

Fijando nuestro objetivo

Anteriormente mencionamos los *paradigmas de programación*. En este capítulo vamos a conocer otra forma de pensar el mundo de la programación.

El paradigma de programación con objetos o programación *orientada a* objetos nos propone tratar con... ¡Adiviná! Sí, nos permite trabajar con objetos.

Estos objetos pueden ser *cualquier* cosa, material o abstracta. Un objeto es cualquier entidad que pueda hacer algo por nosotros para resolver un problema.

Lo importante es que estos objetos viven dentro de un mismo mundo y cada uno de ellos va a tener distintas responsabilidades. Además, van a poder comunicarse entre ellos mandándose mensajes.

Para aprender a hacer todo esto, vamos a utilizar un lenguaje llamado Ruby.

Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL





¡Hola Pepita!

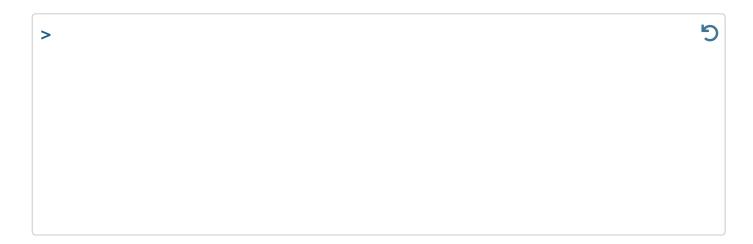
Para empezar en este mundo, conozcamos a Pepita, una golondrina.



Pepita, además de ser un ave que come y vuela (como todo pájaro), es un objeto, que vive en el mundo de los objetos, al cual conocemos como **ambiente**.

¿No nos creés que Pepita está viva y es un objeto? Escribí en la consola Pepita y fijate qué sucede. Cuando te convenzas, pasá al siguiente ejercicio.

♀¡Dame una pista!



Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL





Un mundo de objetos

Como vimos, Pepita es un objeto. Pero Pepita no está sola en este mundo. ¡Hay muchos 3. Un mundo de objetos

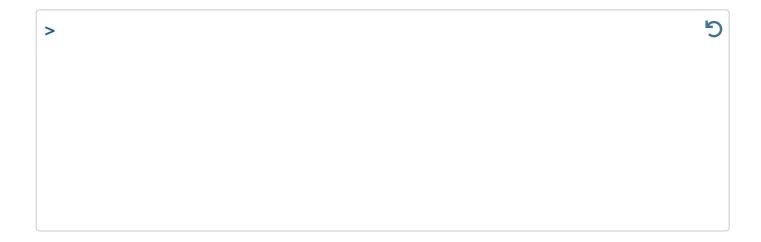
Por ejemplo, existe otra golondrina, llamada Norita, que también vive en este ambiente.

Como ya te vendrás dando cuenta, en este **paradigma** bajo el cual estamos trabajando absolutamente todo es un objeto: también los números, las cadenas de texto (o *strings*) y los booleanos.

¡Probalo! Hacé las siguientes consultas en la consola:

> Pepita
> Norita
> 87
> 'hola mundo'
> true

De todas formas tené cuidado. A Pepita y Norita las creamos por vos. Y los números y booleanos vienen de regalo. Si probás con otra cosa, como por ejemplo Felix, te va a tirar un error.



Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL





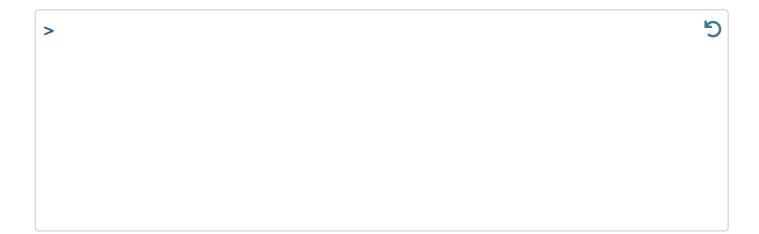
El derecho a la Identidad

•

Un aspecto muy importante de los objetos es que tienen identidad: cada objeto sabe quién es y gracias a esto sabe también que es diferente de los demas. Por ejemplo, Pepita sabe que ella es diferente de Norita, y viceversa.

En Ruby, podemos comparar por identidad a dos objetos utilizando el operador == de la siguiente forma:

♀¡Dame una pista!



Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL





Mensajes, primera parte

Ya entendimos que en un ambiente hay objetos, y que cada uno de ello**≄**tiene *identidad*: sabe que es diferente de otro.

5. Mensajes, primera parte

Pero esto no parece ser muy útil. ¿Qué cosas sabrá hacer una golondrina como Pepita? ¿Sabrá, por ejemplo, cantar!?

Averigualo: enviale un mensaje cantar! y fijate qué pasa...

> Pepita.cantar!

Q i Dame una pista!

> 5

Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL

Información importante

Términos y Condiciones





Mensajes, segunda parte

Ehhhh, ¿qué acaba de pasar acá?

6. Mensajes, segunda parte

Para comunicarnos con los objetos, debemos enviarles **mensajes**. Cuando un objeto recibe un mensaje, este responde *haciendo algo*. En este caso, Pepita produjo el sonido de una golondrina: pri pri pri ...imaginate acá que escuchamos este sonido.....

¿Qué mas sabrá hacer Pepita? ¿Sabrá, por ejemplo, bailar!?



Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL

Información importante

Términos y Condiciones





No entendí...

¡Buu, Pepita no sabía bailar!



En el mundo de los objetos, sólo tiene sentido enviarle un mensaje a un objeto si lo entiende, es decir, si sabe hacer algo como reacción a ese mensaje. De lo contrario, se lanzará un error un poco feo (y en inglés) como el siguiente:



Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL



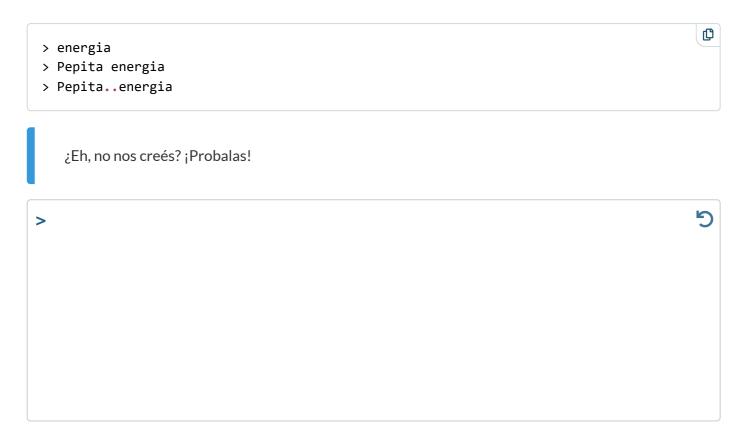


Un poco de sintaxis

¡Pausa! Analicemos la sintaxis del envío de mensajes:

- 1. Pepita. energia es un envío de mensaje, también llamado colaboración;
- 2. energia es el mensaje;
- 3. energia es el nombre del mensaje (en este caso es igual, pero ya veremos otros en los que no);
- 4. Pepita es el objeto receptor del mensaje.

Es importante respetar la sintaxis del envío de mensajes. Por ejemplo, las siguientes NO son colaboraciones validas, porque no funcionan o no hacen lo que deben:



Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la

© 2015-2022 Ikumi SRL





Interfaz

•

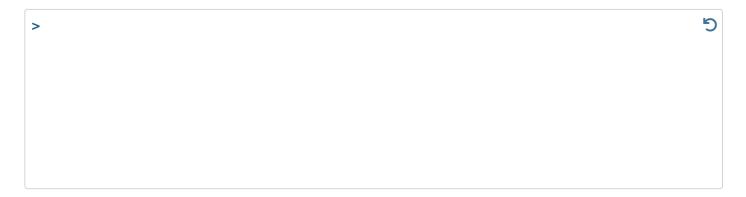
Como vimos, un objeto puede entender múltiples mensajes; a este conjunto de mensajes que podemos enviara lo denominamos interfaz. Por ejemplo, la interfaz de Pepita es:

- energia: nos dice cuanta energía tiene (un número);
- cantar! : hace que cante;
- comer_lombriz!: hace que coma una lombriz;
- volar_en_circulos!: hace que vuele en circulos.

Lo cual también se puede graficar de la siguiente forma:

¡Un momento! ¿Por qué algunos mensajes terminan en ! y otros no? Enviá nuevamente esos mensajes. Fijate qué devuelve cada uno (lo que está a la derecha del =>) y tratá de descubrir el patrón.

?¡Dame una pista!



Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL





Hacer versus Devolver

Cuando se envía un mensaje a un objeto, y este lo entiende, puede reaccionar de dos formas diferentes:

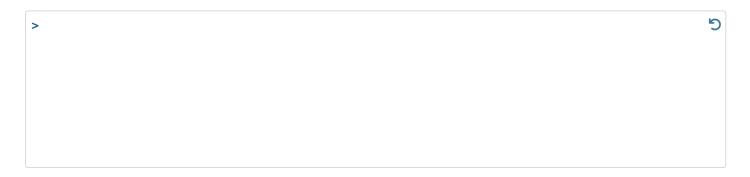
10. Hacer versus Devolver

- Podría producir un efecto, es decir hacer algo. Por ejemplo, el mensaje cantar! reproduce el sonido del canto de Pepita.
- O también podría devolver otro objeto. Por ejemplo el mensaje energia devuelve siempre un número.

En realidad, un mensaje podría reaccionar con una combinación de las formas anteriores: tener un efecto y devolver algo. Pero esto es una **muy** mala idea.

¿Y qué hay de los mensajes como comer_lombriz! y volar_en_circulos!? ¿Hicieron algo? ¿Qué clase de efecto produjeron? ¿Devuelve energia siempre lo mismo?

Descubrilo: enviale a Pepita esos tres mensajes varias veces en distinto orden y fijate si cambia algo.



Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL

Información importante Términos y Condiciones





Tu primer programa con objetos

¡Exacto! El efecto que producen los mensajes comer_lombriz! y volar_en_circulos! es el de alterar la energía de Pepita. En concre#o:

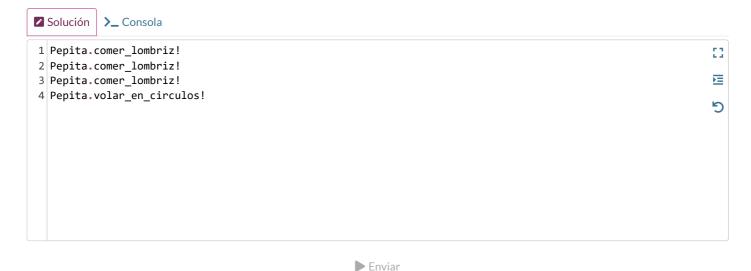
11. Tu primer programa con ob

- comer_lombriz! hace que la energia de Pepita aumente en 20 unidades;
- volar_en_circulos! hace que la energia de Pepita disminuya en 10 unidades.

Como convención, a los mensajes con efecto (es decir, que hacen algo) les pondremos un signo de exclamación ! al final.

Veamos si se entiende: escribí un primer programa que consista en hacer que Pepita coma y vuele hasta quedarse con 150 unidades de energía. Acordate que Pepita arranca con la energía en 100.

🗘 ¡Dame una pista!



¡Muy bien! Tu solución pasó todas las pruebas

Podemos sacar dos conclusiones:

- 1. Los objetos no reaccionan necesariamente siempre igual a los mismos mensajes. Podrían hacer cosas diferentes, o en este caso, devolver objetos distintos.
- 2. ¡Un programa es simplemente una secuencia de envío de mensajes!

 $Esta\ gu\'ia\ fue\ desarrollada\ por\ Federico\ Aloi, Franco\ Bulgarelli, Ariel\ Umansky\ bajo\ los\ t\'erminos\ de\ la\ Licencia\ Creative\ Commons\ Compartir-Igual, 4.0.$

© 2015-2022 Ikumi SRL

Información importante Términos y Condiciones





¿Quién te entiende?

Ya vimos que un objeto puede entender múltiples mensajes, y esos mensajes conforman su interfaz.

12. ¿Quién te

¿Pero podría haber más de un objeto que entienda los mismos mensajes?

A Pepita ya la conocemos bien: canta, come, etc. Su amiga Norita, por otro lado, **no aprendió** nunca a decirnos su energía. Y Mercedes es una reconocida cantora.

Usando la consola, averiguá cuál es la interfaz de cada una de ellas, y completá el listado de mensajes que cada una entiende en el editor.

O;Dame una pista!

```
Solución
           >_ Consola
 1 interfaz_pepita = %w(
                                                                                                                  23
 2
     energia
 3
     cantar!
                                                                                                                  Σ
 4
     comer_lombriz!
                                                                                                                  5
 5
     volar_en_circulos!
 6)
 7
 8 interfaz_norita = %w(
9
     cantar!
10
     comer_lombriz!
11
     volar_en_circulos!
12)
13
14 interfaz_mercedes = %w(
    cantar!
15
16)
```

▶ Enviar

⊘ ¡Muy bien! Tu solución pasó todas las pruebas

¡Así es! Puede haber más de un objeto que entienda el mismo mensaje. Notá que sin embargo no todos los objetos están obligados a reaccionar de igual forma ante el mismo mensaje:

```
Pepita.cantar!
=> "pri pri pri"

Norita.cantar!
=> "priiiip priiiip"

Mercedes.cantar!
=> "\( \) una voz antigua de viento y de sal \( \beta \)"
```

Esto significa que dos o más objetos pueden entender un mismo mensaje, pero pueden **comportarse** de formas diferentes. Ya hablaremos más de esto en próximas lecciones.

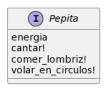
Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.





Interfaces compartidas

Veamos si queda claro, siendo que las interfaces de Norita, Pepita y Mercedes son las siguientes:







Esto significa que comparten algunos mensajes y otros no. ¿Qué interfaces comparten entre ellas?

Completá el código en el editor.


```
Solución
           >_ Consola
 1 # ¿Qué interfaz comparten Mercedes y Norita?
                                                                                                               53
 2 interfaz_compartida_entre_mercedes_y_norita = %w(
 3
                                                                                                               Σ
 4)
                                                                                                               5
 5
 6 # ¿Qué interfaz comparten Pepita y Norita?
 7 interfaz_compartida_entre_pepita_y_norita = %w(
 8
    cantar!
    comer_lombriz!
 9
10
     volar_en_circulos!
11)
12
13 # ¿Qué interfaz comparten Mercedes, Norita y Pepita?
14 interfaz_compartida_entre_todas = %w(
15
    cantar!
16)
```

Enviar

⊘¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL Información importante Términos y Condiciones





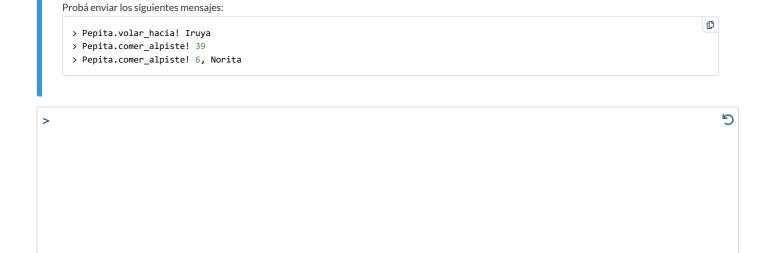
Argumentos

Para hacer las cosas más interesantes, vamos a necesitar mensajes más complejos.

Por ejemplo, si queremos que Pepita coma una cierta cantidad de alpiste que no sea siempre la misma, necesitamos de alguna manera indicar cuál es esa cantidad. Esto podemos escribirlo de la siguiente forma:



Allí, 40 es un *argumento* del mensaje, representa en este caso que vamos a alimentar a pepita con 40 gramos de alpiste. Un mensaje podría tomar más de un argumento, separados por coma.



Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL





Más argumentos

Como ves, si enviás un mensaje con una cantidad incorrecta de argumentos...

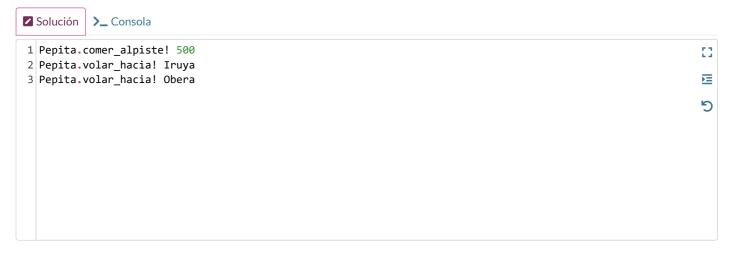
```
> Pepita.comer_alpiste! 6, Norita
# wrong number of arguments (2 for 1) (ArgumentError)
```

...el envío del mensaje también fallará.

Dicho de otra forma, un mensaje queda identificado no sólo por su nombre sino también por la cantidad de parámetros que tiene: no es lo mismo comer_alpiste! que comer_alpiste! 67 que comer_alpiste! 5, 6, son todos mensajes distintos. Y en este caso, Pepita sólo entiende el segundo.

Veamos si va quedando claro: escribí un programa que haga que Pepita coma 500 gramos de alpiste, vuele a Iruya, y finalmente vuelva a Obera.

O ¡Dame una pista!



Enviar

⊘ ¡Muy bien! Tu solución pasó todas las pruebas

¡Perfecto!

Un detalle: en Ruby, a veces, los paréntesis son opcionales. Por eso, cuando no sean imprescindibles los omitiremos.

 $Esta \ gu\'ia \ fue \ desarrollada \ por \ Federico \ Aloi, Franco \ Bulgarelli, Ariel \ Umansky \ bajo \ los \ t\'erminos \ de \ la \ Licencia \ Creative \ Commons \ Compartir-Igual, 4.0.$

© 2015-2022 Ikumi SRL

Información importante Términos y Condiciones





Mensajes por todas partes

Es fácil ver que en Pepita.volar_hacia! Barreal el objeto receptor es Pepita, el mensaje volar_hacia! y el argumento Barreal; pero ¿dónde queda eso de objeto y mensaje cuando hacemos, por ejemplo, 2 + 3?

Como ya dijimos, todas nuestras interacciones en un ambiente de objetos ocurren enviando mensajes y las operaciones aritméticas **no son la excepción** a esta regla.

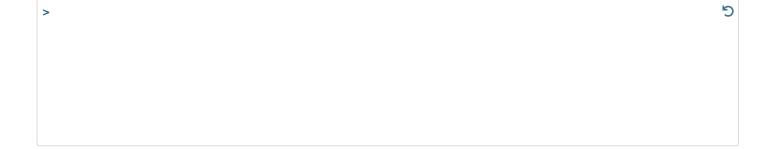
En el caso de 2 + 3 podemos hacer el mismo análisis:

- el objeto receptor es 2;
- el mensaje es +;
- el argumento es 3.

Y de hecho, ¡también podemos escribirlo como un envío de mensajes convencional!

Probá en la consola los siguientes envíos de mensajes:

> 5.+ 6
> 3.< 27
> Pepita.== Norita



Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL





Recapitulando

En un mundo de objetos, todo lo que tenemos son **objetos** y **mensajes**. A estos últimos, podemos distinguirlos según la forma en que se escriben:

- > Mensajes de palabra clave. Su nombre está compuesto por una o varias palabras, puede terminar con un signo de exclamación ! o de pregunta ?, y se envía mediante un punto. Además,
 - pueden no tomar argumentos, como Rayuela.anio_de_edicion;
 - o pueden tomar uno o más argumentos, separados por coma: SanMartin.cruzar! LosAndes, Mula.
- > **Operadores**. Son todos aquellos cuyo "nombre" se compone de uno o más símbolos, y se envían simplemente escribiendo dichos símbolos. En cuanto a los argumentos,
 - pueden no tomar ninguno, como la negación !true;
 - o pueden tomar uno (y solo uno), como Orson == Garfield o energia + 80.

Como vimos, también se pueden escribir como mensajes de palabra clave (aunque no parece buena idea escribir 1.== 2 en vez de 1 == 2).

Vamos a enviar algunos mensajes para terminar de cerrar la idea. Te toca escribir un programa que haga que Pepita:

- 1. Coma 90 gramos de alpiste.
- 2. Vuele a Iruya.
- 3. Finalmente, coma tanto alpiste como el 10% de la energía que le haya quedado.

Este programa tiene que andar sin importar con cuanta energía arranque Pepita.

O¡Dame una pista!



▶ Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Federico Aloi, Franco Bulgarelli, Ariel Umansky bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2022 Ikumi SRL

Términos y Condiciones

