

# Proyectos IV

Diseño Industrial

# Arreglos

- Semana 3 -

# ¿Qué son los arreglos?

- Agrupaciones que almacenan datos del mismo tipo.
- Pueden ser de datos primitivos (int, float, char)
- Pueden ser de datos referenciados (String)

# Ejemplo ...

```
int[] valores;
valores = new int[5];
valores[0] = 1;
valores[1] = 2;
valores[2] = 3;
valores[3] = 4;
valores[4] = 5;

for(int i = 0; i < valores.length; i++){
    valores[i] = i+1;
}

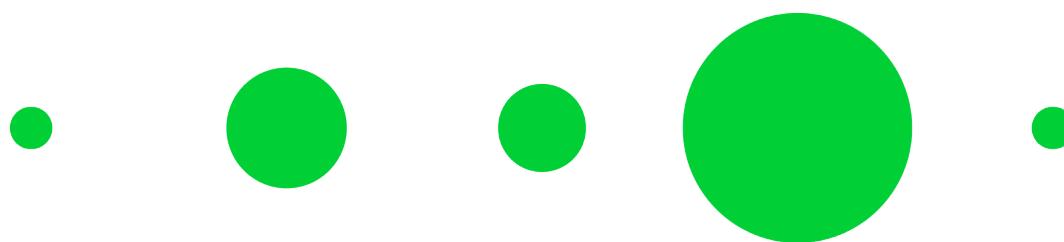
for(int i = 0; i < valores.length; i++){
    println("Valor en "+i+" : "+valores[i]);
}
```

Los arreglos son agrupaciones variables, la utilización de los mismos facilita(simplifica) el manejo de un mayor numero de datos de forma más sencilla al mismo tiempo que disminuye el número de líneas de código.

```
Valor en 0 : 1
Valor en 1 : 2
Valor en 2 : 3
Valor en 3 : 4
Valor en 4 : 5
```

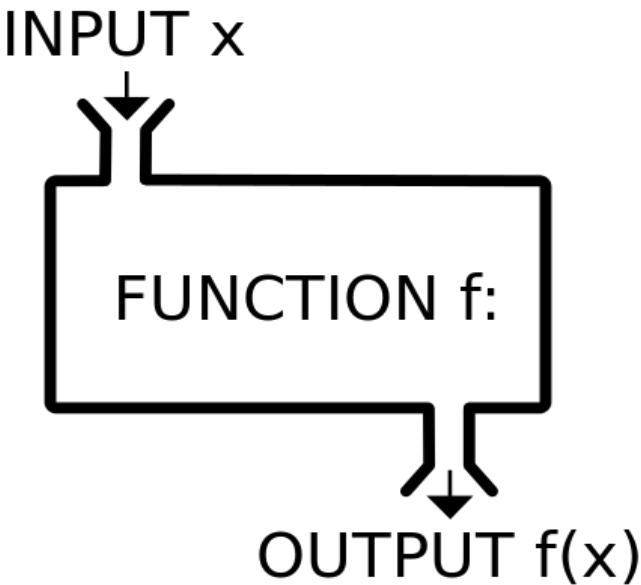
# Ejercicio con Arreglos

- Debe crear 5 elipses en la pantalla que comienzan con un diámetro de 20 pixeles y al hacer clic sobre cada uno crece 5 pixeles.
- El color puede ser seleccionado por usted.



# Métodos

- Semana 3 -



Un método en Java es un conjunto de instrucciones que realizan una determinada tarea y a las que podemos invocar mediante un nombre.

# ¿Qué son los métodos?

- **Instrucciones** que puedo reutilizar, repetir.
- **Cálculos** que dependen de entradas fijas
- **Validaciones** que requieren una respuesta (int, boolean, float, char, String ... etc)

# Ejemplo método ...

```
float areaCuadrado(int lado){  
    return lado*lado;  
}
```

Retorno + Nombre ( parámetros ) {  
 cuerpo de método  
 return valor (si el retorno es diferente de null)  
}

# Ejemplo llamado al método ...

Estructura de una llamada a un método

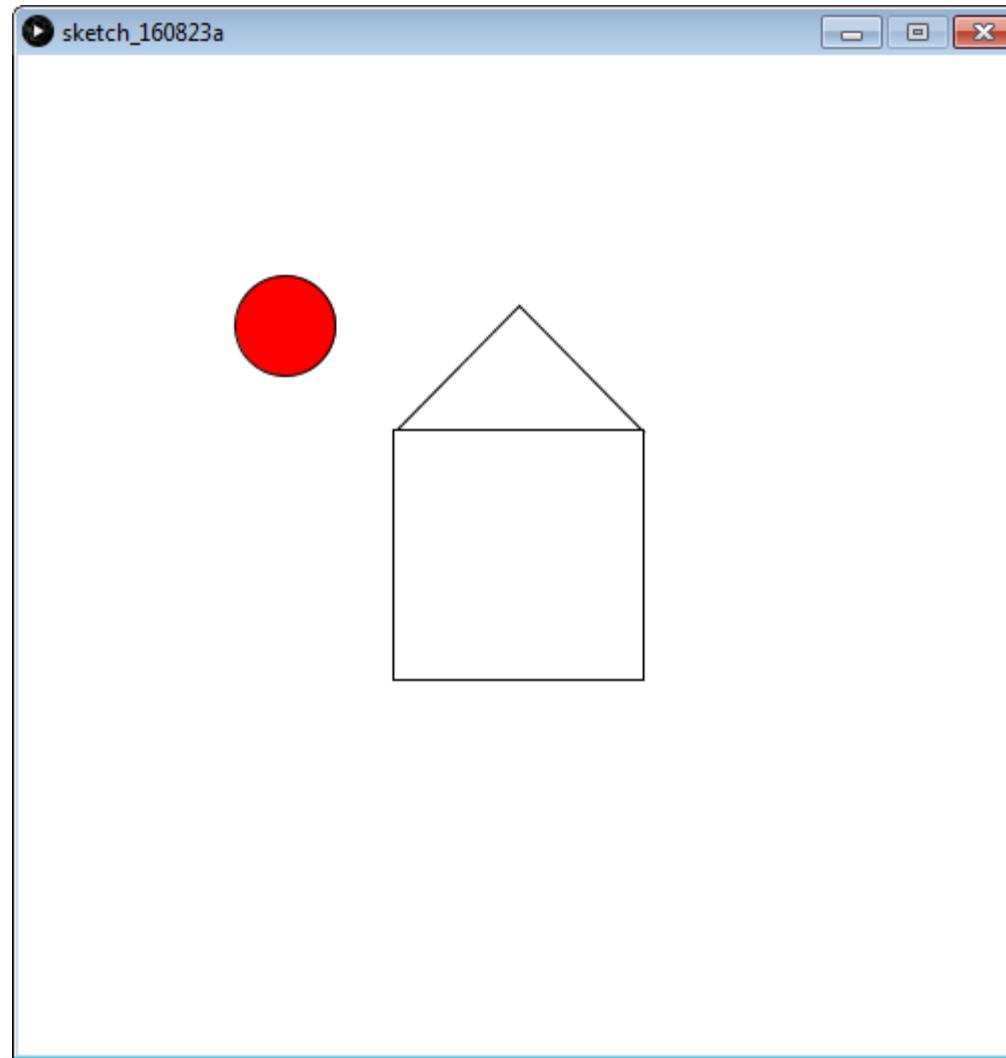
```
variableDelTipoRetorno = NombreMetodo(paramMetodo)
```

```
float valorArea = areaCuadrado(20);  
println("el valor del area es: " + valorArea);
```

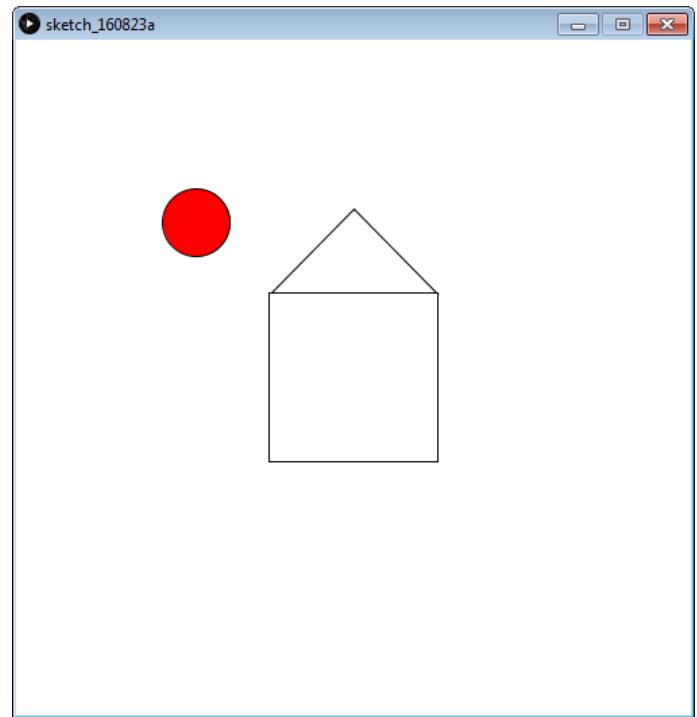
```
float areaCuadrado(int lado){  
    return lado*lado;  
}
```

```
el valor del area es: 400.0
```

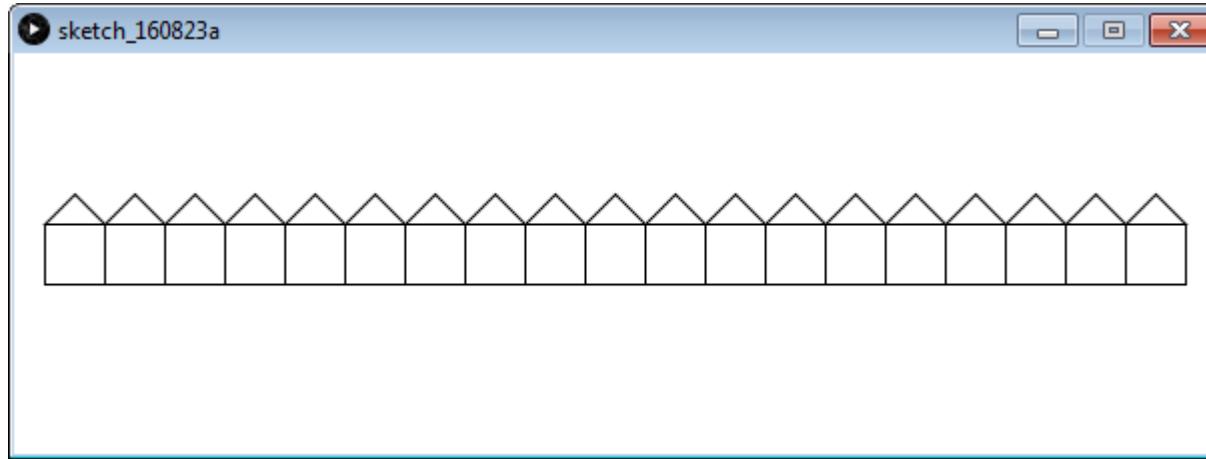
# Ejemplo “Resumen”



```
void setup(){
  size(500,500);
}
void draw(){
  background(255);
  pintarEllipseRojoAleatorio();
  pintarCasa(width/2,height/2,width/4);
}
void pintarEllipseRojoAleatorio(){
  fill(255,0,0);
  ellipse(random(width), random(height), 50, 50);
}
void pintarCasa(int dx, int dy, int ancho){
  fill(255);
  rectMode(CENTER);
  triangle(dx-ancho/2, dy-ancho/2, dx+ancho/2, dy-ancho/2, dx, dy-ancho);
  rect(dx, dy, ancho, ancho);
}
```



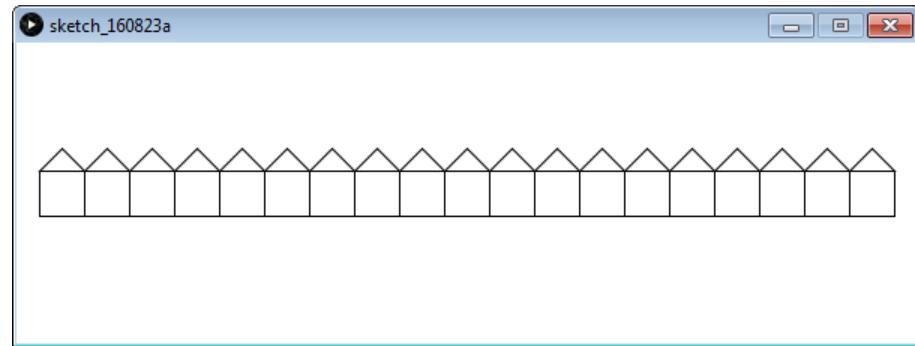
# ¿Cómo logramos esto?



Recuerden la clase pasada (**repetitivas**) y  
piensen en los **métodos**

# ¿Qué podemos ver aquí?

```
void setup(){
  size(600,200);
}
void draw(){
  background(255);
  for(int i = 0 ; i < 19 ; i++){
    pintarCasa(width/20+(i*width/20),height/2,height/20);
  }
}
void pintarCasa(int dx, int dy, int ancho){
  fill(255);
  rectMode(CENTER);
  triangle(dx-ancho/2, dy-ancho/2, dx+ancho/2, dy-ancho/2, dx, dy-ancho);
  rect(dx, dy, ancho, ancho);
}
```



# Programación Orientada a Objetos

- Semana 4 -

# Unidad 1: Interfaces digitales

## *Objetivos de la unidad*

- Comprender los requerimientos básicos para el análisis de problemas que involucren el pensamiento secuencial o algorítmico.
- Entender los fundamentos de programación para la creación de interfaces digitales navegables sensibles al usuario.

# ¿Qué es la programación Orientada a Objetos?

- Es un paradigma de programación.
- No es el único existente.
- Es el más utilizado en el mundo (así como Java).



# ¿En qué se diferencia de lo que venimos trabajando?

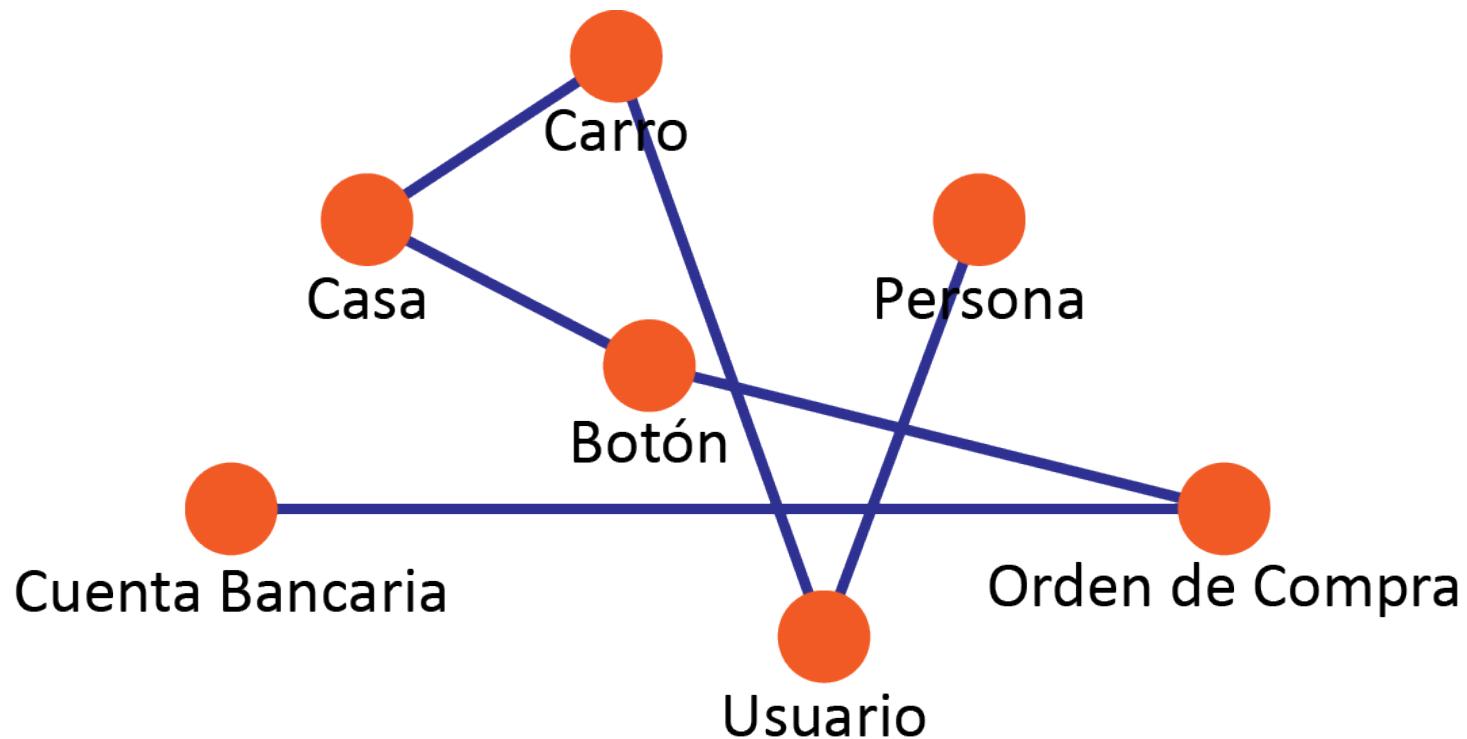
Es más **ordenado** y permite modelar **problemas más complejos** de manera más **simple\***.

\*(en teoría y con algo de práctica)

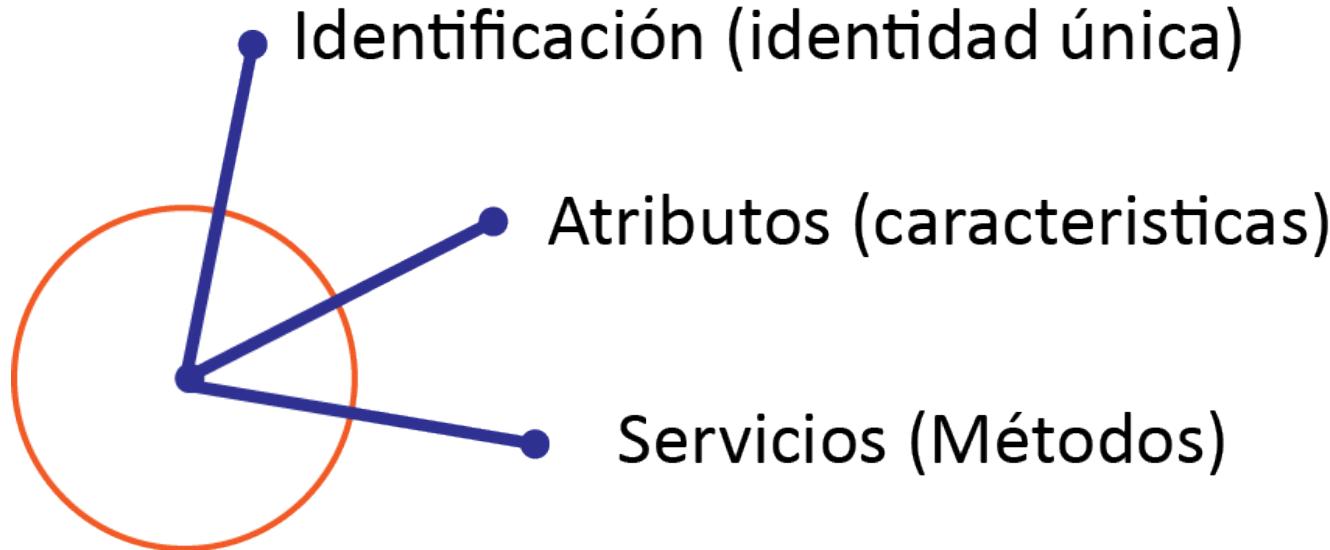
# **¡TODO ES UN OBJETO!**



# ¡TODO ES UN OBJETO!



# Componentes de un Objeto



- La identificación es un sustantivo y mayúscula (inicial)
- Los atributos son variables o constantes
- Los servicios son métodos o funciones

# Book

properties

title  
author  
pages

methods

read()  
buy()

The Giver  
Lois Lowry

179

Fox in Socks  
Dr. Seuss

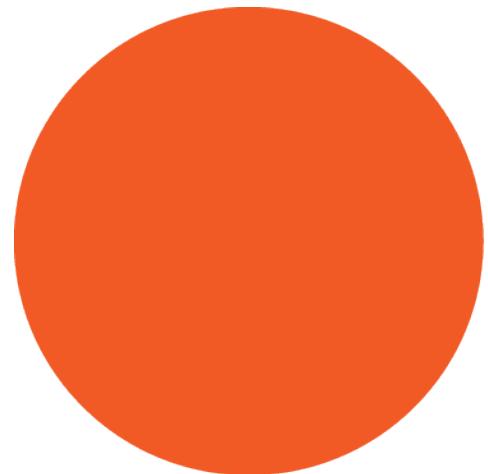
62

A Wrinkle in Time  
Madeline L'Engle

211

# Fundamentos del Paradigma I

Abstracción



Objeto

# Fundamentos del Paradigma II

## Encapsulamiento



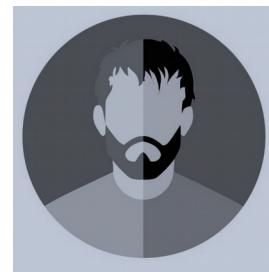
# Fundamentos del Paradigma III

Herencia



# Fundamentos del Paradigma IV

## Polimorfismo



# Estructuras asociadas al paradigma en Java

# ¿Qué son las clases?

En informática, una clase es una **plantilla para la creación de objetos** de datos según un modelo predefinido.

Las clases se utilizan para representar entidades o conceptos, como los sustantivos en el lenguaje.

**Objeto**

**Clase**

**Objeto**

# ¿Cuál es la clase aquí?



# ¿Cuál es la clase aquí?



Vehículo

# ¿Cuál es la clase aquí?



Vehículo **de**  
**construcción**

# ¿Cuál es la clase aquí?



Vehículo **aéreo**



Vehículo  
**acuático**

# ¿Cómo se **crean** las clases en Processing?

```
class Vehiculo{  
    int x;  
    int y;  
  
    Vehiculo(){  
        x = (int) random(width);  
        y = (int) random(height);  
    }  
  
    Vehiculo(int vx, int vy){  
        x = vx;  
        y = vy;  
    }  
  
    void mover(){  
        x++;  
    }  
}
```

**Encabezado**

**Variables**  
(Atributos)

**Métodos**  
(Servicios)

# ¿Cómo se usan las clases en Processing?

```
Vehiculo miVehiculoA;  
Vehiculo miVehiculoB;  
  
void setup(){  
    size(500,500);  
    miVehiculoA = new Vehiculo();  
    miVehiculoB = new Vehiculo(100,100);  
}  
  
void draw(){  
    background(255);  
    miVehiculoA.pintar();  
    miVehiculoB.pintar();  
    miVehiculoA.mover();  
    miVehiculoB.mover();  
}
```

**Declaración**

**Instanciación**  
(asignación)

**Invocación**  
(solicitud servicios)

# Ejemplo en Vivo

- Círculos móviles que responden al clic cambiando de color (aleatorio)

# Ejercicio en Clase

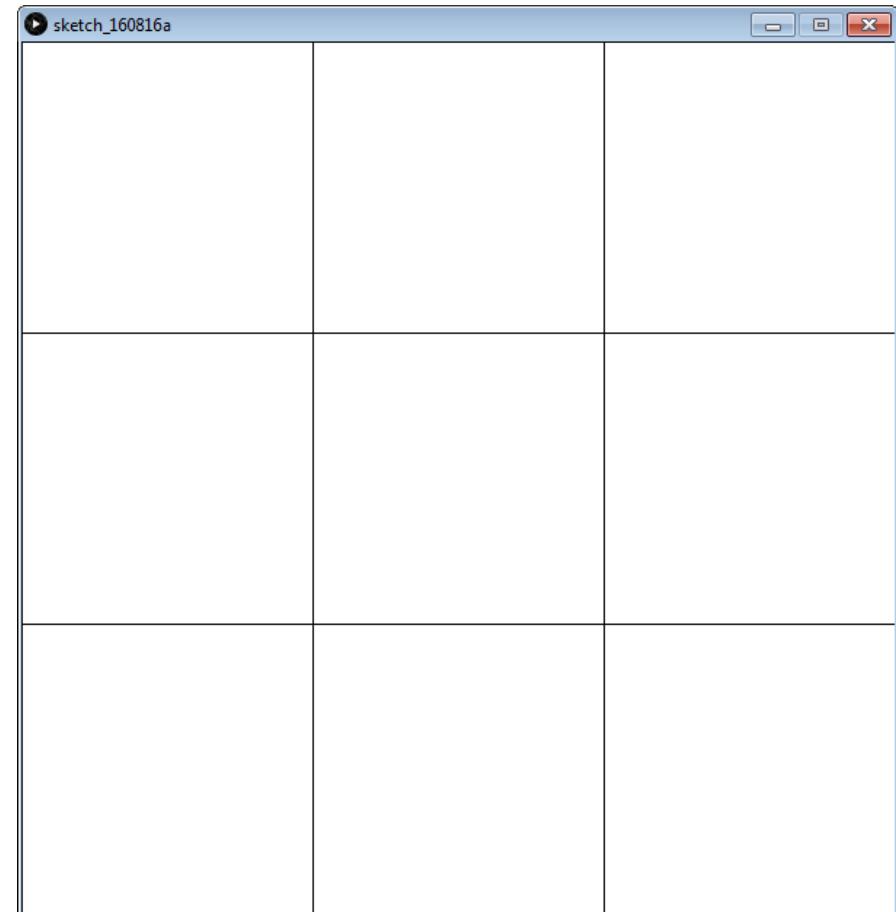
Semana 4

# Ejercicio en Clase (parte 1)

## Usando objetos

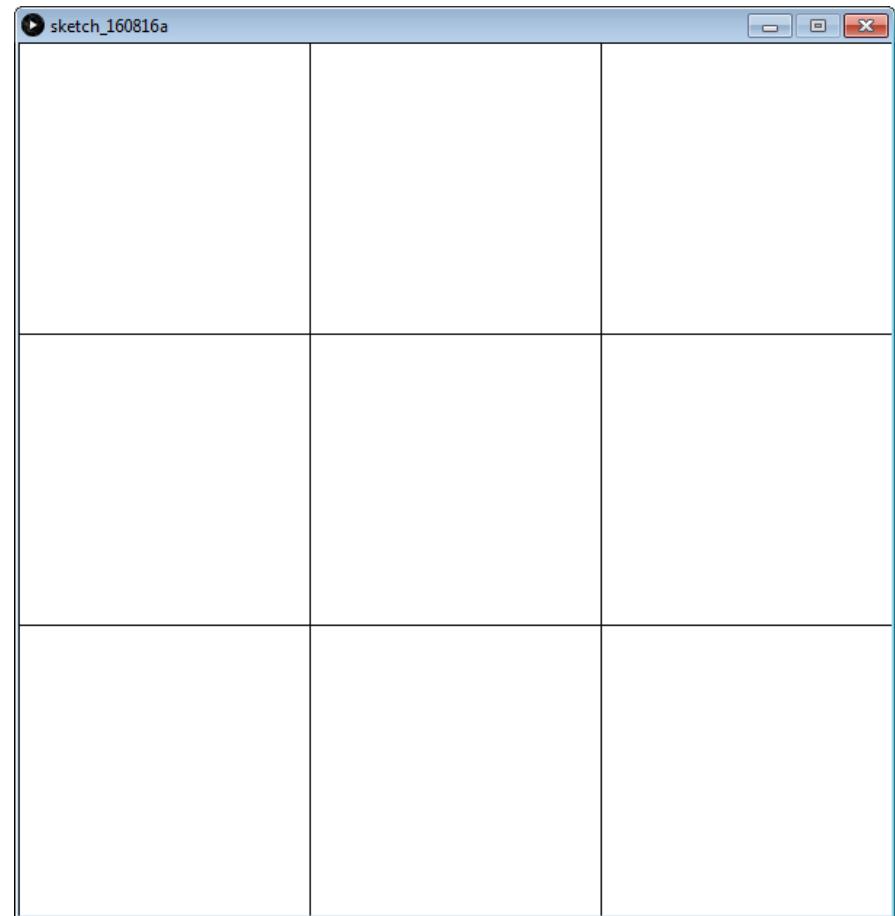
- Construya una clase Casilla
- Cree un método constructor que reciba como parámetros las posiciones x y de la Casilla.
- Incluya un método pintar que dibuje un cuadrado blanco de tamaño width/3
- Cree un arreglo de 9 objetos Casilla:

```
Casilla[] casillas = new  
Casilla[9];
```



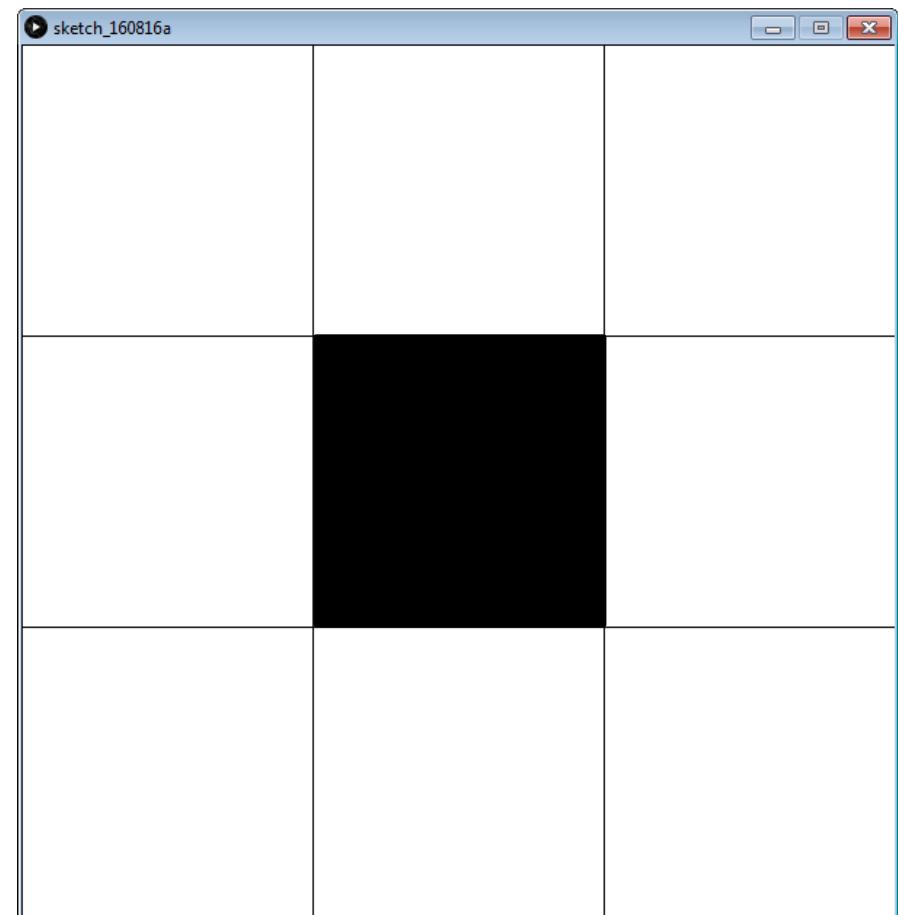
# Ejercicio en Clase (parte 2)

- Agregue un método para validar si se hizo clic sobre una Casilla.
- Genere una repetitiva que recorra el arreglo de casillas preguntando si se hizo clic sobre ellas.  
(dentro del método mousePressed)
- Imprima en consola en qué posición del arreglo está la casilla que respondió.



# Ejercicio en Clase (parte 3)

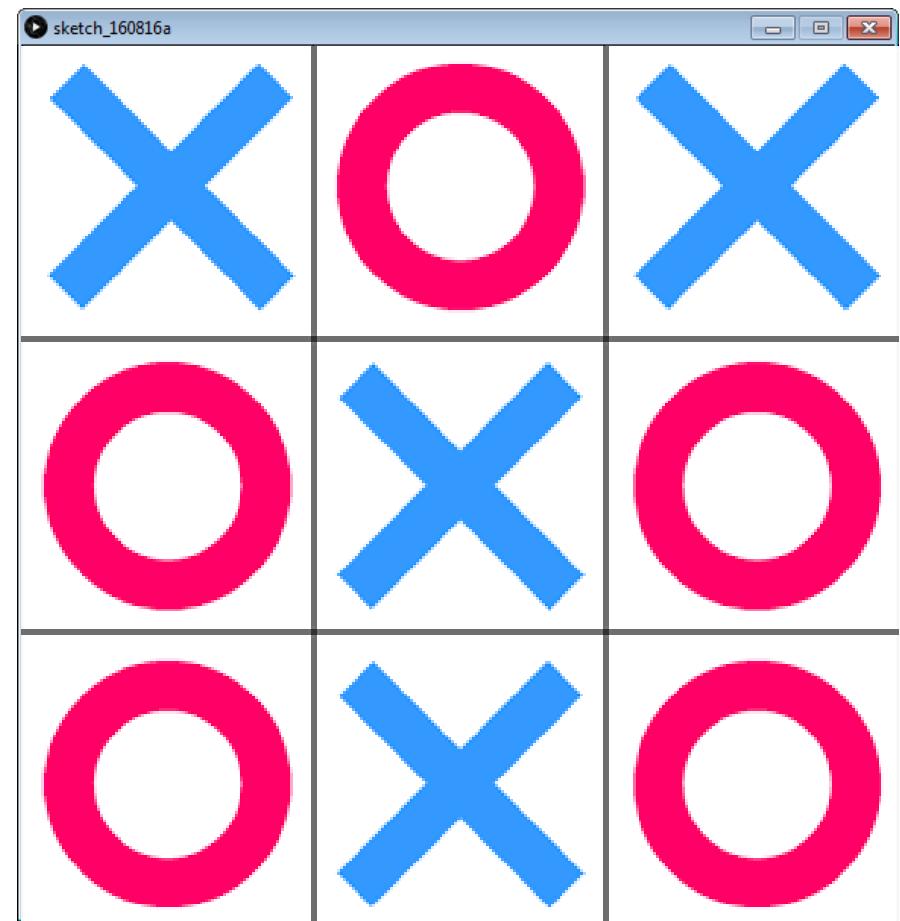
- Agregue una variable estado a la clase Casilla, el valor inicial debería ser 0.
- Si una casilla responde sí cambie el estado por 1.
- Modifique el método pintar para que si el estado de la casilla es 0 se vea blanco y si es uno se vea negro



# Ejercicio en Clase (parte 4)

Construya las estructuras necesarias para garantizar que:

- Si se presiona de nuevo sobre un ítem en estado 1 este pasa a estado 2, si se presiona sobre un estado 2 regresa a 0.
- El estado 0, 1 y 2 estén representados por diferentes colores
- Modifique el método pintar mostrar además del color de fondo una imagen en el centro de la casilla (Equis, Circulo o en blanco). Use imágenes png de 200x200



# Equipos

Equipos de 3 personas

Enlace en Moodle.

**FECHA DE ENTREGA: Semana 7**

# OBJETIVO

**La idea es construir una interfaz para un usuario ficticio para controlar o apoyar su entorno (vivienda, vehículo o similar)**

# Selección de Usuarios (Proyecto 1)

1. Esquimal
2. Vikingo
3. Samurái
4. Muisca
5. Ciudadano del imperio Romano
6. Ciudadano de una colonia lunar
7. Azteca
8. Ninja
9. Señor feudal
10. Cruzado
11. Sarraceno

# Tarea

Finalizar el ejercicio Triqui verificando si hay un ganador, filas, columnas o diagonales.

Propuesta para Proyecto I – Interfaz de Control de Vivienda por usuario.

[https://www.youtube.com/user/shiffman/playlists?shelf\\_id=2&view=50&sort=dd](https://www.youtube.com/user/shiffman/playlists?shelf_id=2&view=50&sort=dd)