



Programación Orientada a Objetos

Índice



Programación Orientada a Objetos

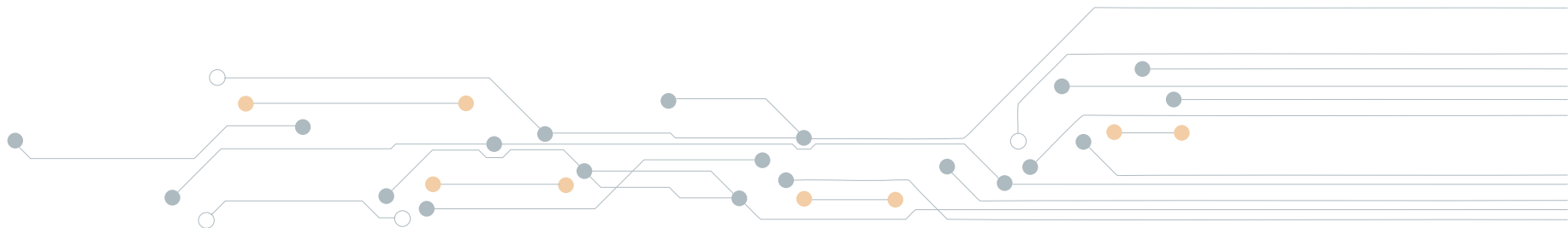
1 Programación Orientada a Objetos	3
1.1 Clases y Objetos	4
Objetos	4
Clases	6
Creación de objetos	8
1.2 Constructores	9

1. Programación Orientada a Objetos

La mayoría de los lenguajes de programación actuales son orientados a objetos. El hecho de que un lenguaje sea orientado a objetos no solo implica que trabaje con objetos, sino que además se pueden aplicar una serie de características propias de la orientación a objetos, encaminadas a reutilizar código en las aplicaciones, a que sean más fáciles de mantener y más fiables.

La programación orientada a objetos (abreviadamente POO), es pues la evolución natural de la programación estructurada. Y es que cuando las aplicaciones son muy grandes, la aplicación de las técnicas básicas de la programación estructurada resultan insuficientes para poder desarrollar programas fiables y fáciles de mantener.

Seguidamente, vamos a presentar los conceptos fundamentales en los que se basa este modelo de programación.



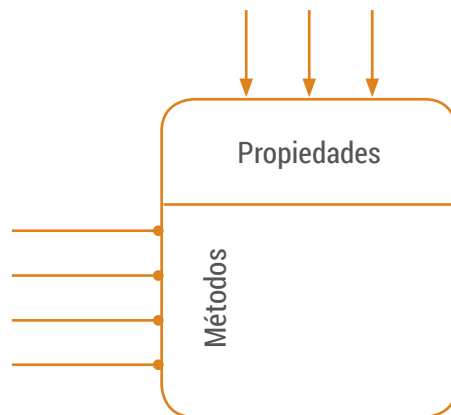
1.1 | Clases y Objetos

Un lenguaje orientado a objetos se basa en el uso de clases y objetos para implementar la funcionalidad de la aplicación, por eso estos conceptos serán los primeros que vamos a abordar en este capítulo.

OBJETOS

Un objeto podemos imaginarlo como una “caja negra” que expone al exterior una serie de funciones con las que poder realizar operaciones, así como una serie de parámetros que permiten configurar características del objeto.

A las funciones y procedimientos que expone el objeto se les llama **métodos**, mientras que a las características se les conoce como **propiedades**.



Para entender mejor el concepto, pongamos un ejemplo del mundo real, que está lleno de objetos. Pensemos, por ejemplo, en un coche. Un coche es un tipo de objeto que expone una serie de propiedades como el color, la matrícula, el peso, etc., además de una serie de métodos para operar con ellos, como arrancar, acelerar, frenar...

Como este tenemos muchos otros ejemplos de objetos del mundo real. Un objeto televisor tiene una serie de propiedades como el número de pulgadas, si es o no 3D, etc., y también unos métodos como encender, apagar o cambiar de canal.

Llevado este concepto al mundo de la programación, el objeto es un elemento residente en memoria y al que accedemos desde el programa a través de una variable. A través de las propiedades podemos configurar características del objeto utilizando la siguiente sintaxis:

```
variable_objeto.propiedad=valor //asignación de valor a una propiedad
```

0

```
a= variable_objeto.propiedad //recuperación del valor de una propiedad
```

En ambos casos, para asignar un valor a una propiedad y para leer el valor de una propiedad, se utiliza la variable que contiene el objeto seguido del **operador punto** (".") y el nombre de la propiedad.

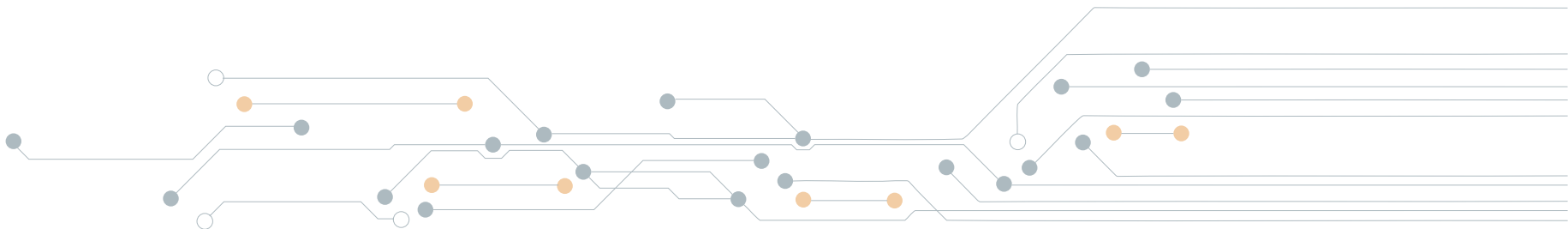
Dado que los métodos representan operaciones sobre el objeto, la llamada a estos ejecutará algún código internamente en el objeto. Para llamar a los métodos de un objeto utilizamos la sintaxis:

```
variable_objeto.metodo(argumento1, argumento2, ..)
```

Donde *argumento1*, *argumento2* son los valores que el método requiere para poder ejecutar la tarea.

Como vemos, la llamada a un método de un objeto es muy similar a las llamadas a procedimientos y funciones, y es que, como veremos a continuación, **un método se implementa mediante procedimientos y/o funciones**. Esto significa que puede haber métodos, aquellos que se creen mediante una función y devuelvan un resultado al punto de llamada, en cuyo caso se deberá recoger el valor devuelto por el mismo:

```
res = variable_objeto.metodo(argumento1, argumento2, ..)
```

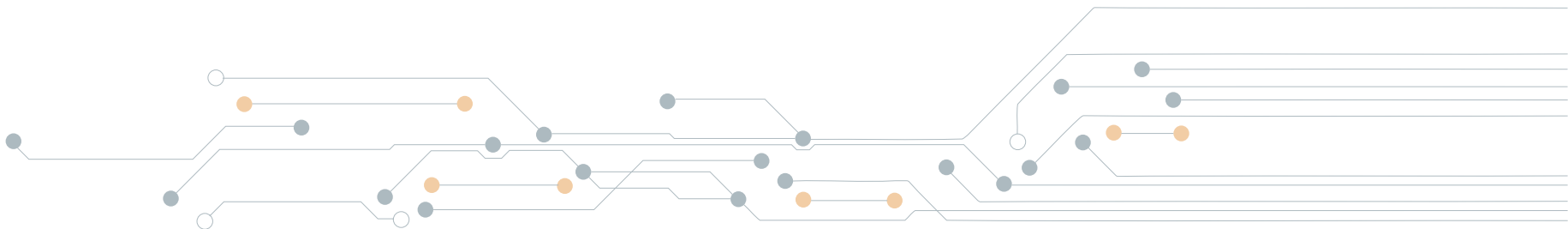
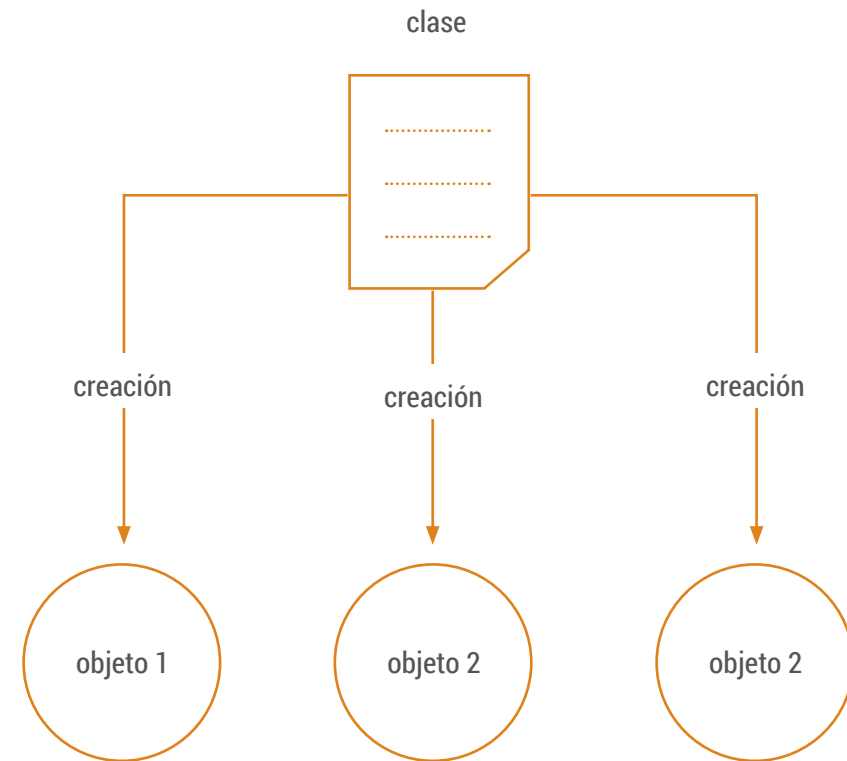


CLASES

Los objetos disponen de métodos, que son implementados mediante funciones y procedimientos, y de propiedades, que vienen a ser variables donde se almacenan las características de los objetos.

El lugar donde se define el código de estos métodos y propiedades es la clase. A través de estos métodos y propiedades, la clase establece el comportamiento de un determinado tipo de objeto

A partir de la clase se crean los objetos y sobre ellos se aplican los métodos y propiedades definidos en la misma. La clase es, por tanto, el molde a partir del cual se pueden construir los objetos, que son los elementos que utilizaremos en los programas para poder hacer uso de los métodos y propiedades.



Cada objeto podrá tener sus propios valores de propiedades, mientras que el comportamiento de los métodos será el mismo en cada objeto.

Para definir una clase mediante pseudocódigo, utilizaremos la siguiente sintaxis:

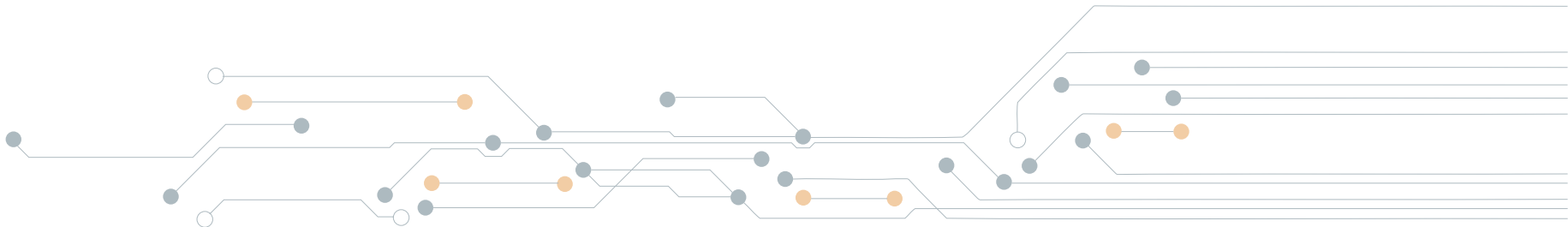
```
Class NombreClase
//definición del contenido de la clase
End Class
```

Veamos un sencillo ejemplo de creación de una clase Calculadora con la que podemos realizar operaciones básicas con dos números, como Sumar, Restar, Multiplicar y Dividir. Además de cuatro métodos para realizar las citadas operaciones, dispondrá de dos propiedades para almacenar los operandos con los que van a trabajar los métodos de la clase.

Esta sería la implementación en pseudocódigo de la clase Calculadora:

```
Class Calculadora
  Property operando1 Integer
  Property operando2 Integer
  Function Sumar()
    Return operando1+operando2
  End Function
  Function Restar()
    Return operando1-operando2
  End Function
  Function Multiplicar()
    Return operando1*operando2
  End Function
  Function Dividir()
    Return operando1/operando2
  End Function
End Class
```

Según se puede apreciar, los cuatro métodos se definen mediante el mismo tipo de funciones que estudiamos durante la programación modular, estas funciones operan con los datos almacenados en las propiedades, que no son más que variables a las que se les añade la palabra **property**



CREACIÓN DE OBJETOS

La clase define los métodos y propiedades, pero para poder hacer uso de los mismos se necesita un objeto sobre el que poder aplicarlos.

Para crear un objeto de una clase, utilizaremos un operador llamado **New**, que es el que se emplea en la mayoría de los lenguajes de programación. La manera de utilizarlo sería la siguiente:

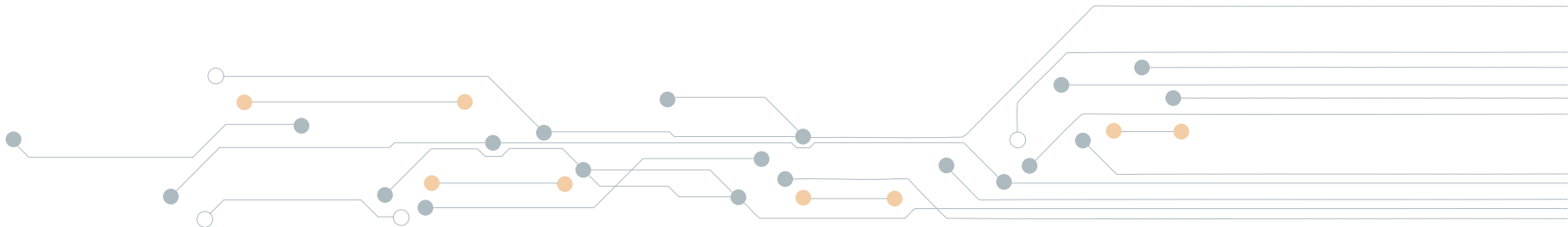
```
variable_objeto=New NombreClase()
```

Como vemos, a continuación del operador se indica el nombre de la clase de la que queremos crear el objeto, seguido de los paréntesis. Esta instrucción creará un objeto en memoria y lo guardará en la variable indicada a la izquierda del signo igual. Esta variable tendrá que ser declarada con el tipo de la clase.

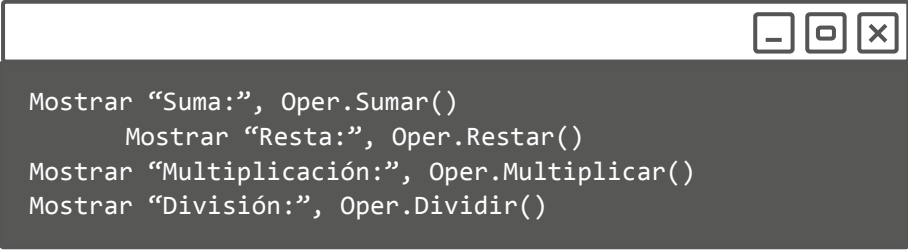
A partir de ahí, ya se puede utilizar la variable objeto para acceder a propiedades y métodos.

Para ilustrarlo con un ejemplo, vamos a hacer un programa principal que se encargue de leer dos números y, utilizando un objeto de la clase Calculadora definida anteriormente, muestre los resultados de las cuatro operaciones básicas sobre los números.

```
Inicio
  Datos:
    Oper Calculadora //define una variable del tipo de
    la clase,
                                //en ella se guardará el objeto
    N1, N2 Integer
    Sum, Res, Multi, Div Integer
  Código:
    Leer N1, N2
    Oper=New Calculadora() //crea el objeto
    //asigna valores leídos a las propiedades
    Oper.operando1=N1
    Oper.operando2=N2
    //llama a los métodos para realizar los cálculos
    Sum=Oper.Sumar()
    Res=Oper.Restar()
    Multi=Oper.Multiplicar()
    Div=Oper.Dividir()
    Mostrar "Suma:", Sum
    Mostrar "Resta:", Res
    Mostrar "Multiplicación:", Multi
    Mostrar "División:", Div
  Fin
```



En el ejemplo podríamos haber prescindido de las variables Sum, Res, Multi y Div:



```
Mostrar "Suma:", Oper.Sumar()  
Mostrar "Resta:", Oper.Restar()  
Mostrar "Multiplicación:", Oper.Multiplicar()  
Mostrar "División:", Oper.Dividir()
```

1.2 | Constructores

Hemos visto en el ejemplo anterior que para poder utilizar los métodos que realizan las operaciones es necesario primero establecer ciertas propiedades del objeto.

En muchos casos estas propiedades se establecen nada más crear el objeto, por lo que sería muy cómodo poder hacerlo en la misma instrucción de creación del objeto, en vez de tener que asignar explícitamente cada propiedad. Esto es posible hacer utilizando **constructores**.

Un constructor es un bloque de código que se define dentro de una clase y que se ejecuta en el momento en que se crea un objeto de dicha clase. Por tanto, es el lugar ideal para inicializar propiedades de los objetos.

Los constructores se definen de la siguiente manera:



```
NombreClase(parametro1, parametro2)
```

Como vemos, son parecidos a los procedimientos, pero sin la palabra Sub. Además, el nombre del constructor debe ser igual al de la clase.

Por ejemplo, si queremos definir la clase Calculadora utilizando constructor, quedaría de la siguiente manera:

```
Oper=New Calculadora(N1, N2)
```

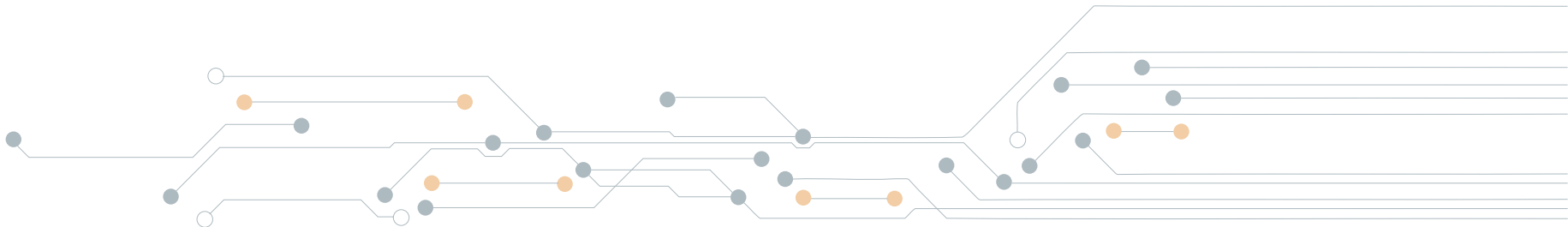
Y ya no sería necesario asignar explícitamente los valores N1 y N2 a las propiedades Operando1 y Operando2.

Una clase puede disponer de más de un constructor, lo que ofrecería la posibilidad de inicializar objetos de distinta forma. Todos los constructores tendrán el mismo nombre, que es el de la clase, pero deberán diferenciarse en el número de parámetros o en el tipo de los mismos.

Por ejemplo, podríamos incluir en la clase Calculadora dos constructores, uno para inicializar cada propiedad con un valor y otro que inicializaría las dos propiedades del objeto con el mismo valor:

```
Calculadora(op1 Integer, op2 Integer)  
    operando1=op1  
    operando2=op2  
End  
Calculadora(op Integer)  
    operando1=op  
    operando2=op  
End
```

Al hecho de poder definir más de un constructor en una clase se le conoce como **sobrecarga de constructores**.



Telefonica

EDUCACIÓN DIGITAL