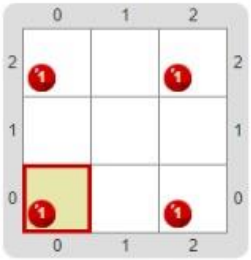
Te quedan 01:08:43

Ejercicio 1: Ejercicio 1

Ale se aburrió de ver vacío su balcón. Así que compró 4 macetas con plantas para ponerlas en cada esquina, de esta forma:



Creá un programa que ponga una maceta (bolita `Rojo`) en cada esquina del balcón. El cabezal empieza en el origen (o sea, en el borde Sur-Oeste) pero no te preocupes por dónde finaliza.

```
program {
  Poner(Rojo)
  Mover(Este)
  Mover(Este)
  Poner(Rojo)

  Mover(Norte)
  Mover(Norte)

  Poner(Rojo)
  Mover(Oeste)
  Mover(Oeste)
  Poner(Rojo)
}
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

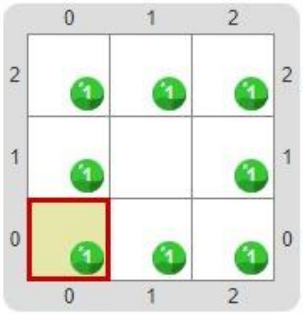
Tablero inicial Tablero final

Te quedan 00:57:13

Ejercicio 1: Ejercicio 1

Ale nos regaló una hermosa pintura para que colguemos en nuestra pared.

Para eso vamos a tener que ponerle un marco el cual pintaremos de Verde:



Creá un programa que pinte de `Verde` los bordes del marco. El cabezal empieza en el origen (o sea, en el borde Sur-Oeste) pero no te preocupes por dónde finaliza.

```
program
{Poner (Verde)
Mover (Norte)
Poner (Verde)
  Mover (Norte)
Poner (Verde)
   Mover (Este)
  Poner (Verde)
  Mover (Sur)
  Mover (Sur)
  Poner (Verde)
  Mover (Este)
  Poner (Verde)
  Mover (Norte)
Poner (Verde)
  Mover (Norte)
Poner (Verde)

```

Te quedan 01:08:33

Ejercicio 2: Ejercicio 2

A Ale no le gustó como quedó ese color en las macetas . ¡Programemos algo para poder probar como quedaría con cualquier color!

> Definí el procedimiento `PonerMacetas` para que ponga macetas del `color` que reciba por parámetro en cada esquina. No te preocupes por donde termina el cabezal.

```
procedure PonerMacetas (color){
  Poner(color)
  Mover(Este)
  Mover(Este)
  Poner(color)
  Mover(Norte)
  Mover(Norte)
  Poner(color)
  Mover(Oeste)
  Mover(Oeste)
  Poner(color)
}
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Resultados de las pruebas:

Te quedan 00:56:47

Ejercicio 2: Ejercicio 2

El color negro del marco del ejercicio anterior no combinó mucho con la pintura . ¡Hagamos algo para poder probar como quedaría con cualquier color!

> Definí el procedimiento `PintarMarco` para que pinte el marco con el `color` que tome por parámetro. No te preocupes por donde termina el cabezal.

```
procedure PintarMarco(color)
{
Poner (color)
Mover (Norte)
Poner (color)
  Mover (Norte)
Poner (color)
   Mover (Este)
  Poner (color)
  Mover (Sur)
  Mover (Sur)
  Poner (color)
  Mover (Este)
  Poner (color)
  Mover (Norte)
Poner (color)
  Mover (Norte)
Poner (color)}

```

Enviar

Te quedan 01:08:25

Ejercicio 3: Ejercicio 3

JS

Dejemos atrás los tableros y... ¡Pasemos a JavaScript!

Cuando hacemos un examen de matemáticas es común verificar más de una vez si las cuentas que hicimos están bien.

Para eso vamos a crear una función que reciba 3 números y nos diga si la multiplicación entre los 2 primeros es igual al tercero. Por ejemplo:

```
> esCorrectaLaMultiplicacion(2, 4, 8)
true //Porque 2 por 4 es igual a 8

> esCorrectaLaMultiplicacion(3, 5, 12)
false //Porque 3 por 5 es 15, no 12
```

Definí la función `esCorrectaLaMultiplicacion`.

Solución | Consola

```
1 function esCorrectaLaMultiplicacion(num1, num2, num3){
2   return (num1*num2) === num3
3 }
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Te quedan 00:56:32

Ejercicio 3: Ejercicio 3

Vamos a un maravilloso mundo... ¡el de la matemática!

Los números se pueden operar y comparar. Nada nos impide hacer un poco de ambas al mismo tiempo.

Para eso vamos a crear una función que reciba 3 números y nos diga si la suma de los 2 primeros es menor al tercero. Por ejemplo:

```
> esMasChicaLaSuma(2, 4, 8)
true //Porque 6 es menor que 8

> esMasChicaLaSuma(3, 5, 7)
false //Porque 8 es mayor a 7
```

Definí la función `esMasChicaLaSuma`.

Solución | Consola

```
1 function esMasChicaLaSuma (num1,num2,num3){return
((num1+num2)<num3)}
```

Enviar

Te quedan 01:08:17

Ejercicio 4: Ejercicio 4

JS

Ya pasamos por el tesoro de la matemática. Otro bien preciado es el tiempo . Es por ello que tratamos de usarlo sabiamente.

Con esto en mente vamos a crear una función que dados un nombre y un apodo nos diga cuál de los dos es más corto.

```
> cualEsMasCorto("Luis", "Lucho")
"Luis"

> cualEsMasCorto("Carolina", "Caro")
"Caro"

> cualEsMasCorto("Ricardo", "Ringo")
"Ringo"
```

Definí la función `cualEsMasCorto`.

Solución | Consola

```
1 function cualEsMasCorto(nombre, apodo){
2   if(longitud(nombre)>longitud(apodo)){
3     return apodo
4   }
5   else {
6     return nombre
7   }
8 }
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Te quedan 01:45:29

Ejercicio 4: Ejercicio 4

Ya pasamos por el tesoro de la matemática. Otro bien preciado es el tiempo . Es por ello que tratamos de usarlo sabiamente.

Con esto en mente vamos a crear una función que dados un nombre y un apodo nos diga cuál de los dos es más corto.

```
> cualEsMasCorto("Luis", "Lucho")
"Luis"

> cualEsMasCorto("Carolina", "Caro")
"Caro"

> cualEsMasCorto("Ricardo", "Ringo")
"Ringo"
```

Definí la función `cualEsMasCorto`.

Solución | >_ Consola

```
1 function cualEsMasCorto(nombre,apodo){
2   if (longitud(nombre)<longitud(apodo)){
3     return nombre
4   }else{
5     return apodo
6   }
7 }
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Siguiente Ejercicio: Ejercicio 5 >

Te quedan 01:51:04

Ejercicio 4: Ejercicio 4

Si hay algo que a Ale le molesta es o pasar frío o abrigarse de más . Pero lo que sí sabe, más allá de la temperatura, es de qué color vestirse ese día. Para eso, pensó en una función que recibe una temperatura y un color y responde qué ropa de ese color ponerse. Si la temperatura es 20 grados o más, se pone una remera de ese color. Si no, se pone una campera de ese color:

```
> vestirseAcorde(20, "negra")
"Remera negra"

> vestirseAcorde(19, "verde")
"Campera verde"

> vestirseAcorde(21, "violeta")
"Remera violeta"
```

Definí la función `vestirseAcorde`.

Solución | Consola

```
function vestirseAcorde (temp,color){
  if(temp >= 20){
    return "Remera "+color
  }else{
    return "Campera "+color
  }
}
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Te quedan 01:08:07

Ejercicio 5: Ejercicio 5

JS

Vale está haciendo un trabajo de investigación y nos pidió ayuda . Necesita poder distinguir las palabras más cortas de una oración . Una palabra se considera corta si tiene menos de 5 letras. Veamos un ejemplo:

```
> palabrasCortas(["Hari", "Seldon", "nacido", "en", "el", "año", "11988", "d
e", "la", "Era", "Galáctica"])

["Hari", "en", "el", "año", "de", "la", "Era"]
```

Definí la función `palabrasCortas`.

Solución | >_ Consola

```
function palabrasCortas(palabras){
  let listaPalabras= []
  for (let palabra of palabras){
    if (longitud(palabra) < 5){
      agregar(listaPalabras,palabra);
    }
  }
  return listaPalabras
}
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Te quedan 01:42:50

Ejercicio 5: Ejercicio 5

JS

Vale está haciendo un trabajo de investigación y nos pidió ayuda . Necesita poder distinguir las palabras más cortas de una oración . Una palabra se considera corta si tiene menos de 5 letras. Veamos un ejemplo:

```
> obtenerPalabrasCortas(["Hari", "Seldon", "nacido", "en", "el", "año", "11988", "de", "la", "Era", "Galáctica"])

["Hari", "en", "el", "año", "de", "la", "Era"]
```

Definí la función `obtenerPalabrasCortas`.

Solución | Consola

```
1  function obtenerPalabrasCortas(palabras){
2    let palabrasFiltradas= []
3
4    for (let palabra of palabras){
5      if(longitud(palabra) < 5){
6        agregar(palabrasFiltradas,palabra)
7      }
8    }
9    return palabrasFiltradas
10
11 }
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Te quedan 00:56:12

Ejercicio 5: Ejercicio 5

Para quienes no suelen leer, la concentración puede variar cuando aparecen palabras largas . Para filtrarlas vamos a crear una función que dada una lista de palabras nos devuelva una lista nueva con las que tengan más de 6 caracteres.

```
> obtenerPalabrasMayores(["jarra", "polilla", "caracol", "gato", "provincia"])
["polilla", "caracol", "provincia"]
```

Definí la función `obtenerPalabrasMayores`.

Solución | Consola

```
1 function obtenerPalabrasMayores (palabras){
2 let palabrasFiltradas= []
3 for (let palabra of palabras){
4 if (longitud (palabra) > 6){
5 agregar (palabrasFiltradas,palabra);}}
6
7
8 return palabrasFiltradas}
```

Enviar

Te quedan 01:07:52

Ejercicio 6: Ejercicio 6

Ale estudia Historia y pensó en crear una función que le ayude a hacer resúmenes . Para eso, consiguió registros de hechos históricos con la siguiente forma:

```
let independenciaArgentina = {
  suceso: "La declaración de la independencia de Argentina",
  anio: 1816,
  ciudad: "San Miguel de Tucumán"
};

let declaracionDerechosHumanos = {
  suceso: "La declaración universal de los Derechos Humanos",
  anio: 1948,
  ciudad: "París"
};
```

La función deberá devolver un resumen de la información registrada de manera simple. Por ejemplo:

```
> resumenHechoHistorico(independenciaArgentina)
"La Declaración de la independencia de Argentina ha sucedido hace 205 años en San Miguel de Tucumán"

> resumenHechoHistorico(declaracionDerechosHumanos)
"La Declaración Universal de los Derechos Humanos ha sucedido hace 73 años en Paris"
```

Definí la función `resumenHechoHistorico` que nos permita obtener la información requerida. Asumí que estamos en 2021.

Solución | Consola

```
1 function resumenHechoHistorico(datos){
2   return datos.suceso + " ha sucedido hace " + (2021 -
  datos.anio) + " años en " + datos.ciudad
3 }
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Te quedan 01:37:31

Ejercicio 6: Ejercicio 6

Ale estudia Historia y pensó en crear una función que le ayude a hacer resúmenes . Para eso, consiguió registros de hechos históricos con la siguiente forma:

```
let independenciaArgentina = {
  suceso: "La declaración de la independencia de Argentina",
  anio: 1816,
  ciudad: "San Miguel de Tucumán"
};

let declaracionDerechosHumanos = {
  suceso: "La declaración universal de los Derechos Humanos",
  anio: 1948,
  ciudad: "París"
};
```

La función deberá devolver un resumen de la información registrada de manera simple. Por ejemplo:

```
> resumenHechoMemorable(independenciaArgentina)
"La Declaración de la independencia de Argentina tuvo lugar hace 205 años en S
an Miguel de Tucumán"
```

Solución | Consola

```
1 function resumenHechoMemorable(suceso){
2
3  return suceso.suceso + " tuvo lugar hace " +
4 (2021 - suceso.anio) + " años en " + suceso.ciudad
5
6 }
```

Enviar

Te quedan 01:19:03

Ejercicio 6: Ejercicio 6

Ale estudia Historia y pensó en crear una función que le ayude a hacer resúmenes . Para eso, consiguió registros de hechos históricos con la siguiente forma:

```
let independenciaArgentina = {
  suceso: "La declaración de la independencia de Argentina",
  anio: 1816,
  ciudad: "San Miguel de Tucumán"
};

let declaracionDerechosHumanos = {
  suceso: "La declaración universal de los Derechos Humanos",
  anio: 1948,
  ciudad: "París"
};
```

La función deberá devolver un resumen de la información registrada de manera simple. Por ejemplo:

```
> resumenHechoMemorable(independenciaArgentina)
"La Declaración de la independencia de Argentina ha sucedido hace 2
```

Solución | Consola

```
1 function resumenHechoMemorable(datos){
2   return datos.suceso + " ha sucedido hace " +
(2021 - datos.anio) + " años en la localidad de " +
datos.ciudad
3 }
```

Enviar

Ejercicio 6: Ejercicio 6

Ale estudia Historia y pensó en crear una función que le ayude a hacer resúmenes . Para eso, consiguió registros de hechos históricos con la siguiente forma:

```
let independenciaArgentina = {
  suceso: "La declaración de la independencia de Argentina",
  anio: 1816,
  ciudad: "San Miguel de Tucumán"
};

let declaracionDerechosHumanos = {
  suceso: "La declaración universal de los Derechos Humanos",
  anio: 1948,
  ciudad: "París"
};
```

La función deberá devolver un resumen de la información registrada de manera simple. Por ejemplo:

Solución | Consola

```
1 function resumenHechoHistorico (resumen){
2   return resumen.suceso + " sucedió hace " + (2021-
resumen.anio)+ " años en la localidad de " +
resumen.ciudad;
3 }
```

Enviar

Te quedan 00:25:16

Ejercicio 6: Ejercicio 6

En una biblioteca guardan registro de todos los libros leídos por las personas que la concurren. Estos registros tienen la siguiente forma:

```
let juan = {
  nombre: "Juan Arrever",
  librosLeidos: ["El conde de Montecristo", "La palabra", "Mi plant
a de naranja lima"],
  anioSuscripcion: 1992
};

let elena = {
  nombre: "Elena Chalver",
  librosLeidos: ["Rabia", "Vida de Bob Marley"],
  anioSuscripcion: 1987
};
```

Solución | Consola

```
1 function resumenInformacion(persona){
2   return persona.nombre+" tiene suscripcion hace "+
(2021 - persona.anioSuscripcion)+" años y leyó "
3   +longitud(persona.librosLeidos)+" libros"}
```

Te quedan 01:07:39

Ejercicio 7: Ejercicio 7

¡Dejemos atrás a JavaScript para pasar a Ruby!

Vamos a modelar la clase `Celular` para poder:

- cargar una cantidad de minutos determinada (si cargamos 10 minutos, su bateria incrementa en 10);
- ver si tiene carga suficiente, es decir, si su bateria es mayor a 50.

> Definí en Ruby, la clase `Celular` que tenga un atributo `@bateria` con su getter. Las instancias de la clase `Celular` entienden los mensajes `cargar!` (que recibe los minutos por parámetro y lo carga esa cantidad) y `suficiente_bateria?`. No te olvides de definir un `initialize` que reciba a la bateria inicial como parámetro.

Solución | Consola

```ruby
class Celular
  def initialize(bateria)
    @bateria = bateria
  end

  def bateria
    @bateria
  end

  def cargar!(minutos)
    @bateria += minutos
  end

  def suficiente_bateria?
    @bateria > 50
  end
end

```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 7: Ejercicio 7

¡Dejemos atrás a JavaScript para pasar a Ruby!

Vamos a modelar `Comida` s para poder:

- agregarle cucharadas de sal;
- ver si tiene demasiada sal, es decir, si tiene más de 4 cucharadas de sal.

> Definí en Ruby, la clase `Comida` que tenga un atributo `@cucharadas_sal` con su getter. Las instancias de esta clase entienden los mensajes `poner_cucharadas!` (que recibe la cantidad a agregar por parámetro) y `mucha_sal?`. No te olvides de definir un `initialize` que reciba las cucharadas de sal iniciales como parámetro.

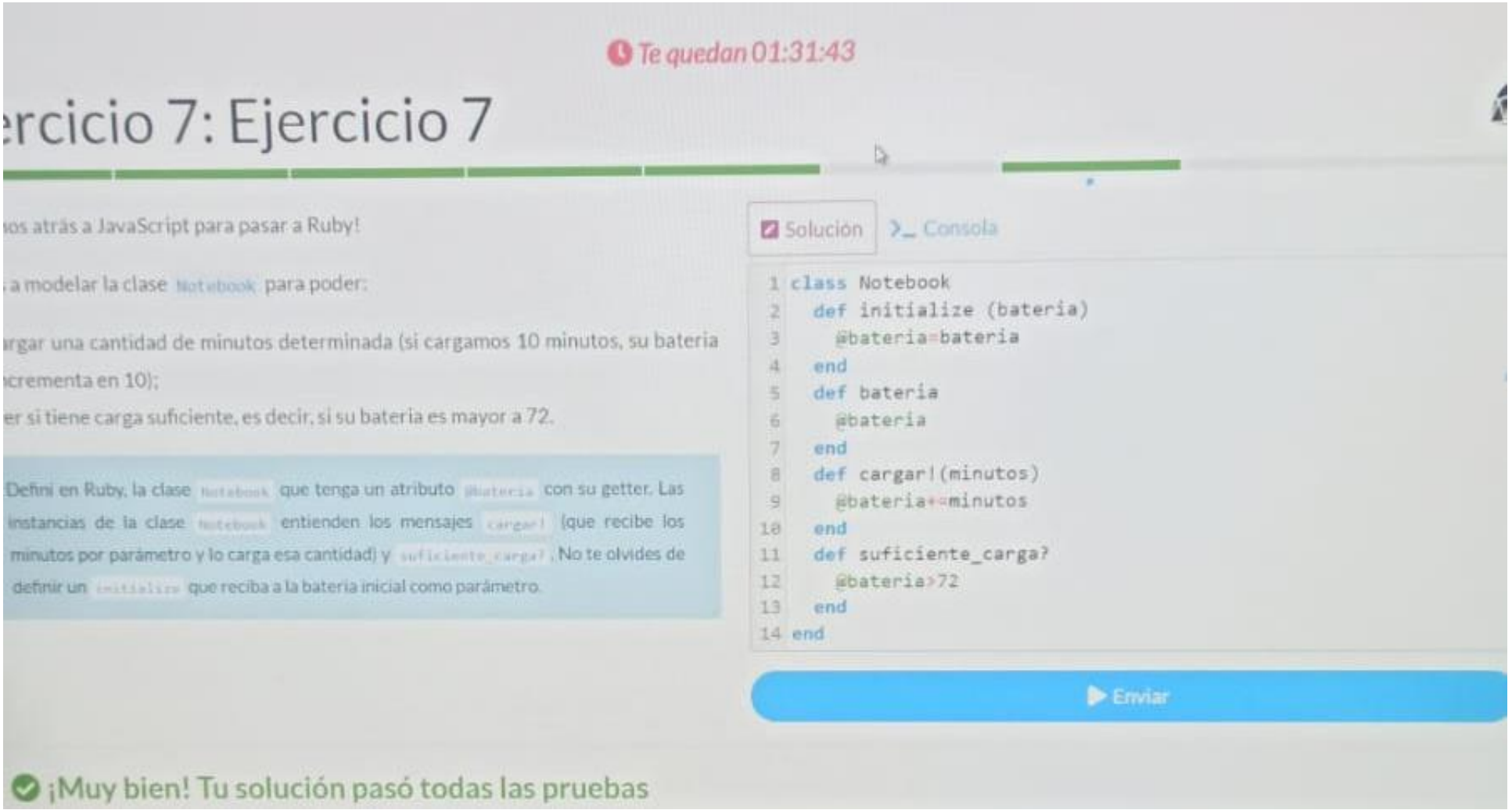
Solución | >_ Consola

```ruby
class Comida
  def initialize(cant)
  @cucharadas_sal = cant
  end
  def cucharadas_sal
    @cucharadas_sal
  end
def poner_cucharadas!(cantidad)
  @cucharadas_sal += cantidad
end

    def mucha_sal?
      return @cucharadas_sal > 4
    end
end
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas


Te quedan 01:31:43
ercicio 7: Ejercicio 7
os atrás a JavaScript para pasar a Ruby!
a modelar la clase Notebook para poder:
rgar una cantidad de minutos determinada (si cargamos 10 minutos, su bateria
crementa en 10);
er si tiene carga suficiente, es decir, si su bateria es mayor a 72.
Defini en Ruby, la clase Notebook que tenga un atributo @bateria con su getter. Las instancias de la clase Notebook entienden los mensajes cargar! (que recibe los minutos por parámetro y lo carga esa cantidad) y suficiente_carga? . No te olvides de definir un initialize que reciba a la bateria inicial como parámetro.
Solución
Consola
1 class Notebook
2 def initialize (bateria)
3 @bateria=bateria
4 end
5 def bateria
6 @bateria
7 end
8 def cargar!(minutos)
9 @bateria+=minutos
10 end
11 def suficiente_carga?
12 @bateria>72
13 end
14 end
Enviar
¡Muy bien! Tu solución pasó todas las pruebas

Te quedan 01:07:26

Ejercicio 8: Ejercicio 8

A `Lu` le gusta mucho leer , pero le interesa saber cuántos de los libros que leyó son largos. Cada uno de los libros sabe responder al mensaje `demasiado_largo?`.

> Definí en Ruby el método `cantidad_de_libros_largos` que responda a cuántos libros largos leyó `Lu`.

Solución | Consola

```ruby
module Lu
  @libros_leidos = [MartinFierro, Fundacion, ElPrincipito, HarryPotter]

  def self.cantidad_de_libros_largos
    @libros_leidos.count {|libro| libro.demasiado_largo? }
  end
end
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Te quedan 00:21:24

Ejercicio 8: Ejercicio 8

A `Oli` le gusta mucho leer , pero le interesa saber cuántos de los libros que leyó son largos. Cada uno de los libros sabe responder al mensaje `largo?`.

> Definí en Ruby el método `cuantos_libros_largos` que responda a cuántos libros largos leyó `Oli`.

Solución | Consola

```ruby
module Oli
  @libros_leidos = [MartinFierro, Fundacion, ElPrincipito, HarryPotter]
  def self.cuantos_libros_largos
  @libros_leidos.count {|libro| libro.largo?}
  end
end
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 8: Ejercicio 8

Los compilados son discos que tienen la característica de recopilar canciones que comparten alguna característica, por ejemplo artista, época o género. Algunas de ellas con mayor duración que otras.

Teniendo en cuenta que las canciones saben responder al mensaje `corta?` ...

> Definí en Ruby el método `cantidad_de_cortas` que responda a cuántas canciones cortas tiene el `Compilado`.

Solución | Consola

```ruby
module Compilado
  @canciones = [AmorAusente, Eco, Agujas, ElBalcon, GuitarrasDeCarton]
  def self.compilado
  @canciones
  end
  def self.cantidad_de_cortas
  compilado.count{|cancion| cancion.corta?}
  end
end

```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Te quedan 00:08:12

Ejercicio 8: Ejercicio 8

Los compilados son discos que tienen la característica de recopilar canciones que comparten alguna característica, por ejemplo artista, época o género. Algunas de ellas con mayor duración que otras.

Teniendo en cuenta que las canciones saben responder al mensaje `corta?`...

> Definí en Ruby el método `cantidad_cortas` que responda a cuántas canciones cortas tiene el `DiscoCompilado`.

Solución | Consola

```ruby
module DiscoCompilado
  @canciones = [AmorAusente, Eco, Agujas, ElBalcon, GuitarrasDeCarton]
  def self. discocompilado
    @canciones
end
  def self. cantidad_cortas
    discocompilado.count{|cancion| cancion.corta?}
  end
end
```

Enviar

Te quedan 01:07:13

Ejercicio 9: Ejercicio 9

Una productora de cine se encarga de invertir en las películas que trabajan con ella.

Cuando esta empresa invierte:

- Las películas de tipo `Documental` duplican su `duracion`;
- las de `Terror` contratan 5 `extras`;
- las de género `Comedia` no se modifican.

> Definí el método `invertir_en_peliculas!` en la clase `Productora` y el método `recibir_inversion!` en los distintos tipos de películas. Definí los getters necesarios en cada una.

Solución | Consola

```ruby
class Productora
  def initialize(peliculas)
    @peliculas = peliculas
  end

  def invertir_en_peliculas!
    @peliculas.each{|pelicula|pelicula.recibir_inversion!}
  end

end

class Documental
  def initialize(duracion)
    @duracion = duracion
  end

  def duracion
    @duracion
  end

  def recibir_inversion!
    @duracion = duracion * 2
  end
end
```

```ruby
class Terror
  def initialize(extras)
    @extras = extras
  end

  def extras
    @extras
  end

  def recibir_inversion!
    @extras += 5
  end
end

class Comedia

  def recibir_inversion!
  end

end
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Ejercicio 9: Ejercicio 9

Una productora de cine se encarga de invertir en las películas que trabajan con ella. Cuando esta empresa invierte:

- Las películas de tipo `Documental` duplican su `duracion`;
- las de `Terror` contratan 12 `extras`;
- las de género `Comedia` no se modifican.

> Definí el método `invertir!` en la clase `Productora` y el método `ser_invertida!` en los distintos tipos de películas. Definí los getters necesarios en cada una.

Solución | Consola

```
1  class Productora
2    def initialize(peliculas)
3      @peliculas = peliculas
4    end
5
6    def invertir!
7      @peliculas.each{|peliculas|
peliculas.ser_invertida!}
8    end
9  end
10
11 class Documental
12   def initialize(duracion)
13     @duracion = duracion
14   end
15   def duracion
16   @duracion
17   end
18   def ser_invertida!
19   @duracion = duracion * 2
20   end
```

```ruby
  def ser_invertida!
  @duracion = duracion * 2
  end

end

class Terror
  def initialize(extras)
    @extras = extras
  end

  def extras
  @extras
  end

  def ser_invertida!
  @extras = extras + 12
  end

end

class Comedia
  def ser_invertida!
  end
end
```

Enviar

A la hora de relajarse muchas `Persona`s juegan con su mascota. Los animales hacen distintas cosas cuando juegan:

- A los `Perro`s les da hambre;
- los `Gato`s incrementan en 3 su nivel de felicidad;
- las `Tortuga`s no hacen nada.

> Definí el método `jugar_con_mascota!` en la clase `Persona` y el método `jugar!` en los distintos tipos de animales. Definí los getters necesarios en cada una.

Solución | Consola

```ruby
class Persona
  def initialize(mascota)
    @mascota = mascota
  end
  def jugar_con_mascota!
  @mascota.jugar!
  end
end

class Perro
  def initialize()
    @tiene_hambre = false
  end
  def tiene_hambre
  @tiene_hambre
  end
  def jugar!
    @tiene_hambre = true
  end
end

class Gato
  def initialize(nivel_de_felicidad)
    @nivel_de_felicidad = nivel_de_felicidad
  end
  def jugar!
  @nivel_de_felicidad+=3
  end
  def nivel_de_felicidad
    @nivel_de_felicidad
  end
end

class Tortuga
  def jugar!
  end
end
```

Te quedan 00:07:50

Ejercicio 9: Ejercicio 9

A la hora de relajarse muchas `Persona`s juegan con su mascota. Los animales hacen distintas cosas cuando juegan:

- A los `Carpincho`s les da hambre;
- los `Gato`s incrementan en 6 su nivel de felicidad;
- las `Tortuga`s no hacen nada.

> Definí el método `jugar_con_mascota!` en la clase `Persona` y el método `jugar!` en los distintos tipos de animales. Definí los getters necesarios en cada una.

Solución | Consola

```
1  class Persona
2    def initialize(mascota)
3      @mascota = mascota
4    end
5    def jugar_con_mascota!
6      @mascota.jugar!
7    end
8  end
9
10 class Carpincho
11   def initialize()
12     @tiene_hambre = false
13   end
14   def tiene_hambre
15     @tiene_hambre
16   end
```

```ruby
  def tiene_hambre
    @tiene_hambre
  end
  def jugar!
    @tiene_hambre = true
end
end
class Gato
  def initialize(nivel_de_felicidad)
    @nivel_de_felicidad = nivel_de_felicidad
end
  def nivel_de_felicidad
    @nivel_de_felicidad
  end
  def jugar!
    @nivel_de_felicidad += 6
end
end
  class Tortuga
    def jugar!
end
  end
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas