Francisco Javier Zaragoza Zepeda ID 151693

Investigación 2do examen

**Variables** **Estáticas**

Una variable estática es una variable que ha sido ubicada estáticamente y cuyo tiempo de vida se extiende durante toda la ejecución del programa. Normalmente una variable estática tiene un ámbito más amplio que otras variables. Los valores de variables estáticas se pueden establecer una vez (durante el tiempo de ejecución) o se pueden cambiar en múltiples ocasiones durante la ejecución del programa. La terminología "variable estática" se basa en C y C++, pero también se usa en muchos lenguajes de programación derivados. En lenguajes de diferente origen el mismo concepto puede denominarse "variable global".

**Ciclo de vida de las variables**

Las variables de instancia 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 no encuentra referencias activas para el objeto.

Las 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).

Las variables locales (o de bloque) son 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).

**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. (En C, el programador es encargado de liberar esta memoria cuando no la utilice más). El uso de memoria dinámica es necesario cuando a priori no conocemos el número de datos/elementos a tratar.

**Clases**

Una clase es una construcción que permite crear tipos personalizados propios mediante la agrupación de variables de otros tipos, métodos y eventos. Una clase es como un plano. Define los datos y el comportamiento de un tipo. Si la clase no se declara como estática, el código de cliente puede utilizarla mediante la creación de objetos o instancias que se asignan a una variable. La variable permanece en memoria hasta que todas las referencias a ella están fuera del ámbito. En ese momento, CLR la marca como apta para la recolección de elementos no utilizados.

**Objeto**

Una definición de clase o struct es como un plano que especifica qué puede hacer el tipo. Un objeto es básicamente un bloque de memoria que se ha asignado y configurado en función del plano. Un programa puede crear muchos objetos de la misma clase. Los objetos también se denominan instancias y se pueden almacenar en una variable con nombre o en una matriz o colección.

**Instanciación**

La instanciación es la creación de una instancia.

Una instancia de un programa es una copia de una versión ejecutable del programa que ha sido escrito en la memoria del computador.

**Herencia**

En programación 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.

**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 la sobrecarga se refiere a la posibilidad de tener dos o más funciones con el mismo nombre pero funcionalidad diferente. Es decir, dos o más funciones con el mismo nombre realizan acciones diferentes. El compilador usará una u otra dependiendo de los parámetros 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.

**Ensombrecimiento (Shadowing)**

El ensombrecimiento de variable ocurre cuando una variable declarada dentro de un método o clase interna tiene el mismo nombre que una variable declarada fuera de ese método.