**Guillermo Alonso Bernal Martínez 148417**

**Paradigmas de la Programación**

**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 (salvo que este no se haya cargado)

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)

**Variable Estática**

Una variable estática es una variable que ha sido definida al principio del programa de manera estática y cuya existencia es útil durante toda la ejecución del programa. 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.

**Memoria Dinámica**

La memoria dinámica es la memoria que no se define dentro de un programa, se desconoce la cantidad que debe considerarse. Únicamente se permite solicitar memoria al mismo tiempo de ejecución. Conforme avance el programa y se vaya requiriendo mas memoria se le ira solicitando al sistema operativo de acorde a lo necesario. Este tipo de memoria se maneja mediante el uso de punteros por lo tanto es 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. La zona libre es donde se ubica la memoria libre para que esta sea utilizada como memoria dinámica.

**Clase**

La clase es una plantilla la cual permite la creación de objetos. Estas son utilizadas como medio de representación de conceptos o entidades. Cada clase busca definir un conjunto de variables. Los objetos creados partir de una clase son denominados instancias de la clase. Las clases son fundamentales para la funcionalidad de la programación orientada a objetos. Las clases están conformadas por campos de datos, métodos o subrutinas, en algunos lenguajes también se manejan las propiedades entre los campos y métodos.

**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. Gracias a las clases se crean los objetos que pertenecen a esa clase.

**Instanciación**

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.

**Herencia**

La herencia en la programación orientada objetos es utilizada para poder crear nuevas clases dentro de un programa tomando como base o referencia una clase o jerarquía de clases que ya existieron antes del programa. La herencia permite la creación de objetos a partir de otros que ya fueron previamente creados, de tal manera la subclase obtiene todos los métodos y variables de la superclase de la cual es heredada.

**Overload**

La sobrecarga en el lenguaje de programación se refiere a la condición que permite nombrar con el mismo identificador a diferentes variables u operaciones. El Overload en la programación orientada a objetos es cuando dos o más funciones usan el mismo nombre. El programa identifica que función llamar o realizar debido a los parámetros que fueron previamente indicados. Esta también puede funcionar con los operadores de la misma manera con que con las funciones.

**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.