PROGRAMACIÓN ORIENTADA A OBJETOS

PROFESOR

CARLOS ALBERTO LONDOÑO LOAIZA

ALUMNO

HERIBERTO DAVID YEPES

UNIVERSIDAD COTECNOVA

TALLER 1 PROGRAMACIÓN ORIENTADA A OBJETOS

CARTAGO – VALLE DEL CAUCA

13 DE FEBRERO DE 2017

SOLUCIÓN TALLER No 1 PROGRAMACIÓN ORIENTADA A OBJETOS

Punto 1 definir los conceptos fundamentales involucrados en el proceso de resolución de un problema

El proceso de resolución de un problema con una computadora conduce a la escritura de un programa y a su ejecución en la misma. Aunque el proceso de diseñar programas es esencialmente un proceso creativo, se pueden considerar una serie de fases o pasos comunes, que generalmente deben seguir todos los programadores.

Tabla de contenidos

1.Definición del problema  
2.Análisis del problema  
3.Selección de la mejor alternativa  
4.Diagramación  
5.Prueba de escritorio  
6.Codificación  
7.Trascripción  
8.Compilación  
9.Pruebas de computador  
10.Documentación externa

1.- **DEFINICIÓN DEL PROBLEMA**

Está dada por el enunciado del problema, el cuál debe ser claro y completo. Es importante que conozcamos exactamente que se desea del computador; mientras qué esto no se comprenda, no tiene caso pasar a la siguiente etapa.

2.- **ANÁLISIS DEL PROBLEMA**

Entendido el problema (que se desea obtener del computador), para resolverlo es preciso analizar:

\*Los datos o resultados que se esperan.  
\*Los datos de entrada que nos suministran.  
\*El proceso al que se requiere someter esos datos a fin de obtener los resultados esperados.  
\*Áreas de trabajo, fórmulas y otros recursos necesarios.

Una recomendación muy práctica es el que nos pongamos en el lugar del computador, y analizar que es necesario que me ordenen y en que secuencia, para poder producir los resultados esperados. También da buenos resultados hacer similitudes con la labor de un empleado que hace el mismo trabajo que deseamos programarle al computador.

3.- **SELECCIÓN DE LA MEJOR ALTERNATIVA**

Analizado el problema, posiblemente tengamos varias formas de resolverlo; lo importante es determinar cuál es la mejor alternativa: la que produce los resultados esperados en el menor tiempo y al menor costo. Claro que aquí también es muy válido el principio de que las cosas siempre se podrán hacer de una mejor forma.

4.- **DIAGRAMACIÓN**  
Una vez que sabemos cómo resolver el problema, pasamos a dibujar gráficamente la lógica de la alternativa seleccionada. Eso es precisamente un Diagrama de Flujo: la representación gráfica de una secuencia lógica de pasos a cumplir por el computador para producir un resultado esperado.

La experiencia nos ha demostrado que resulta muy útil trasladar esos pasos lógicos planteados en el diagrama a frases que indiquen lo mismo; es decir, hacer una codificación del programa pero utilizando instrucciones en Español. Como si le estuviéramos hablando al computador. Esto es lo que denominaremos Algoritmo o Pseudocódigo.

Cuando logremos habilidad para desarrollar programas, es posible que no elaboremos el diagrama de flujo; en su lugar podremos hacer directamente el pseudocódigo del programa.

5.- **PRUEBA DE ESCRITORIO**

Para cerciorarnos de que el diagrama (y/o el pseudocódigo) esta bien, y, para garantizar que el programa que codifiquemos luego también funcione correctamente, es conveniente someterlo a una Prueba de Escritorio. Esta prueba consiste en que damos diferentes datos de entrada al programa y seguimos la secuencia indicada en el diagrama, hasta obtener los resultados.

El análisis de estos nos indicará si el diagrama esta correcto o si hay necesidad de hacer ajustes (volver al paso 4). Se recomienda dar diferentes datos de entrada y considerar todos los posibles casos, aun los de excepción o no esperados, para asegurarnos de que el programa no producirá errores en ejecución cuando se presenten estos casos.

6.- **CODIFICACIÓN**

Una vez que hayamos verificado el diagrama mediante las pruebas de escritorio, codificamos el programa en el lenguaje de computador seleccionado. Esto es, colocamos cada paso del diagrama en una instrucción o sentencia, utilizando un lenguaje que el computador reconoce.

Todos los lenguajes de programación proveen facilidades para incluir líneas de comentarios en los programas. Estos comentarios aclaran lo que se ordena al computador y facilitan entender el programa. Puesto que estos comentarios no son tenidos en cuenta como instrucciones, y aparecen en los listados del programa, resulta muy conveniente agregar abundantes comentarios a todo programa que codifiquemos. Esto es lo que se denomina Documentación Interna.

7.- **TRANSCRIPCIÓN**

El programa codificado es necesario que lo llevemos a un medio que sea aceptado como entrada por el computador: lo perforamos en tarjetas, lo grabamos en un disco flexible o lo grabamos en un disco duro. Este programa es el que se conoce como Programa Fuente (Source).

8.- **COMPILACIÓN**

Utilizamos ahora un programa de computador llamado Compilador o Traductor, el cúal analiza todo el programa fuente y detecta errores de sintaxis ocasionados por fallas en la codificación o en la transcripción. Las fallas de lógica que pueda tener nuestro programa fuente no son detectadas por el compilador. Cuando no hay errores graves en la compilación, el compilador traduce cada instrucción del programa fuente a instrucciones propias de la máquina (Lenguaje de Maquina), creando el Programa Objeto.

Algunos computadores utilizan Interpretadores, (Generalmente para el Lenguaje Basic), en reemplazo de programas compiladores. La diferencia consiste en que el interpretador recibe, desde una terminal, sólo una instrucción a la vez, la analiza y, si esta bien, la convierte al formato propio de la maquina. Si la instrucción tiene algún error, el interpretador llama la atención de la persona para que corrija dicha instrucción.

Como resultado de la corrida del compilador, podemos obtener varios listados:

\*Listado del programa fuente  
\*Listado de los errores detectados  
\*Listado de campos utilizados, etc.

Los errores los debemos corregir sobre el mismo programa fuente, ya sea reemplazando las tarjetas mal perforadas o regrabando en el disco flexible o en el disco duro. Este paso de la compilación lo repetimos hasta eliminar todos los errores y obtener el programa ejecutable.

9.- **PRUEBAS DE COMPUTADOR**

Cuando tenemos el programa ejecutable (en lenguaje de maquina), ordenamos al computador que lo ejecute, para lo cuál suministramos datos de prueba, como lo hicimos en la prueba de escritorio (paso 5). Los resultados obtenidos los analizamos, luego de lo cuál puede ocurrir cualquiera de estas situaciones:

a.- La lógica del programa esta bien, pero hay errores sencillos, los cuales los corregimos modificando algunas instrucciones o incluyendo unas nuevas; el proceso debemos repetirlo desde el paso 6 .

b.- Hay errores ocasionados por fallas en la lógica, lo que nos obliga a regresar a los pasos 4 y 5 para revisión y modificación del diagrama.

c.- Hay errores muy graves y lo más aconsejable es que regresemos al paso 2 para analizar nuevamente el problema, y repetir todo el proceso.

d.- No hay errores y los resultados son los esperados. En este caso, el programa lo podemos guardar permanentemente en una librería o biblioteca del computador, para sacarlo de allí cuando necesitemos ejecutarlo nuevamente.

10.- **DOCUMENTACIÓN EXTERNA**

Cuando el programa ya se tiene listo para ejecutar, es conveniente que hagamos su documentación externa siguiendo las normas de la instalación o las recomendaciones indicadas por el profesor. Una buena documentación incluye siempre:

-Enunciado del problema  
-Diagrama de pasada  
-Narrativo con la descripción de la solución  
-Relación de las variables o campos utilizados en el programa, cada uno con su respectiva función  
-Diagrama del programa  
-Listado de la última compilación  
-Resultados de la ejecución del programa.

Punto 2 definir lenguaje, programa, compilador e intérprete.

**Definición de lenguaje**

El concepto de lenguaje puede ser entendido como un recurso que hace posible la [comunicación](http://definicion.de/comunicacion/).

Un lenguaje de programación" es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo.

**Definición de programa**

Descripción del conjunto de operaciones, y a veces ciclo, que deben o pueden ejecutar ciertas máquinas.

Secuencia de instrucciones escritas en un lenguaje de programación y organizadas siguiendo un esquema lógico, que realiza una determinada acción o resuelve un problema concreto.

El resultante de enlazar uno o varios subprogramas de modo que sea ejecutable.

El resultante de la traducción, mediante un compilador, del programa fuente.

**Definición de compilador**

Un compilador es un pequeño programa informático, que se encarga de traducir (compilar) el código fuente de cualquier aplicación que se esté desarrollando. En pocas palabras, es un [software](http://culturacion.com/el-software-y-su-arquitectura-de-trabajo/" \t "_blank) que se encarga de traducir el programa hecho en lenguaje de programación, a un lenguaje de máquina que pueda ser comprendido por el equipo y pueda ser procesado o ejecutado por este.

**Definición de intérprete**

En el contexto de la informática, se llama intérprete a aquel programa que analiza y ejecuta otros programas pero que se hayan codificados con un lenguaje de mayor complejidad. El intérprete irá realizando la traducción pertinente a medida que sea especialmente necesario, es decir, al momento que el programa se va ejecutando y no es habitual que guarden el resultado de la correspondiente traducción.

**Diferencias entre compilador e interprete**

* Un intérprete traduce instrucciones de alto nivel en una forma intermedia para ser ejecutado. En contraste, un compilador, traduce instrucciones de alto nivel directamente en lenguaje de máquina.
* El intérprete traduce un programa línea a línea mientras que el compilador traduce el programa entero y luego lo ejecuta.
* El intérprete detecta si el programa tiene errores y permite su depuración durante el proceso de ejecución, mientras que el compilador espera hasta terminar la compilación de todo el programa para generar un informe de errores.
* Un programa compilado es más seguro que uno interpretado, porque no contiene el código fuente, que puede ser modificado incorrectamente por el usuario.

Punto 3 identificar los antecedentes de la programación orientada a objetos.

Primera generación: lenguaje de maquina.

Segunda generación: lenguaje ensamblador.

Tercera generación: lenguaje de alto nivel.

Cuarta generación: lenguaje orientado a objetos.

Punto 4 definir los conceptos involucrados en la programación orientada a objetos.

**Objeto**: Los objetos representan una entidad concreta o abstracta del mundo real, en programación básicamente se le conoce como la instancia de una clase en si es lo que da el sentido a estas.

**Estado**: Representa los atributos o características con valores concretos del objeto.

**Comportamiento**: Se define por los métodos u operaciones que se pueden realizar con el.

**Identidad**: Es la propiedad única que representa al objeto y lo diferencia del resto.

**Clase**: Una clase es “Un conjunto de objetos que comparten una estructura en común y un comportamiento común” es simplemente una representación de un tipo de objeto. Es el la plantilla que describe los detalles de un objeto. Una clase es el anteproyecto de la que se crean los objetos individuales Class se compone de tres cosas:. Un nombre, atributos y operaciones

**Nombre**: Contiene el Nombre de la Clase.

**Atributos**: Representan las propiedades que caracterizan la clase.

**Métodos**: Representan el comportamiento u operaciones, la forma como interactúa la clase con su entorno.

**Herencia**: Cuando una clase adquieren la propiedad de otra clase, que se conoce como herencia. La herencia es el proceso de reutilización de objetos. La herencia es “un mecanismo en el cual una clase es definida en términos de una clase existente, y se le agregan características propias”. La posibilidad de una nueva clase que se creará, a partir de una clase existente, haciéndolo extensivo, se denomina herencia.

**Abstracción**: La abstracción es "Representar a la característica esencial y sin representación de los detalles del fondo." Abstracción le permite centrarse en lo que hace el objeto en lugar de cómo lo hace. La abstracción que proporciona una visión generalizada de sus clases u objetos, proporcionando información relevante. La abstracción es el proceso de ocultar el estilo de trabajo de un objeto, y que muestra la información de un objeto de una manera comprensible.

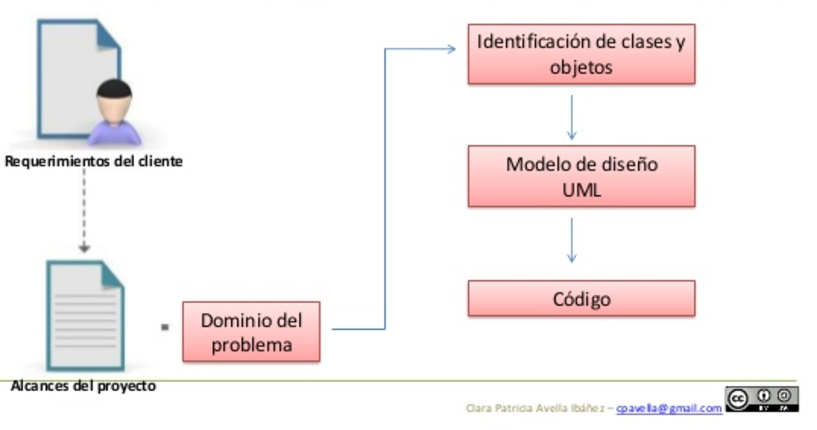
**Constructor**: Un objeto de una clase se crea llamando a una función especial denominada constructor de la clase. El constructor se llama de forma automática cuando se crea un objeto, para situarlo en memoria e inicializar los miembros dato declarados en la clase. El constructor tiene el mismo nombre que la clase. Lo específico del constructor es que no tiene tipo de retorno.

**Encapsulación**: Es una técnica que permite localizar y ocultar los detalles de un objeto. La encapsulación previene que un objeto sea manipulado por operaciones distintas de las definidas. La encapsulación es como una caja negra que esconde los datos y solamente permite acceder a ellos de forma controlada.

**Polimorfismo**:Este tal vez sea uno de los conceptos de la programación orientada a objetos mas usados pero muchas veces sin saber que se aplica ya que el concepto inicialmente puede ser un poco confuso, básicamente mediante el polimorfismo programamos de forma general en lugar de hacerlo de forma especifica, se usa cuando se trabajen con la herencia y objetos de características comunes los cuales comparten la misma superClase y árbol jerárquico, al trabajar con este concepto optimizamos y simplificamos en gran medida nuestro trabajo.

básicamente podemos definirlo como la capacidad que tienen los objetos de comportarse de múltiples formas sin olvidar que para esto se requiere de la herencia, en si consiste en hacer referencia a objetos de una clase que puedan tomar comportamientos de objetos descendientes de esta.

Punto 5 identificar las etapas para la resolución de un problema bajo un enfoque de programación orientada a objetos.



**Etapa de análisis**: El primer paso para tener éxito en el desarrollo de una aplicación de software es tener claridad acerca de lo que quiere el cliente y para ello se identifican los requerimientos de la aplicación, los cuales pueden ser funcionales y no funcionales.

Los requerimientos funcionales corresponden a las acciones que realiza la aplicación de software. Para tomar un ejemplo, piense en el procesador de texto de Office (Word), que corresponde una aplicación de software y mire toda la funcionalidad que ofrece a los usuarios. Algunas de las funcionalidades son: Abrir un archivo, guardar un archivo, imprimir un archivo, cambiar el tipo de letra, adicionar imágenes de archivo y prediseñadas, adicionar tablas, justificar un párrafo, cambiar las márgenes del documento, etc.

Los requerimientos no funcionales corresponden más a condiciones de calidad que debe tener una aplicación o restricciones que el cliente define para que se desarrolle su aplicación. Como ejemplo podemos pensar en que el cliente puede solicitar que su aplicación se construya con cierto lenguaje de programación o programa administrador de base de datos, o si se trata de una aplicación web, puede solicitar que se elabore en algunos idiomas específicos o con los colores de su organización. También pueden ser requerimientos no funcionales la seguridad al acceso de ciertas opciones dentro de la aplicación o los tiempos de respuesta.

**Etapa de diseño**:Una vez se tiene claridad sobre el problema a solucionar, se debe proponer un diseño con la solución. Haciendo una similitud a un proyecto de construcción de una casa, pensemos que el diseño son los diferentes planos que el arquitecto y el ingeniero civil nos elaboran sobre la casa: plano estructural, plano arquitectónico, plano de instalaciones hidráulicas, plano de instalaciones eléctricas, etc. Estos planos son la base para construir la casa.

El diseño se expresa a través de varios modelos (representación del mundo real), los cuales se representan a través de diagramas. Existen varias herramientas de uso libre y comercial, para elaborar los modelos de los diseños de una aplicación de software.

**Etapa de construcción**: En esta etapa se escribe el código fuente de la aplicación que se diseñó en la etapa anterior. Para realizar una buena programación orientada a objetos se debe partir de los modelos diseñados para generar el código a partir de los mismos. Los programas por lo general se elaboran sobre IDE (Integrated Development Environment) Entornos de Desarrollo Integrado. El lenguaje de programación que utilizaremos es Java, un lenguaje orientado a objetos, así que es primordial partir de la concepción del problema y el diseño orientado a objetos antes de pasar al código fuente de la aplicación. Si el problema a solucionar se modela correctamente, el lenguaje de programación utilizado no implica muchos cambios, lo importante es realizar un buen modelado orientado a objetos.

Punto 6 identificar los tipos de datos que se emplean en el desarrollo de un programa.

**Byte**: El tipo de dato byte puede representar datos enteros que se encuentren en el rango de -128 a +127.  
El tamaño de un dato de tipo byte es de 8 bits.  
  
**Short**:El tipo de dato short puede representar datos enteros que se encuentren en el rango de -32768 y +32767.  
El tamaño de un dato de tipo short es de 16 bits.  
  
**Int**:El tipo de dato int puede representar datos enteros que se encuentren en el rango de -2147483648 y +2147483647.  
El tamaño de un dato de tipo int es de 32 bits.  
  
**Long**:El tipo de dato int puede representar datos enteros que se encuentren en el rango de -9223372036854775808 y +9223372036854775807.  
El tamaño de un dato de tipo int es de 64 bits.  
Para indicar de manera explicita que el dato es un long , se agrega una L o l al final del valor de la variable.  
  
**Float**:El tipo de dato float puede representar datos en coma flotante que se encuentren en el rango de 1.40239846e–45f y 3.40282347e+38f.  
El tamaño de un dato de tipo short es de 32 bits.   
Para indicar de manera explicita que el dato es un float , se agrega una F o f al final del valor de la variable.  
  
**Double**:El tipo de dato double puede representar datos en coma flotante que se encuentren en el rango de 4.94065645841246544e–324d y 1.7976931348623157e+308d.  
El tamaño de un dato de tipo short es de 64 bits.   
Para indicar de manera explicita que el dato es un double , se agrega una D o d al final del valor de la variable.  
  
**Boolean**:El tipo de dato boolean puede representar dos valores lógicos: true(verdadero) o false(falso).

**Char**: El tipo de dato char se usa para representar  
caracteres(codigo Unicode).  
Un caracter es representado internamente por un entero.

**String**: A los valores que representan texto se les llama **strings**, y tienen el tipo **str**. Los strings literales pueden ser representados con texto entre comillas simples o comillas dobles.

**Nulo**:Existe un valor llamado **None** (en inglés, «ninguno») que es utilizado para representar casos en que ningún valor es válido, o para indicar que una variable todavía no tiene un valor que tenga sentido.

El valor None tiene su propio tipo, llamado NoneType, que es diferente al de todos los demás valores.

Punto 7 definir los conceptos de variables, expresiones y constantes.

**Expresión:** Una expresión es una combinación de valores y operaciones que son evaluados durante la ejecución del algoritmo para obtener un resultado.

Por ejemplo, 2 + 3 es una expresión aritmética que, al ser evaluada, siempre entrega el valor 5 como resultado. En esta expresión, 2 y 3 son valores literales y + es el operador de adición.

**Variable**: las variables son espacios reservados en la memoria que, como su nombre indica, pueden cambiar de contenido a lo largo de la ejecución de un programa. Una variable corresponde a un área reservada en la memoria principal del ordenador.

Para que nuestro código sea más entendible y claro, el identificador de la variable debe ser [mnemotécnico](https://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Nemot%C3%A9cnicos), es decir que debe reflejar el uso dentro del programa de la misma.

**Constante**: una constante es un valor que no puede ser alterado durante la ejecución de un programa. Corresponde a una longitud fija de un área reservada en la memoria principal del ordenador, donde el programa almacena valores fijos. El nombre de las constantes suele escribirse en mayúsculas en la mayoría de lenguajes.

Punto 8 describir el procedimiento para declarar los formatos de las sentencias de asignación, entrada y salida.

**Asignación**: -> la operación de asignación es el modo de darle valores a una variable. La operación de asignación se representa con el símbolo u operador . la operación de asignación se conoce como *instrucciones o sentencia* de asignación cuando se refiere a un lenguaje de programación.

El formato general de una operación de asignación es:

Nombre de la variable <- expresión

La flecha (operador de asignación) se sustituye en otros lenguajes = (BASIC, FORTRAN) o = Pascal ). Sin embargo es preferible el uso de la flecha en la redacción del algoritmo, para evitar ambigüedades dejando el uso del símbolo = exclusivamente para el operador de igualdad.

La operación de asignación:

A <- 5;

Significa que a la variable A se le ha asignado el valor 5.

Punto 9 ¿que es UML y que proceso se puede realizar con este lenguaje?

**Definición**

UML son las siglas de “Unified Modeling Language” o “Lenguaje Unificado de Modelado”. Se trata de un estándar que se ha adoptado a nivel internacional por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software (programas informáticos).

El término “lenguaje” ha generado bastante confusión respecto a lo que es UML. En realidad el término lenguaje quizás no es el más apropiado, ya que no es un lenguaje propiamente dicho, sino una serie de normas y estándares gráficos respecto a cómo se deben representar los esquemas relativos al software. Mucha gente piensa por confusión que UML es un lenguaje de programación y esta idea es errónea: UML no es un lenguaje de programación. Como decimos, UML son una serie de normas y estándares que dicen cómo se debe representar algo.

UML es una herramienta propia de personas que tienen conocimientos relativamente avanzados de programación y es frecuentemente usada por analistas funcionales (aquellos que definen qué debe hacer un programa sin entrar a escribir el código) y analistas-programadores (aquellos que dado un problema, lo estudian y escriben el código informático para resolverlo en un lenguaje como Java, C#, Python o cualquier otro). Por tanto si estás dando tus primeros pasos en programación, te recomendaríamos que te olvides de UML hasta que tengas unos conocimientos mínimos como uso de condicionales, bucles, y conocimiento de la programación orientada a objetos. Esto es solo una recomendación, en realidad prácticamente cualquier persona puede usar UML, incluso podría usarse para realizar esquemas o documentación de procesos que no tengan que ver con la informática.

**Sirve para**

Sirve para representar visualmente las reglas de creación, estructura y comportamiento de un grupo relacionado de objetos y procesos.

Para visualizar de forma eficiente la complejidad de un sistema u organización en un reducido numero de diagramas.

Para mantener mucho mas ágilmente las especificaciones ante los cambios y mas actualizaciones de arquitectura.

**Los modelos se construyen para**

Para comunicar la estructura deseada y el comportamiento de un sistema

Para ver y controlar la arquitectura de un sistema

Para comprender mejor el sistema que esta en desarrollo

Para descubrir posibilidades de simplificación y reutilización

Para minimizar los riesgos

Punto 10 ¿qué es JAVA, por que es importante para la industria del desarrollo de software y en que proyectos se usa?

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes.

Java ha sido probado, ajustado, ampliado y probado por toda una comunidad de desarrolladores, arquitectos de aplicaciones y entusiastas de Java. Java está diseñado para permitir el desarrollo de aplicaciones portátiles de elevado rendimiento para el más amplio rango de plataformas informáticas posible. Al poner a disposición de todo el mundo aplicaciones en entornos heterogéneos, las empresas pueden proporcionar más servicios y mejorar la productividad, las comunicaciones y colaboración del usuario final y reducir drásticamente el costo de propiedad tanto para aplicaciones de usuario como de empresa. Java se ha convertido en un valor impagable para los desarrolladores, ya que les permite:

* Escribir software en una plataforma y ejecutarla virtualmente en otra
* Crear programas que se puedan ejecutar en un explorador y acceder a servicios Web disponibles
* Desarrollar aplicaciones de servidor para foros en línea, almacenes, encuestas, procesamiento de formularios HTML y mucho más
* Combinar aplicaciones o servicios que utilizan el lenguaje Java para crear aplicaciones o servicios con un gran nivel de personalización
* Escribir aplicaciones potentes y eficaces para teléfonos móviles, procesadores remotos, microcontroladores, módulos inalámbricos, sensores, gateways, productos de consumo y prácticamente cualquier otro dispositivo electrónico

# BIBLIOGRAFÍA

<http://www.conocimientosweb.net/zip/article814.html>

<http://definicion.de/lenguaje/>

[http://es.ccm.net/contents/304-lenguajes-de-programacion#](http://es.ccm.net/contents/304-lenguajes-de-programacion)

<http://es.thefreedictionary.com/programa>

<http://culturacion.com/que-es-un-compilador/>

<http://www.definicionabc.com/general/interprete.php>

<http://www.cavsi.com/preguntasrespuestas/cual-es-la-diferencia-entre-interprete-y-compilador/>

<http://go.yuri.at/juegos/clase1/clase1_introduccion.html>

<http://es.slideshare.net/josuelara792197/conceptos-de-poo-programacion-orientada-a-objetos>

<http://codejavu.blogspot.com.co/2013/05/conceptos-de-programacion-orientada.html>

<http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/clases.htm>

<http://dis.um.es/~jfernand/0506/dai/poo.pdf>

<http://es.slideshare.net/cpavella/1-soluc-problemasprocesos>

<https://sites.google.com/site/programacion1electronica/netbeans/tipos-de-datos>

<http://progra.usm.cl/apunte/materia/tipos.html>

<http://progra.usm.cl/apunte/materia/programas-simples.html>

<https://www.ecured.cu/Constante_(Inform%C3%A1tica>)

<https://www.fing.edu.uy/inco/cursos/fpr/wiki/index.php/Variables_y_Tipos>

<http://es.slideshare.net/BorisSalleg/algoritmo-variables-constantes-tipos-de-datos-y-asignacion>

<http://es.slideshare.net/myle22/qu-es-uml-para-que-sirve-pasos>

<http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46:lenguajes-y-entornos&Itemid=163>

<https://www.java.com/es/download/faq/whatis_java.xml>

<https://www.java.com/es/about/>