MANUAL DE RUBY

(PARTE V)

Luis José Sánchez González

1. VARIABLES DE CLASE

Hemos visto anteriormente las **variables de instancia**. Cuando se crean instancias a partir de una clase invocando al método **new**, las variables de instancia forman parte de los objetos creados.

Por ejemplo, podemos definir la clase **Coche** de la siguiente manera:

```
class Coche
	def initialize(tipo = "turismo", num_plazas = 5)
		@tipo = tipo
		@num_plazas = num_plazas
	end
end

a = Coche.new
b = Coche.new("todoterreno")
c = Coche.new("deportivo",2)
```

En este caso tenemos las variables de instancia **@tipo** y **@num_plazas**. Por cada objeto de la clase Coche que se cree habrá un @tipo y @num_plazas. Hemos creado tres coches: a, b y c. Por tanto para cada uno de esos tres coches tendremos el correspondiente valor de @tipo y @num_plazas.

A diferencia de las variables de instancia, **las variables de clase son únicas dentro de una clase**.

Ejemplo:

```
class Coche

	@@parque_automovilistico = 0

	def initialize(tipo = "turismo", num_plazas = 5)
		@tipo = tipo
		@num_plazas = num_plazas
		@@parque_automovilistico += 1
	end

	def self.numero_de_coches
		return @@parque_automovilistico
	end
end

a = Coche.new
b = Coche.new("todoterreno")
c = Coche.new("deportivo",2)
puts Coche.numero_de_coches
```

En este caso, la variable de clase **@@parque_automovilistico** contiene el número de coches que se han creado. Esta variable es única para toda la clase **Coche**.

Escribiendo **self.** delante del nombre de un método, estamos indicando que definimos un método que se aplicará a una clase y no a una instancia.

2. MÉTODOS SINGLETON

A veces, el comportamiento de una determinada instancia debe ser especial, distinto al del resto de elementos de la clase. En la mayoría de los lenguajes debemos meternos en la problemática de crear otra clase e instanciarla sólo una vez. En ruby se puede asignar a cada objeto sus propios métodos de manera individual.

Un método que pertenece sólo a un objeto se conoce como **método singleton**.

Ejemplo:

```
class Gato

	def maulla
		puts "miau miau"
	end
end

garfield = Gato.new
bolita = Gato.new

def garfield.maulla
	puts "Yo no maullo, yo hablo"
end

def garfield.cuenta_chiste
	puts "Había un inglés, un francés y un español..."
end

bolita.maulla
garfield.maulla
garfield.cuenta_chiste
```

En el ejemplo anterior se ha definido la clase **Gato**. El objeto **garfield** es un tanto especial (es un gato más listo de la cuenta) por tanto hemos redefinido el método **maulla**. Además hemos creado un método que es específico para él, el método **cuenta_chiste**.

3. VARIABLES GLOBALES

Una variable global tiene un nombre que comienza con **$**. Se puede utilizar en cualquier parte de un programa. Antes de inicializarse, una variable global tiene el valor especial **nil**.

La variables globales deben utilizarse con mucho cuidado. Son peligrosas porque se pueden modificar desde cualquier lugar. Una sobreutilización de variables globales puede dificultar la localización de errores.

Una característica importante de las variables globales es que se les puede hacer un seguimiento. Se puede definir un procedimiento que se llame cada vez que se modifique el valor de la variable.

Ejemplo:

```
trace_var:$x, proc{
        puts "la variable $x ha cambiado, ahora vale #{$x}"
        }
$x = 4
$x = 10
$x += 1
```

4. PASO DE PARÁMETROS

Se pueden pasar argumentos a un programa en ruby mediante **$***

$* es un array que contiene todos los argumentos y se puede acceder a ellos mediante el índice (igual que con cualquier otro array).

Ejemplo:

```
if $*[0] == "es"
        puts "hola"
  else
        puts "hello"
end
```

Si grabamos este programa como ejemplo_argumentos.rb, lo podríamos ejecutar así:

```
ruby  ejemplo_argumentos.rb es
```

o bien

```
ruby  ejemplo_argumentos.rb en
```

y nos saludará de una forma u otra en función del argumento introducido.

Ejercicios

1. Crea la clase **bombilla** con el método **encender**. A este método se le debe pasar como argumento el número de horas que estará encendida la bombilla. Crea una variable de clase que controle el gasto de electricidad global. Cuando se crea una bombilla, se debe indicar cuántos watios de potencia tiene. Se supone que si una bombilla es de 60w, en una hora gastará 60w (0'060 kw).

2. Define, al menos, tres métodos singleton en base a alguna/s clase/s creada/s en ejercicios anteriores.

3. Añade el símbolo **$** a alguna de las variables utilizadas en alguno de los ejercicios realizados anteriormente para convertirla en variable global, y haz un seguimiento de su contenido durante el transcurso del programa. Intenta hacer que cada vez que cambie esa variable, se modifique otra diferente, ¿es posible?

4. Realiza un programa que ordene un lista de palabras que se pasan como argumento.

5. Vuelve a realizar el juego de las 21 (blackjack) pero esta vez utilizando clases ¿cuál de las dos versiones es más corta? ¿cuál es más fácil de entender?