**Variable Estática**

En la programación una variable estática es una variable que ha sido definida como constante, por lo que está pensado que a lo largo de la ejecución del programa, no va a cambiar su valor. Ejemplo de esto sería crear una variable “pi” (3.1415…) o “e” (2.7182…). Este tipo de variables se pueden establecer durante el tiempo de ejecución así como pueden ser manipuladas en distintas ocasiones durante la ejecución del programa. Este concepto también puede ser denominado como una variable global. Cabe destacar que estas se crean en las clases, por lo que los objetos de la clase la comparten, y no dejan de existir aun si los objetos son eliminados.

**Ciclo de vida de las variables**

Instancia (objeto)

* Existen cuando se crea un objeto que las contiene
* Se inicializan por defecto si no se hace de modo explicito

Estáticas (clase)

* Se crean cuando la clase se usa por primera vez
* Se inicializan por defecto si no se hace de modo explicito
* Suelen existir por el resto del programa

Locales (bloque)

* Creadas en la sentencia en la que están definidas
* No se inicializan por defecto, contienen datos imprevisibles
* Se destruyen al salir del bloque (llave final)

**Memoria Dinámica**

Es memoria que se reserva en tiempo de ejecución. Su principal ventaja frente a la estática, es que su tamaño puede variar durante la ejecución del programa. El uso de memoria dinámica es necesario cuando no conocemos el número de datos a tratar; sin embargo es algo más lento, ya que el tiempo ejecución depende del espacio que se va a usar.

Su tamaño y forma es variable a lo largo de un programa, por lo que se crean y destruyen en tiempo de ejecución. Esto permite dimensionar la estructura de datos de una forma precisa: se va asignando memoria en tiempo de ejecución según se va necesitando.

Este tipo de memoria se maneja mediante el uso de punteros por lo tanto resulta imposible conocer el tamaño de la memoria al momento de compilar. Cuando se desarrolla un programa el cual hace uso de la memoria dinámica, es necesario dividir el programa en cuatro partes: datos, texto, pila y la zona libre o heap. En el heap es donde queda la memoria libre para que esta sea utilizada como memoria dinámica.

**Clase**

La clase es el bloque de construcción fundamental de un lenguaje de orientado a objetos. Es un tipo abstracto de datos, cuya función principal es la instanciación de objetos, definiendo propiedades y métodos de los mismos. Es la “clasificación” de los elementos que se emplearan en el programa.

**Objeto**

Un objeto se define como una unidad dentro de un programa de la computadora que consiste en un estado y un comportamiento, suelen tener datos almacenados y de tareas realizables durante el tiempo de ejecución. El objeto se puede generan instanciando una clase, esto ocurre en la programación orientada objetos. Los objetos interactúan unos con otros. Los objetos son capaces de procesar datos recibir mensajes y generar los mismos para ser enviados a otros objetos. Un ejemplo podría ser una clase llamada “Vehículo” que crea 3 objetos llamados “bicicleta”, “coche” y “avión” respectivamente. O incluso una clase llamada “Coche” que crea objetos tipo coche llamados “mustang” y “challenger” con propiedades como “color” y “tipodemotor”, y métodos como “acelerar” y “frenar”.

**Instanciación**

Se llama instancia a todo objeto que derive de algún otro. De esta forma, todos los objetos son instancias de algún otro, si bien, decíamos que una clase es como la definición de un objeto, pero no es el objeto en sí, del modo como una idea no es una cosa física.

En un mismo proyecto se puede tener una o más instancias de una misma clase sin problemas.

Cada vez que creamos una nueva instancia, ésta adquiere las propiedades, métodos y eventos de la clase a la que pertenece, sin embargo, cada instancia es independiente de las otras; esto nos da dos ventajas:

1. Si hago algún cambio en la clase, todas las instancias de esta clase se actualizarán automáticamente; esto nos permite hacer cambios sin tener que ir a cada una de las instancias.
2. Al ser independientes de las otras instancias, puedo darles valores diferentes sin que afecten a las demás (como tener una silla negra, una roja, una más alta, etc.). Aunque comparten la misma estructura, pueden programarse individualmente, dando versatilidad y flexibilidad al código.

**Herencia**

Es la capacidad de las clases de heredar métodos y propiedades entre sí. De ahí surgen conceptos como Clase Padre y Clase Hija. La herencia funciona diferente entre lenguajes de programación, por ejemplo, en C es posible heredar de más de una clase a la vez, mientras que en Java, la herencia es lineal, al solo poder heredar de una clase. En la herencia, las clases hijas son capaces de realizar los mismos métodos (o sobrescribirlos) que tienen los padres, sin embargo, los padres no pueden hacer lo de los hijos. Ejemplo de esto podría ser con las clases árbol y manzano, donde todos los manzanos son arboles pero no todos los árboles son manzanos, por lo que no cualquier árbol te va a dar manzanas.

**Sobrecarga**

Es la capacidad de nombrar de la misma manera varios métodos, pero con diferente número de argumentos y por lo tanto la posibilidad de diferentes resultados. Por ejemplo un método suma que recibe 2 argumentos y te suma 2 números puede llevar el mismo nombre que otro método llamado suma que reciba 3 argumentos y sume 3 números.

**Shadowing**

En [la programación de computadoras,](https://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=search&rurl=translate.google.com.mx&sl=en&u=https://en.wikipedia.org/wiki/Computer_programming&usg=ALkJrhiAGIaBfyp87Tybt59-9ipCRy4r1w) sombras variables se produce cuando una variable declarada dentro de un determinado [ámbito](https://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=search&rurl=translate.google.com.mx&sl=en&u=https://en.wikipedia.org/wiki/Scope_(computer_science)&usg=ALkJrhgAiJXFHwZb4BcBtRhM8ADEXEVfCA) (bloque de decisión, el método o clase interna) tiene el mismo nombre que una variable declarada en un ámbito exterior. A nivel de los identificadores (nombres, en lugar de variables), esto se conoce como [nombre de enmascaramiento.](https://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=search&rurl=translate.google.com.mx&sl=en&u=https://en.wikipedia.org/wiki/Name_resolution_(programming_languages)&usg=ALkJrhhlw4zfkQ5gzvXJreQ7O2uN6HK2eA#Name_masking) Se dice que la variable externa está a la sombra de la variable interna, mientras que el identificador interno se sirve para enmascarar el identificador exterior.

Uno de los primeros lenguajes de introducir sombras variables fue [ALGOL,](https://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=search&rurl=translate.google.com.mx&sl=en&u=https://en.wikipedia.org/wiki/ALGOL&usg=ALkJrhg2sYqC35IBbzdi9viy_ml3K3b2KQ) que introdujo por primera vez bloques de establecer ámbitos. También se le permitió por muchos de los lenguajes de programación derivados, incluyendo C++ y [Java.](https://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=search&rurl=translate.google.com.mx&sl=en&u=https://en.wikipedia.org/wiki/Java_(programming_language)&usg=ALkJrhgyvyd2wQBdBUGSQzi9AdZzq3dj5A)