**Axel Bravo Carreón ID: 148182 Tarea: INVESTIGACION DE CONCEPTOS**

**Variables Estáticas**

En informática una **variable estática** es una variable 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). Normalmente una variable estática tiene un [ámbito](https://es.wikipedia.org/wiki/%C3%81mbito_(programaci%C3%B3n)) más amplio que otras variables. 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)".

**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.

**Clase**

En [informática](https://es.wikipedia.org/wiki/Inform%C3%A1tica), 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.

**Objeto**

**E**n el [paradigma de programación](https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n) [orientada a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos) (POO, o bien OOP en [inglés](https://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s)), un **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.

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.

**Instanciación**

La palabra instancia significa: solicitud o instancia.  
Una instancia de un programa es una copia de una versión ejecutable del programa que ha sido escrito en la memoria del computador  
  
una instancia de un programa es creada típicamente por el click del usuario en un icono de interfaz gráfico para usuario.

**Herencia**

En [programación orientada a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos), la **herencia** es, después de la agregación o composición, el mecanismo más utilizado para alcanzar algunos de los objetivos más preciados en el desarrollo de software como lo son la reutilización y la extensibilidad. A través de ella los diseñadores pueden crear nuevas clases partiendo de una clase o de una jerarquía de clases preexistente (ya comprobadas y verificadas) evitando con ello el rediseño, la modificación y verificación de la parte ya implementada. La herencia facilita la creación de objetos a partir de otros ya existentes e implica que una subclase obtiene todo el comportamiento (métodos) y eventualmente los atributos (variables) de su superclase.

Es la relación entre una clase general y otra clase más específica. Por ejemplo: Si declaramos una clase párrafo derivada de una clase texto, todos los métodos y variables asociadas con la clase texto, son automáticamente heredados por la subclase párrafo.

**Sobrecarga**

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

En [programación orientada a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos) 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. El [compilador](https://es.wikipedia.org/wiki/Compilador) usará una u otra dependiendo de los [parámetros](https://es.wikipedia.org/wiki/Argumento_(inform%C3%A1tica)) usados. A esto se llama también sobrecarga de funciones.

También existe la sobrecarga de operadores que al igual que con la sobrecarga de funciones se le da más de una implementación a un operador.

El mismo método dentro de una clase permite hacer cosas distintas en función de los parámetros.

Java no permite al programador implementar sus propios operadores sobrecargados, pero sí utilizar los predefinidos como el +. C++, por el contrario si permite hacerlo.

**Shadowing**

Se llama shadowing al hecho de que en una clase una variable miembro y una variable local definida en un método miembro, se llamen igual.

**Ciclo de vida de las variables**

Las variables tienen un ciclo de vida: 1. En la declaración se reserva el espacio necesario para que se puedan comenzar a utilizar (digamos que se avisa de su futura existencia) 2. Se le asigna su primer valor (la variable nace) 3. Se la utiliza en diversas sentencias (no se debe leer su contenido sin haberla asignado ese primer valor). 4. Cuando finaliza el bloque en el que fue declarada, la variable muere. Es decir, se libera el espacio que ocupa esa variable en memoria. No se la podrá volver a utilizar

**Ciclos de vida de las variables**

Variables de instancia (u objeto):

– Se crean cuando se crea el objeto que las contiene.

– Se inicializan por defecto si no se hace de modo explícito: • 0 para números, "false" para booleano, "null" para objetos.

– Se destruyen cuando el recolector de basura de Java no encuentra referencias activas para el objeto.

Variables estáticas (o de clase):

– Se crean cuando la clase se usa por primera vez.

– Se inicializan por defecto si no se hace de modo explícito: • 0 para números, "false" para booleano, "null" para objetos

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

• Variables locales (o de bloque):

– 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).