¿Qué es una Clase en Java?

[Intermedio](https://javadesdecero.es/categoria/intermedio/) [POO](https://javadesdecero.es/categoria/poo/)[23 Comentarios](https://javadesdecero.es/poo/que-es-una-clase-ejemplos/#comments) por Alex Walton May 5, 2020

[ⓘ](https://www.pixfuture.com/advertisers/?id0348293521d)

Antes de que pueda seguir avanzando en su estudio de Java, necesita aprender sobre la clase. La clase es la esencia de Java. Es la base sobre la cual se construye todo el lenguaje Java porque **la clase define la naturaleza de un objeto**. Como tal, la clase forma la base para la programación orientada a objetos en Java.

Dentro de una clase se definen los datos y el código que actúa sobre esos datos. El código está contenido en métodos. Debido a que las clases, los objetos y los métodos son fundamentales para Java, los detallamos aquí. Tener una comprensión básica de estas características le permitirá escribir programas más sofisticados y comprender mejor ciertos elementos claves de Java que se describen más adelante.

Table de Contenido

* [1. Qué es una Clase](https://javadesdecero.es/poo/que-es-una-clase-ejemplos/#1_Que_es_una_Clase)
* [2. La forma general de una clase](https://javadesdecero.es/poo/que-es-una-clase-ejemplos/#2_La_forma_general_de_una_clase)
* [3. Definición de una Clase](https://javadesdecero.es/poo/que-es-una-clase-ejemplos/#3_Definicion_de_una_Clase)
* [4. ¿Cómo se crea un objeto?](https://javadesdecero.es/poo/que-es-una-clase-ejemplos/#4_%C2%BFComo_se_crea_un_objeto)
* [5. Creación de varios objetos](https://javadesdecero.es/poo/que-es-una-clase-ejemplos/#5_Creacion_de_varios_objetos)
* [6. Variables de referencia y asignación](https://javadesdecero.es/poo/que-es-una-clase-ejemplos/#6_Variables_de_referencia_y_asignacion)

1. Qué es una Clase

Dado que toda la actividad del programa Java ocurre dentro de una clase, hemos estado usando clases desde el comienzo de este curso. Por supuesto, solo se han utilizado clases extremadamente simples, y no hemos aprovechado la mayoría de sus características. Como verá, las clases son sustancialmente más poderosas que las limitadas presentadas hasta ahora.

**¿Qué es una clase?** Una clase es una plantilla que define la forma de un objeto. Especifica los datos y el código que operará en esos datos. Java usa una especificación de clase para construir objetos. Los objetos son instancias de una clase. Por lo tanto, **una clase es esencialmente un conjunto de planes que especifican cómo construir un objeto**.

Es importante tener claro que una clase es una abstracción lógica.

[×](https://javadesdecero.es/poo/que-es-una-clase-ejemplos/)**Nota**: No es hasta que se haya creado un objeto de esa clase, que existe una representación física de esa clase en la memoria.

[×](https://javadesdecero.es/poo/que-es-una-clase-ejemplos/)**Recuerda**: los métodos y variables que constituyen una clase se llaman **miembros de la clase**. Los miembros de datos también se conocen como **variables de instancia**.

2. La forma general de una clase

Cuando defines una clase, declaras su forma y naturaleza exactas. Para ello, especifique las variables de instancia que contiene y los métodos que operan en ellas. Aunque las clases muy simples pueden contener solo métodos o solo variables de instancia, la mayoría de las clases del mundo real contienen ambas.

[ⓘ](https://www.pixfuture.com/advertisers/?id0348293521d)

Una clase se crea al usar la palabra clave . Aquí se muestra una forma general simplificada de una definición de clase:

class NombreClase{

//Declarar variables de instancia

tipo var1;

tipo var2;

//..

//Declarar métodos

tipo metodo1(parámetros){

//Cuerpo del método

}

tipo metodo2(parámetros){

//Cuerpo del método

}

}

Aunque no existe una regla sintáctica que se deba cumplir, **una clase bien diseñada debería definir una y solo una entidad lógica**. Por ejemplo, una clase que almacena nombres y números de teléfono normalmente no almacenará información sobre el mercado de valores, el promedio de precipitaciones u otra información no relacionada. El punto aquí es que una clase bien diseñada agrupa la información conectada lógicamente. ¡**Poner información no relacionada en la misma clase des-estructurará rápidamente tu código**!

Hasta este punto, las clases que hemos estado utilizando tienen solo un método: . Pronto verás cómo crear otros. Sin embargo, observe que la forma general de una clase no especifica un método . Se requiere un método solo si esa clase es el punto de partida para su programa. Además, algunos tipos de aplicaciones Java no requieren un .

3. Definición de una Clase

Para ilustrar las clases, desarrollaremos una clase que encapsula información sobre vehículos, tales como automóviles, furgonetas y camiones. Esta clase se llama *vehículo*, y almacenará tres elementos de la información sobre un vehículo: el número de pasajeros que puede llevar, su capacidad de combustible, y la cantidad media de combustible consumido (en millas por galón).

Entonces, se define tres variables de instancia: *pasajeros, capacidad y mpg*. Observe que el vehículo no contiene ningún método. Por lo tanto, actualmente es una clase de sólo datos.

class Vehiculo{

int pasajeros; //número de pasajeros

int capacidad; //capacidad en galones

int mpg; //consumo de combustible en millas por galón

}

Una definición de clase crea un nuevo tipo de datos. En este caso, el nuevo tipo de datos se llama *Vehiculo*. Usará este nombre para declarar objetos de tipo *Vehiculo*. Recuerde que **una declaración de clase es sólo una descripción de tipo; no crea un objeto real**. Así, el código anterior no hace que ningún objeto de tipo Vehiculo entre en existencia.

4. ¿Cómo se crea un objeto?

Para crear realmente un objeto de vehículo, usará una instrucción como la siguiente:

Vehiculo minivan = new Vehiculo(); //Creando un objeto vehículo llamado minivan

Después de que se ejecute esta declaración, *minivan* se refiere a una instancia del vehículo. Por lo tanto, tendrá la realidad “**física**“. Por el momento, no se preocupe por los detalles de esta declaración.

Cada vez que se crea una instancia de una clase, se crea un objeto que contiene su propia copia de cada variable de instancia definida por la clase.

Así, cada objeto del vehículo contendrá sus propias copias de las variables de la instancia *pasajeros, capacidad, y mpg*. Para acceder a estas variables, se usará el operador *punto* (.). El operador *punto*vincula el nombre de un objeto con el nombre de un miembro. La forma general del operador *punto* se muestra aquí:

Objeto.Miembro

Por lo tanto, el objeto se especifica a la izquierda, y el miembro se pone a la derecha. Por ejemplo, para asignar la variable *capacidad* de *minivan* al valor *16*, utilice la siguiente instrucción:

minivan.capacidad = 16

En general, puede utilizar el operador *punto* para acceder a las variables de instancia y a los métodos. Aquí está un programa completo que utiliza la clase

/\* Un programa que usa la clase Vehiculo

El archivo se llama DemoVehiculo.java

\*/

class Vehiculo {

int pasajeros; //números de pasajeros

int capacidad; //capacidad del combustible en galones

int mpg; //combustible consumido en millas por galon

}

//Esta clase declara un objeto de tipo Vehiculo

class DemoVehiculo {

public static void main(String[] args) {

Vehiculo minivan = new Vehiculo();

int rango;

//asignando valores a los campos de minivan

minivan.pasajeros = 9;

minivan.capacidad = 15;

minivan.mpg = 20;

//Calcular el rango asumiendo un tanque lleno

rango = minivan.capacidad \* minivan.mpg;

System.out.println("La Minivan puede llevar " + minivan.pasajeros + " pasajeros con un rango de " + rango + " millas");

}

}

Salida:

La Minivan puede llevar 9 pasajeros con un rango de 300 millas

Debe llamar al archivo que contiene este programa **VehiculoDemo.java** porque el método *main()* está en la clase denominada *VehiculoDemo*, no en la clase denominada *Vehiculo*.

Cuando compile este programa, encontrará que se han creado dos archivos *.Class*, uno para *Vehiculo* y otro para *VehiculoDemo*. El compilador de Java pone automáticamente cada clase en su propio archivo *.class*. No es necesario que tanto *Vehiculo* como la clase *VehiculoDemo* estén en el mismo archivo fuente. Podría poner cada clase en su propio archivo, llamado *VehiculoDemo.java* y *VehiculoDemo.java*, respectivamente.

5. Creación de varios objetos

Antes de continuar, repasemos un principio fundamental: **cada objeto tiene sus propias copias de las variables de instancia definidas por su clase**. Así, el contenido de las variables en un objeto puede diferir del contenido de las variables en otro.

Lectura Recomendada

Introducción a Objetos

[IR AL ARTÍCULO](https://javadesdecero.es/poo/objetos-java/)

No hay conexión entre los dos objetos excepto por el hecho de que ambos son objetos del mismo tipo. Por ejemplo, si tiene dos objetos de *Vehiculo*, cada uno tiene su propia copia de *pasajeros, capacidad y mpg*, y **el contenido de éstos puede diferir entre los dos objetos**.

El siguiente programa demuestra este hecho. (Observe que la clase con se llama ahora DosVehiculos.)

//Este programa crea dos objetos Vehiculo

class Vehiculo {

int pasajeros; //números de pasajeros

int capacidad; //capacidad del combustible en galones

int mpg; //combustible consumido en millas por galon

}

//Esta clase declara un objeto de tipo Vehiculo

class DosVehiculo {

public static void main(String[] args) {

Vehiculo minivan = new Vehiculo();

Vehiculo sportscar = new Vehiculo();

int rango1, rango2;

//asignando valores a los campos de minivan

minivan.pasajeros = 9;

minivan.capacidad = 15;

minivan.mpg = 20;

//asignando valores a los campos de sportscar

sportscar.pasajeros = 10;

sportscar.capacidad = 25;

sportscar.mpg = 30;

//Calcular el rango asumiendo un tanque lleno

rango1 = minivan.capacidad \* minivan.mpg;

rango2 = sportscar.capacidad \* sportscar.mpg;

System.out.println("La Minivan puede llevar " + minivan.pasajeros + " pasajeros con un rango de " + rango1 + " millas");

System.out.println("El Sportscar puede llevar " + sportscar.pasajeros + " pasajeros con un rango de " + rango2 + " millas");

}

}

Salida:

La Minivan puede llevar 9 pasajeros con un rango de 300 millas

El Sportscar puede llevar 10 pasajeros con un rango de 750 millas

6. Variables de referencia y asignación

En una operación de asignación, las variables de referencia de objeto actúan de forma diferente a las variables de un tipo primitivo, como *int*.

* Cuando se asigna una variable de tipo primitivo a otra, la situación es sencilla. La variable de la izquierda recibe una copia del valor de la variable a la derecha.
* Al asignar una variable de referencia de objeto a otra, la situación es un poco más complicada porque está cambiando el objeto al que se refiere la variable de referencia. El efecto de esta diferencia puede causar algunos resultados contraintuitivos.

Por ejemplo, tenga en cuenta el siguiente fragmento:

Vehiculo auto1 = new Vehiculo();

Vehiculo auto2 = auto1;

A primera vista, es fácil pensar que *auto1* y *auto2* se refieren a diferentes objetos, pero este no es el caso. En su lugar, *auto1* y *auto2* se referirán al **mismo objeto**. La asignación de *auto1*a *auto2* simplemente hace que *auto2* haga referencia al mismo objeto que *auto1*. Así, el objeto puede ser actuado por cualquiera de *auto1* o*auto2*. Por ejemplo, después de la asignación

auto1.mpg = 28;

Si ejecuta, ambas instrucciones con *println()*:

System.out.println(auto1.mpg);

System.out.println(auto2.mpg);

¿Adivina qué? Se muestra el mismo valor: **28**.

Aunque *auto1* y*auto2* se refieren al mismo objeto, no están vinculados de ninguna otra manera. Por ejemplo, una asignación posterior a *auto2*simplemente cambia el objeto al que se refiere *auto2*. Por ejemplo:

Vehiculo auto1 = new Vehiculo();

Vehiculo auto2 = auto1;

Vehiculo auto3 = new Vehiculo();

auto2 = auto3; //Ahora auto2 y auto3 se refieren al mismo objeto

Una vez ejecutada esta secuencia, *auto2* se refiere al mismo objeto que *auto3*.

Si te ha quedado alguna duda por favor coméntale. ¡Hasta pronto!