 1 en la posición y del tamaño indicado,
Incrementar la posición del limón 1.
Mostrar el limón 2 en la posición y del tamaño indicado,
Incrementar la posición del limón 2.

****Piense en el caso de representar 100
'limones'.***

Usando un objeto



La programación orientada a objetos nos permite tomar las variables y funciones fuera del programa principal y almacenarlas dentro de un objeto 'limón':

1. Un objeto 'limón' tiene acceso a sus datos (color, peso, tamaño, posición).
2. La segunda parte del objeto 'limón' son las acciones que puede realizar, los métodos (funciones dentro de un objeto). El objeto 'limón' puede caer y ser mostrado en pantalla. Usando un diseño orientado a objetos el pseudocódigo puede mejorar y verse algo así.

Datos (Variables globales):

Objeto limón,

Setup:

Inicializar objeto limón,

Draw:

Pintar fondo,

Mostrar objeto limón,

Caída de limón.

Usando un objeto

- Representación de dos “limones” usando programación orientada a objetos

Datos (Variables globales):

Objeto limón 1,

Objeto limón 2,

Setup:

Inicializar objeto limón 1,

Inicializar objeto limón 2,

Draw:

Pintar fondo,

Mostrar objeto limón 1,

Mostrar objeto limón 2,

Caída de limón 1

Caída de limón 2

****Piense en el caso de representar 64
'limones'.***

Usando un objeto

- Representación de 64 'limones' haciendo uso de arrays y programación orientada a objetos:

Datos (Variables globales):

Array de limones [64],

Setup:

Para i = 0 hasta 63:

 Inicializar elemento "i" de array de limones

FinPara

Draw:

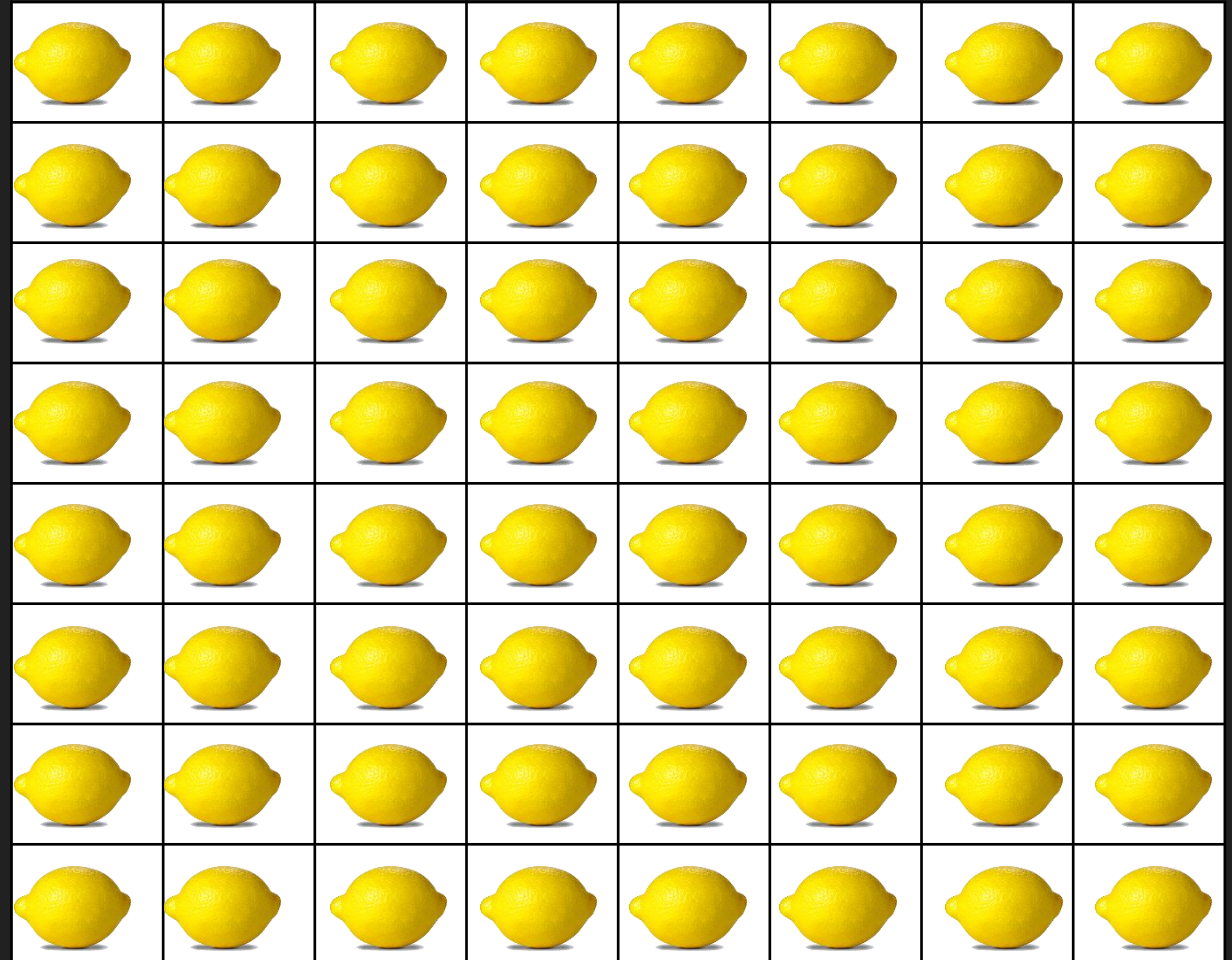
Pintar fondo,

Para i = 0 hasta 63:

 Mostrar objeto limón "i" del array,

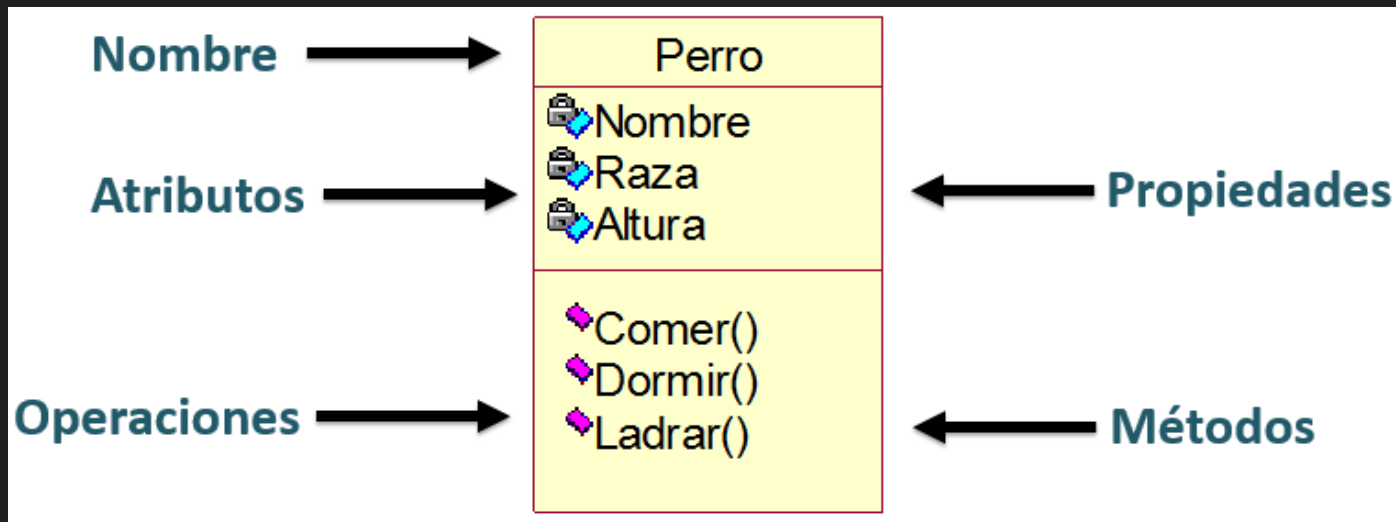
 Caída de limón "i" del array,

FinPara



Escribiendo la clase

- El ejemplo del “limón” demuestra como el uso de objetos en Processing hace un código más reducido y legible. El trabajo difícil consiste en escribir la plantilla que describe las propiedades y funciones del objeto, es decir la clase.
- Todas las clases deben incluir cuatro elementos: *nombre*, *atributos*, *constructor* y *métodos*.



Escribiendo la clase

A continuación se define cada elemento de la clase:

- **Nombre de la clase:** El nombre es determinado por "class ElNombreQueQuieras" (Note que no contiene espacios). Luego encerramos el código de la clase entre llaves, después de la declaración del nombre. El nombre de la clase empieza tradicionalmente con mayúscula (para distinguirlo de los nombres de variables, que tradicionalmente empiezan con minúscula).
- **Atributos:** Es una colección de variables. Estas variables son comúnmente denominadas como *variables de instancia*, ya que cada instancia de un objeto contiene este conjunto de variables.
- **Constructor:** El constructor es una función especial adentro de una clase que crea la instancia del objeto mismo. Es donde se le dan las instrucciones de inicialización al objeto. Es como la función **setup()** de Processing, solo que el constructor se utiliza para crear un objeto individual dentro del sketch, donde sea que un nuevo objeto sea creado a partir de esta clase. Tiene siempre el mismo nombre que la clase y es invocado por el operador **new**: "**Limon miLimon = new Limon();**".
- **Métodos:** Podemos añadir funcionalidad a nuestro objeto por medio de la escritura de métodos. Estos se definen de la misma manera que una función, con un tipo de dato de retorno, nombre, argumentos y el cuerpo de código.

```
// Simple non OOP Car
```

```
color c;  
int xpos;  
int ypos;  
int xspeed;
```

```
void setup() {
```

```
  size(200,200);  
  c = color(255);  
  xpos = width/2;  
  ypos = height/2;  
  xspeed = 1;  
}
```

```
void draw() {  
  background(0);  
  display();  
  drive();  
}
```

```
void display () {  
  rectMode(CENTER);  
  fill(c);  
  rect  
(xpos, ypos, 20, 10);  
}
```

```
void drive () {  
  xpos = xpos + xspeed;  
  if (xpos > width) {  
    xpos = 0;  
  }  
}
```

```
class Car {
```

```
  color c;  
  float xpos;  
  float ypos;  
  float xspeed;
```

```
  Car() {  
    c = color(255);  
    xpos = width/2;  
    ypos = height/2;  
    xspeed = 1;  
  }
```

```
  void display () {  
    rectMode(CENTER);  
    fill(c);  
    rect(xpos, ypos, 20, 10);  
  }
```

```
  void drive() {  
    xpos = xpos + xspeed;  
    if (xpos > width){  
      xpos = 0;  
    }  
  }
```

```
}
```

The class name

Data

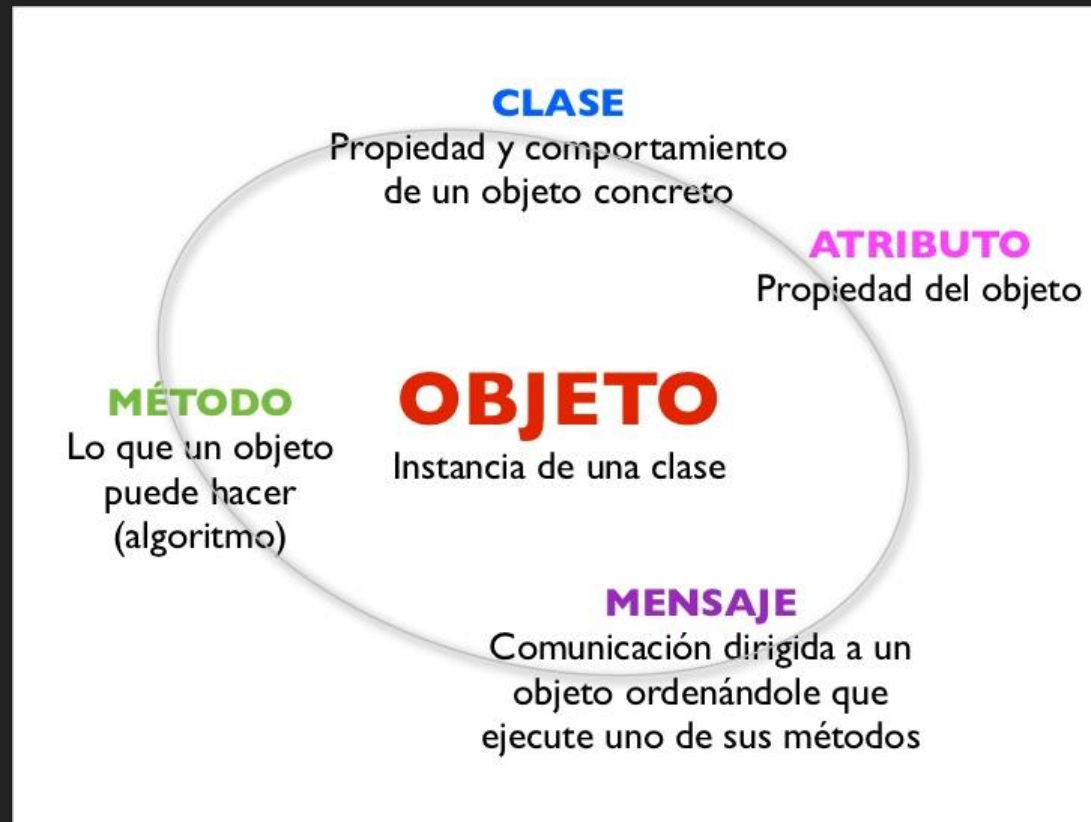
Constructor

Functionality

Ejercicio

```
_____ {  
    color hairColor;  
    float height;  
    _____ () {  
        _____  
        _____  
    }  
    _____ {  
        _____  
        _____  
        _____  
    }  
}
```

En resumen...



Tarea 2

- *En Processing, hacer un programa “no orientado a objetos” representando uno de los objetos que propuso en la tarea anterior.*
- *A partir de ese programa escribir una clase (en la libreta o en un documento de Word).*