Conceptos.

1. **Variable Estática:**

Es una [variable](https://es.wikipedia.org/wiki/Variable_(programaci%C3%B3n)) que ha sido ubicada estáticamente y cuyo tiempo de vida se extiende durante toda la [ejecución](https://es.wikipedia.org/w/index.php?title=Ejecuci%C3%B3n_(inform%C3%A1tica)&action=edit&redlink=1) del [programa](https://es.wikipedia.org/wiki/Programa_inform%C3%A1tico). Los valores de variables estáticas se pueden establecer una vez (durante el [tiempo de ejecución](https://es.wikipedia.org/wiki/Tiempo_de_ejecuci%C3%B3n)) o se pueden cambiar en múltiples ocasiones durante la ejecución del programa. La terminología "variable estática" se basa en [C](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C) y [C++](https://es.wikipedia.org/wiki/C%2B%2B), pero también se usa en muchos lenguajes de programación derivados. En lenguajes de diferente origen el mismo concepto puede denominarse "[variable global](https://es.wikipedia.org/wiki/Variable_global)".

Los [programas informáticos](https://es.wikipedia.org/wiki/Programa_inform%C3%A1tico) pueden almacenar [constantes](https://es.wikipedia.org/wiki/Constante_(programaci%C3%B3n)) en variables constant o en variables static, dependiendo de las posibilidades del [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n). En el [lenguaje de programación C](https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n)) se usa static con variables globales y funciones para restringir su ámbito al archivo donde se definen.

1. **Ciclos de vida de las variables:**

* Variables de instancia: Se crean cuando se crea el objeto que las contiene.

–Se inicializan por defecto si no se hace de modo explicito

* Variables estáticas: Se crean cuando la clase se usa por primera vez.

–Se inicializan por defecto si no se hace de modo explicito

– Suelen existir para el resto del programa (salvo que no esté cargado).

* Variables locales: Creadas en la sentencia en la que están definidas.

–No se inicializan por defecto. Contienen datos imprevisibles. – Se destruyen al salir del bloque (en la llave final).

1. **Memoria Dinámica:**

La memoria dinámica se refiere a aquella [memoria](https://es.wikipedia.org/wiki/Memoria_(inform%C3%A1tica)) que no puede ser definida ya que no se conoce o no se tiene idea del número de la variable a considerarse, la solución a este problema es la memoria dinámica que permite solicitar memoria en [tiempo de ejecución](https://es.wikipedia.org/wiki/Tiempo_de_ejecuci%C3%B3n), por lo que cuanta más memoria se necesite, más se solicita al sistema operativo. El sistema operativo maneja la memoria gracias al uso de [punteros](https://es.wikipedia.org/wiki/Puntero_(inform%C3%A1tica)), por la misma naturaleza del proceso nos impide conocer el tamaño de la memoria necesaria en el momento de [compilar](https://es.wikipedia.org/wiki/Compilaci%C3%B3n).

Un dato importante es que como tal este tipo de datos se crean y se destruyen mientras se ejecuta el programa y por lo tanto la estructura de datos se va dimensionando de forma precisa a los requerimientos del programa, evitándonos así perder datos o desperdiciar memoria si hubiéramos tratado de definir la cantidad de memoria a utilizar en el momento de compilar el programa.

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.( En C, el programador es encargado liberar esta memoria cuando no la utilice más). El uso de memoria dinámica es necesario cuando no conocemos el número de datos, elementos a tratar; sin embargo es algo más lento, ya que el tiempo de ejecución depende del espacio que se va ha usar.

1. **Clase:**

Una **clase** es una plantilla para la creación de [objetos de datos](https://es.wikipedia.org/wiki/Objeto_(programaci%C3%B3n)) según un modelo predefinido. Las clases se utilizan para representar entidades o conceptos, como los [sustantivos](https://es.wikipedia.org/wiki/Sustantivo) en el lenguaje. Cada clase es un modelo que define un conjunto de [variables](https://es.wikipedia.org/wiki/Variable_(programaci%C3%B3n)) (-el estado, y [métodos](https://es.wikipedia.org/wiki/M%C3%A9todo_(inform%C3%A1tica)) apropiados para operar con dichos datos -el comportamiento). Cada objeto creado a partir de la clase se denomina [instancia](https://es.wikipedia.org/wiki/Instancia_(inform%C3%A1tica)) de la clase.

Las clases son un pilar fundamental de la [programación orientada a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos). Permiten [abstraer](https://es.wikipedia.org/wiki/Abstracci%C3%B3n_(inform%C3%A1tica)) los datos y sus operaciones asociadas al modo de una [caja negra](https://es.wikipedia.org/wiki/Caja_negra_(sistemas)). Los lenguajes de programación que soportan clases difieren sutilmente en su soporte para diversas características relacionadas con clases. La mayoría soportan diversas formas de [herencia](https://es.wikipedia.org/wiki/Herencia_(programaci%C3%B3n_orientada_a_objetos)). Muchos lenguajes también soportan características para proporcionar [encapsulación](https://es.wikipedia.org/wiki/Encapsulamiento_(inform%C3%A1tica)), como especificadores de acceso.

Una clase también puede tener una representación (meta objeto) en tiempo de ejecución, que proporciona apoyo en tiempo de ejecución para la manipulación de los metadatos relacionados con la clase.

1. **Objeto:**

Es una unidad dentro de un [programa de computadora](https://es.wikipedia.org/wiki/Programa_(computaci%C3%B3n)) que consta de un estado y de un comportamiento, que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución. Un objeto puede ser creado [instanciando](https://es.wikipedia.org/wiki/Instancia_(programaci%C3%B3n)) una [clase](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)), como ocurre en la programación orientada a objetos, o mediante escritura directa de código y la replicación otros objetos, como ocurre en la [programación basada en prototipos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_basada_en_prototipos).

Estos objetos interactúan unos con otros, en contraposición a la visión tradicional en la cual un programa es una colección de [subrutinas](https://es.wikipedia.org/wiki/Subrutinas) ([funciones](https://es.wikipedia.org/wiki/Funci%C3%B3n_(programaci%C3%B3n)) o [procedimientos](https://es.wikipedia.org/wiki/Procedimiento_(Programaci%C3%B3n))), o simplemente una lista de [instrucciones](https://es.wikipedia.org/wiki/Instrucci%C3%B3n_(inform%C3%A1tica)) para el computador. Cada objeto es capaz de recibir [mensajes](https://es.wikipedia.org/wiki/Mensajes), procesar datos y enviar [mensajes](https://es.wikipedia.org/wiki/Mensajes) a otros objetos de manera similar a un [servicio](https://es.wikipedia.org/wiki/Servicio).

Objeto consiste en ordenar datos en conjuntos modulares de elementos de información del mundo real (denominado un dominio). Estos elementos de datos se llaman objetos. Estos datos se agrupan de acuerdo a las características principales del mundo real de estos elementos (tamaño, color, etc.).

1. **Instanciación:**

Una instancia (en [inglés](https://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s), instance) se refiere a una realización específica de una [clase](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)) o [prototipo](https://es.wikipedia.org/wiki/Prototipo#Desarrollo_Orientado_a_Prototipos) determinados.

Es la realización de una clase o prototipo determinados. En el momento que un programa es ejecutado, esto significa que el programa se instancia. Un objeto también se puede definir como una instancia dentro de la clase. En otras palabras se refiera que un miembro de la clase cuenta con atributos en vez de variables.

1. **Herencia:**

Es una propiedad que permite que los objetos sean creados a partir de otros ya existentes, obteniendo características (métodos y atributos) similares a los ya existentes. Es la relación entre una clase general y otra clase mas especifica. Es un mecanismo que nos permite crear clases derivadas a partir de clase base, Nos permite compartir automáticamente métodos y datos entre clases subclases y objetos. Es la relación entre una clase general y otra clase más específica.

Es uno de los mecanismos de los lenguajes de [programación orientada a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos) basados en clases, por medio del cual una [clase](https://es.wikipedia.org/wiki/Clase_(programaci%C3%B3n_orientada_a_objetos)) se deriva de otra de manera que extiende su funcionalidad. La clase de la que se hereda se suele denominar clase base, clase padre, superclase, clase ancestro (el vocabulario que se utiliza suele depender en gran medida del lenguaje de programación).

1. **Sobrecarga:**

Se refiere a la posibilidad de tener dos o mas funciones con el mismo nombre pero funcionalidad diferente. Es decir dos o mas funciones con el mismo nombre realizan acciones diferentes. El [compilador](http://www.diclib.com/compilador/show/es/es_wiki_10/71910) usara una u otra dependiendo de los parámetros usados. a esto se llama también sobrecarga de funciones.

Es la capacidad de un lenguaje de programación, que permite nombrar con el mismo identificador diferentes variables u operaciones.

La sobrecarga se refiere a la posibilidad de tener dos o más [funciones](https://es.wikipedia.org/wiki/Funci%C3%B3n_(programaci%C3%B3n)) con el mismo nombre pero funcionalidad diferente. Es decir, dos o más funciones con el mismo nombre realizan acciones diferentes.

1. **Ensombrecimiento (Shadowing)**

Al método de shadowing se le conoce cuando una variable de tipo miembro y una variable local definida en un método, ambas reciban el mismo nombre.

1. **Variables:**

* Variables de Clase: especifican cómo serán los objetos de dicha clase, esto es, de que variables y de que métodos constarán.   Indica el nombre de la clase precedido por la palabra clave class.
* Variables de Método: tipo de variable independiente que se manipula o mide para comprobar si modifica la relación entre la variable independiente y la variable dependiente.
* Variables Estáticas: Son propias únicamente de la clase y no de los objetos que pueden crearse de la misma, por lo tanto, sus valores son compartidos por todos los objetos de la clase. Van precedidas del modificador static.

# Bibliografía

Joyanes, L., & Zahonero, I. (2010). *Programacion en C, C++, Java y UML.* McGraw Hill.

Sandi Metz. 2012. Practical Object‐Oriented Design in Ruby: An Agile Primer. Addison‐Wesley Professional.