I.P.N

Instituto Politécnico Nacional

ESCOM

Escuela Superior de Computo

Programación orientada a objetos

Tarea 1

Investigación del paradigma orientado a objetos

Alumno.- Santuario Parra Luis Fernando

Importancia de la programación orientada a objetos (paradigma)

La programación orientada a objetos tiene origen en Simula 67, un lenguaje diseñado para hacer simulaciones, creado por Ole-Johan Dahl y Kristen Nygaard, del Centro de Cómputo Noruego en Oslo. La principal diferencia entre la programación estructurada y la orientada a objetos radica en que en la programación estructurada los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida mientras que en la programación orientada a objetos primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos, ósea, un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.
La mayor importancia de la programación orientada a objetos es que permite solucionar problemas que el paradigma estructurado no puede, teniendo como principal características:

- **Reusabilidad**. Cuando hemos diseñado adecuadamente las clases, se pueden usar en distintas partes del programa y en numerosos proyectos.
- **Mantenibilidad**. Debido a la sencillez para abstraer el problema, los programas orientados a objetos son más sencillos de leer y comprender, pues nos permiten ocultar detalles de implementación dejando visibles sólo aquellos detalles más relevantes.
- **Modificabilidad**. La facilidad de añadir, suprimir o modificar nuevos objetos nos permite hacer modificaciones de una forma muy sencilla.
- **Fiabilidad**. Al dividir el problema en partes más pequeñas podemos probarlas de manera independiente y aislar mucho más fácilmente los posibles errores que puedan surgir.

Historia de la programación Orientada a Objetos

Como ya se había mencionado anteriormente la Programación Orientación a Objetos (P.O.O.) surge en Noruega en 1967 con un lenguaje llamado Simula 67, desarrollado por Krinsten Nygaard y Ole-Johan Dahl, en el centro de cálculo noruego. Simula 67 introdujo por primera vez los conceptos de clases, corrutinas y subclases (conceptos muy similares a los lenguajes Orientados a Objetos de hoy en día).

En los 70’s científicos del centro de investigación en Palo Alto Xerox (Xerox park) crearon Small talk que fue el primer lenguaje Orientado a Objetos puro de los lenguajes Orientados a Objetos, es decir, únicamente utiliza clases y objetos. Quien tuvo la idea fue D. Parnas cuando propuso la disciplina de ocultar la información. Su idea era encapsular cada una de las variables globales de la aplicación en un solo módulo junto con sus operaciones asociadas, sólo mediante las cuales se podía tener acceso a esas variables. El resto de los módulos (objetos) podían acceder a las variables sólo de forma indirecta mediante las operaciones diseñadas para tal efecto.

En los años 80's Bjarne Stroustrup de AT&T Labs., amplió el lenguaje C para crear C++ que soporta la programación Orientada a Objetos. En esta misma década se desarrollaron otros lenguajes Orientados a Objetos como Objective C, Common Lisp Object System (CIOS), object Pascal, Ada y otros.

Posteriores mejoras en herramientas y lanzamientos comerciales de C++ por distintos fabricantes, justificaron la mayor atención hacia la programación Orientada a Objetos en la comunidad de desarrollo de software. El desarrollo técnico del hardware y su disminución del costo fue el detonante final. Con más computadoras al alcance de más personas más programadores, más problemas y más algoritmos surgieron.

En el inicio de los 90's se consolida la Orientación a Objetos como una de las mejores maneras para resolver problemas. Aumenta la necesidad de generar prototipos más rápidamente (concepto RAD Rapid Aplication Developments). Sin esperar a que los requerimientos iniciales estén totalmente precisos. En 1996 surge un desarrollo llamado JAVA (extensión de C++). Su filosofía es aprovechar el software existente. Facilitar la adaptación del mismo a otros usos diferentes a los originales sin necesidad de modificar el código ya existente.

En 1997-98 se desarrollan herramientas 'CASE' orientadas a objetos (como el diseño asistido por computadora).

Del 98 a la fecha se desarrolla la arquitectura de objetos distribuidos RMI, Corba, COM, DCOM. Actualmente la orientación a objetos parece ser el mejor paradigma, no obstante, no es una solución a todos los problemas. Trata de eliminar la crisis del software. Entre los creadores de metodologías orientadas a objetos se encuentran: G. Booch, Rambaught, Ivar Jacobson y Peter Cheng

Cursos de programación

Empresa	Precio
KMMX "México"	**Precio por participante: $8,120 MN** (IVA incluido)
ESCOM "México"	**$650 o Gratis**
Educa Web "España"	**414 €**
Internet(video tutoriales)	**Gratis**
Escuela it "España"	Curso Programación Orientada a Objetos 120€

Cuadro comparativo

Existen más de 24 lenguajes de programación orientados a objetos de los cuales mostrare solo algunos cuantos los más populares con sus características fortalezas debilidades y una opinión.

Lenguaje	características	Fortalezas	Debilidades	Opinión
Java	• Es orientado a objetos • Multiplataforma •	Al ser orientado a objetos permite su modularización Permite la creación de aplicaciones de escritorio Tiene soporte a desarrollo de aplicaciones móviles y web.	• Es un lenguaje interpretado así que es relativamente lento en comparación con otros lenguajes	• Es un lenguaje bastante documentado y fácil de aprender, contiene muchas librerías tiene varias alternativas de framework para un desarrollo más fácil y creación de aplicaciones robustas.
PHP	• Utilizado para generar páginas web dinámicas • Se ejecuta en el servidor • Los usuarios no pueden ver el código PHP únicamente reciben en sus navegadores código HTML • Las páginas que genera son visibles para prácticamente cualquier navegador y computadora o dispositivos móviles que pueda interpretar el HTML. • No se necesita la instalación de PHP en el lado del cliente. • Versiones resiente permiten la POO • Lenguaje de alto nivel	• Su sintaxis es muy similar a otros lenguajes • Fácil • Es un lenguaje muy popular tiene una comunidad muy grande • Rápido • Multiplataforma • Maneja base de datos • Bastante documentado • Libre y gratuito. • Varias funciones • No requiere definición de variables • Puede ser combinado junto a HTML • Tiene muchos frameworks que facilitan el desarrollo en este lenguaje. • Muchos servicios de alojamiento web tienen PHP	• Necesita un servidor para funcionar • La POO es deficiente para aplicaciones grandes • Todo el trabajo se realiza el en servidor y mucha información o solicitudes pueden ser ineficiente.	• Es un lenguaje que está muy bien documentado y se pueden encontrar un sinfín de ejemplos y tutoriales lo cual lo hacer una muy buena opción para aprender y conocer sobre la programación.

RUBY	• Orientado a objetos • Lenguaje de alto nivel • Sintaxis similar a Python y Perl • Opensource • Lenguaje para la creación de aplicaciones de escritorio y aplicaciones web.	• Diferencia entre mayúsculas y minúsculas • Maneja excepciones • Puede cargar librerías si el sistema operativo lo permite • Multiplataforma • Portátil • Desarrollo de bajo costo • Software libre • multiplataforma	• es relativamente nuevo y no cuenta con mucha documentación en comparación con otros lenguajes de programación • no está muy difundido en relación a otros lenguajes.	• Sus sintaxis es muy simple y fácil de aprender y posible utilizarlos en baria plataformas, además es Opensource y libre.
ASP.ET	• Sucesor de ASP • Creada por Microsoft • De paga • Orientado a objetos	• Controles de usuarios y personalizados • Fácil mantenimiento • Incremento en velocidad • Mayor seguridad	• Mayor consumo de recursos	• es un lenguaje que tiene con mejores características que su primera versión
JavaScript	• es un lenguaje interpretado • es similar a java • es orientado a objetos	• los script tiene capacidad limitada por razones de seguridad • se ejecuta del lado del cliente • lenguaje de scripting seguro y fiable	• No soporta herencias • Código visible por cualquier usuario • El código debe ser descargado completamente • Puede poner en riesgo la seguridad del sitio con el actual problema llamado XSS (significa en inglés Cross Site Scripting renombrado a XSS por su similitud con las hojas de estilo CSS).	• Es un lenguaje fácil de aprender y que puede hacer un sitio web muy dinámico y grandes efectos. Además cuenta con múltiples librerías de terceros las cuales pueden facilitar el desarrollo de scripts.

C++	• Orientado a objetos • Rápido	• Ideal para sistemas robustos • IDEs de desarrollo son DEV C++, BORLAND C, TURBO C • Es multiplaforma	• No soporta creación de aplicaciones web • Complejo visualmente	• Al ser multiplaforma y rápido es una buena alternativa para el desarrollo de aplicaciones para escritorio
C#	• Está orientado a objetos • Esta estandarizado por Microsoft como parte de su plataforma net.	• Se desempeña de forma plena en los sistemas operativos Windows. • Sintaxis más en comparación con C y C++ • Posibilidad de realizar aplicaciones web, de escritorio y móviles.	• Requiere un mínimo de 4 gb para su instalación.	• Es un lenguaje ideal para desarrollar aplicaciones para los entorno de Windows.