Manual del docente

Última revisión: noviembre 2015

Tabla de contenidos

INTRODUCCIÓN

ABC de RITA

RITA (Robot Inventor to Teach Algorithms) es una herramienta gratuita y de código abierto que le permitirá acercar a sus alumnos al mundo de la programación mediante el desarrollo de actividades lúdicas. Con RITA sus alumnos serán capaces de "programar" la estrategia de un robot, pero sin el uso directo de un lenguaje de programación en particular, sino con el uso de bloques de encastre, al estilo Lego. De esta manera le indican al robot que hacer y cómo reaccionar ante la ocurrencia de eventos de su entorno.

A medida que se avanza en el aprendizaje de RITA, se introducirán conceptos propios de la programación, como son la *secuencia*, los *condicionales* y *decisiones* y las *iteraciones* de una manera natural y otros conceptos relacionados a la programación orientada a objetos..

Las estrategias de los robots creados por sus alumnos podrán ser puesto a prueba frente al resto de los robots. Durante nuestra experiencia en las aulas, notamos que éste último punto funciona como elemento motivador en los alumnos, quienes tratarán de construir un robot que pueda ganarle a los robots de sus compañeros de aula.

Instalación

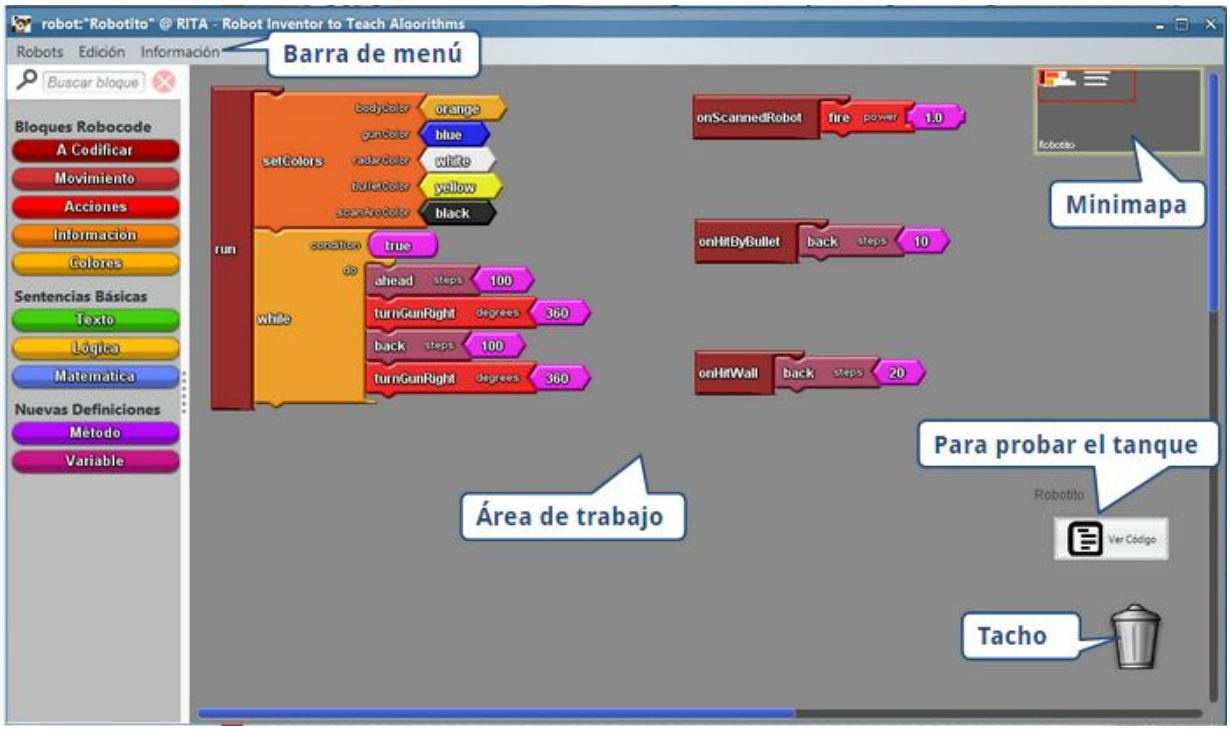
El instalador de RITA le guiará paso a paso en el proceso de instalación. Sólo en caso que no quiera instalar RITA en el directorio propuesto, será necesario que indique un directorio de instalación.

- En caso que su equipo use el sistema operativo Windows se recomienda usar un directorio sin restricciones de acceso, se sugiere como directorio de instalación C:\Users\UserXYZ\
- En caso que la instalación se realice sobre Windows XP setee la variable de entorno LOCALAPPDATA con el valor c:\windows\temp
- En caso que su equipo use el sistema operativo Huayra u otro sistema operativo basado en linux, no hay restricciones en el proceso de instalación.

Introducción al entorno

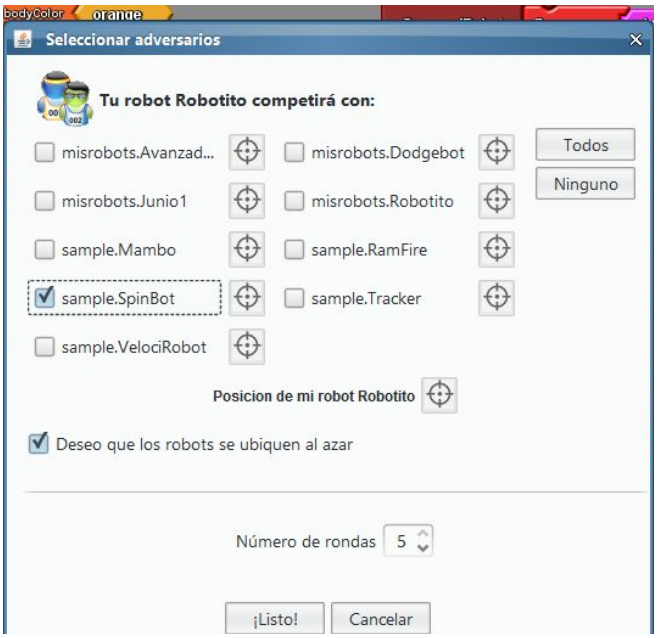
El entorno de trabajo de RITA es muy sencillo. La primera pantalla le permitirá elegir un nombre para su robot y el tipo de Robot a crear. RITA permite crear 2 tipos de robots: Junior o Avanzado, la explicación de este documento se basa en un robot Junior.

Una vez que hace click en “Crear Robot”, pasará a la pantalla principal donde organizará la estrategia de su robot, el área de trabajo.
El área central, con fondo gris, contiene los bloques que conforman la estrategia de un robot. Como mencionamos antes, RITA permite crear 2 tipos de robots: Junior o Avanzado, para cualquiera de los dos casos, cuando ud. crea un robot, RITA le propone un conjunto de bloques básico ya organizado que ud. puede o no usar. A modo de prueba, abra la aplicación y verá los componentes indicados en el siguiente gráfico.



Una simple prueba
Una vez dentro de RITA, y con el robot básico propuesto, vamos a ponerlo a competir. Haga click sobre el botón “Ver Código” (en el área inferior derecha de la pantalla). Al hacer click se mostrará la siguiente ventana:

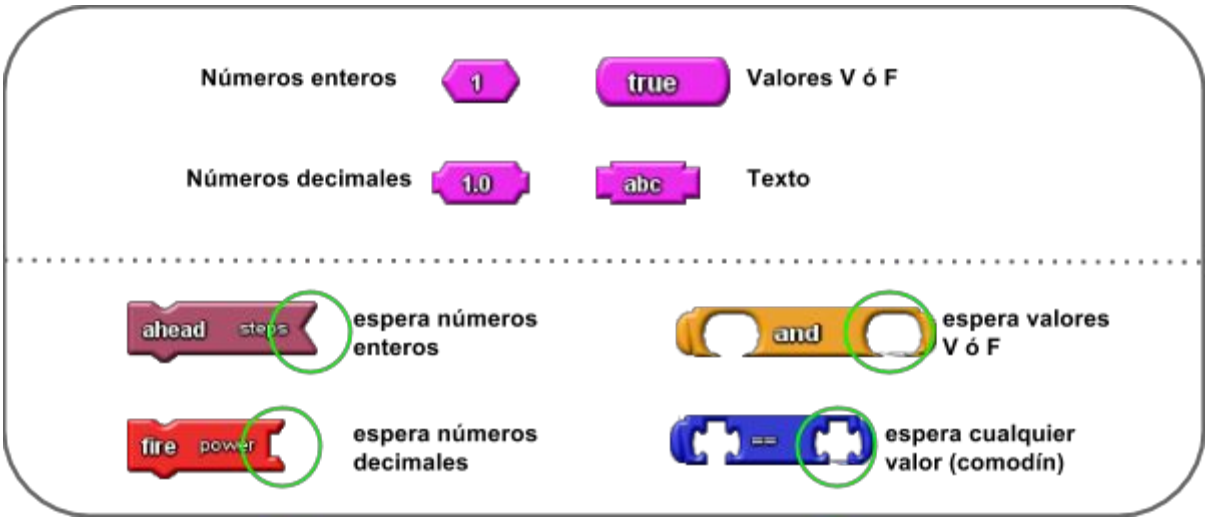
Seleccione cualquiera de los robots listados y luego haga click en “Listo”. A continuación se abrirá la ventana que muestra a su robot midiéndose en el campo de batalla con el robot seleccionado.



Si desea cerrar la ventana de la batalla, haga click en el botón “Stop” que se encuentra en la parte inferior de la misma.

Formas de los bloques

RITA provee distintos bloques que pueden ser encastrados entre sí. Preste atención a la forma de los mismos, dado que sólo pueden ser encastrados si los bloques “se complementan” ó si se va a encastrar en un bloque tipo ”comodín”, es decir, que acepta más de una forma.
A continuación un listado las formas de bloques que encontrará en RITA.



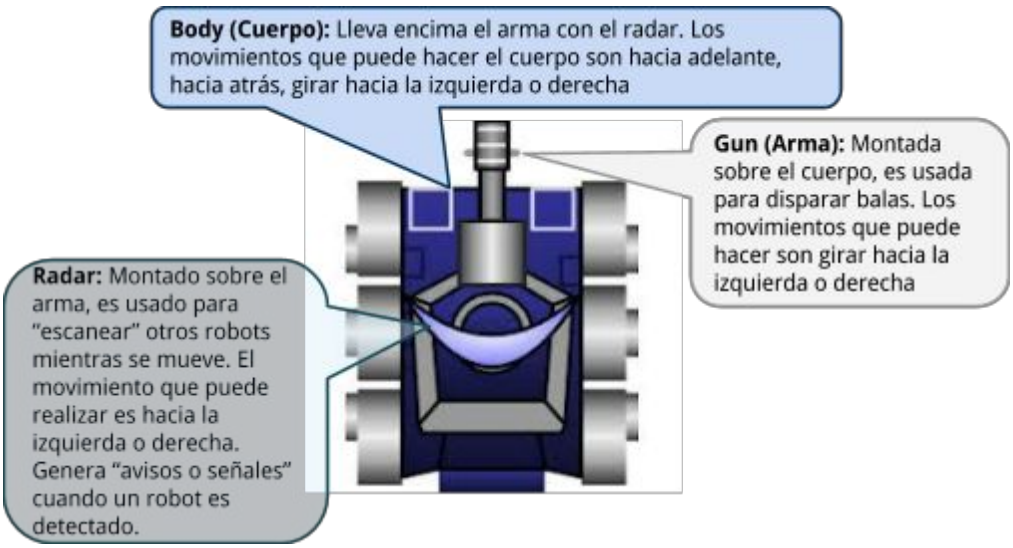
Instrucciones que nuestro robot ejecutará

Durante toda la primer parte de este documento, por una cuestión de facilitar el entendimiento del alumno, se trabajará agregando bloques dentro del bloque **run**. Este bloque **run** aparece por defecto cuando creamos un nuevo robot. El bloque run contiene los bloques que representan acciones a realizar por defecto, es decir, cuando el robot no está ocupado reaccionando a otras cuestiones de su entorno (como tratando de defenderse al ser atacado), que veremos al final del documento.

CONCEPTOS BÁSICOS

El Tanque

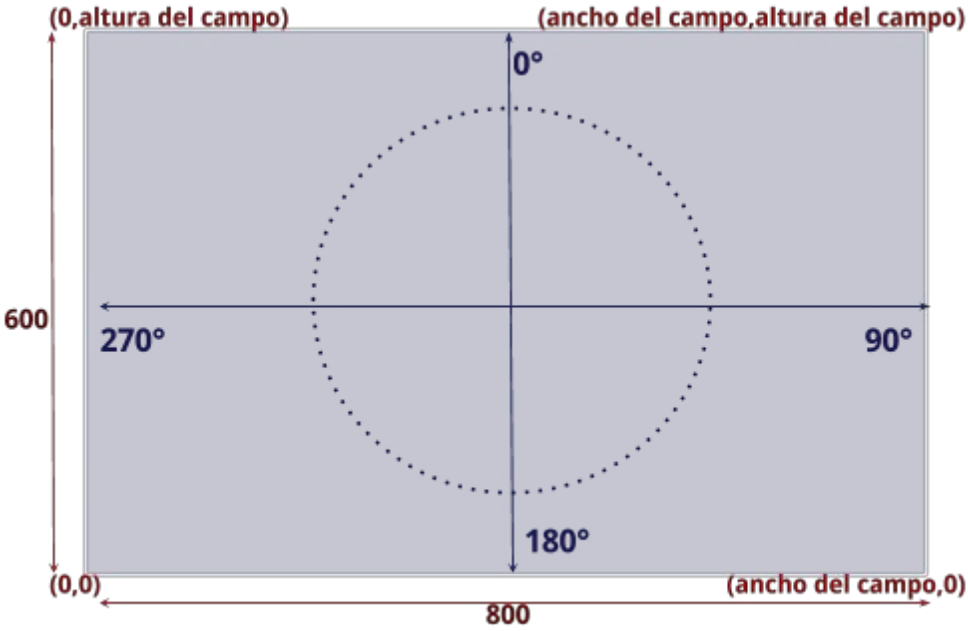
Sus alumnos escribirán la estrategia de un robot que posee la forma de un tanque. Observe en el siguiente gráfico la forma del mismo.



Cabe destacar que como el radar se encuentra posicionado sobre el arma, cuando el arma gira, automáticamente se mueve el radar del robot, revisando el área en busca de otros tanques. Por ende, si hacemos girar el arma 360º, estamos revisando toda el área con el radar.

El campo de batalla

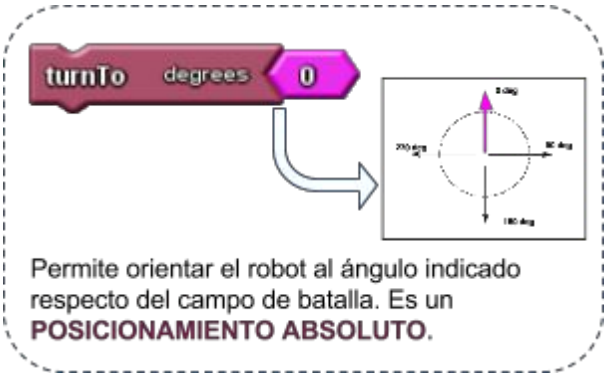
Las especificaciones del campo de batalla son las que se muestran en el gráfico. Su robot puede orientarse y desplazarse sobre el mismo. A continuación veremos qué bloques debemos usar para lograr el desplazamiento deseado.



Orientación

RITA provee el bloque **turnTo** que permite orientar al robot en el campo de batalla usando un posicionamiento absoluto:

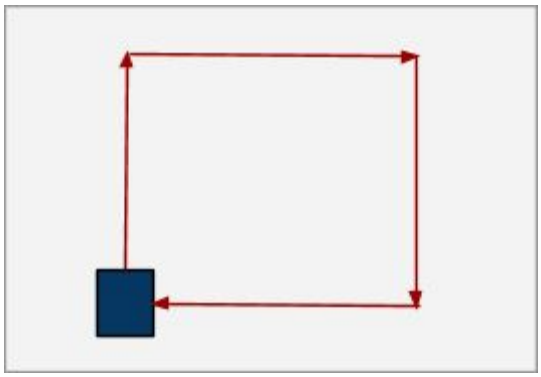
- al NORTE: 0º
- al OESTE: 90º
- al SUR: 180º
- al ESTE: 270º



Desplazamiento

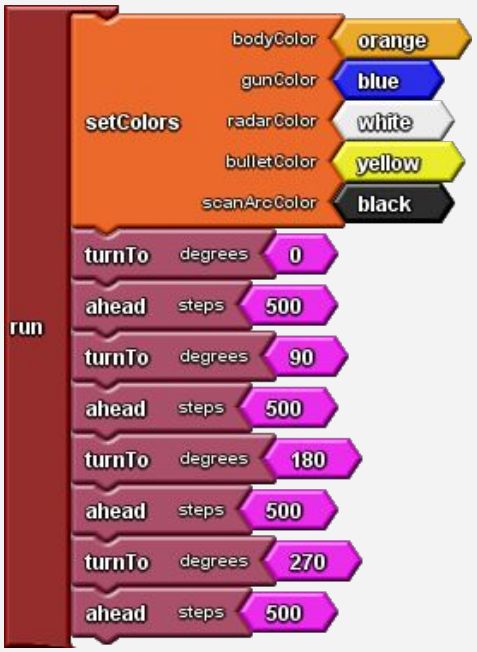
Los bloques **ahead** y **back** nos permitirán desplazarnos hacia adelante o hacia atrás en el campo de batalla. Tenga en cuenta que con un desplazamiento excesivo, existe la posibilidad de chocar contra alguno de los muros del campo de batalla.

Ejercicio 1: Usando los bloques **turnTo** y **ahead** realizar un robot que dibuje un cuadrado con su desplazamiento y giros.



1. Crear un robot llamado Cuadrado
2. Orientarse hacia el NORTE (0º)
3. Avanzar hacia adelante 500 pasos
4. Orientarse hacia el OESTE (90º)
5. Avanzar hacia adelante 500 pasos
6. Orientarse hacia el SUR (180º)
7. Avanzar hacia adelante 500 pasos
8. Orientarse hacia el ESTE (270º)
9. Avanzar hacia adelante 500 pasos

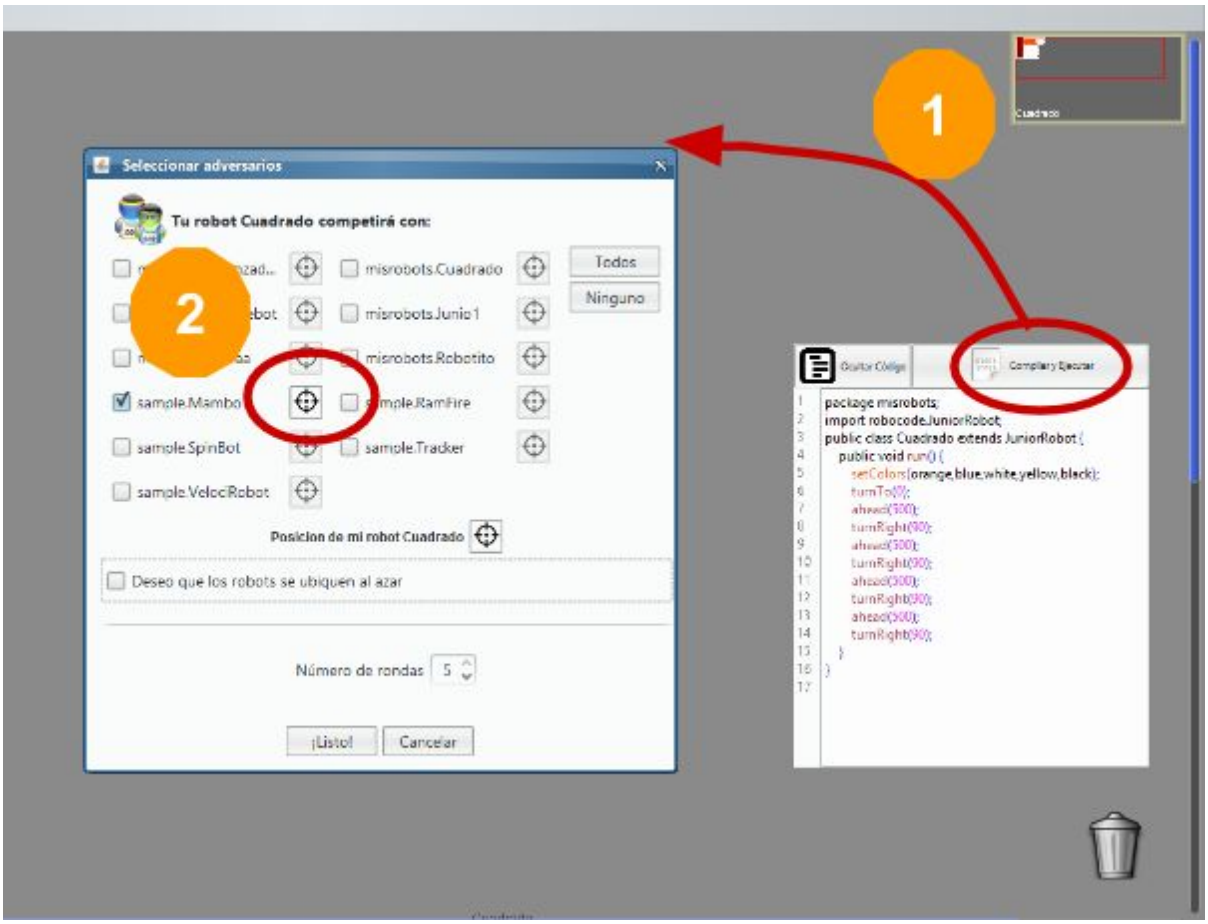
Finalmente, el código que permita realizar la siguiente figura se muestra en bloques y su traducción a código Java.

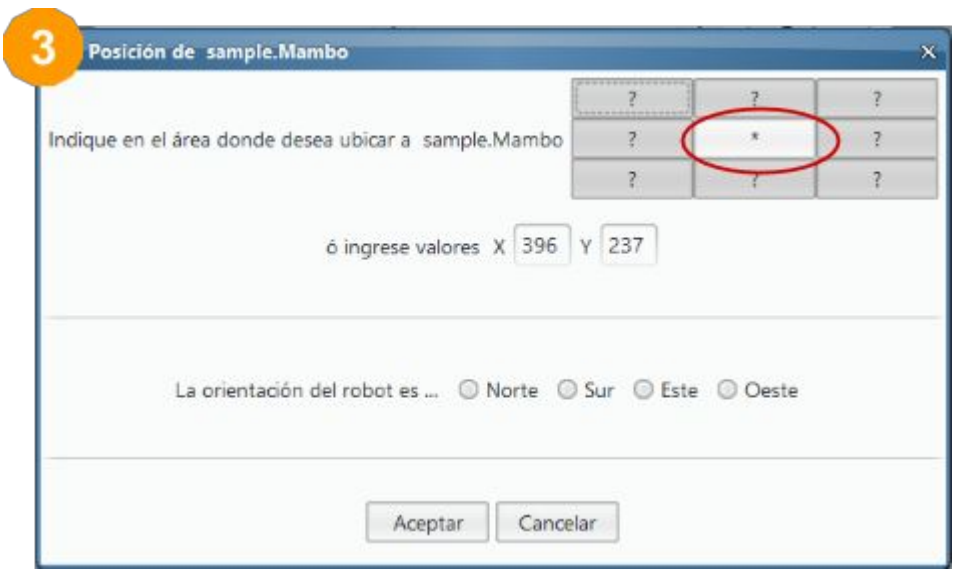



```
1  package misrobots;
2  import robocode.JuniorRobot;
3  public class Cuadrado extends JuniorRobot {
4     public void run() {
5        setColors(orange,blue,white,yellow,black);
6        turnTo(0);
7        ahead(500);
8        turnTo(90);
9        ahead(500);
10       turnTo(180);
11       ahead(500);
12       turnTo(270);
13       ahead(500);
14    }
15 }
16
```

código de bloques	código Java

Ponga a prueba su robot, llevándolo al campo de batalla. En la ventana que muestra el código Java del robot, haga click en “Compilar y ejecutar” (1). Aparecerá una ventana donde puede seleccionar con qué otro robot competir (2). Elija a Mambo también para que esté en el centro del campo (3).





PROGRAMACIÓN ESTRUCTURADA VS. PROGRAMACIÓN ORIENTADA A OBJETOS

Los lenguajes de programación como Pascal ó C están basados en el paradigma de programación estructurada, donde se tienen a los datos y las instrucciones sin ninguna integración particular estipulada por el paradigma. Por otro lado, el lenguaje Java (lenguaje de programación en el que se traduce la estrategia del robot) es un lenguaje basado en el paradigma de programación orientada a objetos (POO), donde las distintas piezas que conforman un programa son "objetos", y cada objeto posee atributos (datos) y comportamiento (instrucciones). Siguiendo nuestro ejemplo, cuando el robot Cuadrado esté ejecutándose en una batalla, será un objeto de nuestro programa. Y más aún, el arma y el radar son objetos, cada uno con sus características propias. El paradigma de POO busca representar los elementos del mundo real como objetos que participan en la solución de un problema.

La Herencia es uno de los principios de la programación orientada a objetos, y se refiere a un tipo de relación entre objetos, mediante la cual, un objeto "hereda" atributos y comportamiento de otra clase que funciona como padre. Es decir, la clase padre le da un marco general de atributos y comportamiento que luego el objeto hijo puede completar con más atributos ó más comportamiento, ó modificar parte de lo que está "heredando" de su padre.

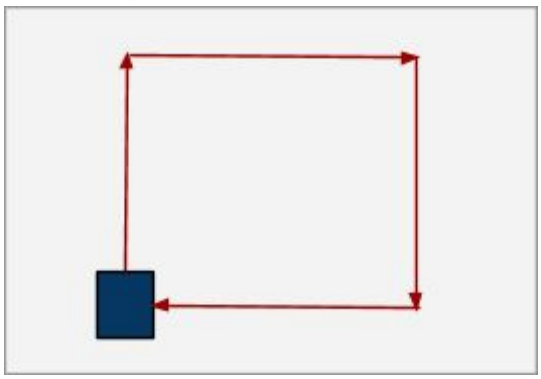
En nuestro ejemplo, el robot Cuadrado no tiene atributos propios sino que hereda atributos de JuniorRobot (vea la palabra clave *extends* en Java), pero con seguridad, heredará de su padre atributos básicos como la cantidad de energía ó la posición actual. Acerca del comportamiento, en el código del gráfico aparece un método "run", mientras que el resto del comportamiento lo *hereda* de JuniorRobot, por ejemplo, cuando se hace quiere realizar un *turnTo* (girar), Cuadrado no tiene la definición de qué es "girar", sino que está heredando este comportamiento de JuniorRobot que sí sabe cómo girar en un ángulo determinado.

OTROS MOVIMIENTOS DEL ROBOT

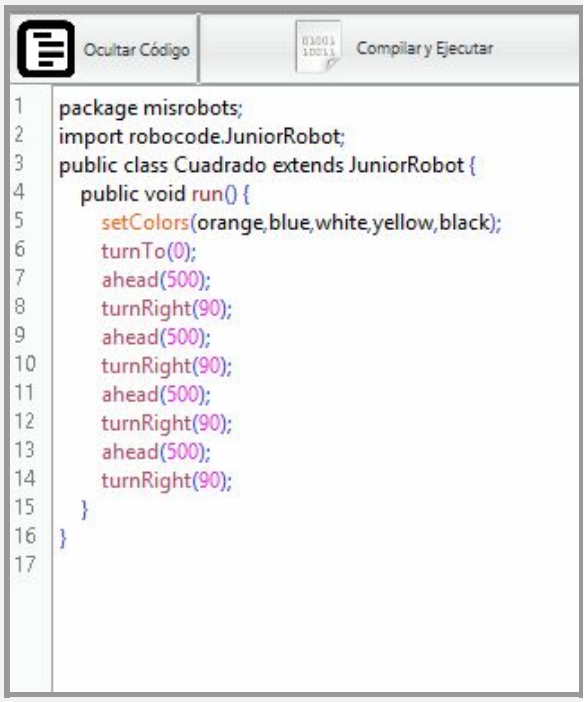
Más bloques de orientación

RITA provee otros bloques como el **turnLeft** ó el **turnRight**, que permiten girar una determinada cantidad de grados a izquierda o a derecha a partir de la orientación actual de nuestro robot, es decir, ahora no estamos especificando un posicionamiento absoluto como vimos anteriormente, sino que el giro es relativo a la orientación actual de nuestro robot.

Ejercicio 2. Cree una robot llamado Cuadrado usando los bloques **turnRight** y **ahead**. Sólo podrá usar el bloque **turnTo** para orientar el robot hacia el norte inicialmente



1. Crear un robot llamado Cuadrado
2. Primero debe orientar al robot mirando al norte (posicionamiento absoluto)
3. Avanzar hacia adelante 500 pasos
4. Girar a la derecha 90º
5. Repetir 4 veces los puntos 3 y 4
6. Finalmente, el código que permita realizar la siguiente figura se muestra en bloques y su traducción a código Java



código en bloques	código Java

Finalmente ponga a prueba su robot en el campo de batalla como lo hizo en el primer ejercicio.

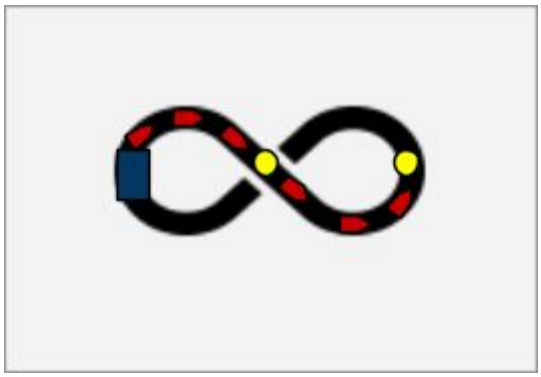
Orientación y desplazamiento simultáneos

RITA provee otros bloques que le permiten desplazarse hacia adelante o atrás a la vez que realiza un giro a izquierda o derecha simultáneamente. Este tipo de desplazamientos permitirán que el robot se mueva con forma de semicírculos o incluso círculos.

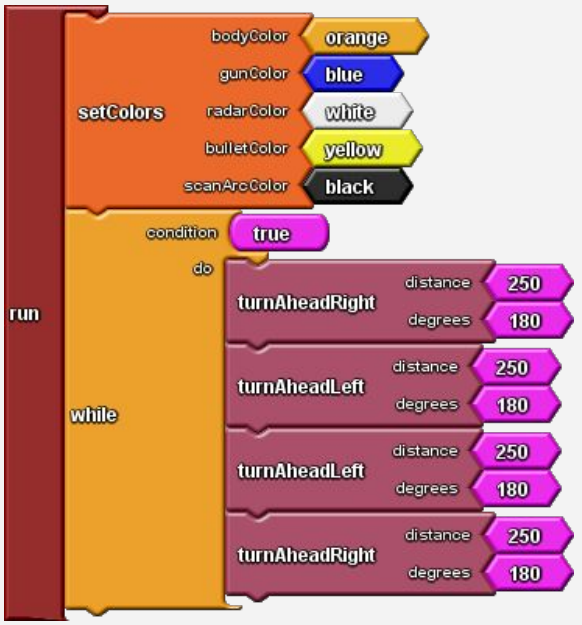
Ejercicio 3. Cree un robot llamado “Infinito” que dibuje la siguiente figura:

Coloque a Mambo en un lugar donde no lo incomode.

Si ubica a “Infinito” en un extremo de la pantalla, deberá “avanzar y girar” simultáneamente. Note que va de un extremo al otro dibujando una forma curva.



1. Primero debe orientar al robot mirando al norte
2. Avanzar y girar simultáneamente a la derecha (avanzar 250 pasos y girar 180º)
3. Avanzar y girar simultáneamente a la izquierda (avanzar 250 pasos y girar 180 º)
4. Avanzar y girar simultáneamente a la izquierda (avanzar 250 pasos y girar 180º)
5. Avanzar y girar simultáneamente a la derecha (avanzar 250 pasos y girar 180º)



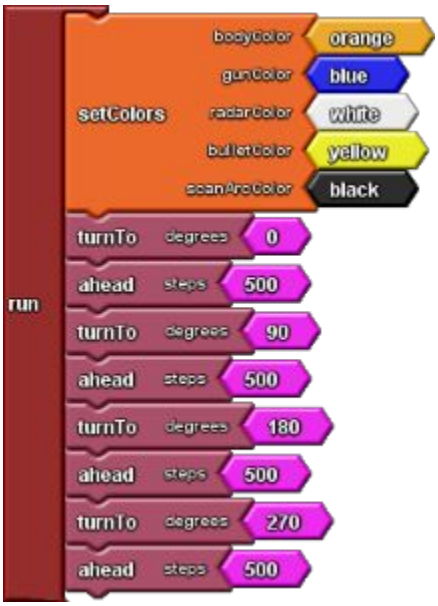

```
1  package misrobots;
2  import robocode.JuniorRobot;
3  public class Ocho extends JuniorRobot {
4     public void run() {
5        setColors(orange,blue,white,yellow,black);
6        while (true) {
7           turnAheadRight(250,180);
8           turnAheadLeft(250,180);
9           turnAheadLeft(250,180);
10          turnAheadRight(250,180);
11       }
12    }
13 }
14
```

código en bloques	código Java

Esta no es la única solución al ejercicio, otra forma de realizarlo sería realizando 2 círculos con giros de 360 grados. Aunque en este caso, el robot debería estar ubicado en la parte central del dibujo esperado.

EL CONCEPTO DE SECUENCIA

Recordemos la solución en bloques del ejercicio 1:



Nosotros indicamos paso a paso cuales son las acciones que nuestro robot Cuadrado debe realizar para dibujar un cuadrado en el campo de batalla. Estas acciones tienen un orden determinado, si cambiamos el orden obtendremos un resultado diferente. Aquí aparece el concepto de secuencia aplicado a nuestra solución.

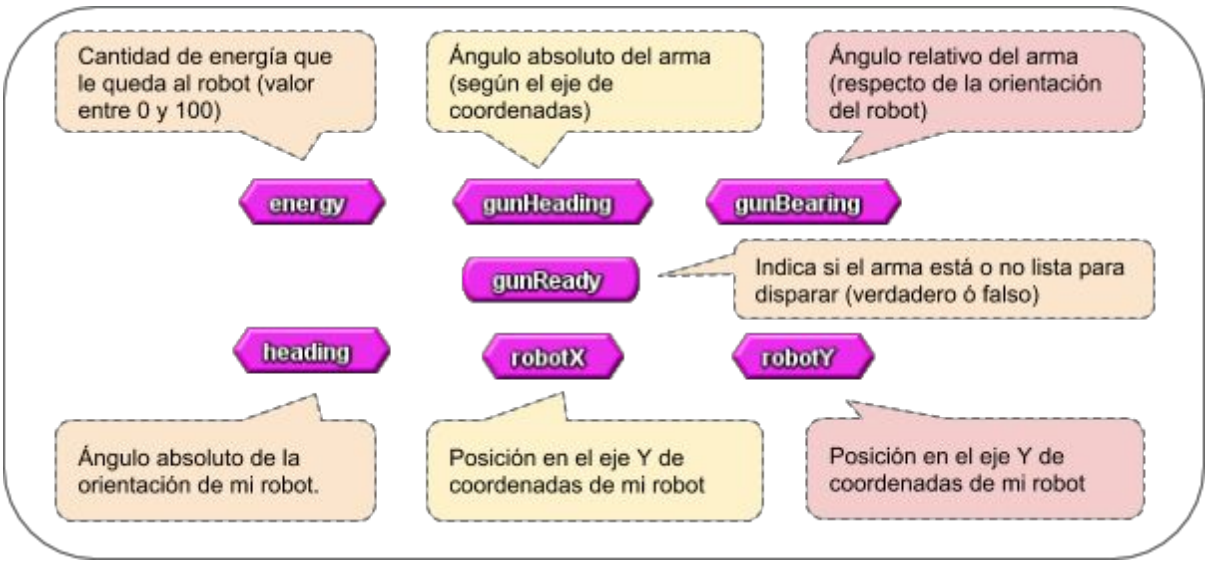
¿QUÉ ES UNA SECUENCIA?

Una secuencia no es más que una serie de acciones ordenadas que permiten alcanzar un objetivo. Estas acciones se ejecutarán paso a paso, una a la vez, no hay opciones de decisión, simplemente ocurrirán. En el mundo de la programación llamamos a cada acción como una "instrucción" que debe ejecutarse.
El concepto de secuencia no sólo se aplica en el ámbito de la programación, normalmente cualquier tarea que realizamos cotidianamente, ya sea arreglarnos para iniciar nuestro día, ó realizar un viaje de la casa a la escuela conlleva un serie de acciones, y aquí también aparece el concepto de **secuencia**.

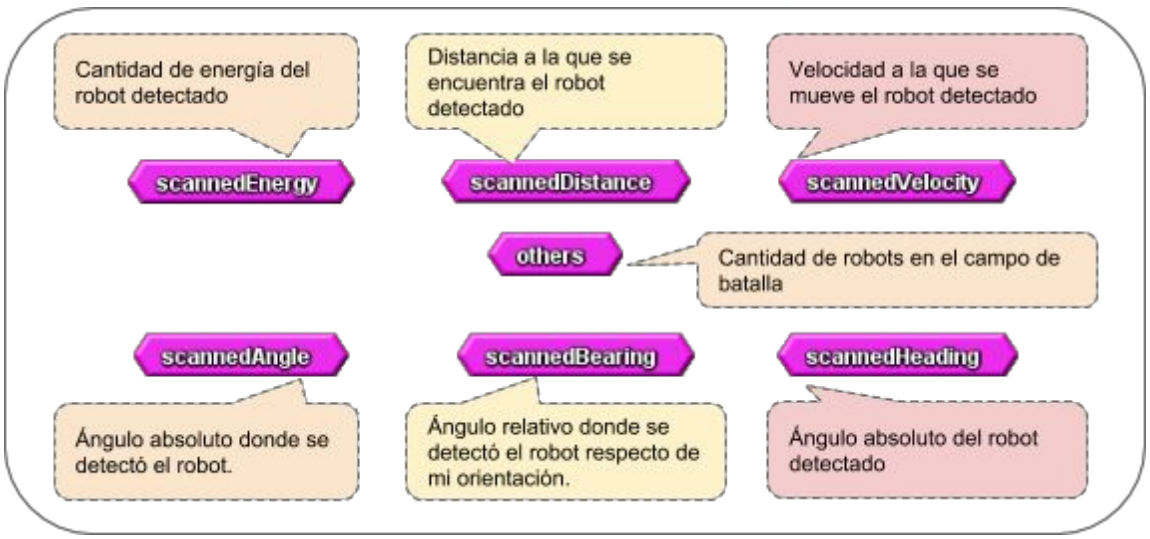
BLOQUES CON INFORMACIÓN ÚTIL

Nuestro robot mantiene información de su propio estado ó desempeño en el campo de batalla. Esta información puede ser recuperada y es útil para la toma de decisiones de nuestro robot, dado que devuelve valores “verdadero” ó “falso” ó puede usarse como parte de la evaluación de una condición . A continuación se explican qué información representa cada uno de estos bloques.

Información propia del estado del robot



Información acerca de otros robots



Información que afectó a mi robot relacionada a otros robots

Información acerca del campo de batalla

BLOQUES QUE INDICAN VERDAD O FALSEDAD
SECCIÓN LÓGICA

Cotidianamente estamos evaluando si se cumplen o no determinadas condiciones y muy probablemente en función de si éstas se cumplen o no, realicemos otras acciones en consecuencia. En RITA tenemos la posibilidad de evaluar si se cumplen o no ciertas condiciones, es decir, si son **verdaderas** o **falsas**.

De hecho, existen 2 pequeños bloques que representan el valor **verdadero** y el valor **falso**:

Además el resultado de una evaluación puede devolver un valor de verdad o falsedad, a modo de ejemplo, si quisiera saber si la energía de mi robot es menor a 10, necesitaría comparar el valor actual de mi energía frente al valor 10. Para esto, RITA provee varios bloques que permiten realizar comparaciones matemáticas. El ejemplo planteado se resolvería de la siguiente manera:

Note que la forma del bloque azul tiene los lados redondeados. Esto es porque todos los bloques que representan un valor de verdadero o falso adoptan esta forma.

Si quisiera comprobar que la ubicación de mi robot en el campo de batalla, respecto del eje de las X es igual a 500 construiría la siguiente estructura:

Y se pueden construir estructuras más complejas. Por ejemplo, podríamos no sólo saber si es cierto que la energía de nuestro robot es menor a 10 sino que además, la posición de nuestro robot en el eje X de coordenadas es igual a 500. Es decir, queremos saber si las dos condiciones mencionadas previamente se cumplen. Para estos casos, RITA provee el bloque **AND,** que evalúa si dos expresiones son ciertas.

A modo de ejemplo, la expresión planteada en el párrafo anterior se vería así construída con bloques:

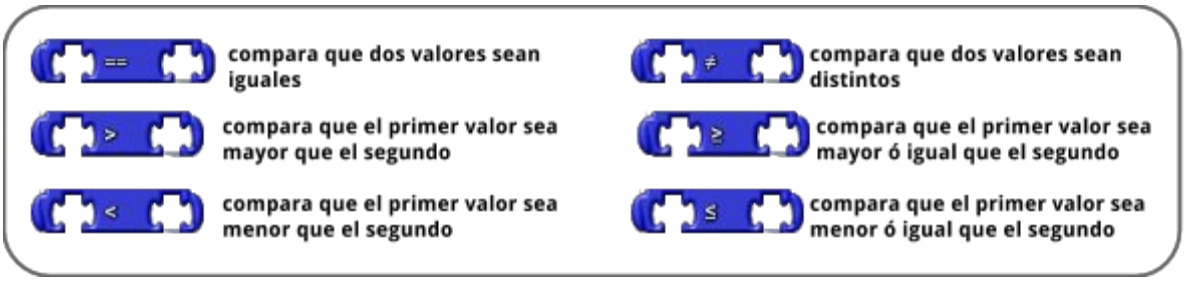
Y también existe el bloque **O** que evalúa si alguna de las dos condiciones, o ambas, son verdaderas, en cuyo caso devuelve un valor de verdad. Sólo devuelve falso en caso de que las dos condiciones sean falsas.

A modo de ejemplo, la misma expresión del bloque **AND** pero evaluada con un bloque **O** se construiría de la siguiente manera:

Cabe resaltar que el bloque AND y el bloque O tienen los extremos redondeados, debido al tipo de valor que retornan.

Estas construcciones condicionales son importantes porque como veremos en la próxima sección, nos ayudarán a completar otros bloques que dotarán de cierto grado de inteligencia a nuestro robot.

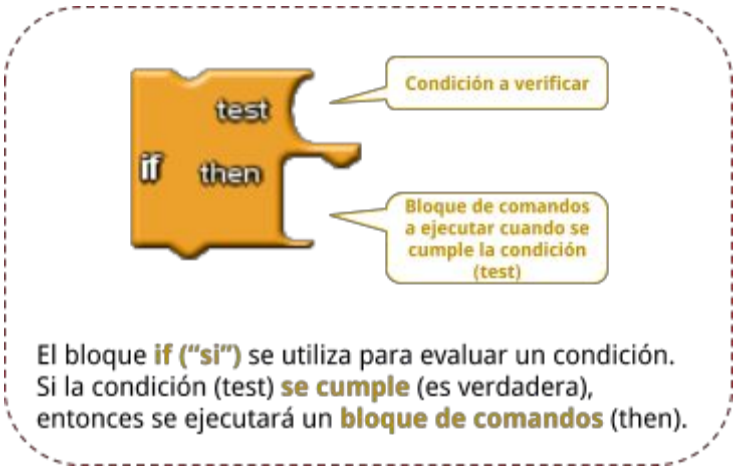
Bloques de comparación que devuelven valor verdadero o falso.



BLOQUES QUE PERMITEN LA TOMA DE DECISIÓN

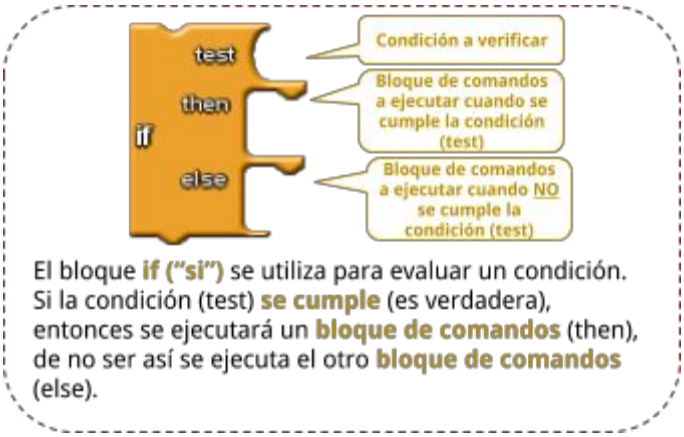
RITA provee bloques que le permiten al robot realizar determinadas acciones sólo si se cumplen o no determinadas condiciones (que pueden estar relacionadas con las condiciones propias del robot o externas). Por ejemplo, el robot podría decidir en función de la cantidad de energía que le queda si le conviene o no atacar al adversario, dado que si intenta atacar a su adversario y ese ataque no resulta certero, terminará perdiendo energía.

El bloque IF nos permitirá realizar lo explicado previamente



Por ejemplo, si quisiéramos indicarle a nuestro robot que cuando la energía sea menor que 10 no realice ninguna acción, debería formar la siguiente estructura:

Podemos armar estructuras más complejas con otro bloque IF más completo, donde podemos indicar que si se cumple una condición realice determinadas acciones, pero si no se cumple esa condición, también puede realizar otro conjunto de acciones:

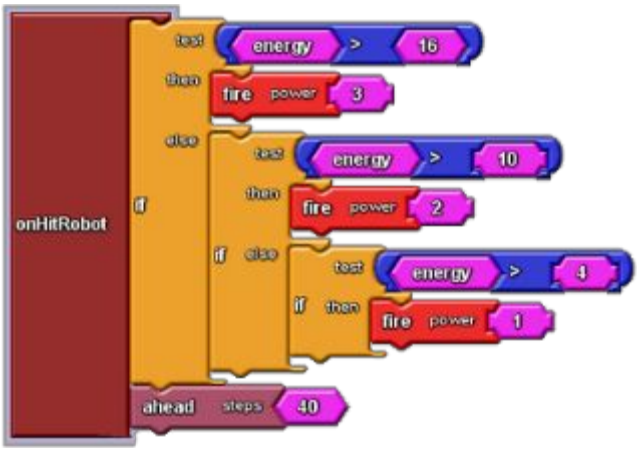


El bloque **if ("si")** se utiliza para evaluar un condición. Si la condición (test) **se cumple** (es verdadera), entonces se ejecutará un **bloque de comandos** (then), de no ser así se ejecuta el otro **bloque de comandos** (else).

Un ejemplo para completar el visto previamente sería evaluar la energía, si es menor a 10 el robot no hace nada, mientras que si tiene 10 de energía o más, avance 50 pasos hacia adelante.

Y ya con estos bloques podemos construir estructuras más complejas que permiten realizar un robot más inteligente.

Analicemos la siguiente construcción de bloques:



En función de la cantidad de energía de nuestro robot realizamos distintas acciones. Por ejemplo, la potencia del disparo varía en función de la cantidad de energía que le queda a mi robot. Podría resultar interesante esta evaluación de condiciones porque al realizar un disparo con más potencia, si bien causa más daño a mi enemigo, también hace que pierda más energía si no logro dar en el blanco.

EL CONCEPTO DE "CONDICIONAL"

Un condicional es una expresión que puede resultar verdadera o falsa. A diario realizamos la evaluación de condicionales: "si llueve entonces salgo con paraguas", "si salgo tarde del trabajo entonces tomo un taxi", etc. Nuestro robot también es capaz de evaluar determinadas condiciones, por ejemplo: dada la expresión "*la energía del robot es menor a 10*", esta puede resultar verdadera o falsa dependiendo de la cantidad de energía actual de nuestro robot y dependerá del resultado de esta evaluación las acciones que realice.

Los condicionales nos permitirán dotar de cierto "grado de inteligencia" a nuestro robot, de modo que sea capaz de "tomar decisiones" durante la batalla. Por ejemplo:

Dada la siguiente expresión que representa el uso de un condicional

si se cumple una condición entonces realizar determinadas acciones

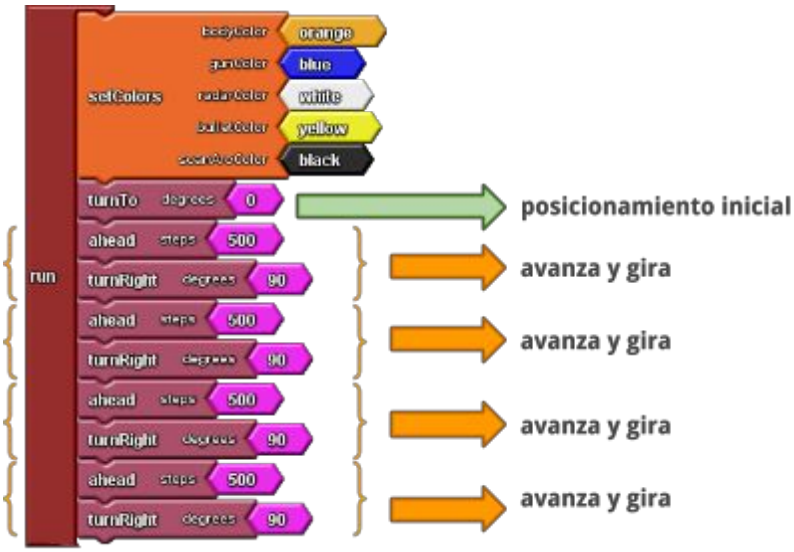
podríamos indicarle a nuestro robot...

si *la energía del robot es menor a 10*, entonces no hacer nada"

lo que en código de bloques equivale a ...

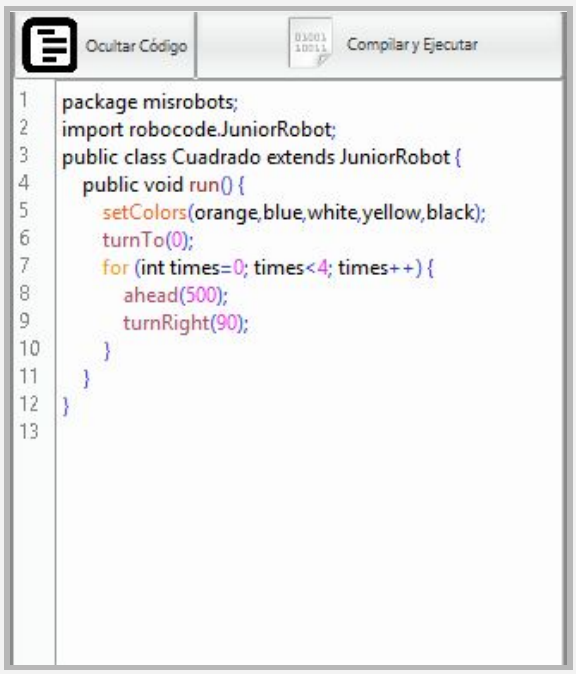
EL CONCEPTO DE ITERACIÓN

Recordemos el Ejercicio 2, donde nuestro **Robot Cuadrado** logró cumplir su objetivo de dibujar un cuadrado en el campo de batalla usando los bloques **ahead** y **turnRight**.



Si analizamos con atención, veremos que el robot "avanza y gira" exactamente igual 4 veces consecutivas, por tanto, podríamos indicarle al robot Cuadrado que realice 4 veces ese mismo par de acciones.

La iteración usando bloques



La iteración escrita en código Java

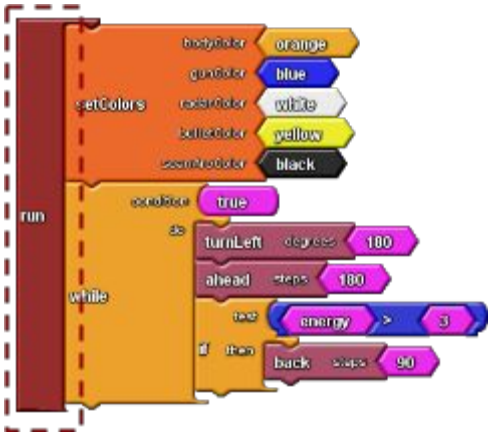
De esta forma además simplificamos la lectura de las acciones que realiza nuestro robot.

¿Qué es una iteración?

Una **iteración** no es más que la repetición de acciones o “instrucciones” ya sea una cantidad fija de veces (bloque **for**) o mientras se cumpla una determinada condición (bloque **while**). Para ejemplificar este último caso, necesitamos tener entendido el concepto de qué es un condicional.

ACCIONES Y REACCIONES DEL TANQUE

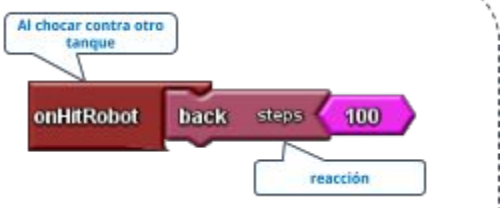
Nuestro tanque tiene un bloque especial donde se encuentran las acciones que realizará siempre que nuestro robot no tenga otra tarea que realizar.



Además, puede reaccionar ante algunos eventos que pueden sucederle durante el transcurso de la batalla, como puede ser:

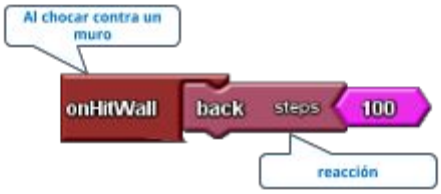
- Al **chocar** contra otro **tanque**
- Al ser **alcanzado por una bala** enemiga
- Al **chocar** contra un **muro**
- Al **escanear** (detectar) un tanque

Para poder indicarle a nuestro robot qué acciones realizar ante cada uno de estos eventos, vamos a completar los bloques que representan eventos (los puede recuperar desde la sección “A Codificar”).

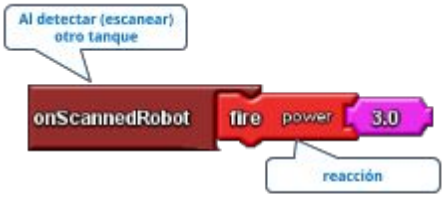


Cuando **nuestro tanque choque a otro**, realizará las acciones que le indiquemos. En el ejemplo, el tanque retrocederá 100 pasos.

Cuando **nuestro tanque sea alcanzado por una bala enemiga**, realizará las acciones que le indiquemos. En el ejemplo, el tanque retrocederá 100 pasos.



Cuando **nuestro tanque choque contra un muro**, realizará las acciones que le indiquemos. En el ejemplo, el tanque retrocederá 100 pasos.



Cuando **nuestro tanque detecte a otro tanque**, realizará las acciones que le indiquemos. En el ejemplo, el tanque disparará con potencia máxima(3).