PARADIGMAS DE PROGRAMACION

programming paradigms

Autor 1: Angie Paola Ramírez Villada Autor 2: Brayan de Jesús Orrego
Risaralda, Tecnológica, Pereira, Colombia
Correo-e: paola.ramirez2@utp.edu.co
Correo-e: b.orrego1@utp.edu.co

***Resumen*—Paradigmas; consiste en un método para llevar acabo cómputos y formas en las que debe estructurarse y tareas que debe realizar un programa, es decir un modelo para realizar problemas también se utilizan los llamados lenguajes de programación necesariamente se encuadran en uno o varios paradigmas a la vez a partir del tipo de órdenes que permiten implementar algo que tiene una relación directa con una sintaxis.**

- **Imperativo. Los programas se componen de un conjunto de sentencias que cambian su estado. Son secuencias de comandos que ordenan acciones a la computadora.**
- **Declarativo. Opuesto al imperativo. Los programas describen los resultados esperados sin listar explícitamente los pasos a llevar a cabo para alcanzarlos.**
- **Lógico. El problema se modela con enunciados de lógica de primer orden.**
- **Funcional. Los programas se componen de funciones, es decir, implementaciones de comportamiento que reciben un conjunto de datos de entrada y devuelven un valor de salida.**
- **Orientado a objetos. El comportamiento del programa es llevado a cabo por objetos, entidades que representan elementos del problema a resolver y tienen atributos y comportamiento.**

Otros son de aparición relativamente reciente y no forman parte del grupo principal:

- **Dirigido por eventos. El flujo del programa está determinado por sucesos externos (por ejemplo, una acción del usuario).**
- **Orientado a aspectos. Apunta a dividir el programa en módulos independientes, cada uno con un comportamiento bien definido.**

***Palabras clave*— Algoritmos, computadoras, lenguajes, programación**
Algorithms, computers, languages, programming

Fecha de Recepción: (Letra Times New Roman de 8 puntos)
Fecha de Aceptación: Dejar en blanco

I. INTRODUCCIÓN

Hablaremos de los paradigmas de programación. ¿Qué es?, sus ventajas, desventajas, lenguajes y daremos algunos ejemplos.

¿Qué es?

Un paradigma de programación es un modelo básico de diseño y desarrollo de programas, que permite producir programas con un conjunto de normas específicas, tales como: estructura modular, fuerte cohesión, alta rentabilidad, etc.

Los paradigmas pueden ser considerados como patrones de pensamiento para la resolución de problemas. Desde luego siempre teniendo en cuenta los lenguajes de programación, según nuestro interés de estudio.

No es mejor uno que otro sino que cada uno tiene ventajas y desventajas. También hay situaciones donde un paradigma resulta más apropiado que otro.

Hay multitud de ellos atendiendo a alguna particularidad metodológica o funcional Cuando un lenguaje refleja bien un paradigma particular, se dice que soporta el paradigma, y en la práctica un lenguaje que soporta correctamente un paradigma, es difícil distinguirlo del propio paradigma, por lo que se identifica con él.

PROGRAMACIÓN FUNCIONAL

Características

- Los programas escritos en un lenguaje funcional están constituidos únicamente por definiciones de funciones
- La no existencia de asignaciones de variables y la falta de construcciones estructuradas como la secuencia o la iteración
- Existen dos grandes categorías de lenguajes funcionales: los funcionales *puros* y los *híbridos*
- En contraste, los lenguajes funcionales puros tienen una mayor potencia expresiva, conservando a la vez su transparencia referencial. [1]

Ventajas

- Ausencia de efectos colaterales
- Proceso de depuración menos problemático
- Pruebas de unidades más confiables
- Mayor facilidad para la ejecución concurrente

Desventajas

- Falta de estandarización
- Bajo rendimiento de los programas

PROGRAMACIÓN ESTRUCTURADA

Características

- La estructura de los programas es clara, puesto que las instrucciones están más ligadas o relacionadas entre sí.
- Los programas son más fáciles de entender, pueden ser leídos de forma secuencial.
- Un programa escrito de acuerdo a los principios de programación estructurada no solamente tendrá una mejor estructura sino también una excelente presentación.

Ventajas

- Reducción de los costos de mantenimiento
- Reducción del esfuerzo en las pruebas y depuración
- Los bloques de código son casi auto-explicativos, lo que reduce y facilita la documentación.
- Se incrementa el rendimiento de los programadores

Desventajas

- El principal inconveniente de este método de programación es que se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar problemático el manejo de su código fuente
- Se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar difícil su manejo

PROGRAMACIÓN MODULAR

Características

- Este paradigma también se conoce como principio de ocultación de procedimientos y datos

- Consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable.
- Se presenta históricamente como una evolución de la programación estructurada para solucionar problemas de programación más grandes y complejos de lo que ésta puede resolver.

Ventajas

- Al aplicar la programación modular, un problema complejo debe ser dividido en varios subproblemas más simples, y estos a su vez en otros subproblemas más simples.
- En caso de que un módulo necesite de otro, puede comunicarse con éste mediante una interfaz de comunicación que también debe estar bien definida.
- Es fácil de mantener y modificar
- Es más fácil de escribir y depurar
- Facilidad de controlar es decir descompone un problema en estructuras jerárquicas, de modo que se puede considerar cada estructura desde dos puntos de vista

Desventajas

- No se dispone de algoritmos formales de modularidad, por lo que a veces los programadores no tienen claras las ideas de los módulos
- La programación modular requiere más memoria y tiempo de ejecución

ABSTRACCIÓN DE DATOS

Características

- Proporciona un conjunto completo de operaciones válidas y útiles para cada tipo de dato
- La abstracción de datos se logra mediante los tipos de datos que suministra el lenguaje y los subprogramas que van a implementar las operaciones permitidas
- Se encapsula la representación interna de un dato junto con las implementaciones de todas las operaciones que se pueden realizar con ese dato.

Ventajas

- Produce código reutilizable
- Favorece la ausencia de errores, al reutilizar código ya probado y forzar a utilizar la estructura de datos correctamente
- Aumenta la facilidad de uso y la legibilidad del código
- Favorece la extensibilidad del código
- Facilita y hace más rápido el desarrollo de aplicaciones

Desventajas

- Las operaciones requieren mayor tiempo de procesamiento
- En la mayoría no existen herramientas predefinidas, por lo tanto es el programador quien implementa las estructuras
- Al agregar un nuevo elemento en algún lugar que no sea el último, se tiene que desplazar los elementos hacia abajo que están después de la posición de inserción deseada logrando que se genere un espacio para poder agregar un nuevo elemento

PROGRAMACIÓN ORIENTADA A OBJETOS

Características

- Abstracción: denota las características esenciales de un objeto, donde se capturan sus comportamientos
- Encapsulamiento: significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción
- Modularidad: propiedad que permite subdividir una aplicación en partes más pequeñas
- Polimorfismo: comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando.

Ventajas

- Permite crear sistemas más complejos
- Agiliza el desarrollo de software
- Proporciona conceptos y herramientas con las cuales se modela y representa el mundo real tan fielmente como sea posible.
- Fomenta la reutilización y extensión del código.

Desventajas

- Complejidad para adaptarse
- Mayor cantidad de código [2]

http://www.slideshare.net/ARMANDO1022/m-o-d-u-l-a-r-i-d-a-d
http://tareas.org/apuntes/EI/1/EDI/teoria/07-08/tad_-_ventajas.pdf
http://paradimas/programación.org/apuntes/ventajas_desventajas

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

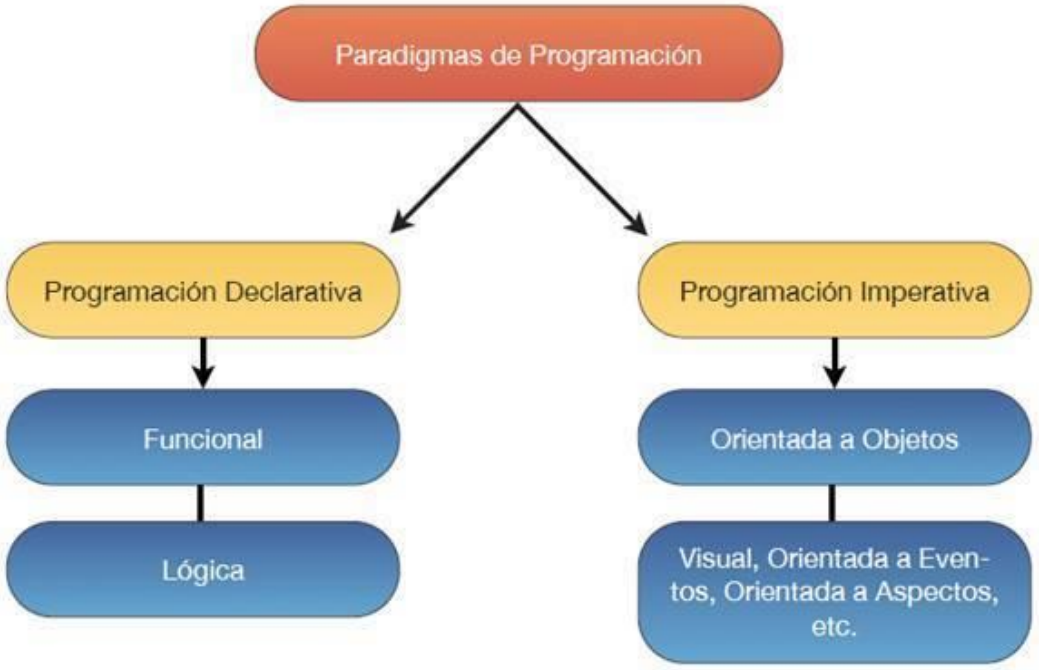
II. CONTENIDO

PARADIGMAS DE PROGRAMACION:

A lo largo de la historia, el término «paradigma» fue objeto de muchas interpretaciones. En su origen griego, significaba «modelo», «ejemplo» o «patrón». Sobre este punto de partida, podemos hablar de un paradigma como un conjunto de creencias, prácticas y conocimientos que guían el desarrollo de una disciplina durante un período de tiempo. En diversas ramas de la ciencia, un conjunto de ideas en vigencia puede ser reemplazado drásticamente por otro que entre en conflicto con él y se demuestre más acertado. La programación tiene sus propios paradigmas, pero el término «paradigma de programación» no necesariamente representa un modelo único que deba ser respetado hasta que aparezca otro mejor. De hecho, actualmente muchos paradigmas coexisten en armonía.

Un paradigma de programación es un estilo de desarrollo de programas. Es decir, un modelo para resolver problemas computacionales. Los lenguajes de programación, necesariamente, se encuadran en uno o varios paradigmas a la vez a partir del tipo de órdenes que permiten implementar, algo que tiene una relación directa con su sintaxis.

Paradigma de programación es una propuesta tecnológica que es adoptada por una comunidad de programadores cuyo núcleo central es incuestionable en cuanto a que unívocamente trata de resolver uno o varios problemas claramente delimitados. La resolución de estos problemas debe suponer consecuentemente un avance significativo en al menos un parámetro que afecte a la ingeniería de software. Tiene una estrecha relación con la formalización de determinados lenguajes en su momento de definición.



LENGUAJES:

Los primeros lenguajes de programación de alto nivel se diseñaron durante los años 1950. Desde entonces los lenguajes de programación han sido una fascinante y prolífica área de estudio para los científicos de la computación y los ingenieros. [3]

El estudio de los lenguajes de programación, es llamado a veces lingüística de la programación, por analogía con la lingüística de los lenguajes naturales. La analogía se basa en el hecho en que ambos; lenguajes naturales y lenguajes de programación,

poseen sintaxis (forma) y semántica (significado). La analogía no puede tomarse en todo el contexto, Los lenguajes de programación no pueden ser comparados con los lenguajes naturales en términos de su rango de expresividad y subjetividad. Por otro lado, un lenguaje natural no es más ni menos que un grupo de personas que hablan y escriben, así que la lingüística natural está restringida al análisis de los lenguajes existentes; mientras que los lenguajes de programación son concienzudamente diseñados y se pueden implementar en computadoras.

CONCEPTOS BÁSICO

Cada lenguaje de programación es una creación y como tal ha sido cuidadosamente diseñado. Algunos lenguajes han sido diseñados por personas únicas, como por ejemplo Pascal. Otros, han sido diseñados por un grupo grande de personas, tales como PL/l y Ada. La experiencia sugiere que aquellos lenguajes diseñados por personas únicas o grupos pequeños, tienden a ser más compactos y coherentes que aquellos lenguajes diseñados por grandes grupos.

Un lenguaje de programación, digno de su nombre, debe reunir ciertos requisitos.

El lenguaje de programación debe ser universal. Es decir, cualquier problema debe tener una solución que puede ser programada en el lenguaje y dicha solución ser implementada en cualquier computador. Este requisito es uno de los más fuertes y pocos lenguajes lo poseen. Se dice que cualquier lenguaje en el cual pueden definirse funciones recursivas se considera universal. De otro lado, un lenguaje sin recursión ni iteración no puede ser universal. Existen ciertos lenguajes de aplicación que no son universales, pero sí podrían ser razonablemente descritos así mismos, como lenguajes de programación.

El lenguaje de programación debe ser impleméntale en una computadora, es decir; debe ser posible ejecutar un programa en términos del lenguaje en cualquier máquina. La notación matemática generalmente no es implementable porque en su notación es posible formular problemas que no pueden ser resueltos por cualquier computador. Los lenguajes naturales tampoco son implementables por razones totalmente diferentes: ellos son tan imprecisos y tienden a ser muy ambiguos.

SINTAXIS Y SEMÁNTICA

Cada lenguaje tiene sintaxis y semántica:

- **La sintaxis de un lenguaje de programación está relacionada con la forma de los programas, por ejemplo, como es que las expresiones, comandos, declaraciones, etc. son puestos juntos en un programa.**

- **La semántica de un lenguaje de programación está relacionada con el significado de los programas; por ejemplo, cómo ellos se comportarán cuando se ejecutan en una computadora.**

La sintaxis de un lenguaje influye en cómo los programas son escritos por el programador, leídos por otro programador y traducidos por el computador. La semántica de un lenguaje determina como los programas son compuestos por el programador, entendidos por otros programadores e interpretados por el computador. La sintaxis es importante; pero la semántica es más importante aún. Figura 1

ENFOQUE HISTÓRICO

Los lenguajes de programación de hoy son el producto de un desarrollo que se inició en los 1950's. Numerosos conceptos de lenguajes han sido inventados, examinados e implementados en sucesivos lenguajes. Con muy pocas excepciones, el diseño de cada lenguaje ha sido fuertemente influenciado por la experiencia con los lenguajes iniciales. Los lenguajes de hoy no son el producto final del desarrollo del diseño del lenguaje; nuevos conceptos y paradigmas están siendo desarrollados y el escenario de los lenguajes de programación de los próximos diez años podría ser un poco diferente al de hoy.

Para que una computadora realice una tarea, debe programársela para que lo haga colocando en la memoria principal un algoritmo apropiado el cual es expresado en lenguaje máquina. En los inicios de la programación, esta tarea era onerosa por lo laborioso y difícil de diseñar cada algoritmo (sin contar los errores en que se podría incurrir). El gran paso se dio cuando se empezó a dar mnemónicos a los diversos códigos de operación y a los operandos del lenguaje de máquina. Con esto, los programadores pudieron aumentar considerablemente la comprensibilidad de las secuencias de instrucciones máquina. Por

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos

ejemplo, la siguiente rutina, empleando el método mnemónico:

CRG	**R2**	**TARIFA**
CRG	**R3**	**HORAST**
MULTI R0	**R2 R3**	
ALM	**R0**	**PAGO**
STOP		

Se puede interpretar como: cargar en el registro R2 al valor de TARIFA, cargar en el registro R3 el valor de HORAST, en la tercera sentencia, MULTI RO, 192,193 significa multiplique el contenido de R2 por R3 y póngalo en RO. A este tipo de lenguaje de programación se convino en llamarlo lenguaje ensamblador debido a que justamente un programa llamado ensamblador se encargaba de traducir estos mnemónicos a una forma más compatible con la máquina. (Se le llamó ensamblador porque su tarea era ensamblar instrucciones en lenguaje máquina a partir de los códigos de operación y operandos obtenidos al traducir nombre mnemónicos e identificadores). Al lenguaje ensamblador se le conoce también como lenguaje de bajo nivel. Una desventaja importante del lenguaje ensamblador es el ser dependiente de la máquina, es decir, si se cambia la máquina, cambia el programa ensamblador.

Al estudio de los lenguajes en cuanto al enfoque del proceso de programación se le denomina paradigmas de la programación, entendiéndose el término paradigma como la forma de ver y hacerlos programas. Bajo este enfoque se tienen cuatro paradigmas los cuales son:

- **paradigma por procedimientos o paradigma imperativo**
- **paradigma declarativo**
- **paradigma funcional**
- **paradigma orientado a objetos**

El paradigma por procedimientos, es tal vez el más conocido y utilizado en el proceso de programación, donde los programas se desarrollan a través de procedimientos. Pascal C y BASIC son tres de los lenguajes imperativos más importantes. La palabra latina imperare significa "dar instrucciones". El paradigma se inició al principio del año 1950 cuando los diseñadores reconocieron que las variables y los comandos o instrucciones de asignación constituían una simple pero útil abstracción del acceso a memoria y actualización del conjunto de instrucciones máquina. Debido a la estrecha relación con la arquitectura de la máquina, los lenguajes de programación imperativa pueden ser implementados muy eficientemente, al menos en principio.

El paradigma imperativo aún tiene cierto dominio en la actualidad. Una buena parte del software actual ha sido desarrollado y escrito en lenguajes imperativos. La gran mayoría de programadores profesionales son principalmente o exclusivamente programadores imperativos (Hay que añadir que los paradigmas de la programación concurrente y orientada al objeto son en realidad sub-paradigmas de la programación imperativa, así que sus adeptos también son programadores imperativos).

El paradigma declarativo o paradigma de programación lógica se basa en el hecho que un programa implementa una relación antes que una correspondencia. Debido a que las relaciones son más generales que las correspondencias (identificador - dirección de memoria), la programación lógica es potencialmente de más alto nivel que la programación funcional o la imperativa. El lenguaje más popular enmarcado dentro de este paradigma es el lenguaje PROLOG. El auge del paradigma declarativo se debe a que el área de la lógica formal de las matemáticas ofrece un sencillo algoritmo de resolución de problemas adecuado para, usarse en un sistema de programación declarativo de propósito general.

Si la programación imperativa se caracteriza por el uso de variables, comandos y procedimientos, la programación funcional se caracteriza por el uso de expresiones y funciones. Un programa dentro del paradigma funcional, es una función o un grupo de funciones compuestas por funciones más simples estableciéndose que una función puede llamar a otra, o el resultado de una función puede ser usado como argumento de otra función. El lenguaje por excelencia ubicado dentro de este paradigma es el LISP. Por ejemplo si se desea obtener la nota promedio de un alumno podría construirse una función promedio la cual se obtendría a partir de otras funciones más simples: una (sumar) la cual obtiene la suma de las entradas de la lista, otra (contar) la cual cuenta el número de entradas de la lista y la tercera (dividir) que obtiene el cociente de los valores anteriores, su

sintaxis será:

(dividir (sumar notas) (contar notas))

Obsérvese que la estructura anidada refleja el hecho de que la función dividir actúa sobre los resultados de suma y contar.

El paradigma orientado a objetos, se basa en los conceptos de objetos y clases de objetos. Un objeto es una variable equipada con un conjunto de operaciones que le pertenecen o están definidas para ellos. El paradigma orientado a objetos actualmente es el paradigma más popular y día a día los programadores, estudiantes y profesionales tratan de tomar algún curso que tenga que ver con este paradigma, podría decirse, que programar orientado a objetos está de moda.

Alrededor de 1970 David Parnas planteó el ocultamiento de la información como una solución al problema de gerenciar grandes proyectos software. Su idea fue encapsular cada variable global en un módulo con un grupo de operaciones (al igual que los procedimientos y las funciones) que permitan tener un acceso directo a la variable. Otros módulos pueden acceder a la variable sólo indirectamente, llamando a estas operaciones. Hoy se usa el término objeto para tales módulos o variables encapsuladas a sí mismas. Lenguajes imperativos como Pascal Y C han sido modificados (o añadidos) para que soporten el paradigma orientado a objetos para dar Delphi en el caso de Pascal y C++ en el caso de C.

Una de las bondades importantes de los lenguajes orientados a objetos es que las definiciones de los objetos pueden usarse una y otra vez para construir múltiples objetos con las mismas propiedades o modificarse para construir nuevos objetos con propiedades similares pero no exactamente iguales.

El lenguaje orientado a objetos por excelencia es Smaltalk desarrollado en Palo Alto Research Center durante los 1970's.

Pero que es exactamente un lenguaje orientado a objetos- Los siguientes conceptos señalan las características generalmente aceptadas acerca de los lenguajes orientados a objetos.

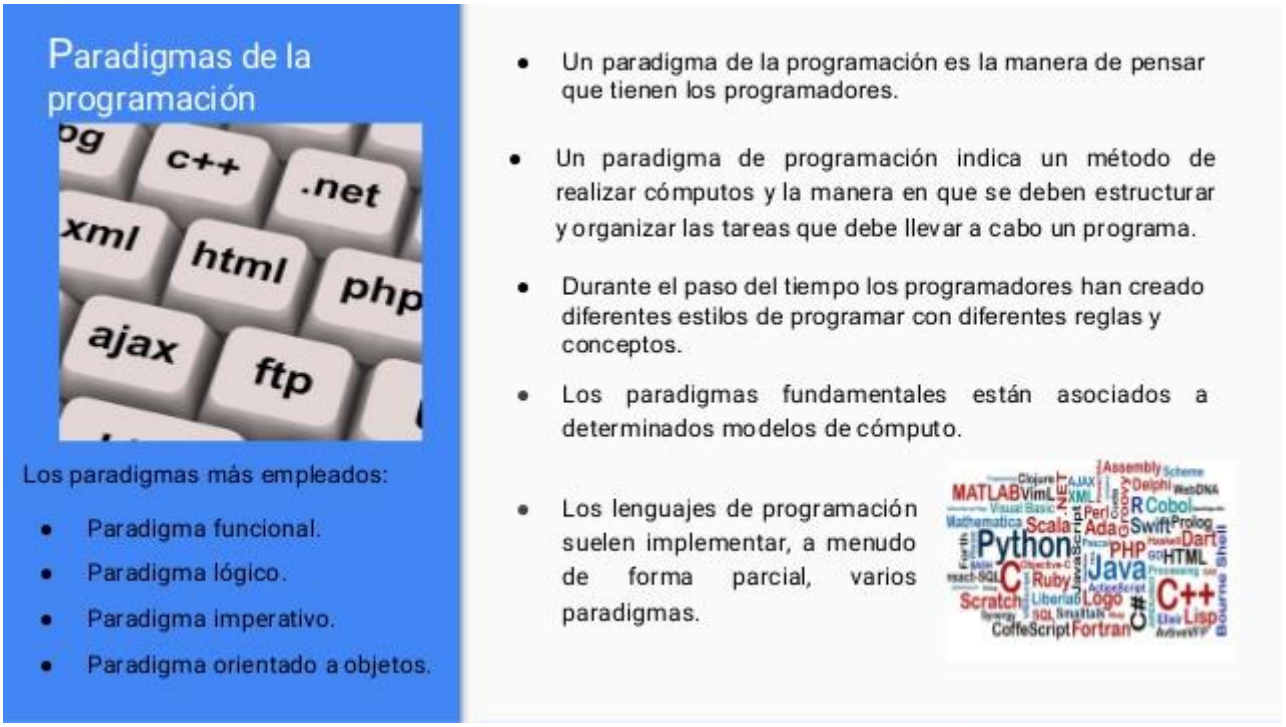
- **Objetos y clases son obviamente los conceptos fundamentales. Una clase es un conjunto de objetos que comparten las mismas operaciones.**

- **Objetos (o al menos referencia a objetos) deben ser valores de la clase base. Así, cualquier operación puede tomar un objeto como un argumento y puede devolver un objeto como resultado. De esta manera el concepto de clase de objetos está relacionado con el concepto de tipo de dato.**

- **Herencia es también vista como un concepto clave dentro del mundo de los objetos. En este contexto, la herencia es la habilidad para organizar las clases de objetos en una jerarquía de subclases y superclases y las operaciones dadas para una clase se pueden aplicar a los objetos de la subclase.**

EJEMPLOS:

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos



[4]

III. CONCLUSIONES

La comprensión básica de los conceptos de los lenguajes de programación y los diferentes paradigmas son necesarios para todos los ingenieros de software, no tanto para los especialistas en un lenguaje de programación. Esto se debe a que los lenguajes de programación son una herramienta fundamental.

Los lenguajes de programación influencian notablemente la manera en que pensamos acerca del diseño y construcción del software y los algoritmos y estructuras de datos que utilicemos para desarrollar software.

RECOMENDACIONES

Agradecimientos a todos los que puedan leer esta página espero les guste y les aclare sus dudas.

REFERENCIAS:

[1] En contraste, los lenguajes funcionales puros tienen una mayor potencia expresiva

[2] http://www.slideshare.net/ARMANDO1022/m-o-d-u-l-a-r-i-d-a-d
[3] http://tareas.org/apuntes/EI/1/EDI/teoria/07-08/tad_-_ventajas.pdf
[4] http://paradimas/programación.org/apuntes/ventajas_desventajas

1. Las notas de pie de página deberán estar en la página donde se citan. Letra Times New Roman de 8 puntos