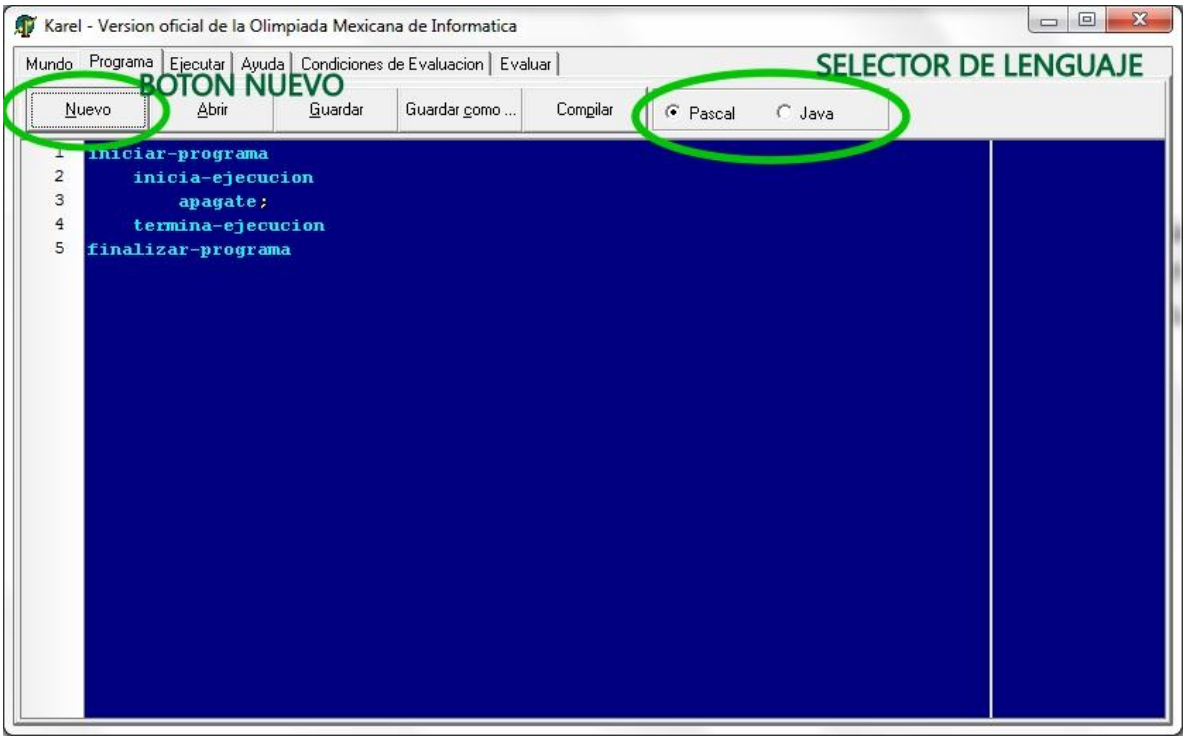
Como programar en Karel

Karel tiene un editor en el cual podemos programar, para acceder a él hay que dar click en la pestaña **Programa**. Despues de esto, debemos seleccionar el lenguaje a utilizar (Pascal o Java) y por ultimo dar click en **Nuevo**. Esto hará que el editor nos coloque la estructura básica de un programa de Karel.



Dependiendo del lenguaje que queramos utilizar, el editor nos puede mostrar lo siguiente:

PASCAL		**JAVA**	
1	iniciar-programa	1	class program {
2	inicia-ejecucion	2	
3	apagate;	3	program() {
4	termina-ejecucion	4	;
5	finalizar-programa	5	turnoff();
		6	}
		7	
		8	}

A pesar de lo diferentes que parecen a primera vista, ambos lenguajes son muy similares y solo debes aprender a usar uno de ellos, elige aquel que te parezca más fácil de utilizar. Veamos las partes que componen a estos códigos.

Delimitación del programa

PASCAL	JAVA

1 iniciar-programa
2 inicia-ejecucion
3 apagate;
4 termina-ejecucion
5 finalizar-programa
	1 class program {
	2
	3 program() {
	4 ;
	5 turnoff();
	6 }
	7
	8 }

Las partes marcadas en rojo son las que definen donde comienza y donde termina el código del programa.

Procedimiento principal

PASCAL	JAVA

1 iniciar-programa
2 inicia-ejecucion
3 apagate;
4 termina-ejecucion
5 finalizar-programa
	1 class program {
	2
	3 program() {
	4 ;
	5 turnoff();
	6 }
	7
	8 }

Las partes marcadas en azul delimitan el procedimiento principal, que es el primera en ejecutarse. Entre estos marcadores debemos escribir lo que queremos que haga Karel. Sin embargo para poder ordenar a Karel, primero hay que conocer las instrucciones que entiende Karel.

Instrucciones de Karel

Como Karel es un robot pequeño, solo entiende 5 instrucciones.

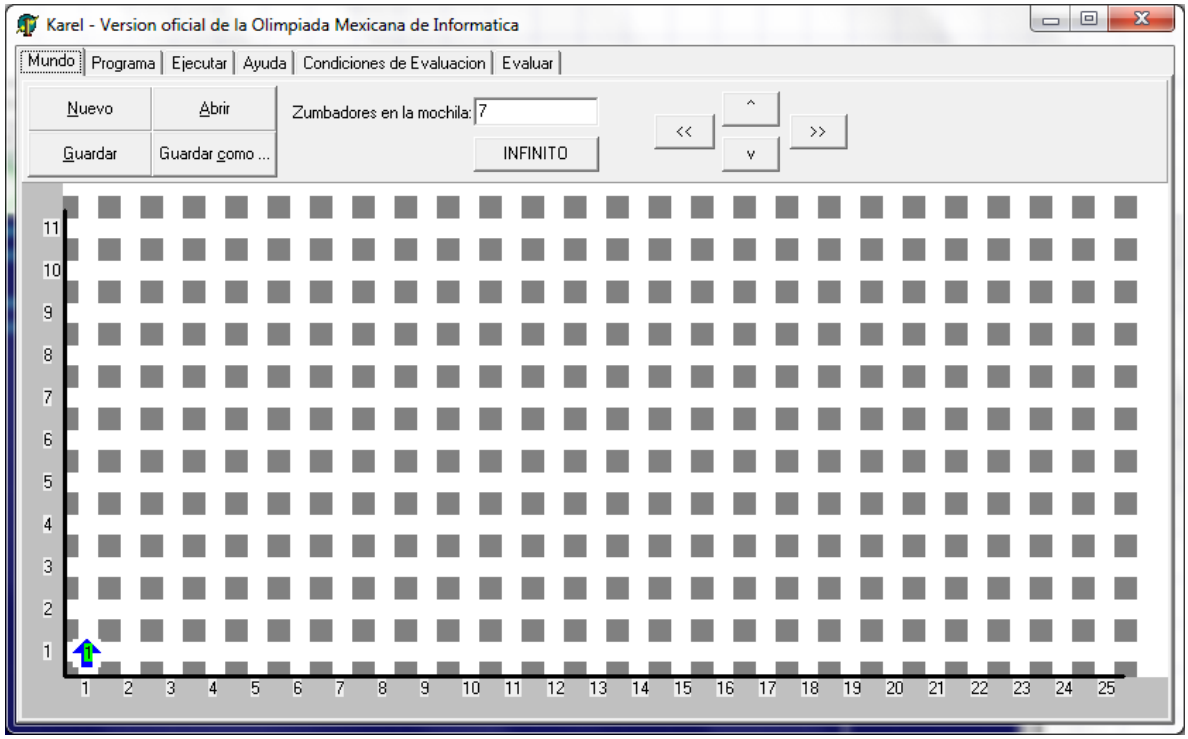
PASCAL	JAVA
avanza	move
gira-izquierda	turnleft
coge-zumbador	pickbeeper
deja-zumbador	putbeeper
apagate	turnoff

- **avanza / move** Esta instrucción le indica a Karel que debe avanzar a la posición que tenga enfrente. Sin embargo hay que recordar que en el mundo de Karel hay paredes, por lo que si le decimos a Karel que avance y hay una pared, entonces nuestro programa terminará con un error puesto que Karel intenta atravesar una pared.

- **gira-izquierda / turnleft** Esta instrucción le indica a Karel que debe girar 90° hacia la izquierda, esto es, si Karel está orientado al norte, girará hacia el oeste; si está orientado al oeste, girará hacia al sur; si está orientado al sur, girará hacia el este, y si está orientado al este, girará hacia el norte.
- **coge-zumbador / pickbeeper** Si recordamos un poco del mundo de Karel, en el mundo de Karel hay zumbadores (beepers) los cuales Karel puede recoger. Esta instrucción sirve para indicarle a Karel que recoja un zumbador que se encuentre en su misma posición y lo guardará en su mochila, sin embargo si no hay un zumbador en la misma posición que Karel, entonces nuestro programa terminará con un error.
- **deja-zumbador / putbeeper** También Karel puede colocar zumbadores en el mundo, para indicarle a Karel que deje un zumbador en su posición, se utiliza esta instrucción. Sin embargo si Karel no tiene zumbadores en su mochila, entonces nuestro programa terminará con un error.
- **apagate / turnoff** Esta instrucción nos sirve para indicarle a Karel que termine. Al momento de ejecutarla nuestro programa tendrá una terminación normal y nuestros programas siempre deben terminar con esta instrucción.

Ejemplo

Hagamos un programa que tome un zumbador, haga que Karel avance una posición, gire tres veces a la izquierda, avance de nuevo y deje el zumbador. Para esto necesitamos un mundo que nos permita ejecutar el programa sin problemas. Hagamos un mundo como el siguiente.



Como Karel es muy ordenado(a) debemos indicarle las cosas en orden para que pueda ejecutarlas así que las instrucciones en orden serían:

1. **coge-zumbador / pickbeeper**
2. **avanza / move**
3. **gira-izquierda / turnleft**
4. **gira-izquierda / turnleft**
5. **gira-izquierda / turnleft**
6. **avanza / move**
7. **deja-zumbador / putbeeper**
8. **apagate / turnoff**

Ahora veamos cómo queda el código.

PASCAL

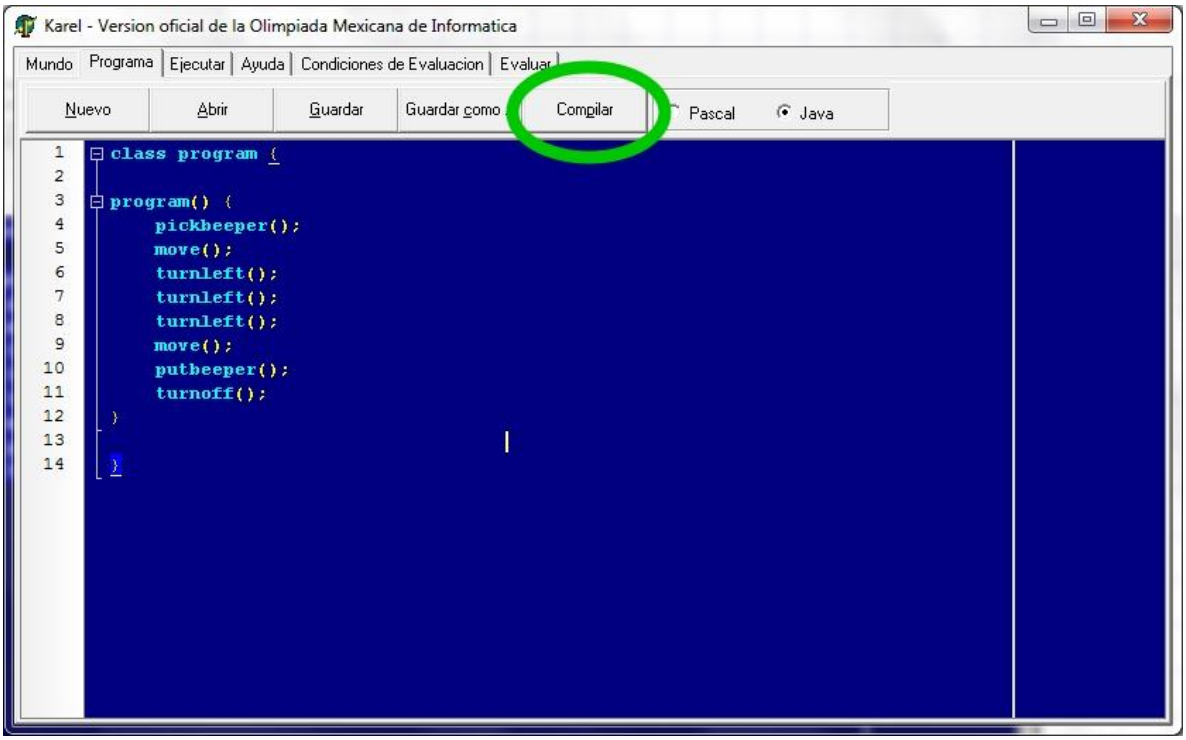
```
1  iniciar-programa
2      inicia-ejecucion
3          coge-zumbador;
4          avanza;
5          gira-izquierda;
6          gira-izquierda;
7          gira-izquierda;
8          avanza;
9          deja-zumbador;
10         apagate;
11     termina-ejecucion
12 finalizar-programa
```

JAVA

```
1  class program {
2
3  program() {
4      pickbeeper();
5      move();
6      turnleft();
7      turnleft();
8      turnleft();
9      move();
10     putbeeper();
11     turnoff();
12 }
13
14 }
```

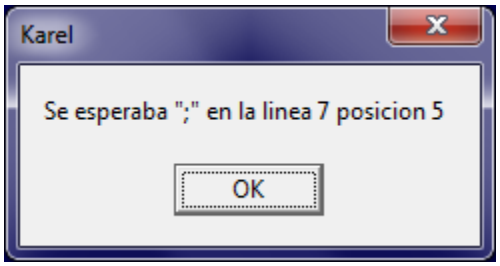
Hay que notar que si utilizamos java, las instrucciones llevan paréntesis, y que tanto en pascal como en java es necesario utilizar punto y coma (;) después de cada instrucción.

Continuando con el programa, una vez que está codificado, debemos compilarlo antes de que Karel pueda ejecutarlo, para hacer esto, debemos dar click en el botón **Compilar** de la pestaña **Programa**.



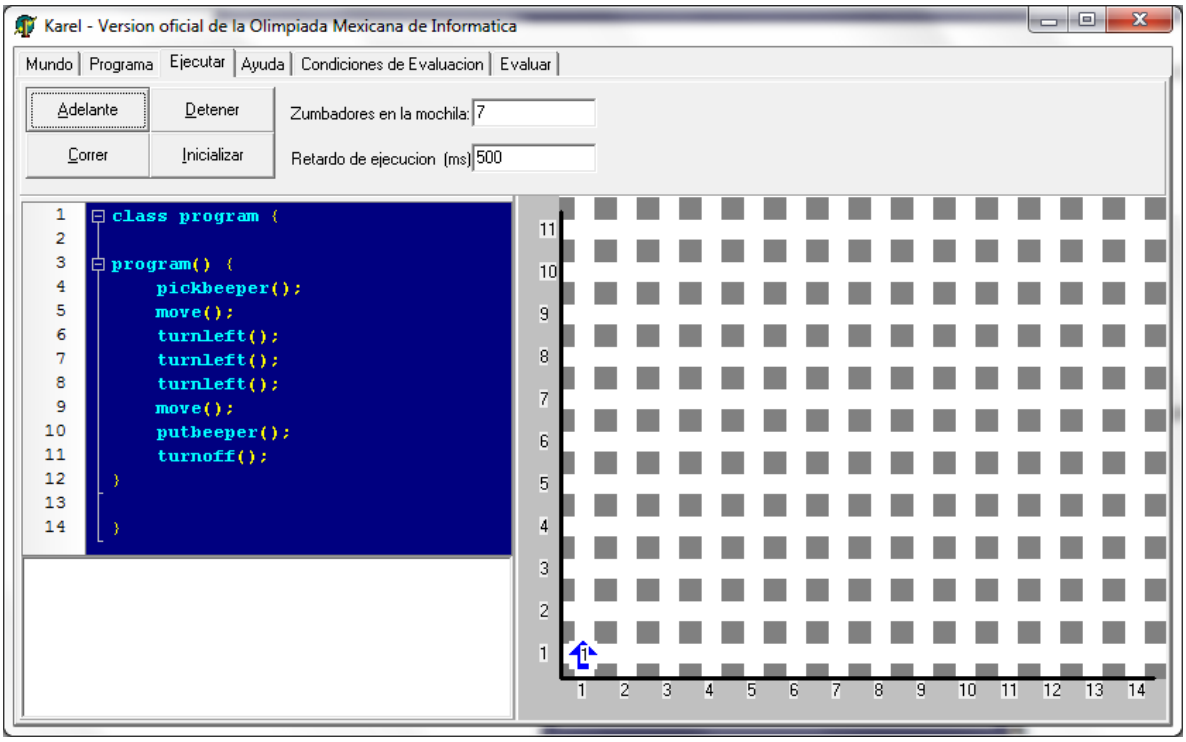
En caso de que nuestro programa esté bien escrito nos aparecerá la siguiente ventana.

En caso contrario nos parecerá una ventana indicando el error que detectó el compilador. Por ejemplo, si nos faltó un “;”, el compilador mostrará lo siguiente.



Nos indicará la posición y la línea donde encontró el error.

Una vez compilado el programa hay que probarlo para esto, existe la pestaña **Ejecutar** en la cual podemos probar el programa.

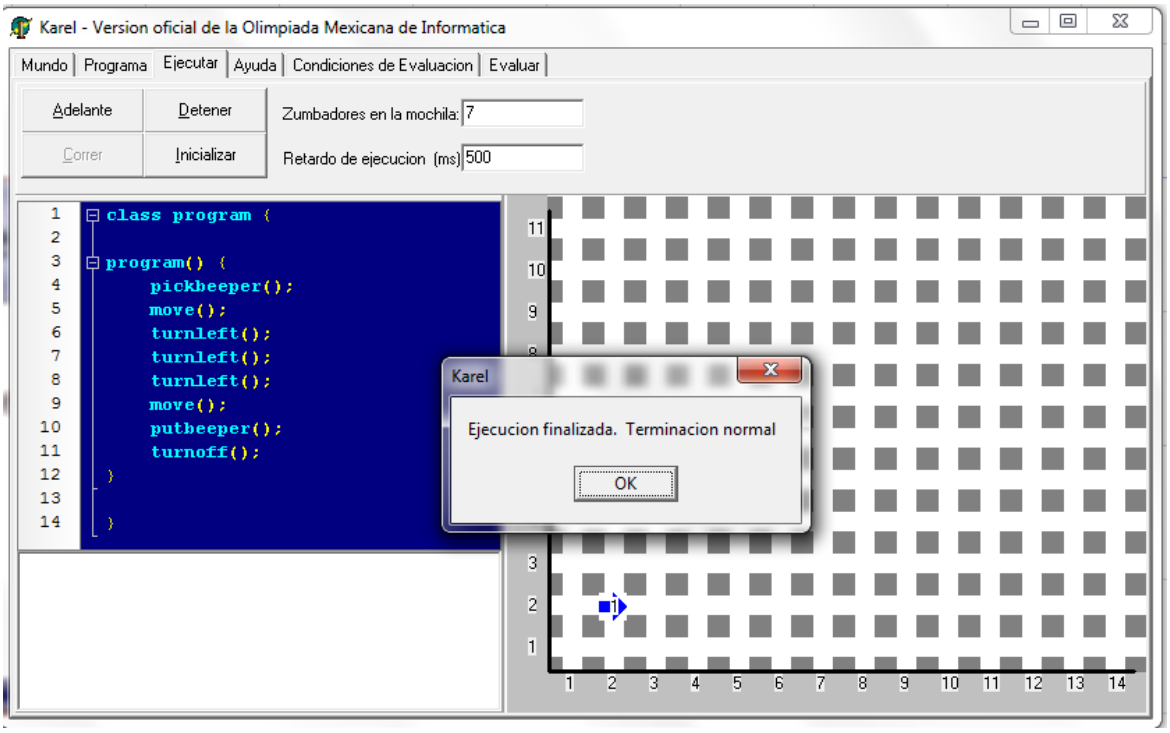


Esta pestaña tiene 4 botones:

- **Adelante.** Hace que se ejecute una instrucción.
- **Correr.** Hace que se ejecute de manera continua el programa.
- **Detener.** Detiene el programa en el punto que se encuentra.
- **Inicializar.** Regresa el programa al inicio, con el mundo inicial.

La parte que dice “Retardo de ejecución (ms)” nos indica cada cuántos milisegundos se ejecutará cada instrucción, podemos modificar este valor, siendo 0 la máxima velocidad de ejecución.

Si le damos correr a nuestro ejemplo, veremos como se ejecuta y al final terminaremos con una ventana como la siguiente.



Corre el programa paso por paso con el botón **Avanza** para que veas como Karel ejecuta las instrucciones.