UNIVERSIDAD DE LAS AMÉRICAS PUEBLA



DEPARTAMENTO DE ELECTRÓNICA, MECATRÓNICA Y SISTEMAS

LENGUAJES Y PARADIGMAS DE LA PROGRAMACIÓN

M. JUAN CARLOS GALÁN

“TAREA 3: CONCEPTOS DE PROGRAMACIÓN”

ALAN PÉREZ CASTILLO

150294

SECCIÓN 1

SAN ANDRÉS CHOLULA, A 01 DE OCUBRE DE 2015

**Variables estáticas**

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). 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 (lenguaje c)**

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, 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 se hubiese tratado de definir la cantidad de memoria a utilizar en el momento de compilar el programa.

**Clase**

Lenguajes [orientados a objetos](https://es.wikipedia.org/wiki/Lenguaje_orientado_a_objetos) usan clases y [objetos](https://es.wikipedia.org/wiki/Objeto_(programaci%C3%B3n)). En este caso, una variable de clase es una variable que no está asociada con instancias de la clase. Sólo hay una única copia de la variable que se comparte entre los métodos de todas las instancias independientemente de su cantidad. En C++, las variables de clase se conocen como miembros de datos estáticos.

**Objeto**

Es el resultado de la instanciación de una [clase](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)). Una [clase](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)) es el anteproyecto que ofrece la funcionalidad en ella definida, pero ésta queda implementada sólo al crear una [instancia](https://es.wikipedia.org/wiki/Instancia_(programaci%C3%B3n)) de la [clase](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)), en la forma de un objeto. Por ejemplo: dado un plano para construir sillas (una clase de nombre clase\_silla), entonces una silla concreta, en la que podemos sentarnos, construida a partir de este plano, sería un objeto de clase\_silla. Es posible crear (construir) múltiples objetos (sillas) utilizando la definición de la clase (plano) anterior. Los conceptos de clase y objetos son análogos a los de [tipo de datos](https://es.wikipedia.org/wiki/Tipo_de_datos) y [variable](https://es.wikipedia.org/wiki/Variable_(programaci%C3%B3n)); es decir, definida una clase podemos crear objetos de esa clase, igual que disponiendo de un determinado tipo de dato.

**Instanciación**

En [lenguajes](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) que crean objetos a partir de clases, un objeto es una instancia de una clase. Esto es, un miembro de una clase que tiene [atributos](https://es.wikipedia.org/wiki/Atributo_(inform%C3%A1tica)) en lugar de variables. En un contexto del mundo real, podríamos pensar en "Perro" como una clase y en un perro concreto es una instancia de esta clase.1 En este caso no nos importa la raza del perro. Si fuese de nuestro interés modelarla, y diferenciásemos entre un dóberman y un chihuahua, no solo cada instancia sería diferente, sino que pertenecerían a clases o prototipos diferentes.

**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.

La herencia 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).

En los lenguajes que cuentan con un [sistema de tipos](https://es.wikipedia.org/wiki/Sistema_de_tipos) fuerte y estrictamente restrictivo con el tipo de datos de las variables, la herencia suele ser un requisito fundamental para poder emplear el [Polimorfismo](https://es.wikipedia.org/wiki/Polimorfismo_(programaci%C3%B3n_orientada_a_objetos)), al igual que un mecanismo que permita decidir en tiempo de ejecución qué método debe invocarse en respuesta a la recepción de un mensaje, conocido como enlace tardío (late binding) o enlace dinámico (dynamic binding).

**Sobrecarga**

En [lenguajes](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) que crean objetos a partir de clases, un objeto es una instancia de una clase. Esto es, un miembro de una clase que tiene [atributos](https://es.wikipedia.org/wiki/Atributo_(inform%C3%A1tica)) en lugar de variables. En un contexto del mundo real, podríamos pensar en "Perro" como una clase y en un perro concreto es una instancia de esta clase.1 En este caso no nos importa la raza del perro. Si fuese de nuestro interés modelarla, y diferenciásemos entre un dóberman y un chihuahua, no solo cada instancia sería diferente, sino que pertenecerían a clases o prototipos diferentes, c.f. [herencia (informática)](https://es.wikipedia.org/wiki/Herencia_(inform%C3%A1tica)).

**Ensombrecimiento (shadowing)**

Se le llama shadowing al efecto de llevar el mismo nombre a una variable de clase con una variable local definida, donde el acceso o instancia de la variable puede cambiar de resultado cuando se lleva a cabo la ejecución en orden. Para prevenir este tipo de problemas, se debe utilizar la palabra “this” seguido de la variable de la clase para que el apuntador haga referencia a la variable de instancia.

**Ciclo de vida de:**

**Variable de clase:**

1. Pertenece la variable a la clase, donde se define si la variable será pública, privada, vacía o global dentro de la clase.

**Variable de método**

1. Pertenece a la variable del método, donde éste recibe un valor para que se lleve la ejecución a cabo.

**Variables estáticas**

1. Es una variable donde el valor que sea asignado, es único y no puede ser cambiado por otro valor.

**Variables de instancia (objeto):**

1. Se crean cuando se crea el objeto que las contiene.
2. Se inicializan por defecto si no se hace de modo explícito: • 0 para números, "false" para booleano, "null" para objetos.
3. Se destruyen cuando el recolector de basura no encuentra referencias activas para el objeto.

**Variables estáticas (clase):**

1. Se crean cuando la clase se usa por primera vez.
2. Se inicializan por defecto si no se hace de modo explícito: • 0 para números, "false" para booleano, "null" para objetos
3. Suelen existir para el resto del programa (salvo que no esté cargado).
4. Variables locales (odebloque):  
   Creadas en la sentencia en la que están definidas.  
   No se inicializan por defecto. Contienen datos imprevisibles.
5. Se destruyen al salir del bloque (en la llave final).

**Variables locales**

Existen desde el punto de definición hasta el final del bloque, donde estos se definen mediante llaves “{ }”. Los bloques se suelen utilizar para definir:

El cuerpo del método

1. Sentencias múltiples en operaciones de tipo if-else o en forma de bucles
2. Las variables locales solo existen dentro del propio método.
3. No se da acceso a ningún otro método
4. Si la variable local y de instancia tiene el mismo nombre, la variable local oculta a la de instancia

**Las variables y métodos de instancia y estáticos (en una clase) tienen 4 modificadores de acceso:**

1. Prívate: acceso solo a los métodos de su propia clase • Los campos de datos casi siempre tienen que ser privados
2. Public: acceso a todos los métodos y clases. Los métodos que van a utilizar otras clases son públicos. Los métodos solo para uso interno son privados
3. Package: acceso a los métodos de las clases del mismo paquete (un paquete es un grupo de clases). Este es el valor por defecto. Especifique siempre el ámbito. No hay palabra clave 'package', es el valor por defecto sin clave
4. Protected: utilizado con herencia • Es como una variable privada.

**Referencias bibliográficas**

* http://www.javatutoriales.com/2010/10/sun-certified-java-programmer-6-cx-310\_10.html
* http://profesores.fi-b.unam.mx/carlos/java/java\_basico2\_6.html
* https://www.fdi.ucm.es/profesor/jpavon/poo/1.1.Objetos%20y%20Clases.pdf
* http://www.edu4java.com/es/conceptos/clases-instancias-prototipos-herencia-poo.html
* http://elvex.ugr.es/decsai/builder/intro/5.html