



# CS Bridge

# ¿Quién es un programador?



# ¿Quién soy yo?



# Un gran equipo

## Profesores



Colin



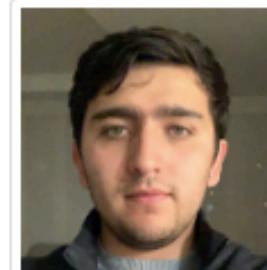
Mario



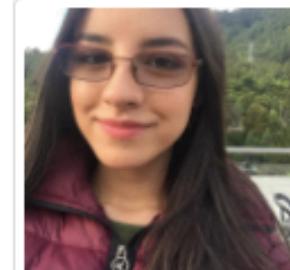
Ana



Evani



Jorge



Juliana



Sierra



Michael



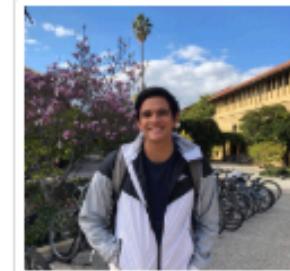
Nathan



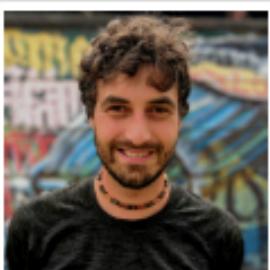
Nicolás



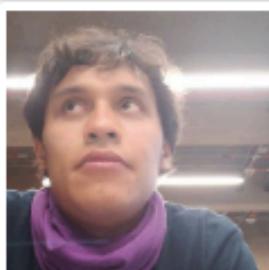
Pietro



Eddie



Chris



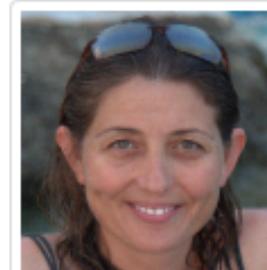
Santiago



Dario



Nick



Asena

# Organizadores



**Asena**



**Darío**



**Nick**

# Stanford?



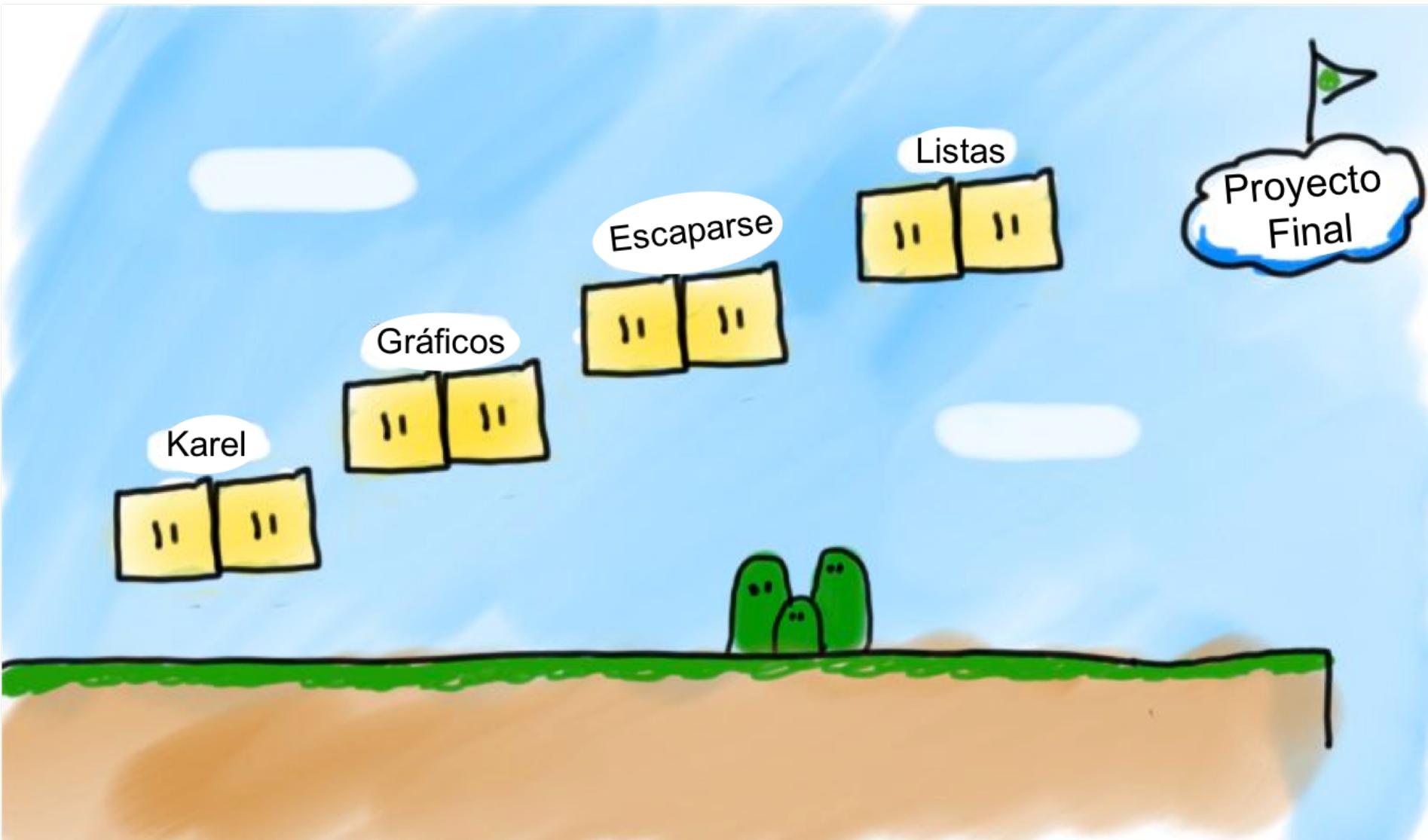
# Stanford



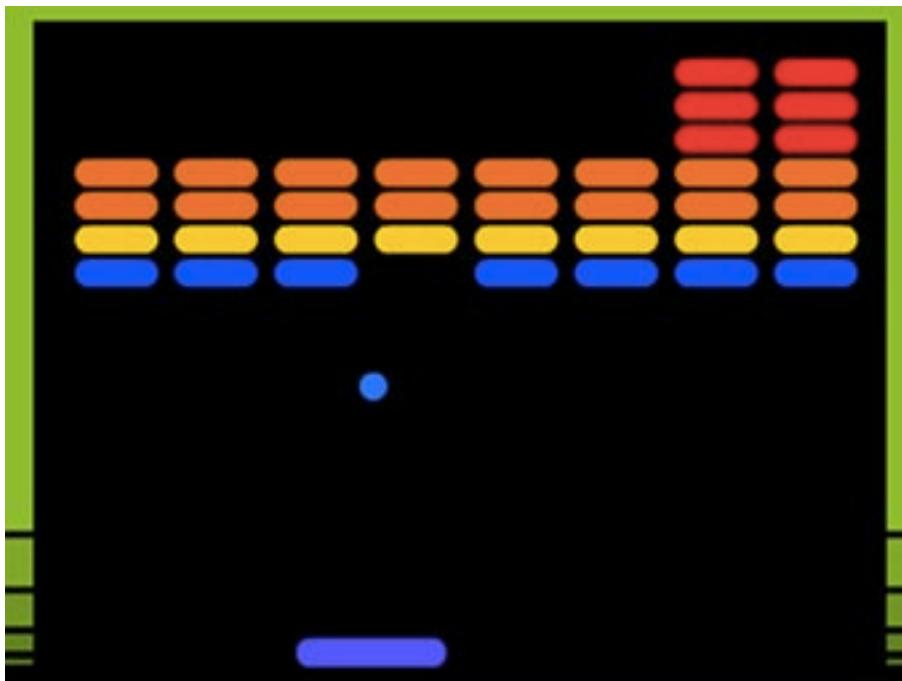
# Trabajo colaborativo



# Nuestra aventura



# Escaparse



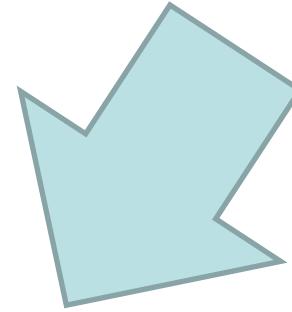
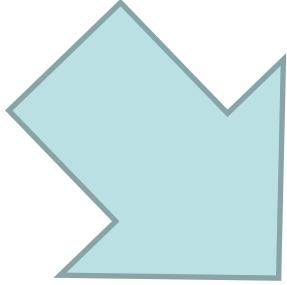
# Prerequisites



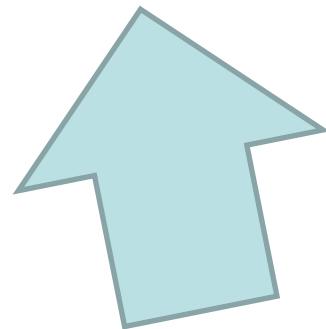
# ¿Está prendido?



# Sitio web



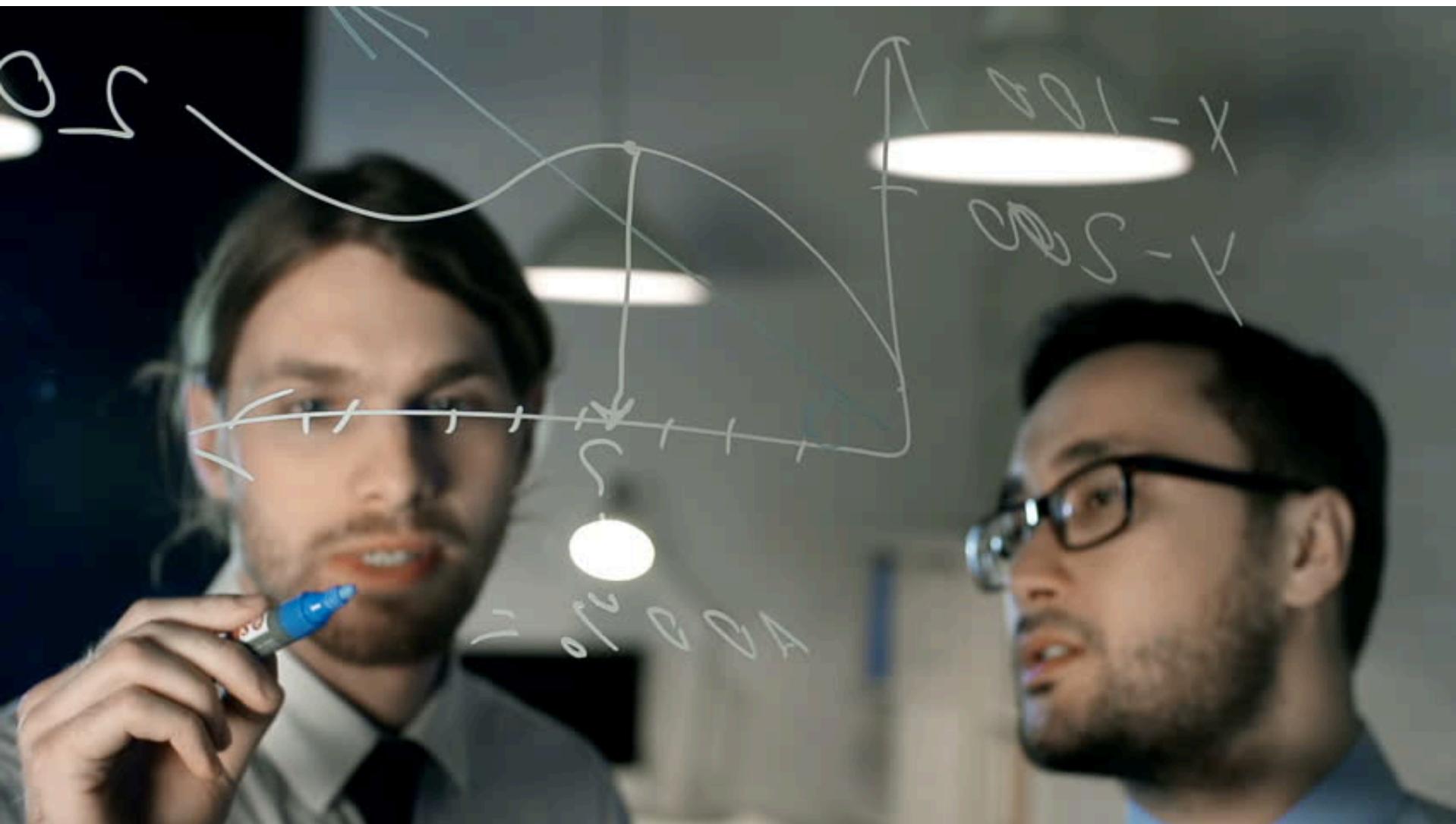
<http://uniandes.csbridge.org>



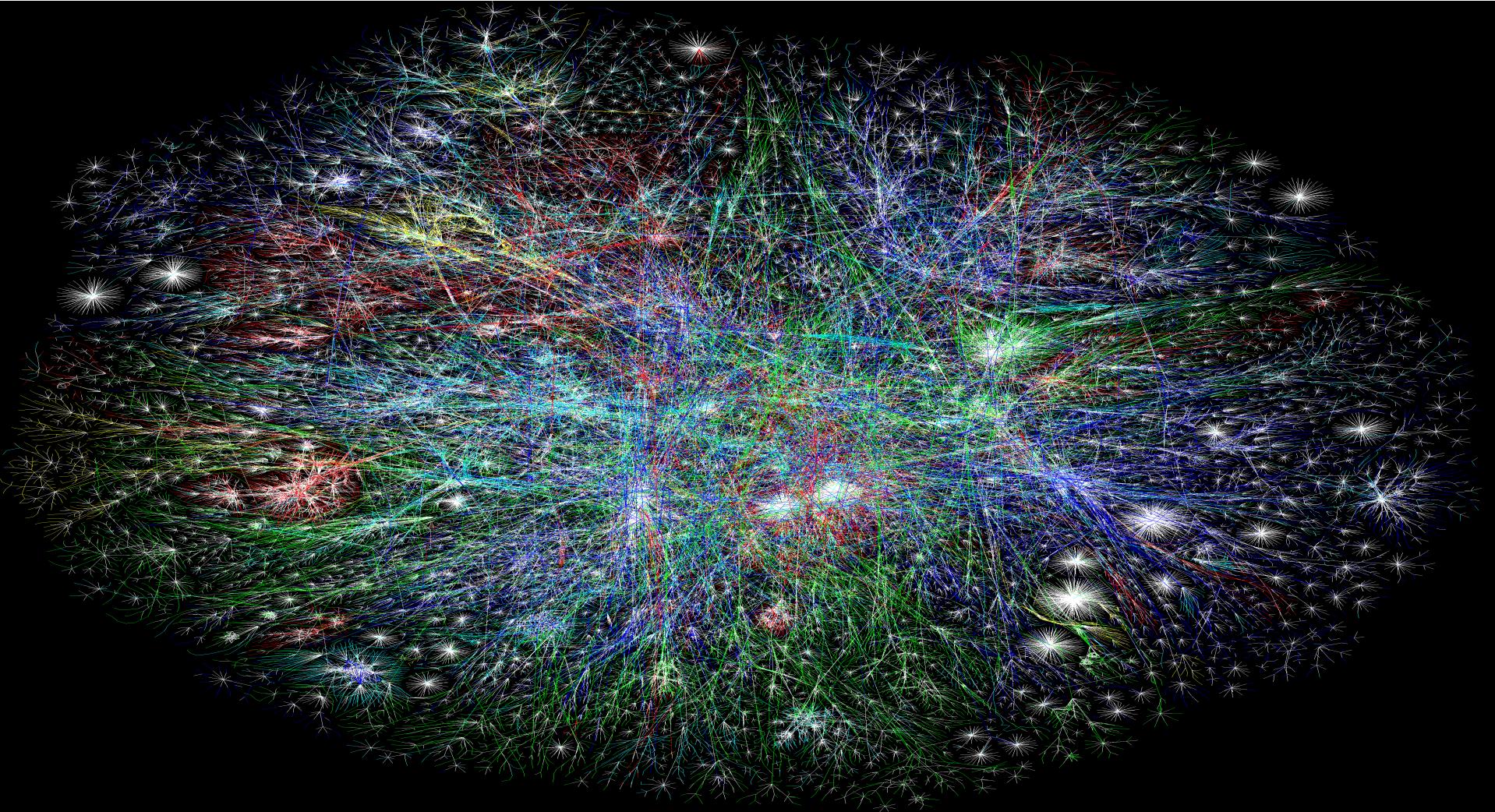
# ¿Y si me quedo atrás?



# Comparte ideas sin hablar del código



# El poder de la computación



# Conoce a Karel el robot

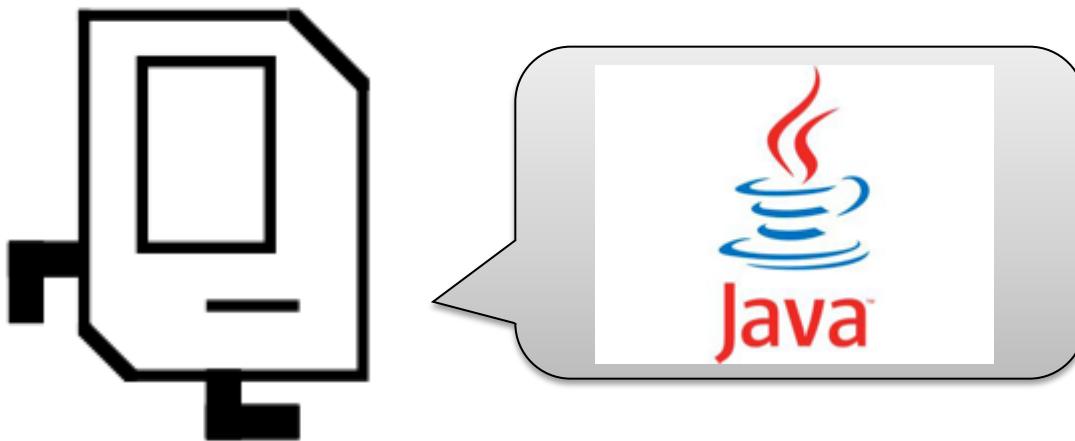


# Conoce a Karel el robot



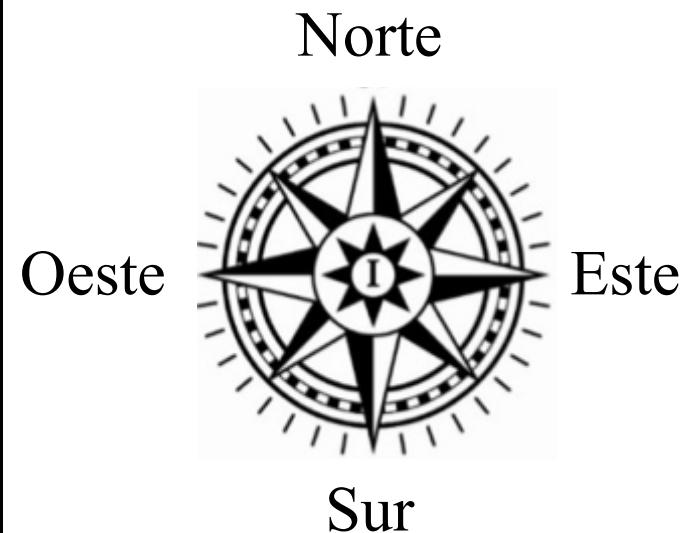
¡Hola mundo!

# Karel habla Java

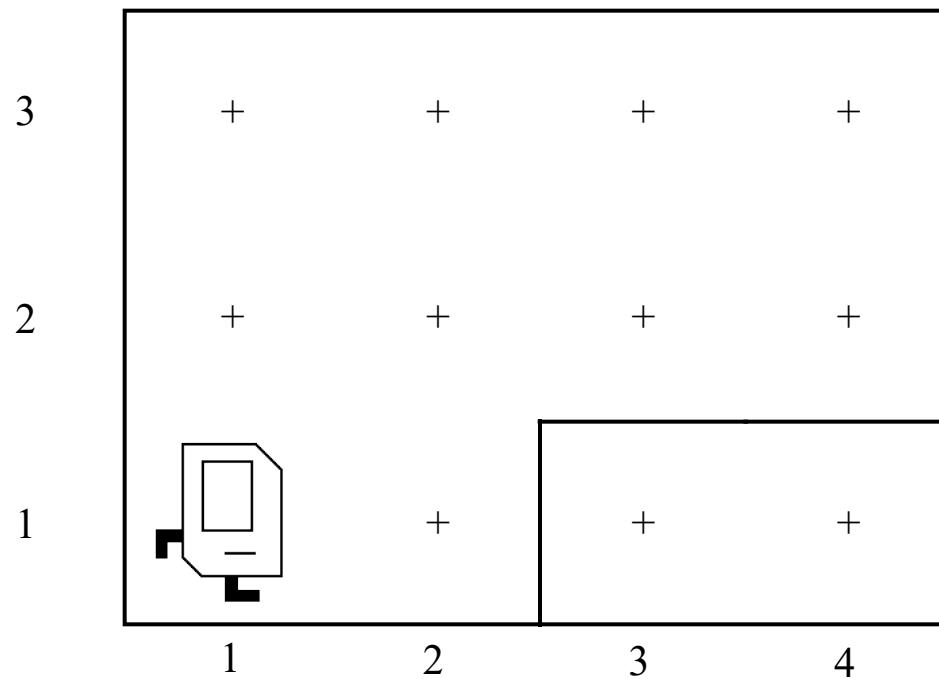


# El mundo de Karel

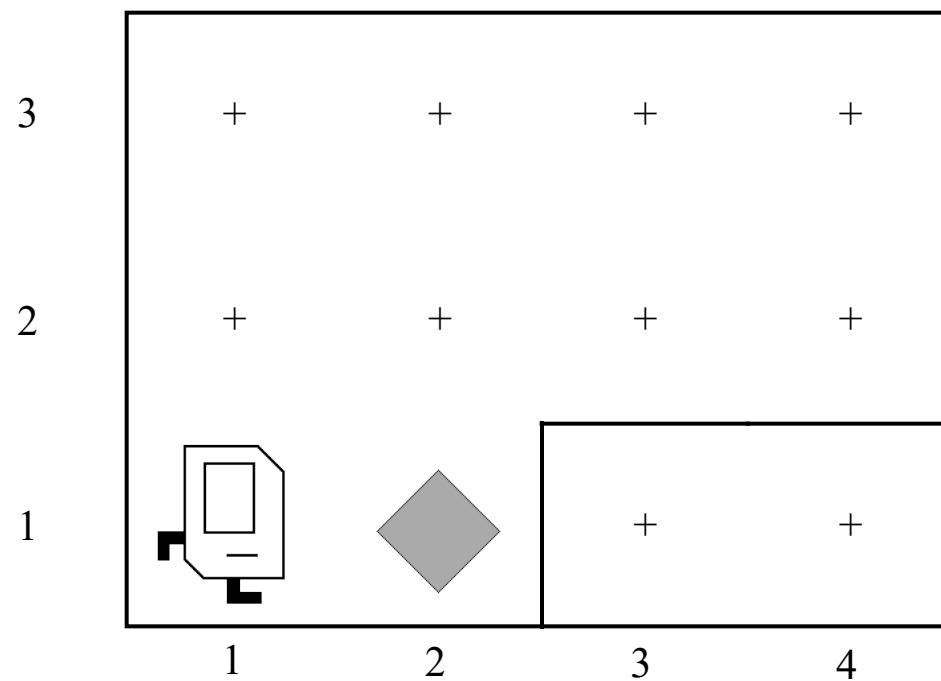
3	+	+	+	+	
2	+	+	+	+	
1		+	+	+	
	1	2	3	4	5



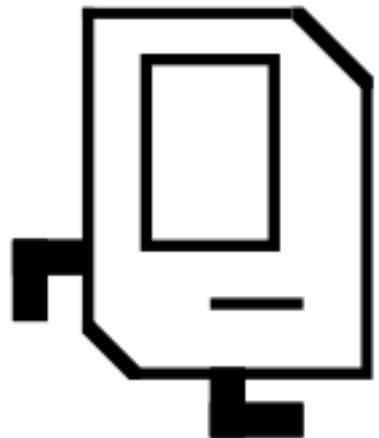
# Paredes



# Conos



# Entiende cuatro comandos



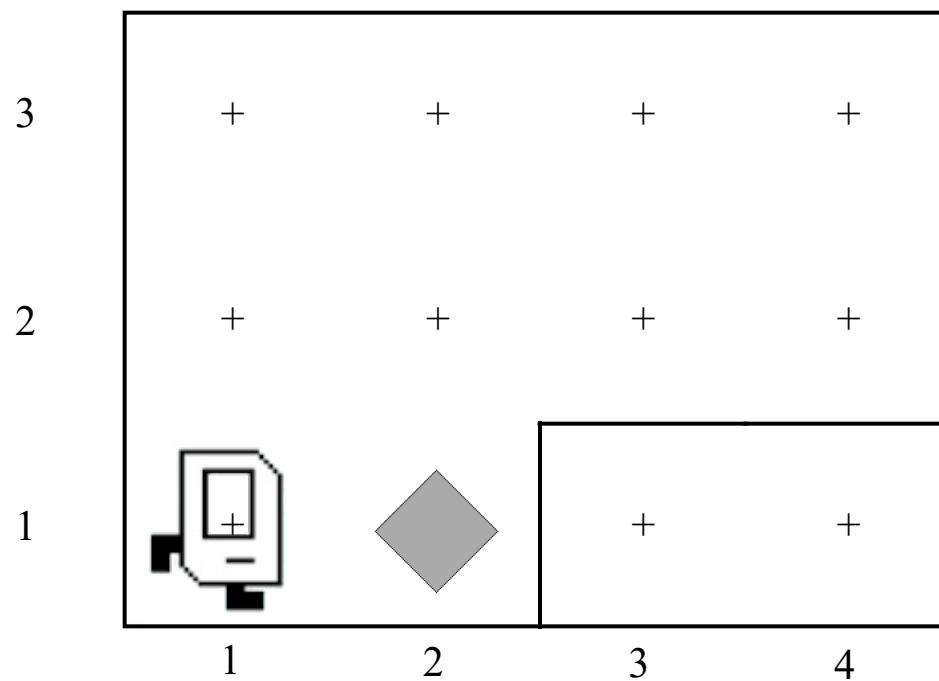
moverse( );

girarIzquierda( );

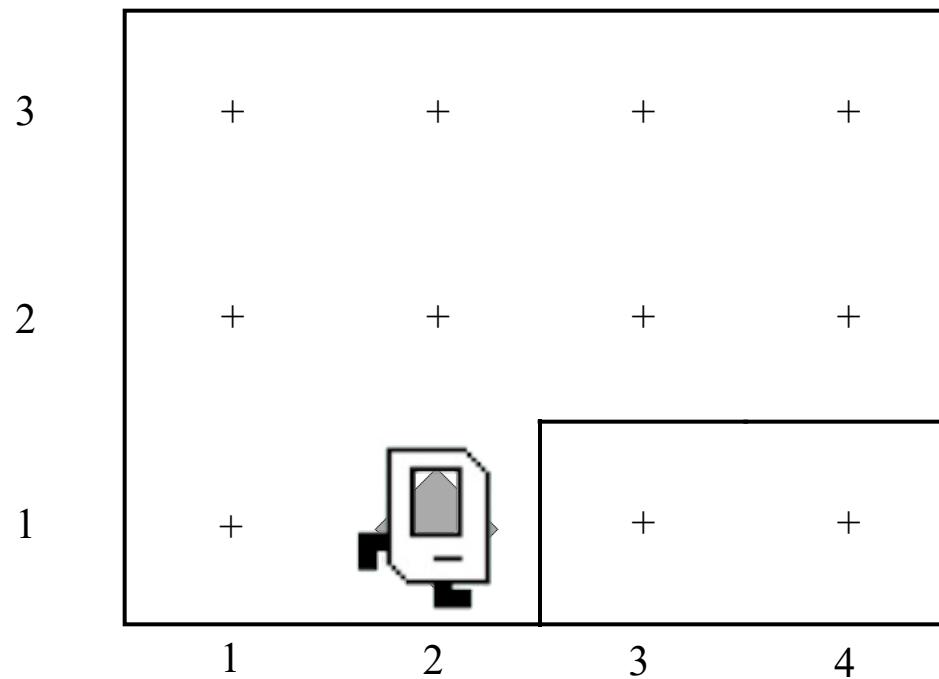
recogerCono( );

ponerCono( );

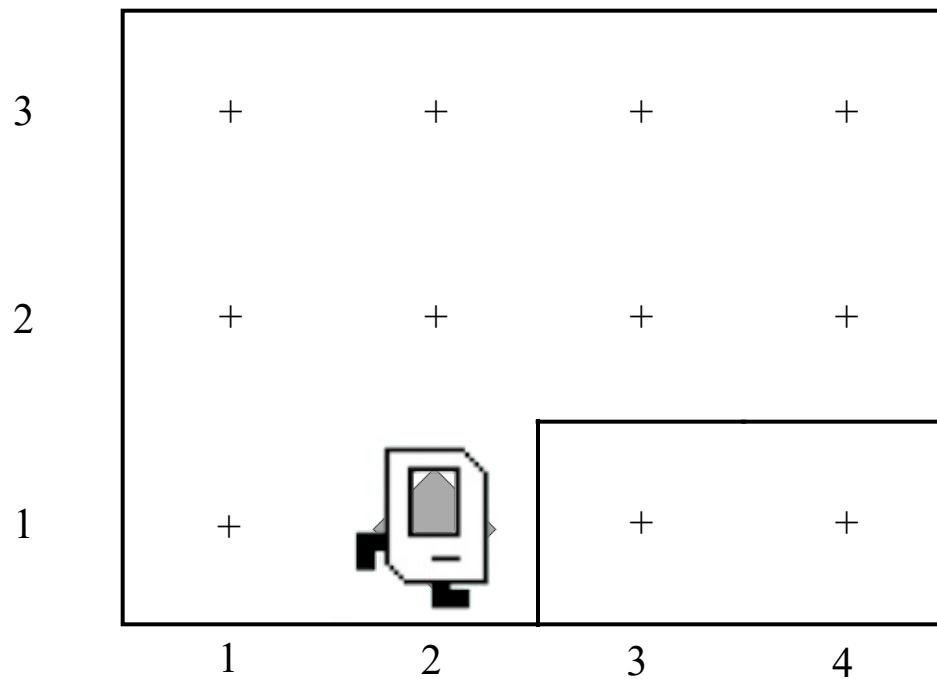
# moverse();



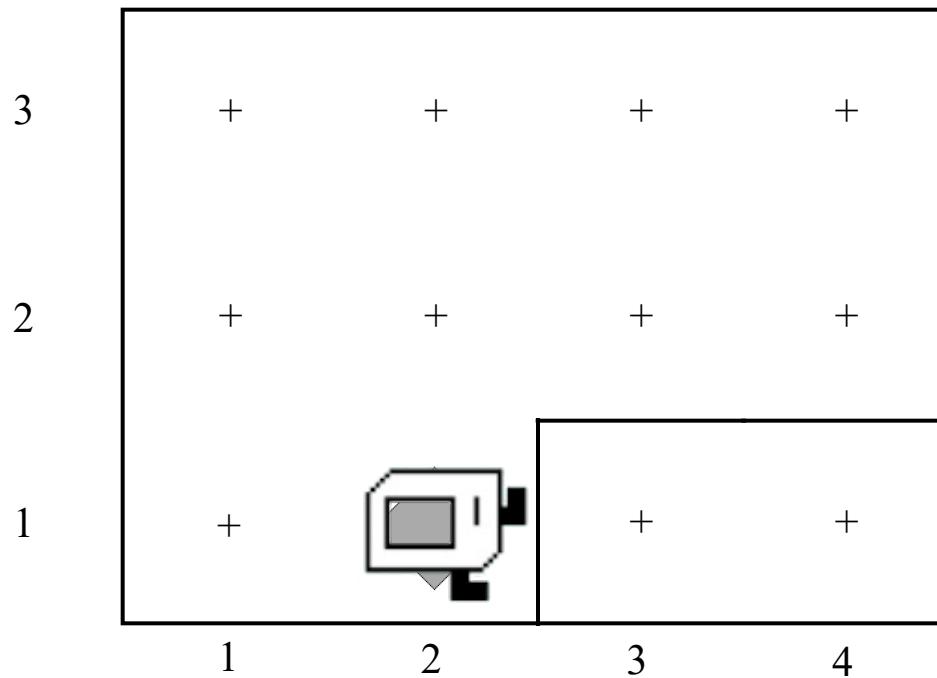
# **moverse();**



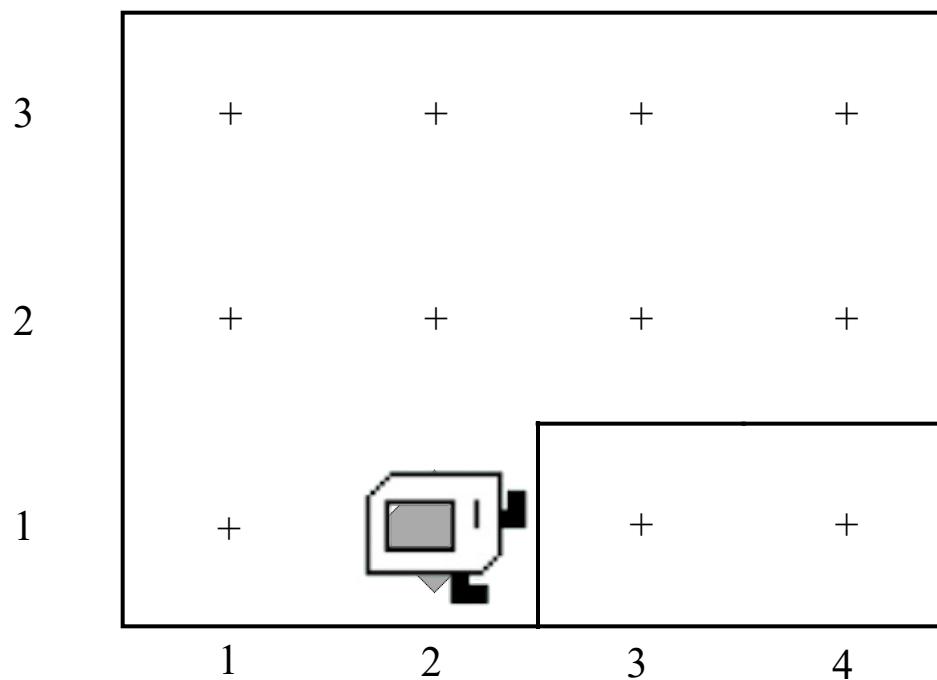
# **girarIzquierda();**



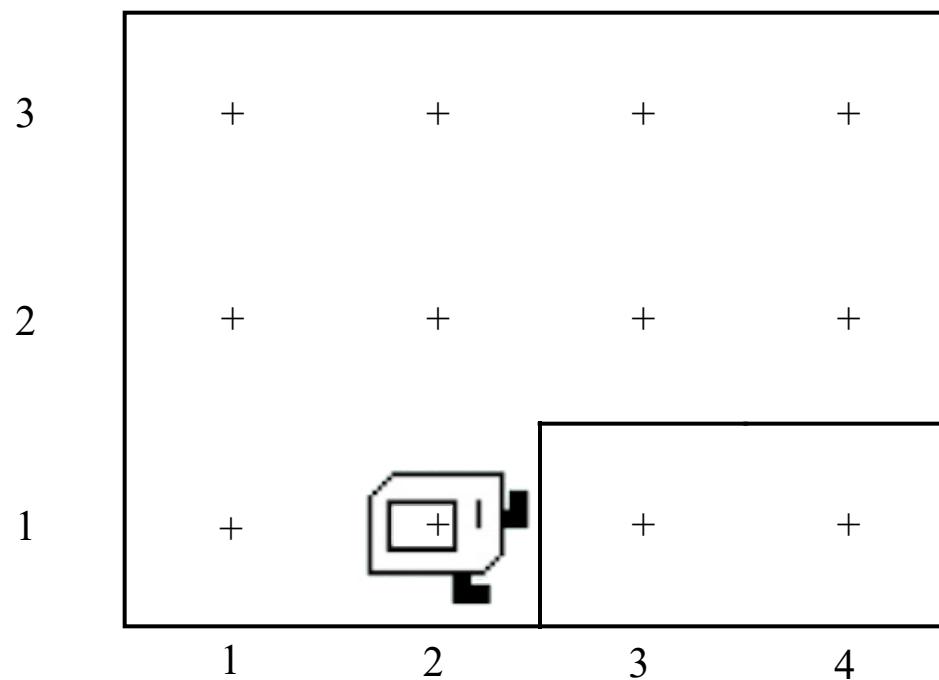
# **girarIzquierda();**



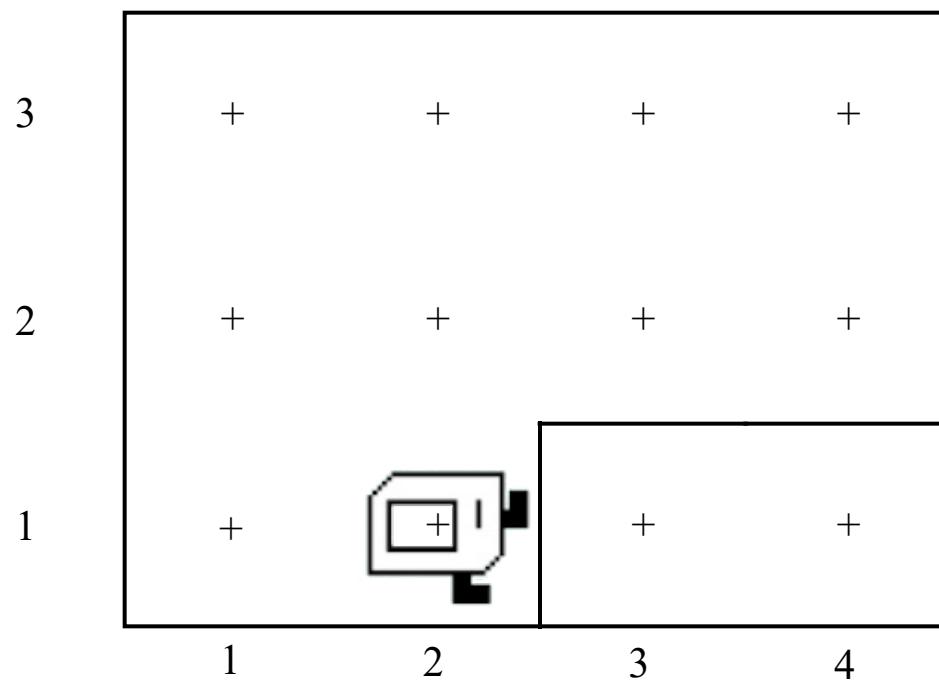
# **recogerCono();**



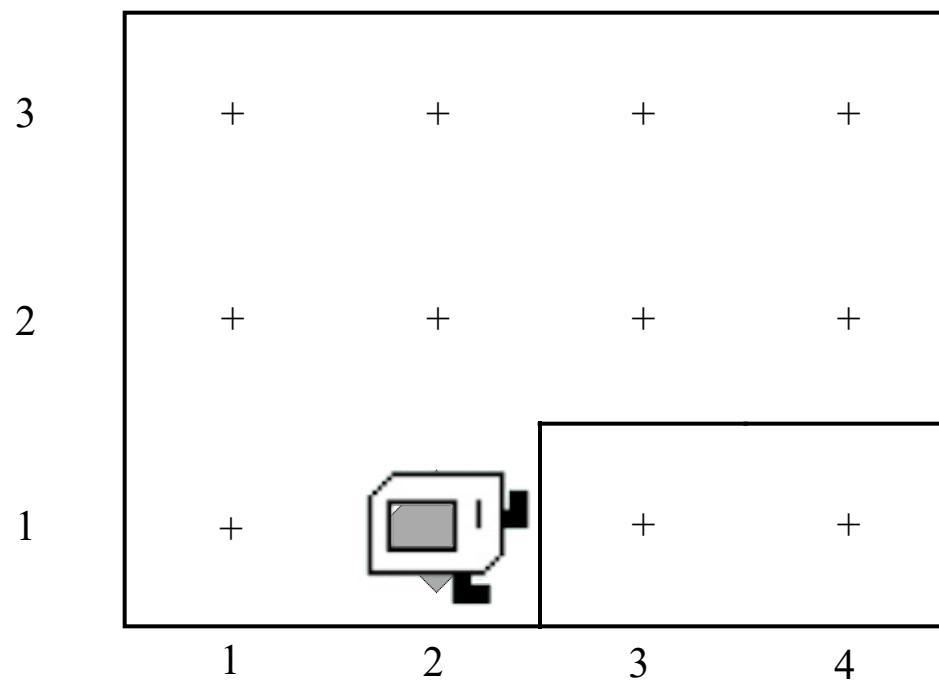
# **recogerCono();**



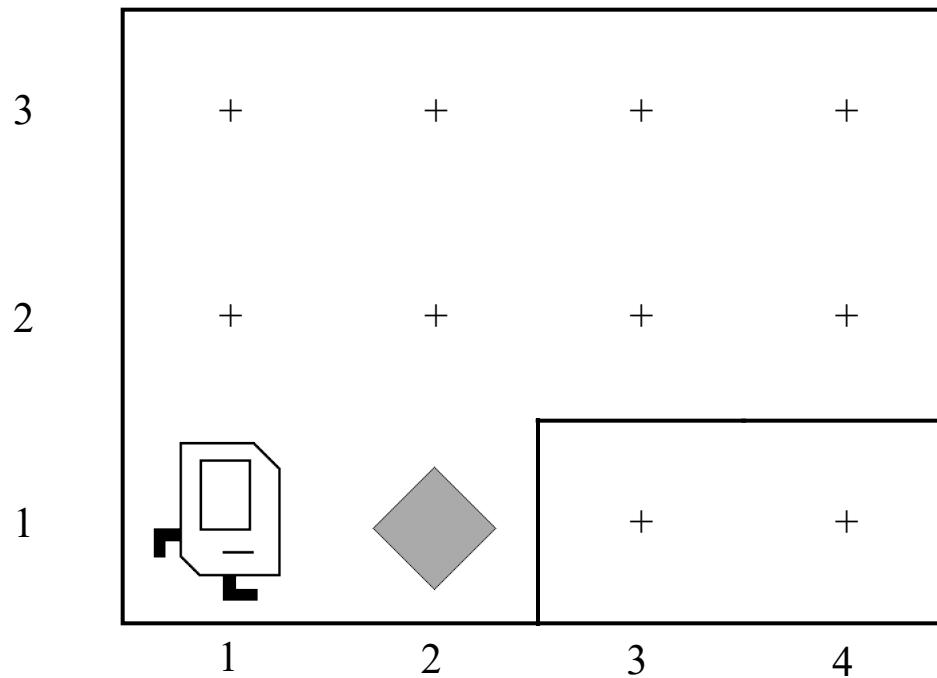
# **ponerCono( );**



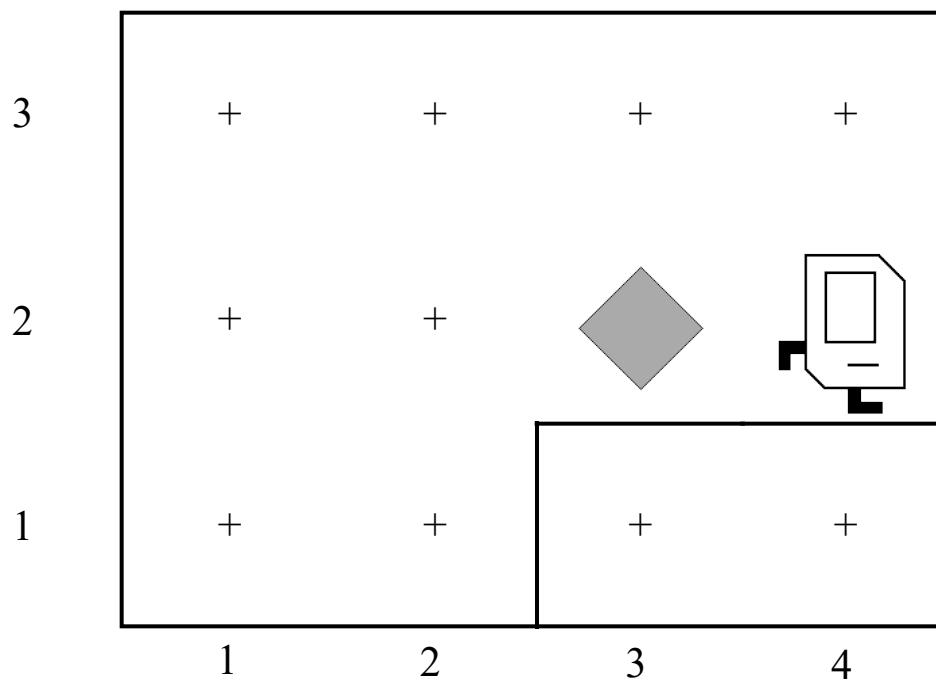
# **ponerCono( );**



# Primer reto: antes



# Primer reto: después



# Necesito un voluntario



# ¡Patos al agua!

eclipse

# Definición de un método

```
private void nombre() {  
    espacio para el cuerpo  
    (las instrucciones) del método  
}
```

Esta definición adiciona  
un nuevo comando al  
vocabulario de Karel

# Anatomía de un programa

```
import stanford.karel.*;

public class NuestroPrograma extends EsKarel {
    public void run() {
        moverse();
        recogerCono();
        moverse();
        girarIzquierda();
        moverse();
        girarDerecha();
        moverse();
        ponerCono();
        moverse();
    }

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

# Anatomía de un programa

```
import stanford.karel.*;  
  
public class NuestroPrograma extends EsKarel {  
    public void run() {  
        moverse();  
        recogerCono();  
        moverse();  
        girarIzquierda();  
        moverse();  
        girarDerecha();  
        moverse();  
        ponerCono();  
        moverse();  
    }  
  
    private void girarDerecha() {  
        girarIzquierda();  
        girarIzquierda();  
        girarIzquierda();  
    }  
}
```

Este es el **código fuente** del programa

# Anatomía de un programa

```
import stanford.karel.*;  
  
public class NuestroPrograma extends EsKarel {  
    public void run() {  
        moverse();  
        recogerCono();  
        moverse();  
        girarIzquierda();  
        moverse();  
        girarDerecha();  
        moverse();  
        ponerCono();  
        moverse();  
    }  
  
    private void girarDerecha() {  
        girarIzquierda();  
        girarIzquierda();  
        girarIzquierda();  
    }  
}
```

Esta parte del **código fuente** se llama un **método**.

# Anatomía de un programa

```
import stanford.karel.*;  
  
public class NuestroPrograma extends EsKarel {  
    public void run() {  
        moverse();  
        recogerCono();  
        moverse();  
        girarIzquierda();  
        moverse();  
        girarDerecha();  
        moverse();  
        ponerCono();  
        moverse();  
    }  
  
    private void girarDerecha() {  
        girarIzquierda();  
        girarIzquierda();  
        girarIzquierda();  
    }  
}
```

Esta línea del código define el **nombre** del método  
(en este: run)

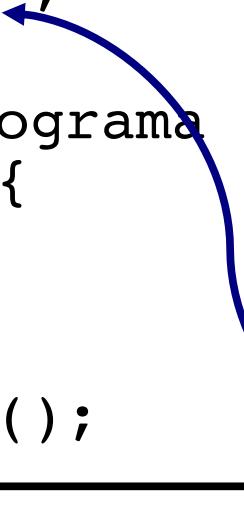
# Anatomía de un programa

```
import stanford.karel.*;  
  
public class NuestroPrograma extends EsKarel {  
    public void run() {  
        moverse();  
        recogerCono();  
        moverse();  
        girarIzquierda();  
        moverse();  
        girarDerecha();  
        moverse();  
        ponerCono();  
        moverse();  
    }  
  
    private void girarDerecha() {  
        girarIzquierda();  
        girarIzquierda();  
        girarIzquierda();  
    }  
}
```

Esta línea del código define el **nombre** del método  
(en este: girarDerecha)

# Anatomía de un programa

```
import stanford.karel.*;  
  
public class NuestroPrograma extends EsKarel {  
    public void run() {  
        moverse();  
        recogerCono();  
        moverse();  
        girarIzquierda();  
        moverse();  
        girarDerecha();  
        moverse();  
        ponerCono();  
        moverse();  
    }  
  
    private void girarDerecha() {  
        girarIzquierda();  
        girarIzquierda();  
        girarIzquierda();  
    }  
}
```



Este se llama una **instrucción import**. Le dice a Java qué es Karel.

# Anatomía de un programa

```
import stanford.karel.*;  
  
public class NuestroPrograma extends EsKarel {  
    public void run() {  
        moverse();  
        recogerCono();  
        moverse();  
        girarIzquierda();  
        moverse();  
        girarDerecha();  
        moverse();  
        ponerCono();  
        moverse();  
    }  
  
    private void girarDerecha() {  
        girarIzquierda();  
        girarIzquierda();  
        girarIzquierda();  
    }  
}
```

Este se llama un  
***bloque de código***

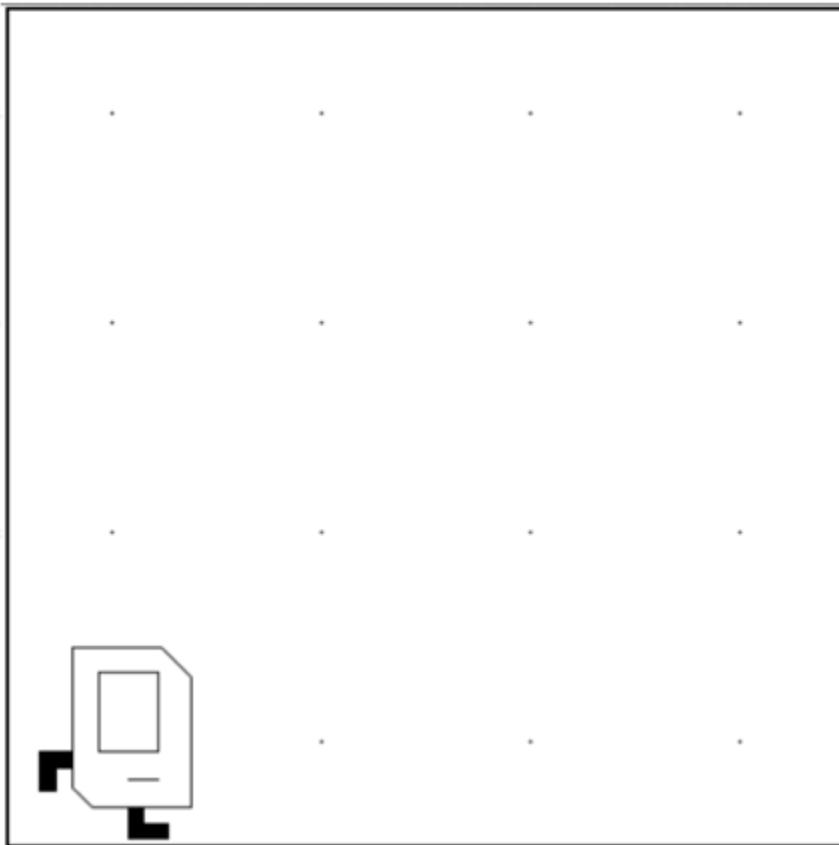
# Estilo de los programas

```
import stanford.karel.*;
```

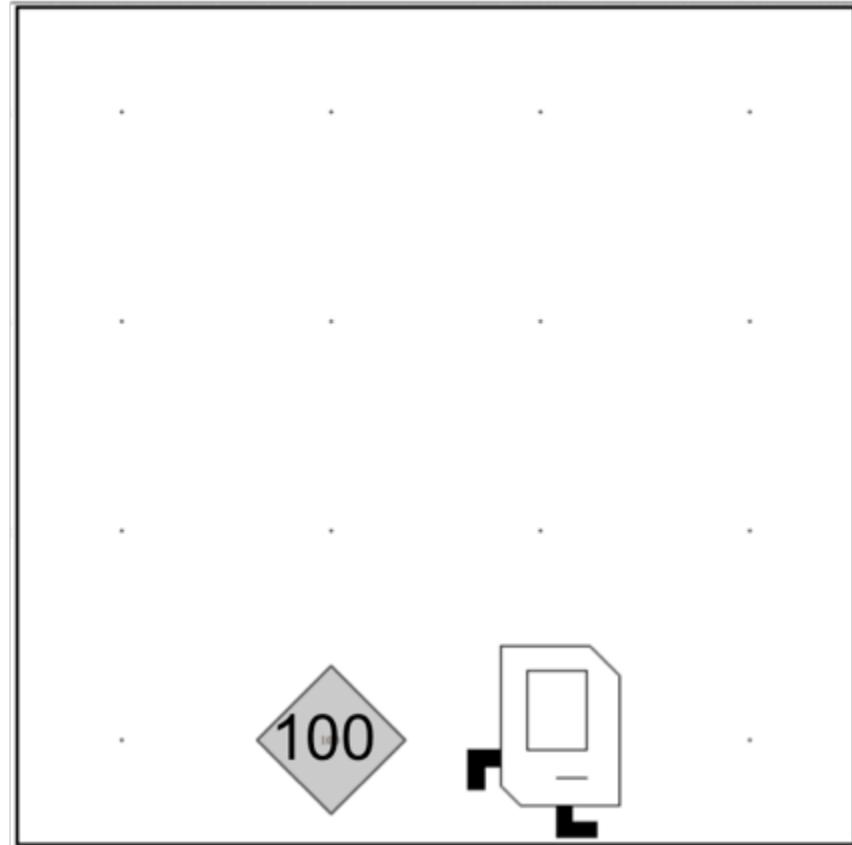
```
public class NuestroPrograma extends EsKarel {  
    public void run() {  
        moverse();  
  
        recogerCono();  
        moverse();  
        girarIzquierda(); moverse();  
        girarDerecha();  
        moverse();  
        moverse();  
        ponerCono();  
        moverse();  
    }  
  
    private void girarDerecha() {  
        girarIzquierda();  
        girarIzquierda();  
        girarIzquierda();  
    }  
}
```

# ¿Poner 100 conos?

Antes



Después



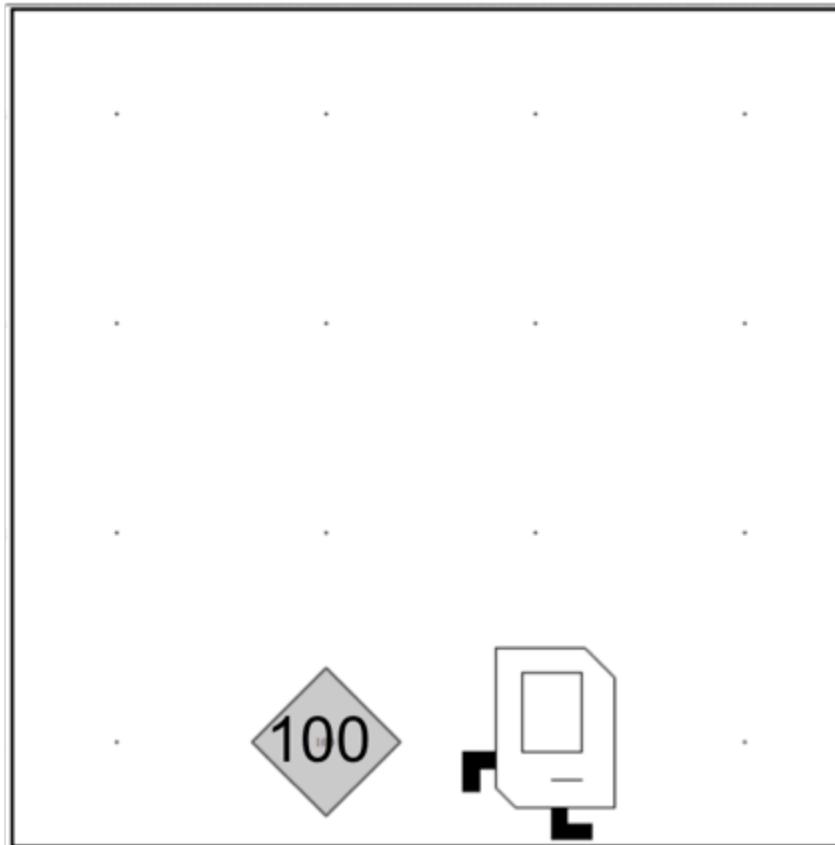
# ¿Poner 100 conos?

Podríamos decir:

```
moverse();  
ponerCono();  
ponerCono();  
ponerCono();
```

...

```
moverse();
```



Pero, es muy repetitivo. Además, es difícil generalizar el programa.

# Ciclos for

En cambio, usa un ciclo **for**:

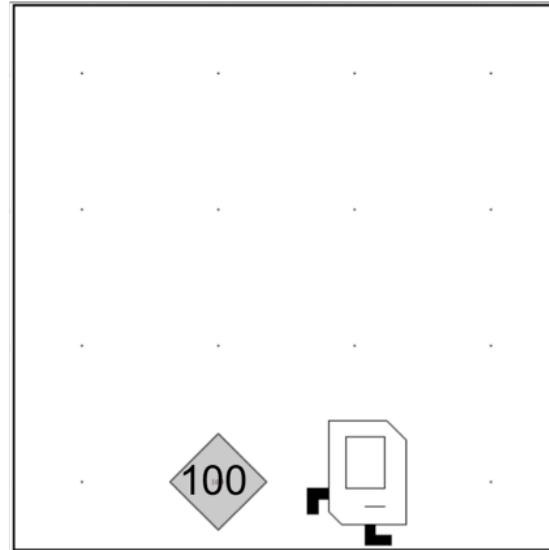
```
for (int i = 0; i < contar; i++) {  
    instrucción;  
    instrucción;  
    ...  
}
```

Repite las instrucciones en el cuerpo del ciclo **contar** veces.

# Poner 100 conos

Ahora podemos decir:

```
moverse();  
for (int i = 0; i < 100; i++) {  
    ponerCono();  
}  
moverse();
```



Es menos repetitivo y más fácil cambiar (por ejemplo:  
poner 25 conos en vez de 100)

# Ciclos for

Unos ejemplos del uso de ciclos **for**:

```
// Karel gira a la derecha
for (int i = 0; i < 3; i++) {
    girarIzquierda();
}

// Karel se mueve en una escuadra
for (int i = 0; i < 4; i++) {
    moverse();
    girarIzquierda();
}
```

# ¡Tu primer programa!

