UNIREMINGTON®
CORPORACIÓN UNIVERSITARIA REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996

FUNDAMENTOS DE PROGRAMACIÓN

INGENIERÍA DE SISTEMAS

FACULTAD DE CIENCIAS BÁSICAS E INGENIERÍA

Vicerrectoría de Educación a Distancia y virtual

2016

AUTOR

Cesar Augusto Jaramillo Henao
Tecnólogo en sistemas
cesar.jaramillo@remington.edu.co

Nota: el autor certificó (de manera verbal o escrita) No haber incurrido en fraude científico, plagio o vicios de autoría; en caso contrario eximió de toda responsabilidad a la Corporación Universitaria Remington, y se declaró como el único responsable.

RESPONSABLES

Jorge Mauricio Sepúlveda Castaño
Decano de la Facultad de Ciencias Básicas e Ingeniería
jsepulveda@uniremington.edu.co

Eduardo Alfredo Castillo Builes
Vicerrector modalidad distancia y virtual
ecastillo@uniremington.edu.co

Francisco Javier Álvarez Gómez
Coordinador CUR-Virtual
falvarez@uniremington.edu.co

GRUPO DE APOYO

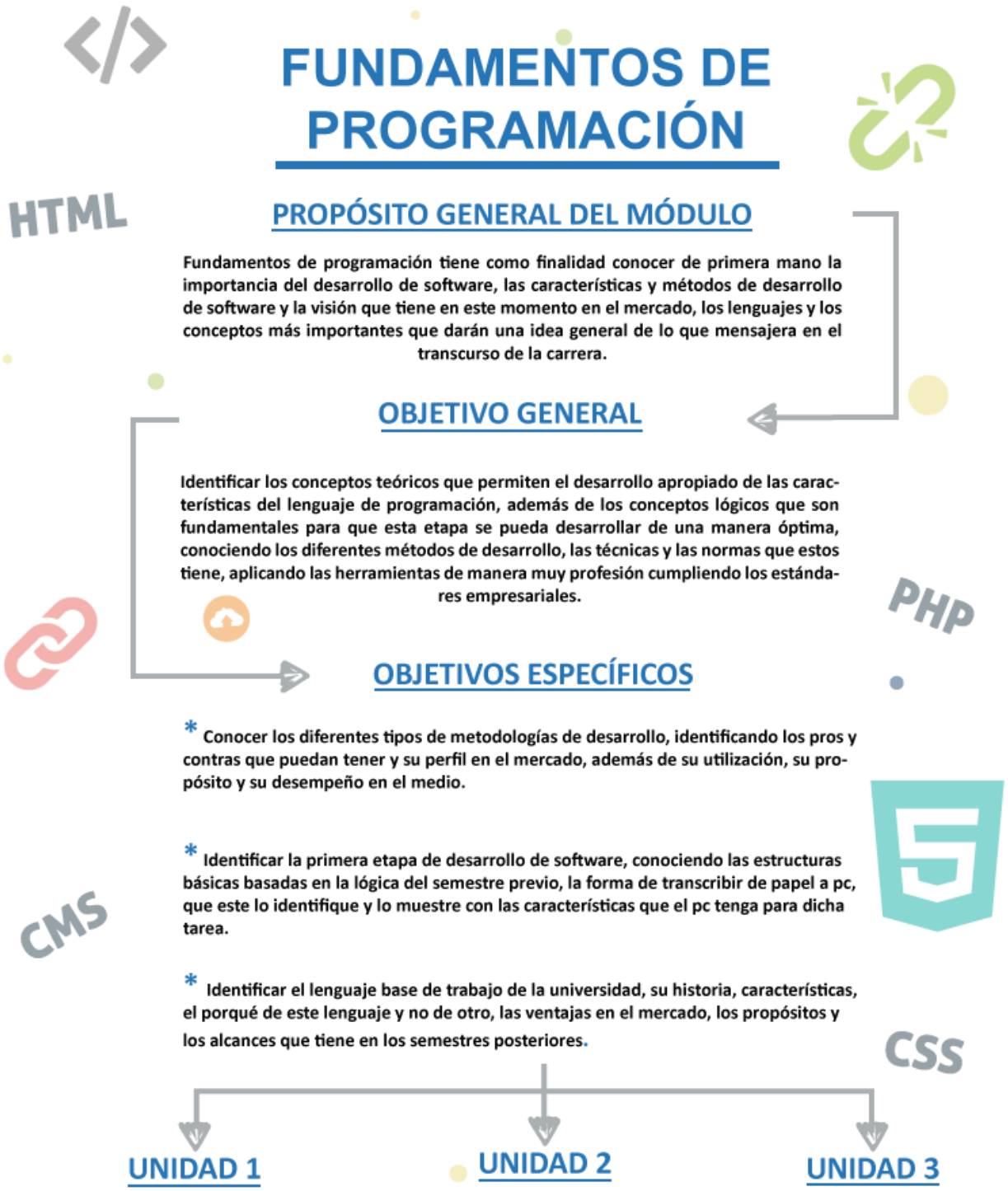
Personal de la Unidad CUR-Virtual
EDICIÓN Y MONTAJE

Primera versión. Febrero de 2011.
Segunda versión. Marzo de 2012
Tercera versión. noviembre de 2015
Cuarta versión. 2016

TABLA DE CONTENIDO

Pág.

1 MAPA DE LA ASIGNATURA



2 UNIDAD 1 PERFILES DEL DESARROLLO DE SOFTWARE

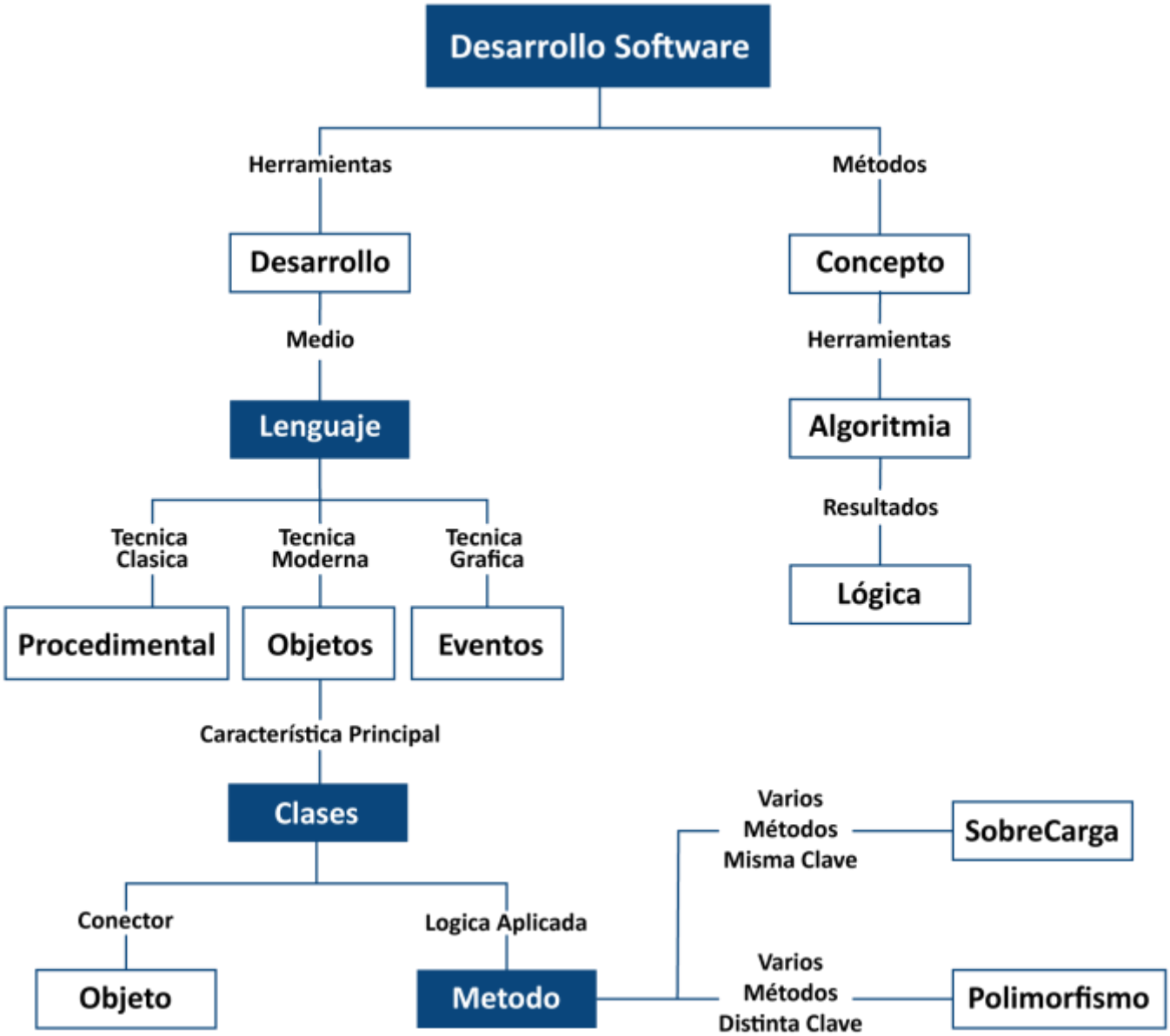


TABLA DE CONCEPTOS		EJEMPLO	
Concepto	Definición		
Software	Conjunto de instrucciones lógicas que permiten la optimización de los recurso de un equipo de cómputo, dando solución a problemas que puedan tener los usuarios o empresas.	Las empresas y personas requieren permanentemente la utilización de recursos de los sistemas de cómputo, la utilización ágil, rápida y confiable de la información, el desarrollo de software permite este tipo de tareas que de una manera adecuada procesara los datos y arrojara los resultados que se estén requiriendo, apoyado en las necesidades que serán la base de la lógica del manejo.	
Algoritmia	Es la ciencia que estudia todo lo relacionado con los aspectos lógicos y su solución mediante un lenguaje de programación.		
Lógica	Es la representación de instrucciones que un pc puede interpretar y arrojar resultados según la necesidad del usuario o empresa.		
Desarrollo	Creación de un aplicativo aplicando conceptos lógicos y desarrollados sobre un lenguaje de programación.		
Lenguaje	Aplicativo que permite interpretar situaciones lógicos con las necesidades de un usuario.	Los lenguaje de programación avanzan en su metodología para hacer de este un proceso más rápido y confiable, las diferentes herramientas procesas la información y darán solución a las necesidades de un cliente.	
Procedimental	Forma de programas clásica que hace recorridos lógicos en forma vertical, la más utilizada desde el inicio de los lenguajes de programación.		
Objetos	Forma de programar moderna, permite la reutilización de recursos y el mejoramiento de tareas y procesos.		

Eventos	Forma de programar en ambientes gráficos, permitiendo utilizar los recursos de mouse y teclado, con las características y limitantes que esto pueda tener.	La orientación a objetos en la herramienta más potente y más común en los lenguajes actuales, permitiendo que sea más confiable, rápida, reutilizable todos los recursos y aprovechar las herramientas creadas, evitando redundancias lógicas dentro del desarrollo.
Clase	Representación abstracta de un procesos, administra los recursos de lo que se está creando, da prioridades y restricciones, almacena los métodos y los objetos que un aplicativo orientado a objetos puedan tener.	
Objeto	Representación lógica de un método, determina el uso, los parámetros y los procesos que se van a ejecutar.	
Método	Representación lógica de un conjunto de instrucciones, es el encargado de los procesos del aplicativo.	
Sobrecarga	Conjunto de métodos que tienen el mismo nombre dentro de una clase, pero que tienen distinto propósito o distinta cantidad de parámetros.	
Polimorfismo	Conjunto de métodos que tienen el mismo nombre dentro de clases distintas del mismo aplicativo.	

RELACIÓN DE CONCEPTOS

OBJETIVO GENERAL

Conocer los diferentes tipos de metodologías de desarrollo, conociendo sus pros y contras que puedan tener y su perfil en el mercado, además de su utilización, su propósito y su desempeño en el medio.

OBJETIVOS ESPECÍFICOS

Conocer de primera mano los conceptos que envuelven el desarrollo de software, sus propósitos, el objetivo de sistematizar y automatizar tareas, todas acompañadas de un sistema de cómputo y de herramientas lógicas que darán como resultado un aplicativo, que permitirá mejorar y agilizar los procesos que de otra manera tardarían mucho tiempo.

Conocer los conceptos entre las diferentes metodologías de desarrollo, los pros y los contras que cada uno tiene, sus características presentes y las tendencias a futuro, las necesidades y el movimiento que tiene la industria actualmente.

Enfatizar los conceptos de orientación a objetos como herramienta de alto nivel, muy utilizada y de gran poderío por sus recursos, es el único método que permite combinar los conceptos con la teoría para un adecuado uso de las herramientas de los lenguajes que tienen esta tendencia.

2.1 TEMA 1 QUE ES EL DESARROLLO DE SOFTWARE

El desarrollo de software es un arte que se aprende mediante conceptos lógicos, adquiridos estos durante toda nuestra vida, estos conceptos lógicos se complementan con herramientas de algoritmia que permiten identificar estas situaciones en una secuencia de procesos y pasos, con normas que dan una formación adecuada, estos procesos habitualmente se establecen para suplir una necesidad de un usuario o empresa, estas necesidades son de distinta índole que van desde el entretenimiento, hasta mejorar los recursos actuales y buscar que lo que se maneja en un momento dado sea reducido en tiempos o sea más completo el resultado esperado.

El desarrollo de software permitirá tomar todo lo anterior y con herramientas apropiadas se establezcan de una manera lógica que comprenderá el equipo de cómputo y procesara la información hasta dar el resultado esperado, todo lo que vemos en nuestro sistema es software y todo ha pasado por las necesidades que se han planteado y vemos como día a día encontramos más alternativas que nos suplen el manejo engorroso de tareas para hacerlas más rápidamente.

Este proceso también lo podemos asociar a la sistematización de tareas, aunque no siempre la sistematización está asociada a equipo de cómputo si comparte el mejoramiento de los procesos y/o tareas.

2.1.1 EJERCICIO

- Crear un ejemplo en el que se aplique sistematización sin relación de equipos de computo
- Crear un ejemplo con el uso de sistematización aplicado a equipos de computo

> El desarrollo de software nos permite el mejoramiento de tareas o procesos así como el entretenimiento y aprovechamiento de los recursos que de otra manera nos demandarían más tiempo (http://www.monografias.com/trabajos39/desarrollo-del-software/desarrollo-del-software.shtml)

2.2 TEMA 2 METODOLOGIAS DEL DESARROLLO DE SOFTWARE

Existen múltiples herramientas, métodos, procesos que nos permiten una adecuada comprensión del desarrollo de software, la pregunta a esto es ¿Cuál es el mejor?, muchos nos dirán que el que esté de moda, otros nos dirán que el más utilizado y otros no dirán el más antiguo, y podemos decir que todos tiene la razón, desarrollar software no es un proceso simple, y cualquiera de las metodologías son válidas, ahí veremos pros y contras de cada uno de ellos, sus alcances, su actualidad, además de características que se manejan y las tendencias del presente y futuro.

Programación Estructurada:

Es el tipo de programación más común, además de ser la forma más antigua, tiene dentro de sus características básicas el crear procesos de manera vertical y sin aprovechamiento de rutinas que se puedan reutilizar, tiende a ser extenso, de múltiples archivos, cada uno para un propósito distinto de la misma aplicación, algunos lo catalogan como código spaguetti, por su cantidad de código que en algunos cosas lo vuelve "enredado" por los procesos que hay que realizar, su único requerimiento es la lógica que cada desarrollador desee aplicar y una herramienta apropiada para esta tarea, para dicha tarea se puede utilizar lenguajes como Basic, C o Cobol

Programación Orientada a Eventos

Esta programación Orientada a Eventos es una mejora de la programación Estructurada, su principal característica la encontramos en ambientes gráficos, esto hace mucho más fácil el desarrollo con herramientas RAD (Rapid Application Development) Desarrollo de Aplicaciones Rápidas, este tipo de lenguajes de programación permiten que el diseño del aplicativo se construya rápidamente por su cantidad de opciones que permiten que tenga una presentación adecuada en corto tiempo, además de que permiten un desarrollo no secuencial, esto nos indica que el desarrollador puede iniciar su tarea en cualquier componente del diseño, sin cumplir un orden especifico como lo hace la programación estructurada, esta programación va ligado también a periféricos como el mouse y el teclado, en el que encontramos eventos como click, doble click, arrastrar mouse, seleccionar, y la presión de

ciertas tablas como enter, esc, tabulador, todo este tipo de opciones permiten que el usuario trabaje de una forma cómoda con opciones comunes de su equipo de cómputo.

Programación Orientada a Objetos:

Es la forma de desarrollo más común en la actualidad, se inició a finales de los años 80's, y sigue vigente por su gran cantidad de herramientas, su principal característica es la reutilización de código, situación que permite hacer de la programación más corta y más eficiente que cualquier otro método de desarrollo, es además el método utilizado por todos los lenguajes de programación de última generación, tiene procesos que mezclan lo mejor de la programación orientada a eventos y los recursos de la orientación estructurada que se utiliza en todos los métodos de desarrollo, simplemente que utilizándolos de la manera apropiada. Hay que tener presente que es un lenguaje conceptual a diferencia de los anteriores, depende de un buen manejo de sus características para aprovechar su potencial, de no ser así estamos incurriendo en programación estructurada o de orientación a eventos, el inconveniente en este caso es su falta de unificación de algunos criterios lo que hace que se considere todavía un paradigma.

Programación Orientada a aspectos

Este tipo de programación o arquitectura es relativamente nuevo, es uno de los lenguajes con características que se pretenden implementar a futuro, teniendo en cuanta que son complemento de las metodologías anteriores, la orientación a aspectos tiene como finalidad la eliminación de incumbencias para gestionar la complejidad del desarrollo de software separando los procesos principales del aplicativo de otros procesos que lo conformen, se tienen algunas características importante como un nivel de abstracción más alto, mayor facilidad en su estructura, mayor reusabilidad, mejores opciones en el mantenimiento del código, flexibilidad y productividad,

2.2.1 EJERCICIO

- Identifique en su medio laboral cuales aplicativos tiene y su metodología de desarrollo según lo que se pueda apreciar en su funcionamiento.

- Cree un cuadro comparativo en el que pueda ver claramente los pro y contras de cada una de estas metodologías.

> Existen múltiples tipos de programación, todas son validad y todas se pueden utilizar de distintos modos dentro de las tendencias nuevas, identifique un poco más de este tema. (http://www.desarrolloweb.com/articulos/2477.php)

2.3 TEMA 3 COMPONENTES DE LA ORIENTACIÓN A OBJETOS

Como se ha mencionado anteriormente la programación orientada a objetos, es el único de los métodos de programación que para ser utilizada correctamente se deben utilizar y conocer los conceptos básicos de sus componentes, dentro de los cuales podemos mencionar.

Clase: es un procesos abstracto que representa la función de una tarea, además de ser el administrador de recursos como variables públicas, protegidas y privadas, métodos y objetos, es la tarea principal dentro de la programación orientada a objetos en lo administrativo por contener y distribuir los procesos de la manera adecuada.

Herencia: es la herramienta por la cual se pueden "compartir" procesos, métodos, objetos con el fin de optimizar las características haciendo uso de componentes ya creados, esto permite una reducción de código y aprovechamiento de tareas ya establecidas.

Objeto: es una entidad que contiene parámetros o atributos que interactúa directamente con los métodos, esta consecuentemente reaccionan a eventos, su característica principal es que tienen un conjunto de características que los hace únicos dentro del tipo de programación.

Método: es el algoritmo como tal, es la secuencia lógica que se desea aplicar al desarrollo, puede contener procesos como variables, condiciones y procesos repetitivos.

Mensaje: es la comunicación dirigida a un objeto, indica que se invoquen los métodos con los parámetros asociados a él.

Polimorfismo: es la posibilidad de que un aplicativo contenga varias clases, y cada una de estas tenga métodos con el mismo nombre, con distintos parámetros y funcionalidad.

Sobrecarga: es la posibilidad de que una clase tenga varios métodos con el mismo nombre y distinta cantidad de parámetros o funcionalidad.

2.3.1 EJERCICIO

- Identifique con las palabras más comunes de la orientación a objetos casos de la vida cotidiana sin asociarlo a sistemas de cómputo.

> **Dentro de la programación orientada a objetos se encuentra un sinnúmero de opciones que debidamente aplicadas darán un resultado óptimo y adecuado de la tarea, la no aplicación de estos conceptos nos puede llevar a un desarrollo estructurado.**
> **(http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos)**

3 UNIDAD 2 INTRODUCCIÓN A PERL

RELACIÓN DE CONCEPTOS

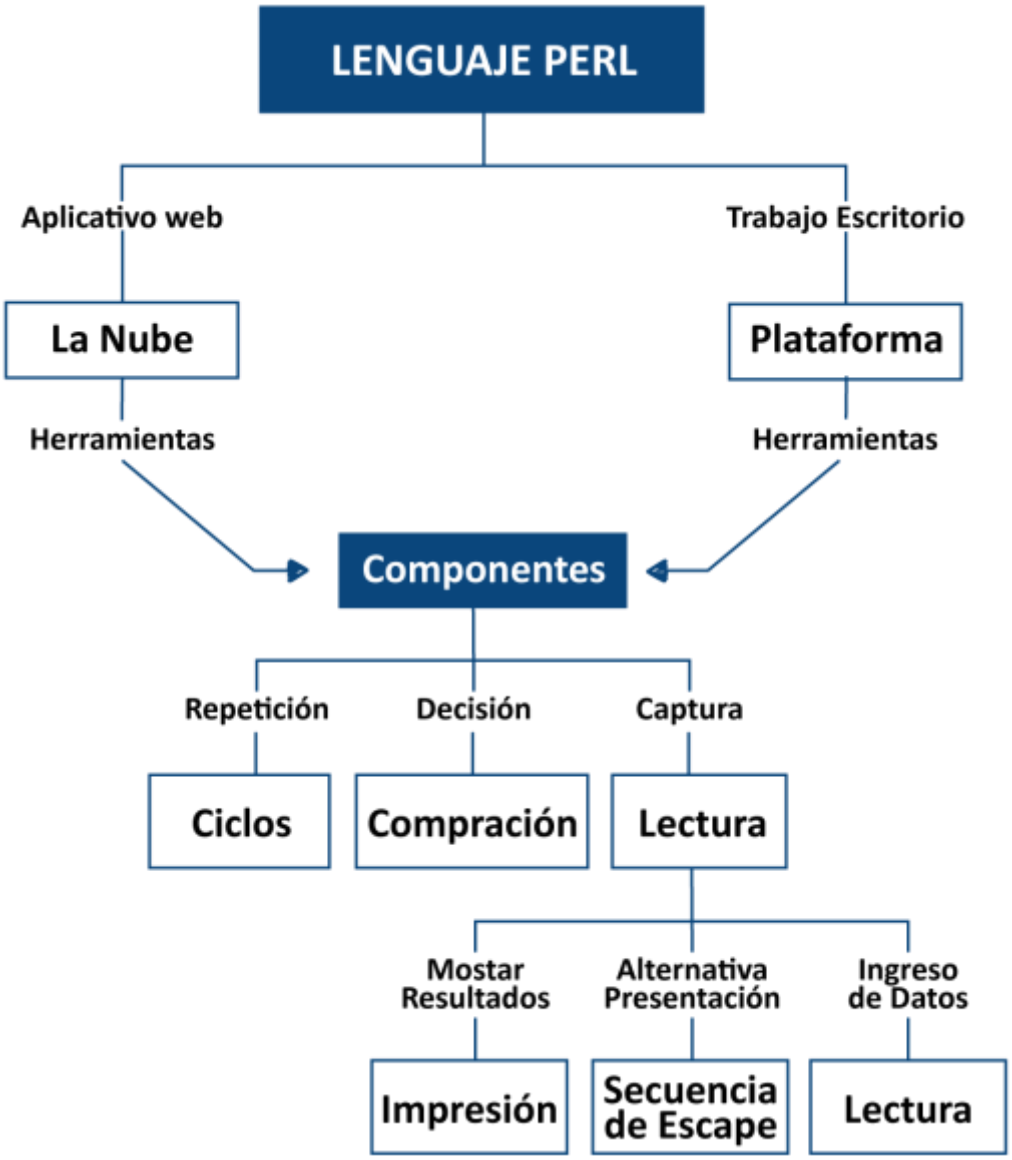


TABLA DE CONCEPTOS		EJEMPLO	
Concepto	Definición		
Perl	Perl es un lenguaje de programación creado en los años 80's, su principal características es la orientación a objetos y el uso en la web	Muchas de las aplicaciones que hoy se usan están orientadas a la web, transacciones bancarias, pago de servicios, consultas, bibliotecas, reportes empresariales, consulta de notas, etc, es la tendencia que hoy vivimos trabajar almacenando en espacios virtuales	
La Nube	Termino que se está utilizando con frecuencia a aplicación que operan desde la web en lugar de discos duros locales		
Concatenación	Es la posibilidad de unir un texto con una variable, aprovechando espacio y tiempo	Mediante procesos lógicos encontramos la solución a ciertos problemas, mediante la utilización de comandos y funciones de un lenguaje de programación podemos realizar procesos repetitivos y condicionales, esto facilita la tarea que vamos a aplicar.	
For	Ciclo para, proceso de repetición automática		
While	Ciclo mientras que, proceso de repetición condicionada		
If	Si, función de evaluación de condiciones		
STDIN	Comando de lectura o captura de información		
Secuencias escape	Conjunto de opciones que permiten realizar tareas que de otra manera implicaría más comandos o funciones		
operadores	Se clasifican normalmente en aritméticos, relacionales y booleanos, es una forma		

	práctica de operar y comprara información.	

OBJETIVO GENERAL

Identificar la primera etapa de desarrollo de software, conociendo las estructuras básicas basadas en la lógica del semestre previo, la forma de transcribir de papel a pc, que este lo identifique y lo muestre con las características que el pc tenga para dicha tarea.

OBJETIVOS ESPECÍFICOS

Conocer las características más importantes del lenguaje, su alcance, y las herramientas más comunes de uso para desarrollar sobre este lenguaje de programación.

Utilizar las herramientas propias del lenguaje, situaciones típicas de manejo de un lenguaje de programación como son la captura (leer información) y la impresión de resultados (mostrar), esta es parte de la razón de ser de muchos de los aplicativos que debemos desarrollar a lo largo de nuestra experiencia en el desarrollo de software.

Conocer las palabras claves (restringidas) que tiene el lenguaje de programación Perl, esto nos ayudara a conocer características, métodos, formas de escritura, y comandos o funciones que no debemos tomar como variables propias, situación que ocasionaría inconvenientes en su etapa inicial, además de la posibilidad de realizar concatenaciones utilizando temas previamente vistos y las sentencias de escape que permitirán la distribución apropiada de la información en la impresión de resultados.

Identificar los operadores típicos en programación como son los aritméticos, relacionales y lógicos o booleanos, estos nos permitirán la comparación, los cálculos que deseamos aplicar a nuestro ambiente lógico.

Conocer de primera mano los procesos lógicos que podemos aplicar tales como ciclos y condiciones que nos permitirán utilizar funciones repetitivas en distintas formas y condicionales simples o compuestas.

3.1 TEMA 1 HISTORÍA, CARACTERÍSTICAS Y HERRAMIENTAS DE USO

Perl es un lenguaje de programación orientado a objetos (basado en clases), creado en 1987 por Larry Wall, un programador de Unisys, está influenciado por características de lenguaje C, y ha influenciado a otros más recientes como Python o PHP, es además un ambiente multiplataforma, que da gran potencia de uso por los múltiples sistemas operativos en los que puede operar sin cambio alguno en la codificación, se estipula que puede operar en más de 100 sistemas operativos pero con una base muy sólida en Linux, hay que tener presente que es además un ambiente gratuito.

Perl es un lenguaje de propósito general, aunque parte de su éxito se debe a la web, es quizás el lenguaje más utilizado actualmente para propósitos de seguridad bancaria por su conjunto de elementos y símbolos, estas características han creado una denominación adiciona que se conoce como CGI que tiene una licencia GPL **(**GNU General Public License)

El CGI (Common Gateway Interface*)*, es una de las características más importantes en la web por la posibilidad de transferencia de datos de un cliente a un aplicativo.

Una de las características más importantes en el procesamiento de su código se refiere a los pocos recursos que requiere para su implementación, solo se necesita del aplicativo que se puede descargar del sitio http://www.perl.org/ y seleccionando el activeState Perl para el sistema operativo que desee implementar, este lenguaje no tiene un IDE (integrated development environment o ambiente de desarrollo integrado) pero permite que se codifique en cualquier tipo de editor partiendo de un block de notas hasta editores más profesionales que permiten explotar todas sus características.

El Perl tiende a ser un lenguaje muy simple en comandos y funciones lo que facilita la programación a individuos que estén iniciando en esta labor y da bases para lenguajes de mayor envergadura por los comandos y funciones comunes que tiene con ambientes como el C.

3.1.1 EJERCICIO

- Descargar el aplicativo e instalarlo en el equipo de cómputo, igualmente busque y descargue un editor de su agrado, configúrelo para un uso adecuado del lenguaje de programación.

> El lenguaje Perl ha sido uno de los más importantes dentro de industria Web, permitiendo dinamismo entre usuario y equipo de trabajo. (http://perlenespanol.com/)

3.2 TEMA 2 IMPRESIÓN Y CAPTURA DE INFORMACIÓN

Cuando nos introducimos en un ambiente de programación hay que tener presente que la principal característica que nos dará éxito en la herramienta es la lógica, la lógica vista desde el punto de análisis, de adecuación a necesidades de la vida cotidiana, la que nos ayuda a resolver problemas, esta lógica nos da suficientes alternativas de trabajo y un aprovechamiento de las tareas y formas de procesar la información en un sistema de cómputo, pero cuando se llegó a esta etapa se dieron unas pautas que nos darán el camino para un buen desarrollo de software.

Estos pasos son:

Planteamiento del problema: este es quizás el más importante porque nos dará las pautas y necesidades, en el analizaremos las entradas, los cálculos, las salidas de información.

Clasificación: la clasificación nos permite identificar que desea el cliente, como lo desea y para que lo desea, con esto podemos "imaginarnos "que procesos lógicos vamos a aplicar.

Diagrama: el diagrama es una herramienta lógica que permite en muchas situaciones llevar un proceso lógico coherente, con el paso a paso del problema a resolver, pensando en una forma simplificada de realizar la tarea que deseamos.

Prueba de escritorio: fundamental para un buen desarrollo futuro, las pruebas de escritorio nos dan el aval de continuar con las tareas pendientes, son los que nos indican si un proceso cumple la normatividad del planteamiento, que los resultados y procesos cumplan la condición apropiada antes de continuar con cualquier otra tarea, nos indicara los posibles erros o fallas que se pueden presentar en comparaciones, cálculos o resultados.

Codificación: teniendo conocimiento en un lenguaje de programación, podemos traducir nuestro diagrama a uno de estos con sus palabras claves, comandos y funciones.

Digitación: mediante un editor de texto simple o uno más elaborado digitamos la información previamente codificada, esto proceso va acompañado de todas la palabras claves que el lenguaje reconozca para la codificación que tengamos.

Compilación: es el proceso mediante el cual, el lenguaje de programación corrige la sintaxis, los comandos, las funciones, busque que todas sus normas se estén cumpliendo, hay que tener presente que si no hay errores de compilación, no quiere decir que la lógica sea la correcta, viéndolo desde otro punto, si tenemos un editor de texto como Word, si nos aparecen palabras mal escritas (mala ortografía) y lo corregimos mediante el asistente, no nos indica que el texto sea comprensible, solo que no tiene errores ortográficos, esto es lo que sucede con un compilador, es de anotar que se depende de una buena lógica, diagrama y prueba de escritorio para determinar esto.

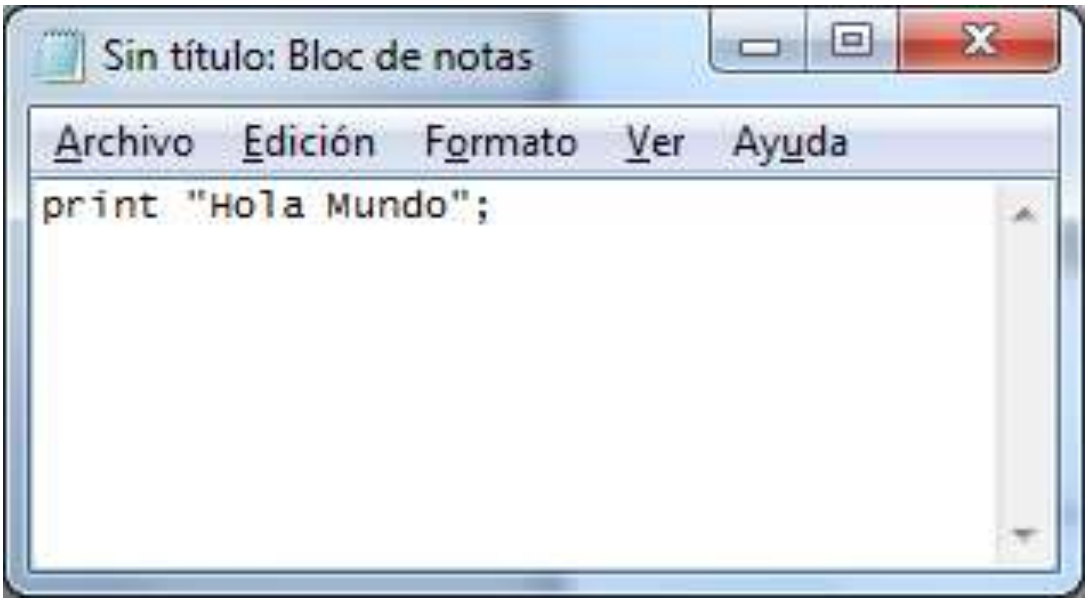
La gran mayoría de lenguajes discriminan muy bien los archivos fuentes (código generado) y los archivos objeto (archivo compilado), pero los lenguajes interpretados no generan este segundo proceso, se interpreta mediante el lenguaje pero no crea archivos adicionales.

Ejecución: la ejecución es el fin de los procesos, nos permite poner a prueba todo el proceso que ya hemos creado mediante un lenguaje de programación, en este podremos ingresar información según la lógica aplicada al programa, realizar cálculos y ver los resultados en pantalla.

Perl es un ambiente multiplataforma y muchos de estos ambientes tienen una serie de normas muy rigurosas y muy similares a el lenguaje de programación C, dentro de estas normas encontramos que la gran mayoría del comando se escriben en minúscula, diferencia entre mayúsculas y minúsculas, y las instrucciones terminadas en punto y coma (;).

La captura e impresión de resultados o mensajes en un lenguaje de programación es crucial, no por el proceso lógico como tal, sino por la orientación que se le da al usuario e indicarle que debe de realizar.

Después de instalar el Perl, ingrese al block de notas (notepad) o cualquier editor de texto plano, dentro de él utilizaremos el comando print que nos permitirá mostrar la información que deseamos.



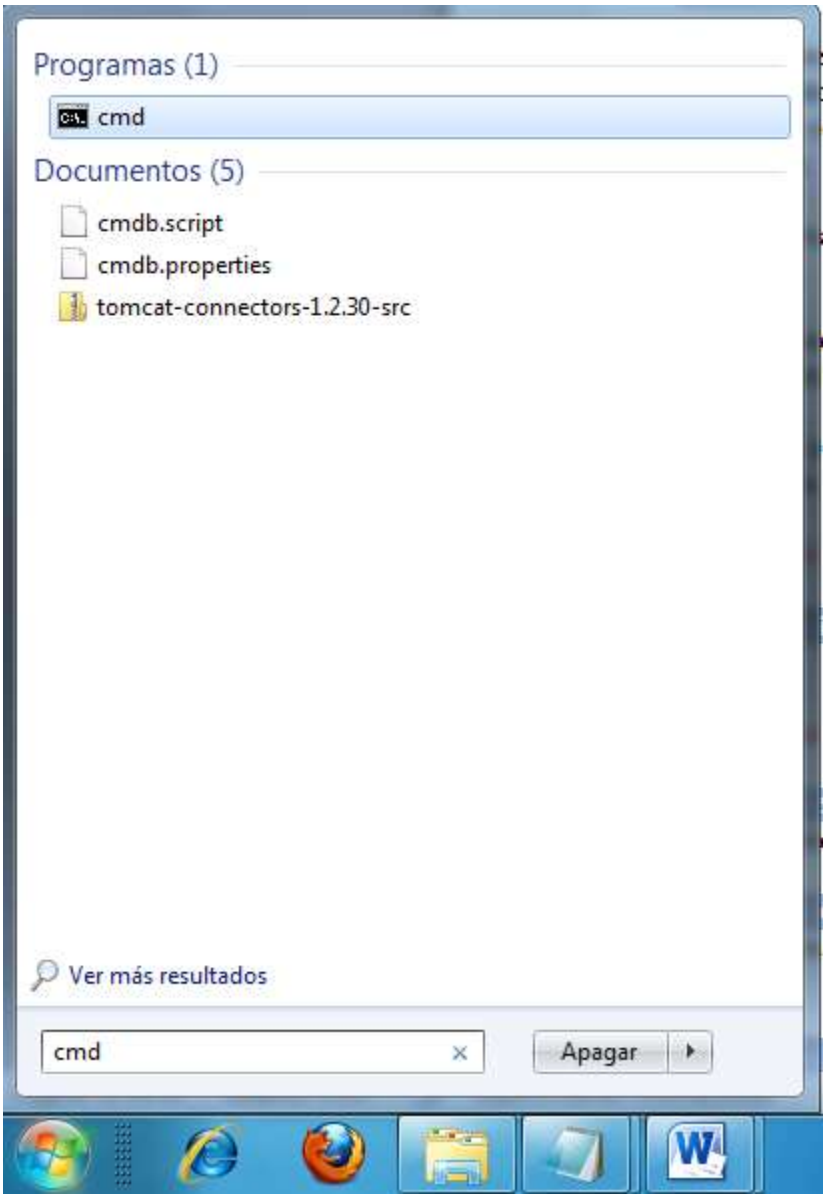
Elaborado por: César A. Jaramillo H.

Observemos que es un proceso muy simple la opción impresa, ahora, el Perl es un lenguaje interpretado que permite que muestre la información de una forma simple, teniendo el texto digitado o codificado, lo almacenamos, este debe tener la extensión .pl, la ubicación puede ser cualquiera.

Cuando almacenamos un archivo .pl, este genera un icono que representa los archivos de Perl.

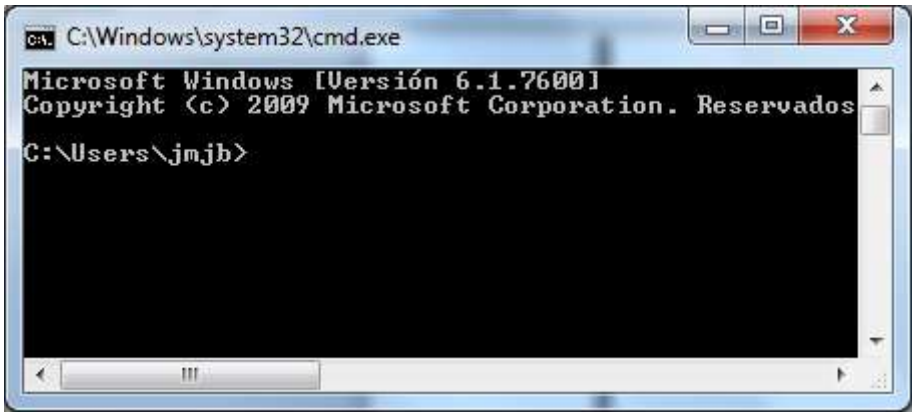
Puede aparecer de distintos colores pero veremos este, en caso de que no aparezca icono no hay problema si la extensión cumple el formato.

Ejecución: para la ejecución de este aplicativo ingresamos al ambiente de texto D.O.S, en inicio del sistema, digitamos cmd.



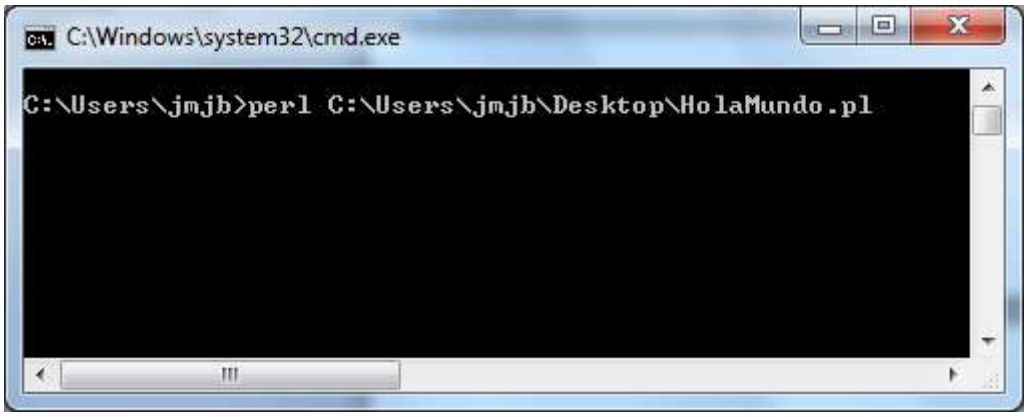
Elaborado por: César A. Jaramillo H.

Cuando ingresamos al sistema veremos una pantalla así:



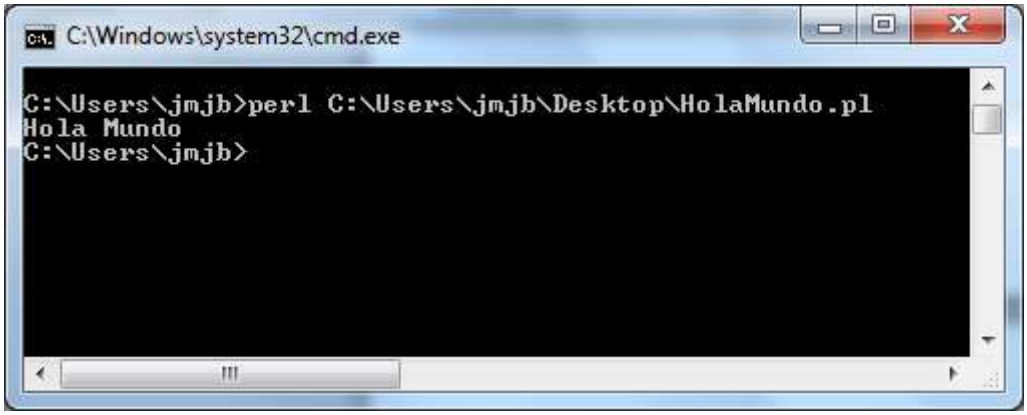
Elaborado por: César A. Jaramillo H.

Este, será nuestro ambiente de trabajo, en el colocamos la palaba Perl, y el archivo con su ubicación respectiva.



Elaborado por: César A. Jaramillo H.

Y procedemos a presionar enter



Elaborado por: César A. Jaramillo H.

Y observamos el resultado según lo que hayamos digitado como mensaje de salida o impresión.

En caso de que aparezca algo distinto debemos observar el archivo original creado en block de notas (notepad) y revisar la sintaxis de escritura.

Captura de datos:

La captura de información hace parte fundamental del proceso de programación, no es que todos los programas lo lleven, pero un gran porcentaje lo requiere para indicarle al usuario que debe realizar.

Planteamiento:

Crear un programa que lea su nombre y lo imprima

Lógica:

Inicio

Mostrar “digite su nombre”

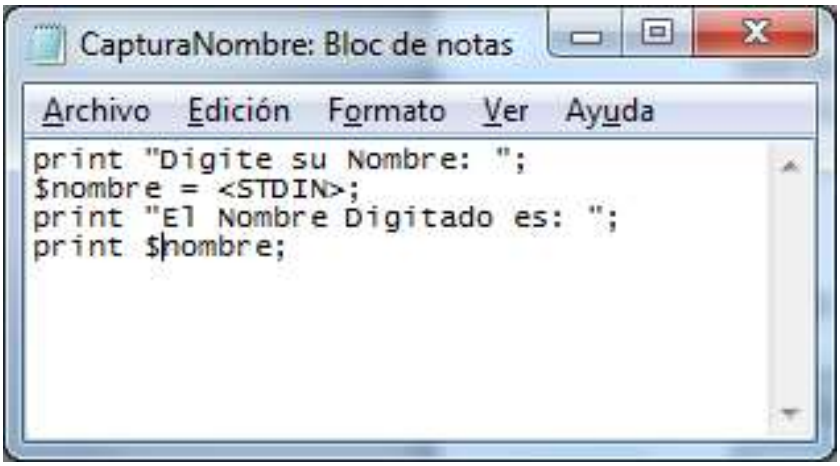
Lea nombre

Mostrar “el nombre digitado es”

Mostrar nombre

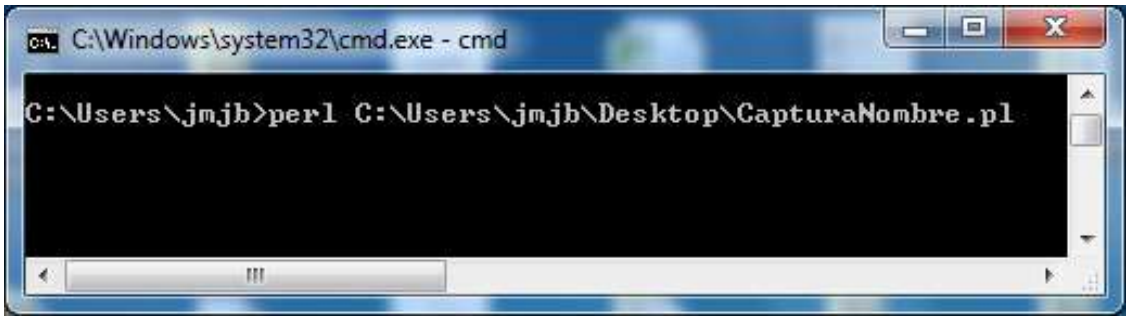
Fin

Luego codificamos el programa así

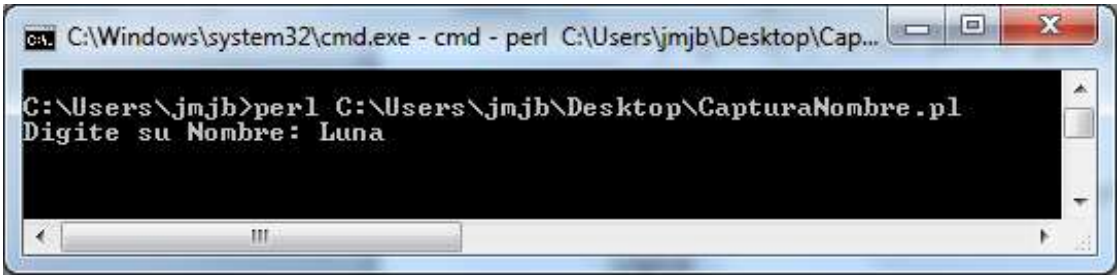


Elaborado por: César A. Jaramillo H.

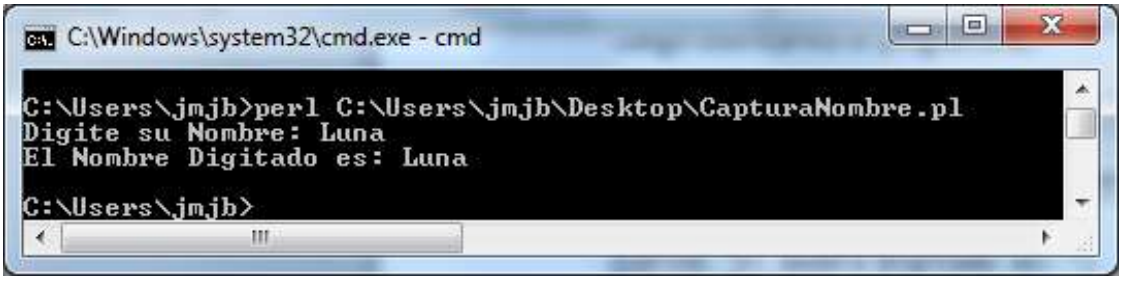
Lo guardamos con extensión .pl y lo ejecutamos



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.

Tenemos la captura de una variable, observemos varios aspectos, primero, el programa se amplía en codificación, a mayor número de variables mayor cantidad de procesos por realizar, segundo, observemos que utilizamos la sentencia <STDIN>, mediante este proceso podemos capturar información, sus 2 ultimas letras indica IN (ingreso), y esto se asigna a una variable llamada nombre, es muy importa reconocer que Perl distingue mayúsculas de minúsculas, esto indica que no es lo mismo colocar nombre que colocar NOMBRE que colocar Nombre, son 3 variables completamente distintas, si cometemos el error de capturar de una forma e imprimir de otra no nos mostrara ningún resultado, todas las variables tienen la característica de que inician por el carácter pesos ($), esto indica que es una variable de usuario, si no se coloca, el sistema no nos mostrar ningún error pero tampono nos procesara la información.

Las variables pueden iniciar por $ acompañado de un underline (_), o de $ con el nombre que deseemos darle, los demás caracteres para iniciar una variable no son válidos.

3.2.1 EJERCICIO

- Crear un programa que lea su nombre, edad y estatura, mostrar los resultados de los datos ingresados.
- Crear un programa que lea su profesión, cargo, salario y empresa donde labora, imprimir estos datos.

> La programación Perl es muy diversa, permitiendo dentro de sus características más básicas el ingreso e impresión de información.
> (http://perl.astalaweb.net/_inicio/Portada.asp).

3.3 TEMA 3 PALABRAS CLAVES, SECUENCIAS DE ESCAPE Y CONCATENACIONES

Los lenguajes de programación tienen dentro de su amplio portafolio una seria de palabras que son reservadas y que no pueden ser utilizadas por el usuario desarrollador para tareas propias como variables, estas palabras tienen una finalidad específica y se debe respetar, de lo contrario nos arrojara un error que no es de sintaxis ni de lógica y en desarrollares noveles será un inconveniente encontrar el error.

abs: devuelve el valor absoluto de la expresión pasada.

chmod: cambia los permisos de los ficheros dados.

chop: recorta y retorna el ultimo carácter de una cadena.

chown: cambia el propietario de los ficheros dados.

close: cierra un fichero. cos: devuelve el coseno del ángulo dado en radianes.

defined: sirve para comprobar si existe una variable, formato, subrutina, etc..

delete: borra un valor de un array asociativo a través de su clave.

die: imprime en la salida del error estándar un mensaje pasado como parámetro

Cuando ocurre un error en la ejecución de una sentencia.

eof: retorna verdadero si el final del fichero dado.

eval: evalúa la expresión pasada como si se tratase de un pequeño programa Perl.

exec: ejecuta lo que pasemos como parámetro y sale del programa.

exit: hace que salgamos del Perl script devolviendo al sistema operativo el valor pasado

Como argumento.

exp: retorna el numero e elevado a la potencia pasada como parámetro.

fileno: devuelve el descriptor del manejador del fichero pasado como parámetro.

fork: realiza una llamada fork. getc: lee el siguiente carácter del fichero especificado.

hex: devuelve el valor decimal del numero hexadecimal pasado como parámetro.

index: devuelve la posición de la primera ocurrencia de una cadena en otra.

int: devuelve la parte entera del parámetro pasado.

join: une las cadenas pasadas como argumento con un separador también pasado

como argumento.

keys: devuelve todas las claves de un array asociativo.

length: devuelve la longitud en caracteres del parámetro pasado.

local: declara como locales las variables pasadas como argumentos.

log: devuelve el logaritmo del numero dado.

mkdir: crea un directorio en el camino dado.

oct: devuelve el valor decimal del numero octal pasado como parámetro.

open: abre el fichero dado asociándole un manejador de fichero especificado también

como parámetro.

pop: retorna y borra el último elemento del array dado.

print: muestra en la salida standard o en el fichero especificado la expresión dada.

push: añade el valor dado al final del array pasado como parámetro.

rand: devuelve un numero aleatorio entre 0 y el valor pasado como argumento.

read: un determinado número de caracteres desde el fichero pasado como argumento.

rename: sirve para renombrar un fichero.

require: sirve para incluir código externo en nuestro guion.

return: devuelve un valor desde una subrutina.

rmdir: borra un directorio.

seek: sitúa un puntero a fichero en un lugar determinado.

select: sirve para seleccionar el manejador de fichero que será utilizado por defecto para la salida de los comandos o funciones que no especifiquen un determinado manejador de fichero como parámetro.

shift: devuelve el primer valor del array dado borrándolo posteriormente.

sin: devuelve el seno del ángulo pasado en radianes.

sleep: causa que el Perl script o guion se detenga el número de segundos especificados.

sort: ordena el array dado.

split: divide una cadena en subcadenas según el separador especificado.

sqrt: devuelve la raíz cuadrada del numero pasado.

system: igual que exec pero no se sale del Perl script.

tell: devuelve la posición actual del puntero a fichero del manejador de fichero especificado.

values: devuelve todos los valores del array asociativo dado.

write: escribe un registro con formato en el fichero asociado a ese formato.

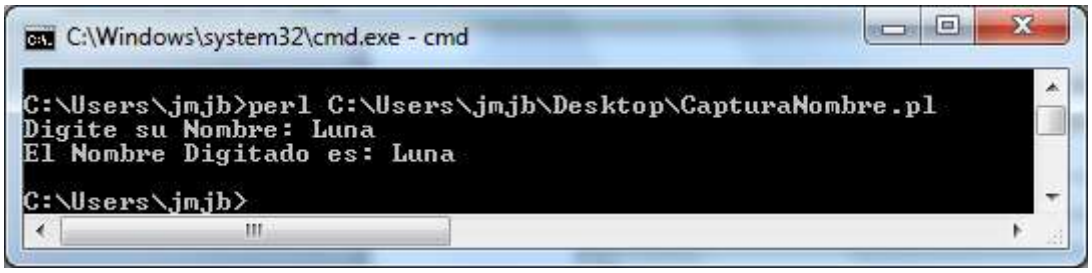
Tomado del sitio: http://www.programacionfacil.com/perl_script:palabras_reservadas

Secuencias de escape: las secuencias de escape son herramientas que permiten realizar tareas como espacios, tabulación, mostrar comillas o valores que de otra manera no se podrían realizar.

CARÁCTER	FUNCIÓN
\t	Tabulación
\r	Salto de línea sin retorno de carro
\n	Salto de línea con retorno de carro
\f	Salto de pagina
\a	Sonido
\e	Escape
\b	Retroceso del cursos
\0nn	Carácter en octal
\xnn	Carácter en hexadecimal
\”	Comillas
\\	Slash
\’	Comilla sencilla

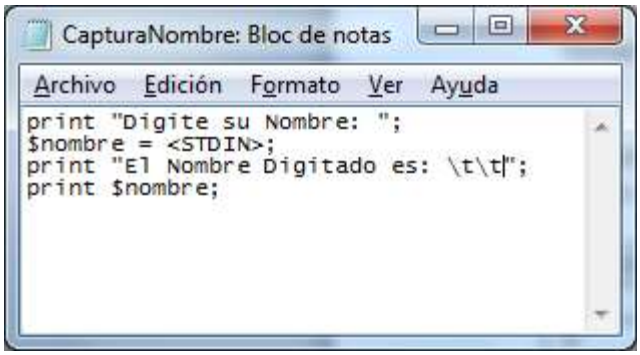
3.3.1 EJEMPLO

Este es el formato por defecto que toma el sistema



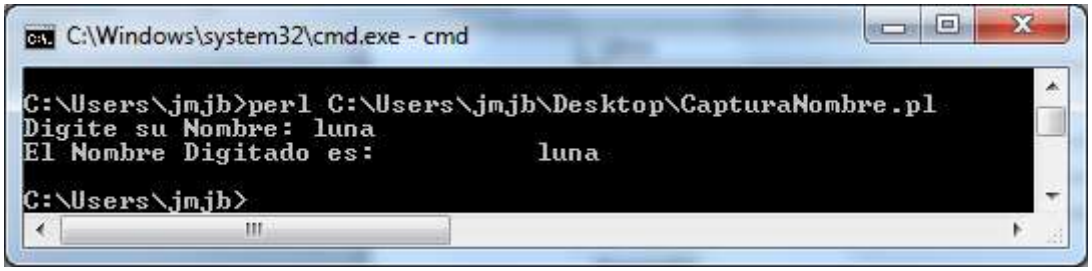
Elaborado por: César A. Jaramillo H.

Este es el formato que toma con las secuencias de escape



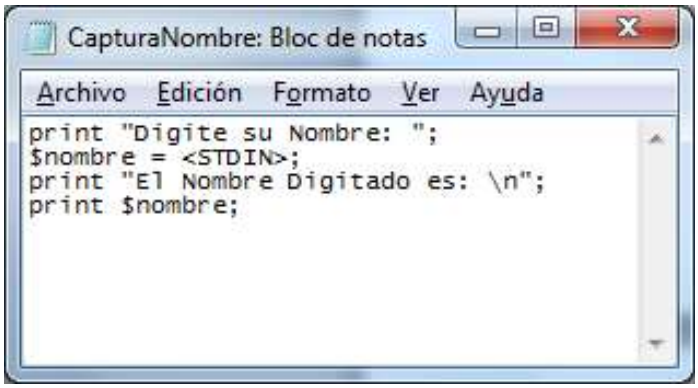
Elaborado por: César A. Jaramillo H.

Aplicamos un doble tabulador y quedaría así



Elaborado por: César A. Jaramillo H.

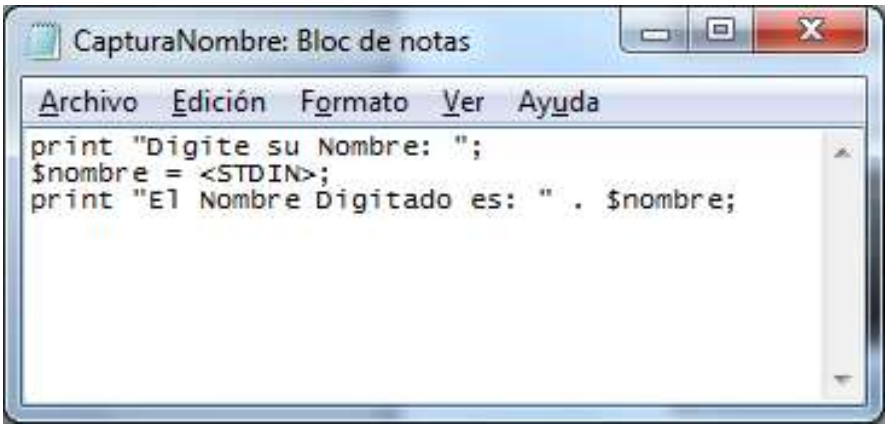
Si deseamos que el nombre se muestre en la línea anterior utilizaríamos lo siguiente:



Elaborado por: César A. Jaramillo H.

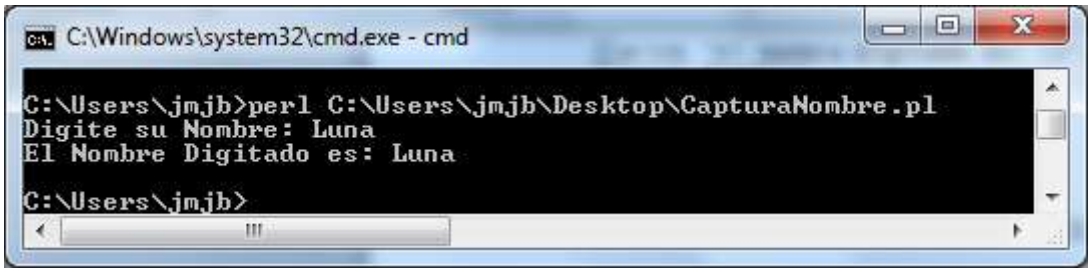
Elaborado por: César A. Jaramillo H.

Concatenaciones: la concatenación es la posibilidad de unir varios procesos en uno solo, podemos unir un mensaje con una variable y con otros mensajes, el símbolo para esto es el punto (.) y parte de su funcionamiento nos permite ahorra espacio y líneas de código.



Elaborado por: César A. Jaramillo H.

Observemos que el nombre está en la misma línea del mensaje, sin utilizar un nuevo print



Elaborado por: César A. Jaramillo H.

3.3.2 EJERCICIO

- Crear un programa que lea su nombre, sexo y edad, mostrar los mensajes de impresión y debajo de estos los valores.

- Crear un programa que lea su nacionalidad, provincia y ciudad de nacimiento, capturar la información debajo del mensaje que indique esta operación y mostrar los resultados al frente del mensaje correspondiente.

> Las secuencias de escape y otras funciones son de vital importancia por la presentación que se da en ambientes simples como el manejo de texto.
>
> (http://www.programacionfacil.com/perl_script:palabras_reservadas)

3.4 TEMA 4 OPERADORES ARITMETICOS, RELACIONALES Y BOOLEANOS

Los diferentes procesos que se presentan tanto en el manejo lógico como en la programación, nos llevan a manejar una serie de alternativas que permite que se puedan realizar más adelante comparación o cálculos, estos procesos se llaman operadores y los más comunes son los aritméticos, relacionales, y los lógicos o booleanos.

Operadores Aritméticos

OPERADOR	FUNCIÓN
+	Suma
-	Resta
*	Multiplicación
/	División
%	Modulo
Sqr	Potencia al cuadrado
**	Potenciación
Sqrt	Raíz cuadra

Relacionales

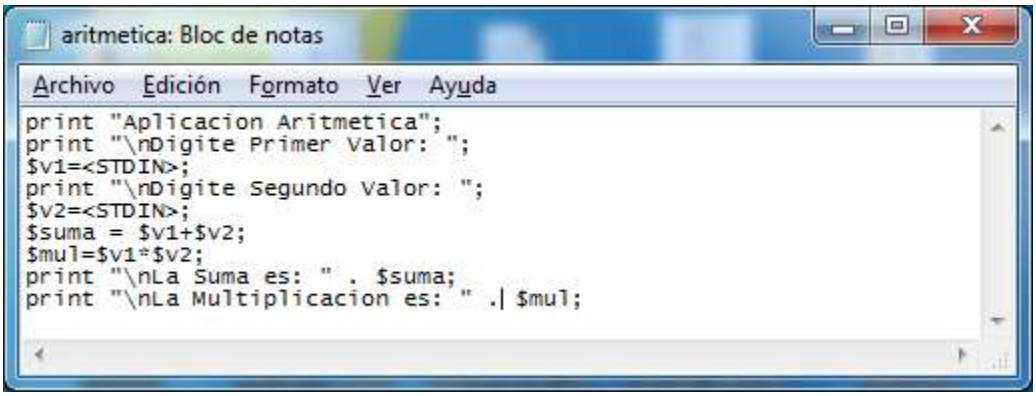
OPERADOR	NUMÉRICO	CADENA
Igualdad	==	eq
Diferente	¡=	ne

Menor que	<	lt
Menor o igual que	<	le
Mayor	>	gt
Mayor o igual que	>=	ge

Operadores Booleanos o Lógicos

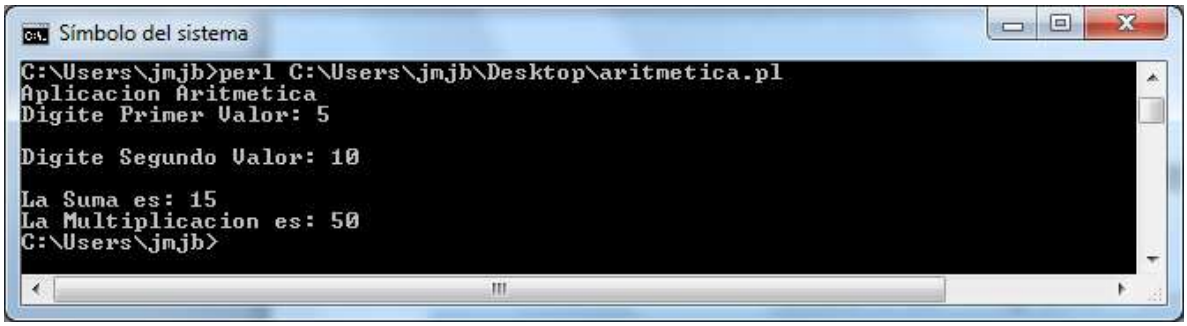
OPERADOR	SIGNIFICADO
&&	Y
||	O
¡	No

Ejemplo: se desean capturar 2 valores numéricos, calcular la suma y la multiplicación de ellos.



Elaborado por: César A. Jaramillo H.

Acá aplicamos comando print, secuencias de escape y cálculos matemáticos



Elaborado por: César A. Jaramillo H.

3.4.1 EJERCICIO

- Consultar los operadores adicionales que se pueden aplicar a un desarrollo en Perl, tales como incrementos unitarios, prefijos y sufijos

> Los operadores de Perl, nos permiten un manejo de procesos de comparación y cálculos matemáticos (http://flanagan.ugr.es/perl/datos.htm).

3.5 TEMA 5 ESTRUCTURAS LÓGICAS

Los procesos lógicos nos permitirán utilizar comandos y funciones que evalúan y repiten tareas, procesos por los cuales será más fácil realizar alguna operación y cálculos, los procesos lógicos nos permitirán ahorrar codificación y procesos esenciales en la aplicación que estamos construyendo.

Pls – Plc (preposiciones lógicas simples – preposiciones lógicas compuestas)

Cuando se habla de una preposición se determina una condición o una pregunta, mediante esta herramienta se puede evaluar procesos y tomar decisiones, y aplica de la siguiente manera

Sintaxis

if (condición)

{

Procesos

}

[else

{

Procesos

}]

Preposición lógica Simple

la sentencia if se encargara de evaluar los procesos que deseemos, partiendo de variables o constantes que tengamos en nuestro aplicativo, contiene entre paréntesis una condición que puede ser simple o puede ser compuesta (operadores booleanos), se abren argumentos (llaves) para indicar el inicio y se cierran argumentos (llaves) para indicar el fin del proceso correspondiente a la verdad de la condición expuesta, los corchetes del siguiente proceso indican que una evaluación o condición es opcional, no todas las condiciones tienen una negación, en caso de tenerla se aplica el comando else y cumple las condiciones de argumentos de la primera parte. Dentro de un aplicativo pueden existir múltiples condiciones y se especifica que la misma cantidad de llaves o argumentos que abren debe de ser la misma que cierra.

Ejemplo

Se desea leer para un empleado, el nombre y su edad, determinar si este es mayor de edad o menor mediante un mensaje

Algoritmo

Inicio

Mostrar “ingrese nombre del empleado”

Lea nombre

Mostrar “ingrese edad del empleado”

Lea edad

Si (edad >= 18) entonces

Mostrar “es mayor de edad”

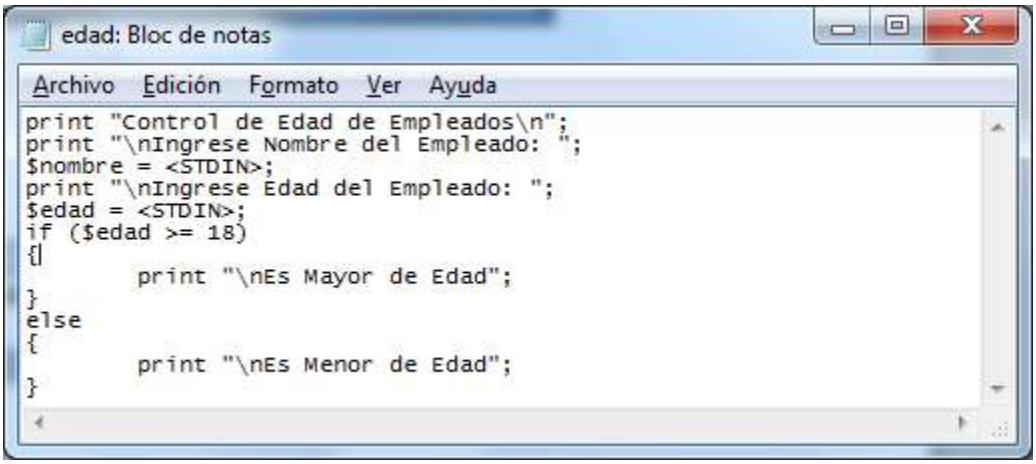
Sino

Mostrar “es menor de edad”

Finsi

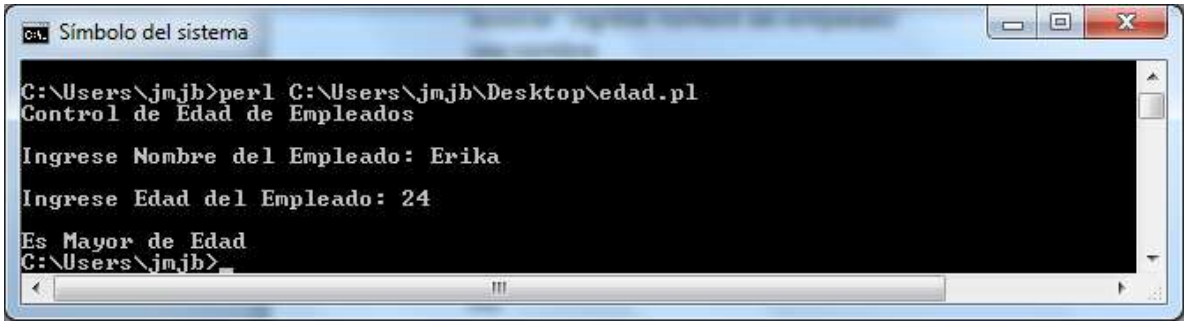
Fin

Codificación

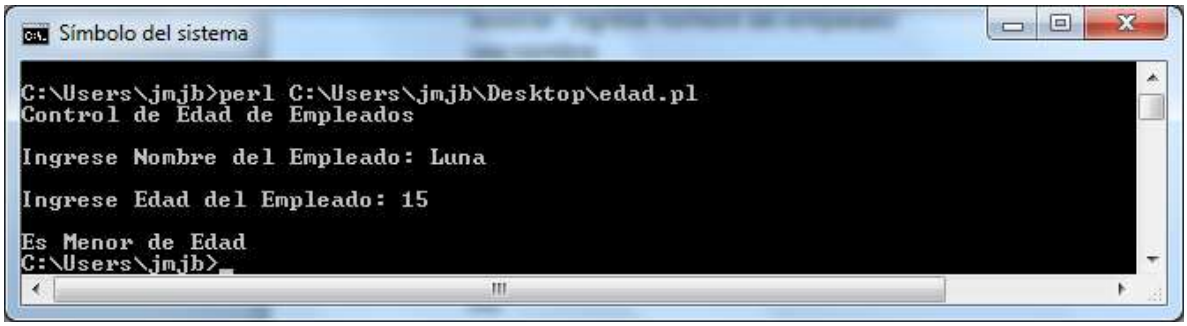


Elaborado por: César A. Jaramillo H.

Ejecución



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.

Preposición Lógica Compuesta

Ejemplo

Una empresa de la ciudad requiere contratar un empleado, la condición que tiene es que este tenga edad entre 20 y 25 años

Algoritmo

Inicio

Mostrar “ingrese nombre del empleado”

Lea nombre

Mostrar “ingrese edad del empleado”

Lea edad

Si (edad >= 20 y edad <= 25) entonces

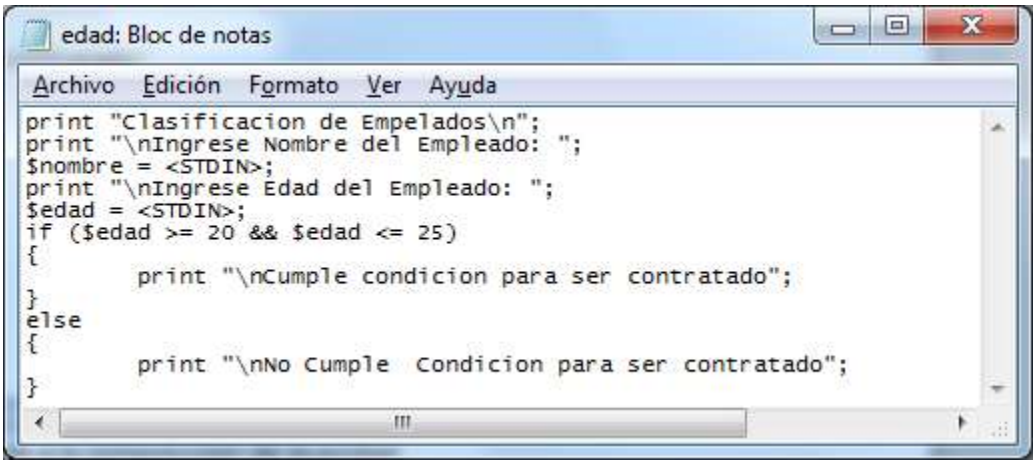
Mostrar “Cumple condición para ser contratado”

Sino

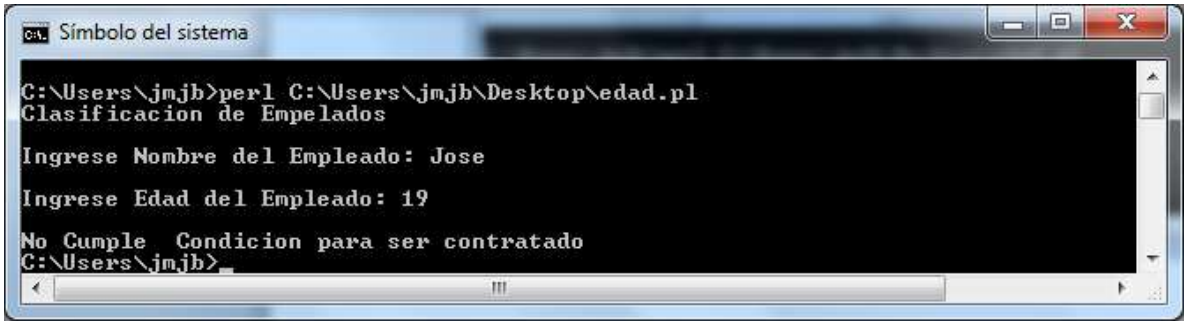
Mostrar “No cumple condición para ser contratado”

Finsi

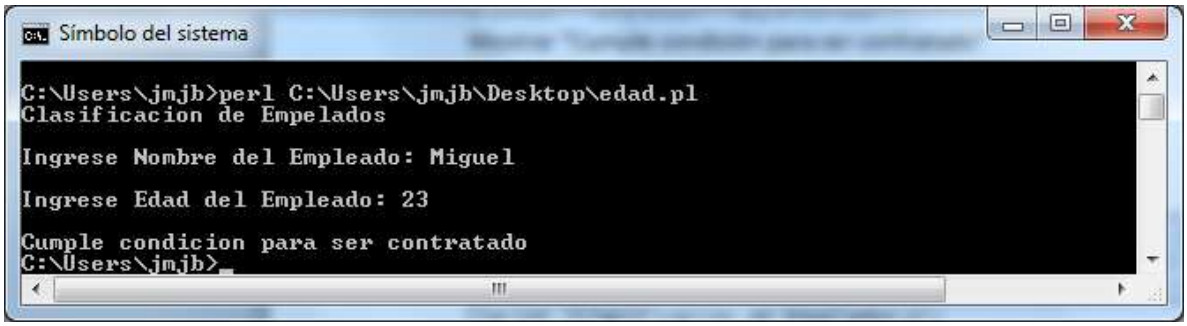
Fin



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.

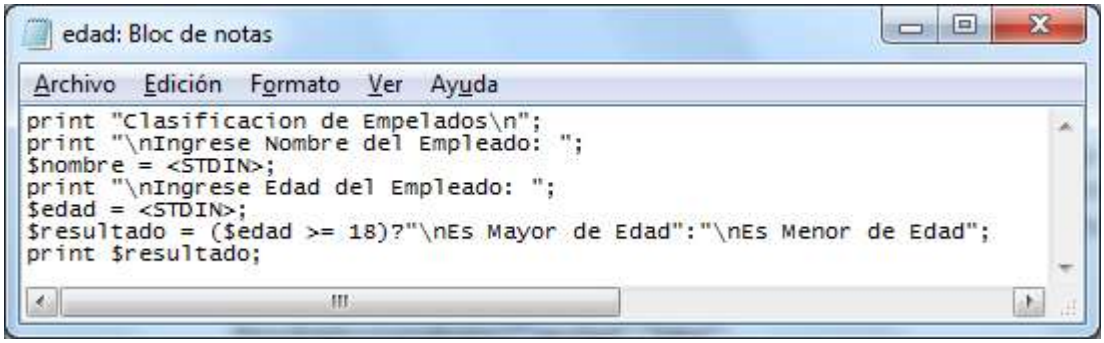
Operador Ternario

El operador ternario es una forma de evaluar mucho más simple que las anteriores, en menor cantidad de líneas, habitualmente utilizan solo una línea, y menor cantidad de argumentos, son procesos muy básicos que permiten establecer una condición de una manera agil.

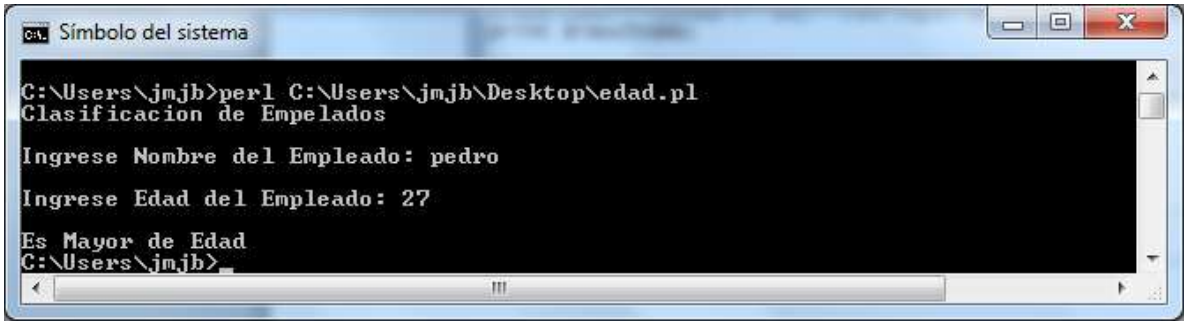
Sintaxis

¿Resultado=(condición)?" verdad": "falso";

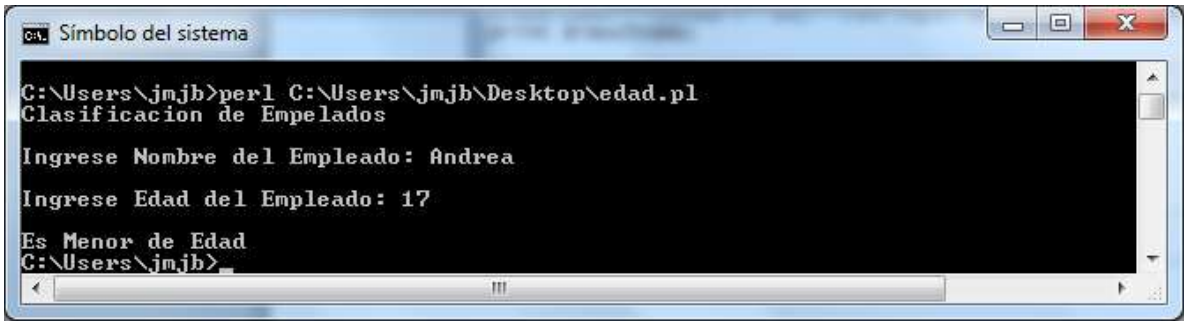
Comprende 4 partes, la primera es la variable donde se almacenara el resultado, la segunda es la condición, la tercera la respuesta en caso de ser verdadera la respuesta y la cuarta la respuesta en caso de ser falso, se presentas 2 símbolos, el primero signo de interrogación para determinar el fin de la condición y el inicio de las respuestas y dos puntos (:) para indicar el inicio de la opción de negación



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.

Bucles – Ciclos – Loops

Hablar de bucles, ciclos o loops es tratar procesos que se pueden repetir, tareas que no evaluaran un una sola condición, sino muchas en las mismas líneas de código que un proceso tradicional.

En los lenguajes de programación existen múltiples ciclos, pero los más comunes son los conocidos como ciclo para y ciclo mientras que, el primero de ellos cumple una condición de ser automático y para una cantidad conocida de datos y el segundo un poco más versátil, es más manual o condicional, además de permitirnos trabajar para una cantidad conocida de datos o desconocida.

Ciclo para

Sintaxis lógica

Para índice = inicio, fin, incrementos

Ciclo para que va desde 1 hasta n con incrementos de 1

Para i = 1, n, 1

Ciclo para que va desde n hasta 1 con decrementos de 1

Para i = n, 1, -1

Ciclo para que va desde 3 hasta 30 con incrementes de 2

Para i = 3, 30, 2

Sintaxis de lenguaje

for (i=1; i<=n; i= i + 1)

En esta opción podemos evaluar que (i) es el índice que inicia en 1, hasta que la i sea menor o igual a n con i llevándole i + 1, esta es la forma de representar el ciclo para

Ciclo para que va desde 1 hasta n con incrementos de 1

for (i = 1;i<=n; i= i + 1)

Ciclo para que va desde n hasta 1 con decrementos de 1

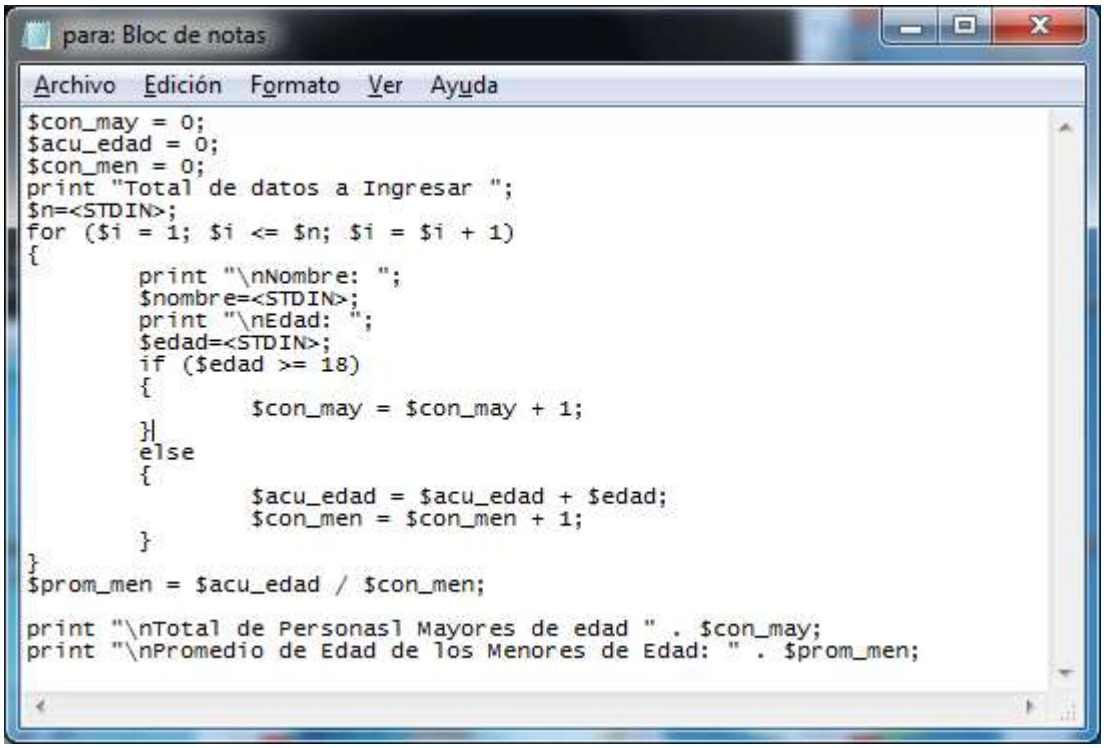
for (i = n; i>= 1; i= i -1)

Ciclo para que va desde 3 hasta 30 con incrementes de 2

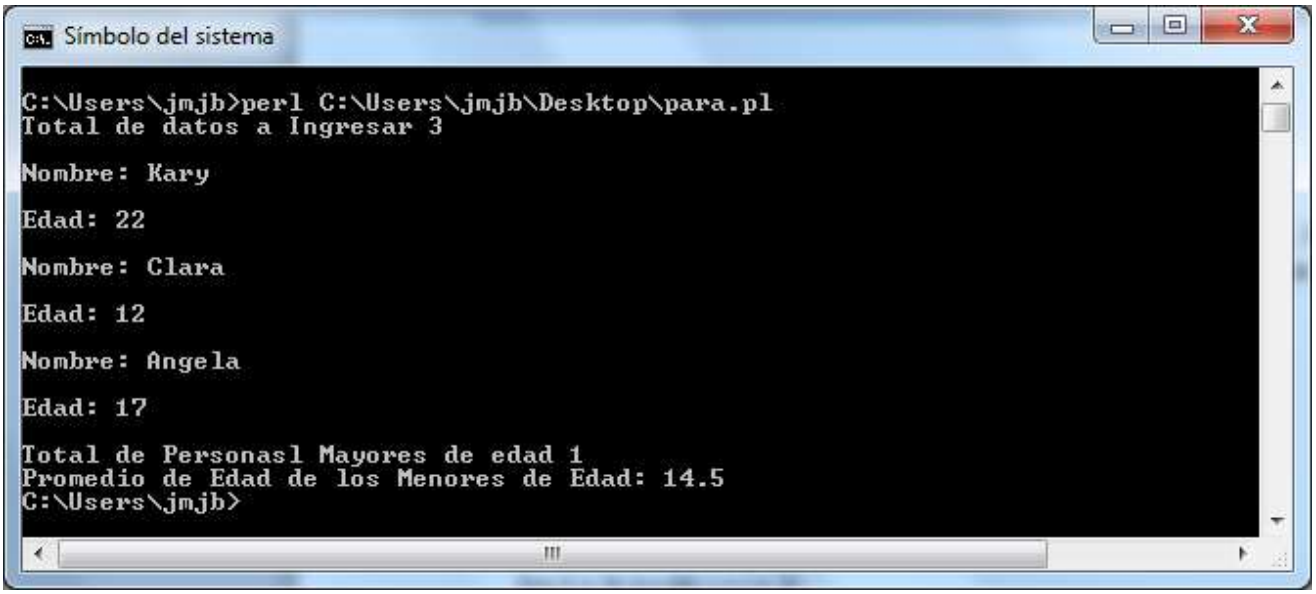
for (i = 3; i<=30; i = i + 2)

Ejemplo

Leer para un número conocido de personas el nombre y la edad, calcular e imprimir total de personas mayores de edad y promedio de edad de los menores.



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.

Ciclo mientras que

El ciclo mientras que es un proceso condicional que se repite o ingresa al ciclo solo cuando la condición de evaluación es verdadera, cuando esta es falsa el ciclo finaliza y arroja los resultados.

Su sintaxis es

Mientras que (condición)

En el lenguaje quedaría así

while (condición)

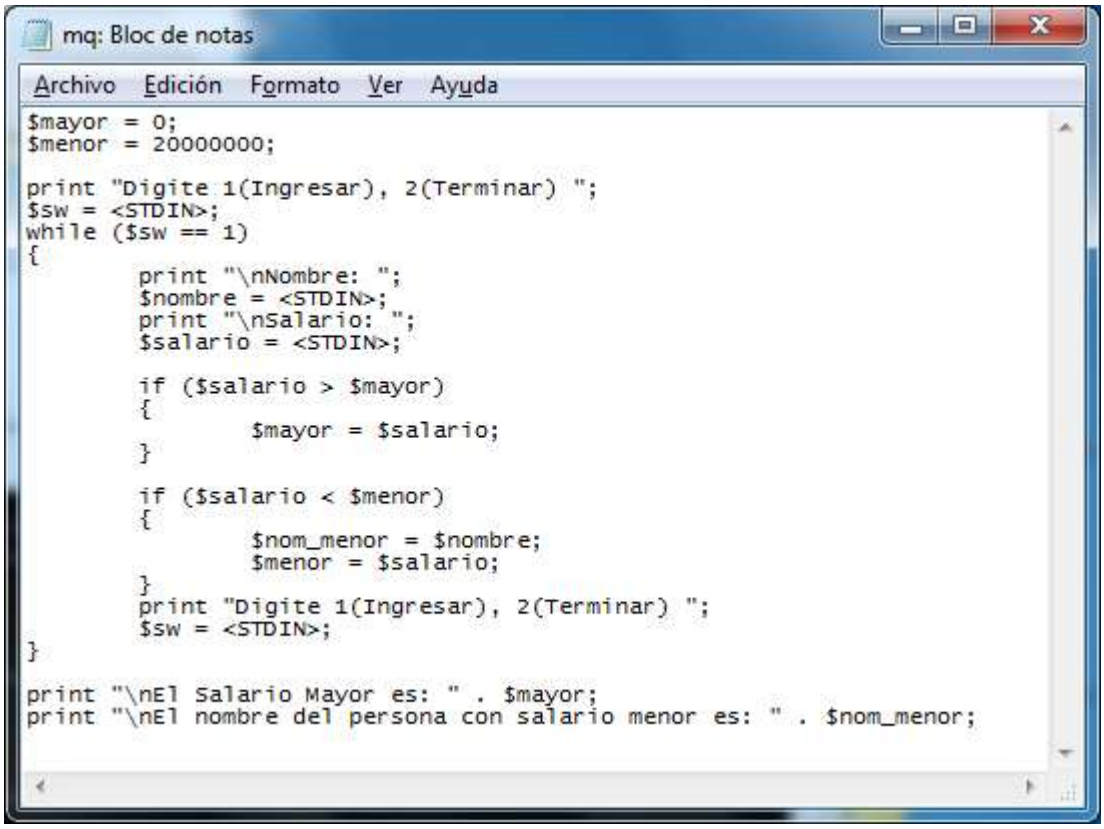
{

Procesos

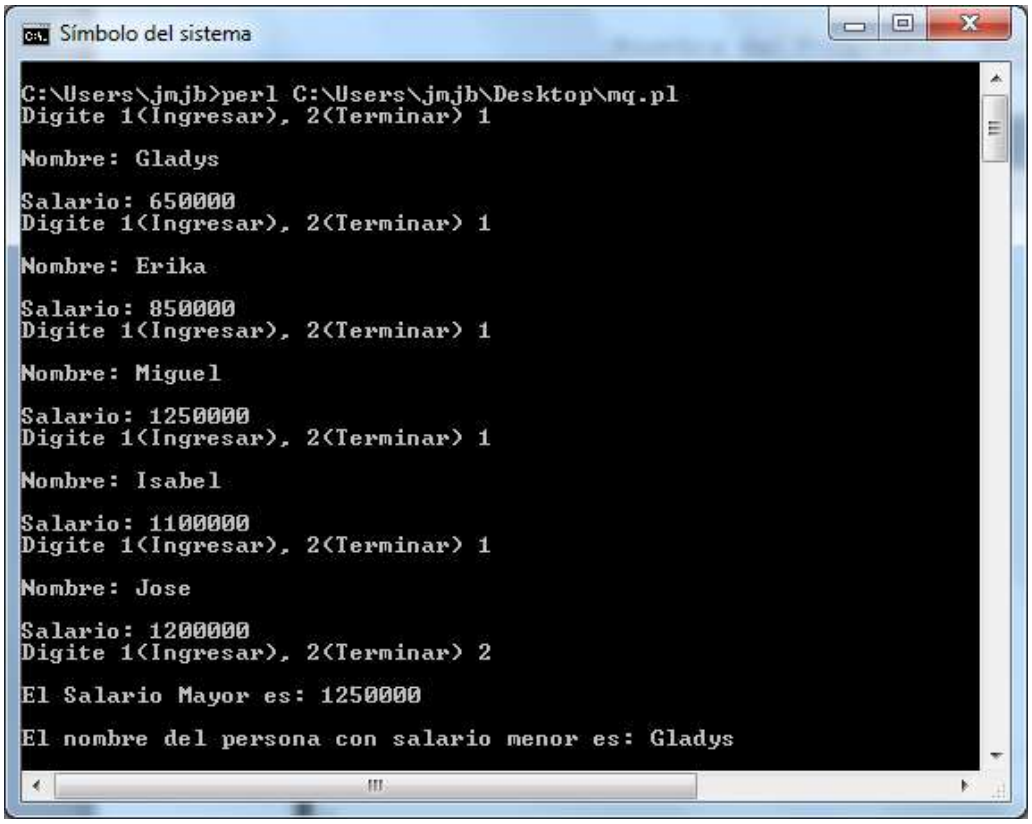
}

Ejercicio

Leer para una cantidad desconocida de datos, nombre, salario, calcular e imprimir el salario más alto, y el nombre de la persona con salario mas bajo



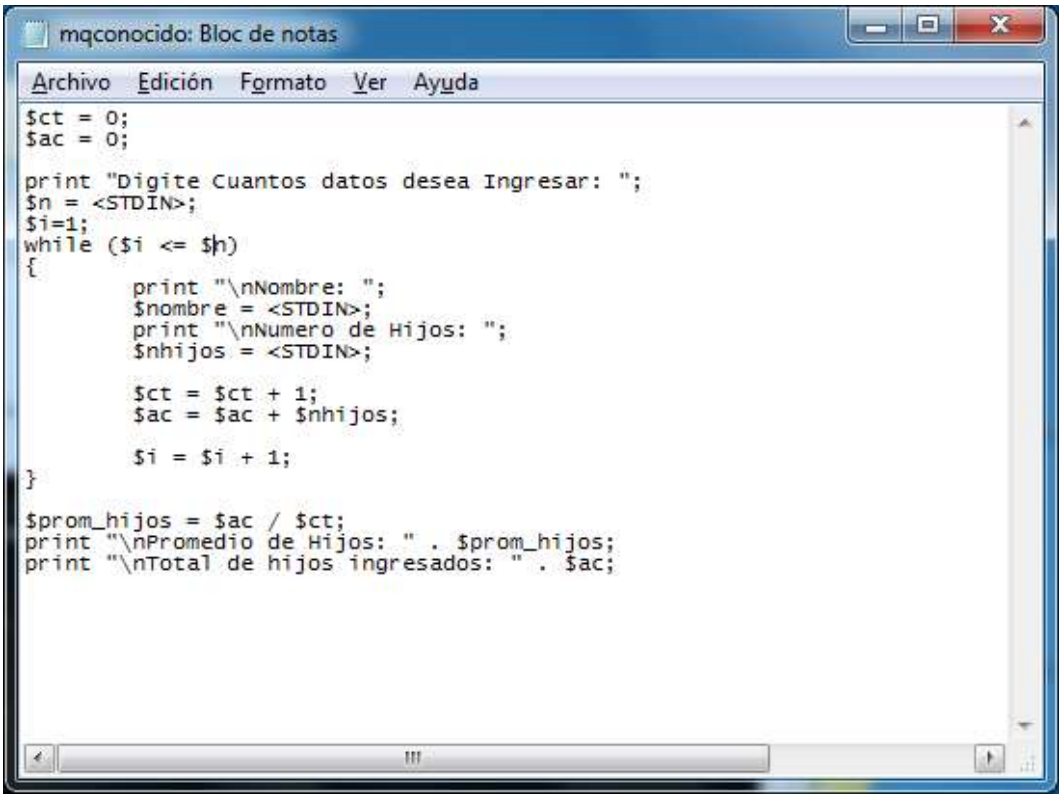
Elaborado por: César A. Jaramillo H.



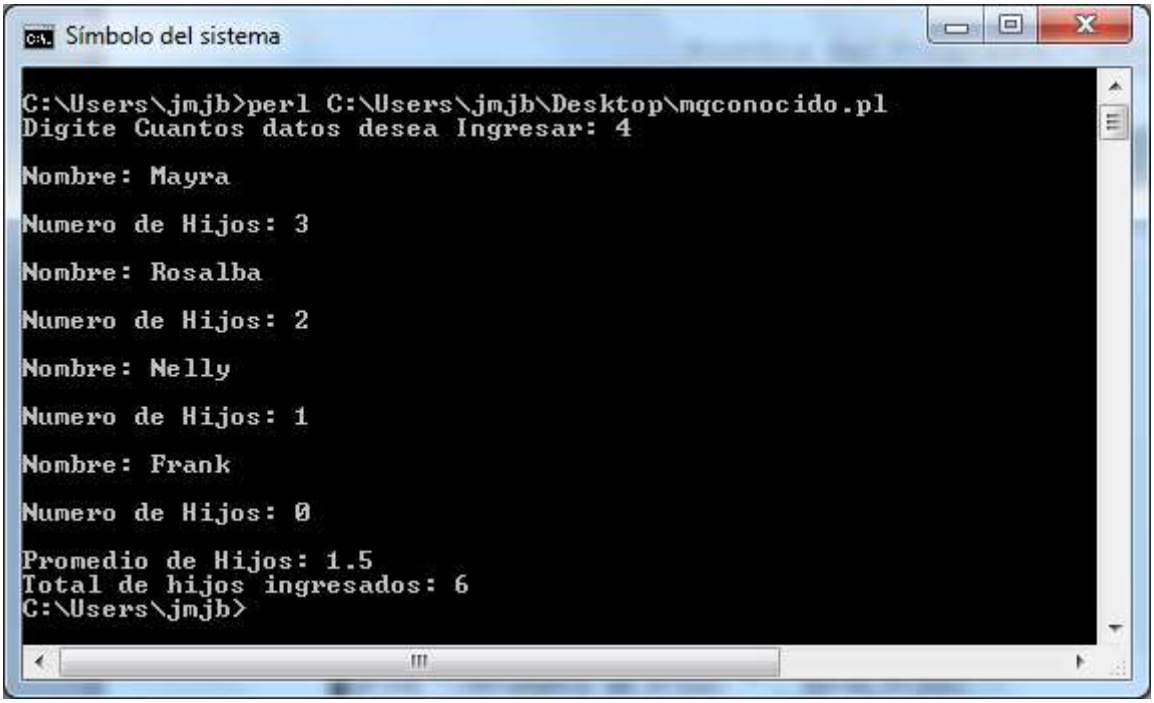
Elaborado por: César A. Jaramillo H.

Para un tamaño conocido se puede realizar así:

Se desea leer para una cantidad conocida de datos el nombre, el número de hijos, calcular e imprimir promedio de hijos y total de hijos ingresados.



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.

3.5.1 EJERCICIO

- Crear la serie de Fibonacci con una cantidad conocida de datos (0,1,1,2,3,5,8,13,21,34,)
- Leer para un grupo de estudiantes la nota definitiva, calcular e imprimir cuantos ganan, pierden y habilitan.

- Leer un valor y generar la tabla de multiplicar de este valor del 1 al 10

> **Los ciclos dentro del lenguaje de programación son fundamentales para ahorrar espacio y trabajo, repitiendo solo un bloque del código una cantidad de veces ilimitado.**
> **(http://www.programacionfacil.com/perl:for)**

4 UNIDAD 3 FUNDAMENTOS DE JAVA

RELACIÓN DE CONCEPTOS

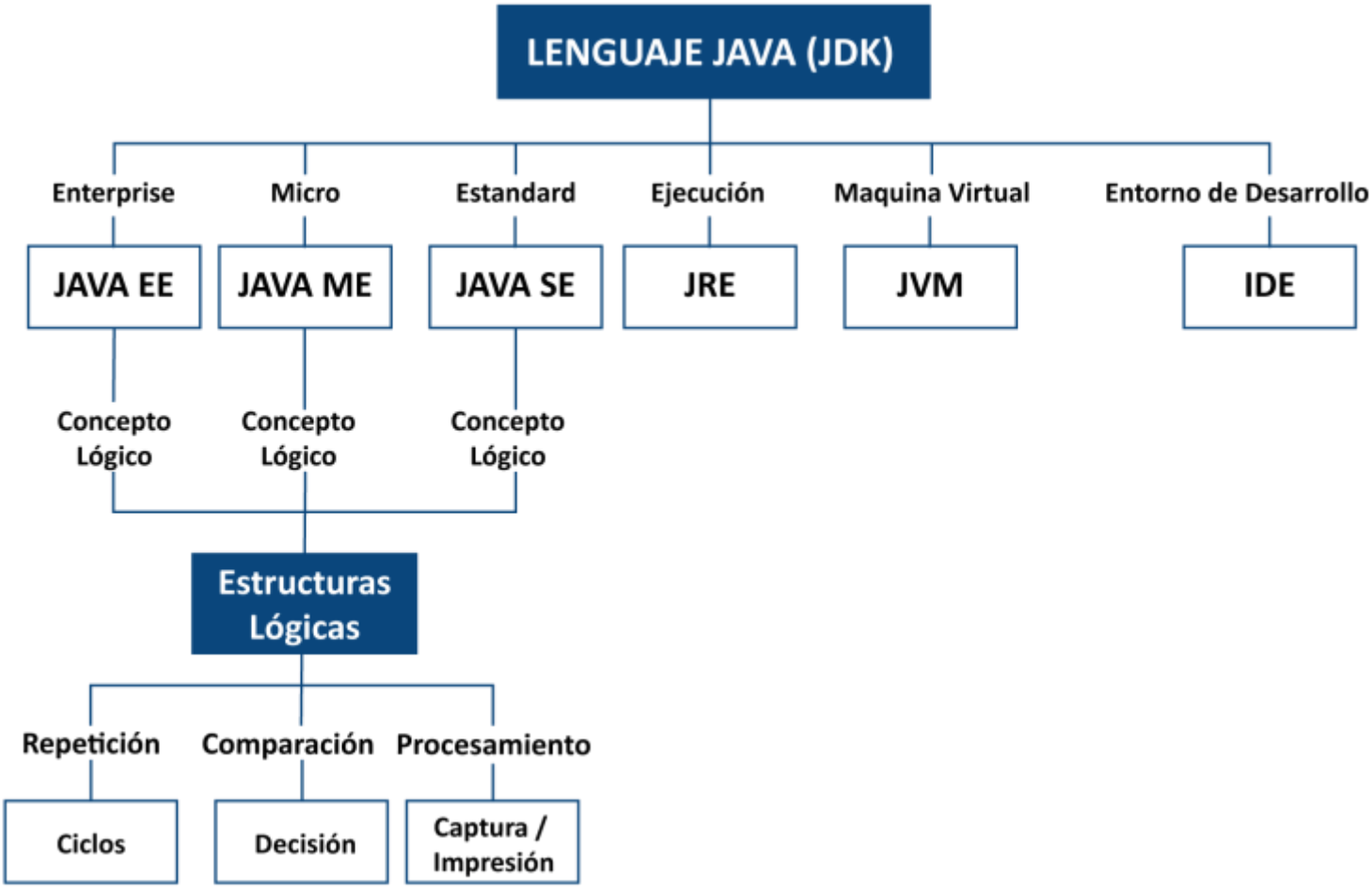


TABLA DE CONCEPTOS		EJEMPLO	
Concepto	Definición		
JDK	Java Development Kid, conjunto de herramientas de desarrollo de java, es el lenguaje de programación de la empresa Oracle, uno de los dos más utilizados en los últimos años		
IDE	Integrated development environment (Entorno de Desarrollo Integrado), es un conjunto de herramientas que sirven de apoyo a un lenguaje de programación, estas herramientas proporcionan configuración, ambiente visual, elementos que ayudan a que la programación sea más rápida y efectiva.		
JVM	Java Virtual Machine (Máquina Virtual de Java) es una herramienta que proporciona el lenguaje para le interpretación de aplicación en java, esta máquina virtual es una librería de JRE		
JRE	Java Runtime Environment (Ambiente de Ejecución de Java) es un aplicativo intérprete que sirve de soporte para cualquier tipo de aplicación java, esto evita que el usuario tradicional tenga que instalar y configurar el lenguaje de programación.		
Java SE	Java stándard Edition, es la versión mas común de las proporcionadas por Oracle, para desarrollo de propósito general.		

Java ME	Java Micro Edition, es una distribución específica para aplicaciones de dispositivos móviles	
Java EE	Java Entrerprese Edition, aplicativo diseñado para aplicativos web.	

OBJETIVO GENERAL

Identificar el lenguaje base de trabajo de la universidad, su historia, características, el porqué de este lenguaje y no de otro, las ventajas en el mercado, los propósitos y los alcances que tiene en los semestres posteriores.

OBJETIVOS ESPECÍFICOS

Conocer la historia del lenguaje, sus características fundamentales, sus fortalezas, las características de configuración, las herramientas que se pueden aplicar para desarrollar el potencial del ambiente de trabajo.

Identificar los tipos de operadores y tipos de datos que se pueden aplicar al lenguaje de programación, teniendo el concepto de la utilización adecuada de los datos, su finalidad, su aplicación y sus ventajas, además del uso de operadores que permitan el aprovechamiento de recursos, cálculos y comparaciones.

Conocer las palabras o comandos más comunes que el lenguaje tiene reservados, con el fin de que el usuario desarrollador no utilice estas que pueden ocasionar inconvenientes en la creación de un aplicativo, las secuencias de escape, muy utilizas en ambiente texto y las concatenaciones para uso de impresión y aprovechamiento de recursos.

Identificar los componentes de un programa en java, el método principal con sus características que lo conforman, acompañado de las herramientas que proporciona el lenguaje de programación para evaluar o repetir procesos.

Conocer las normas más comunes establecidas en su momento por Sun Microsistema y aplicadas por Oracle, normas que permiten un desarrollo ordenado que identifica de cada tarea que se crea.

Conocer las clases más comunes que se aplican en java, su finalidad y su utilización adecuada, herramientas que nos permitirán aprovechar de manera más amplia el lenguaje, pero ante todo clasificar según el tipo de aplicativo a desarrollar.

Identificar los conceptos más elementales de los arreglos en su parte introductoria, arreglos unidimensionales, sus características, inicio y fin de ellos, el potencial lógico que proporcionan y la posibilidad de combinaciones que darán al usuario en aplicaciones basadas en memoria.

4.1 TEMA 1 HISTORIA, CARACTERÍSTICAS DEL LENGUAJE JAVA

Java es un lenguaje creado por Sun MicroSystem, a mediados de los años 90's, se creó con un propósito de comunicación de dispositivos, con el fin de que al activar un dispositivo electrónico este se comunicara con otros dispositivos del hogar, este proyecto no tuvo la suficiente acogida, y se enfocó más adelante en la creación de componentes para la web que en ese momento estaba en su etapa más creciente, a partir de este momento sale la versión 1.0 a principios de 1996, un año más adelante sale la versión 1.1, en esta versión aparece un de las clases más utilizadas durante mucho tiempo para darle un ambiente grafico a las aplicaciones, el AWT y se integra de JDBC, a finales del año siguiente sale la versión 1.2 y es en este momento cuando toma el nombre de Java 2 y aparecen las distribuciones de Java SE, Java ME y Java EE, igualmente aparece el ambiente grafico Swing y JVM, que complementa los aplicativos, permitiendo su interpretación entre programa y máquina, en el año 2000 aparece la versión 1.3, se incluye en esta versión el JNDI, que es un paquete de bibliotecas principales, en el año 2002, sale la versión 1.4, permitió la integración con XML, en el 2004 sale la versión 5, en el 2006 lanzan la versión 6, y en el año 2011 sale la versión 7.

Dentro de sus características más importantes está la de orientación a objetos, que es una filosofía de programación que se basa en el aprovechamiento de los recursos pero ante todo en la reutilización de los procesos creados, esta filosofía da como resultado una serie de conceptos fundamentales para el aprovechamiento de las funciones y características de lenguaje, no aprovechar estas herramientas nos lleva a una programación estructurada que no aplica las ventajas de este lenguaje de programación.

Además de estas ventajas esta la independencia de la plataforma, situación que lleva a muchos desarrolladores a crear aplicaciones independientes del sistema operativo, de sus características o restricciones, existen ambientes de una sola plataforma lo que limita su campo de acción.

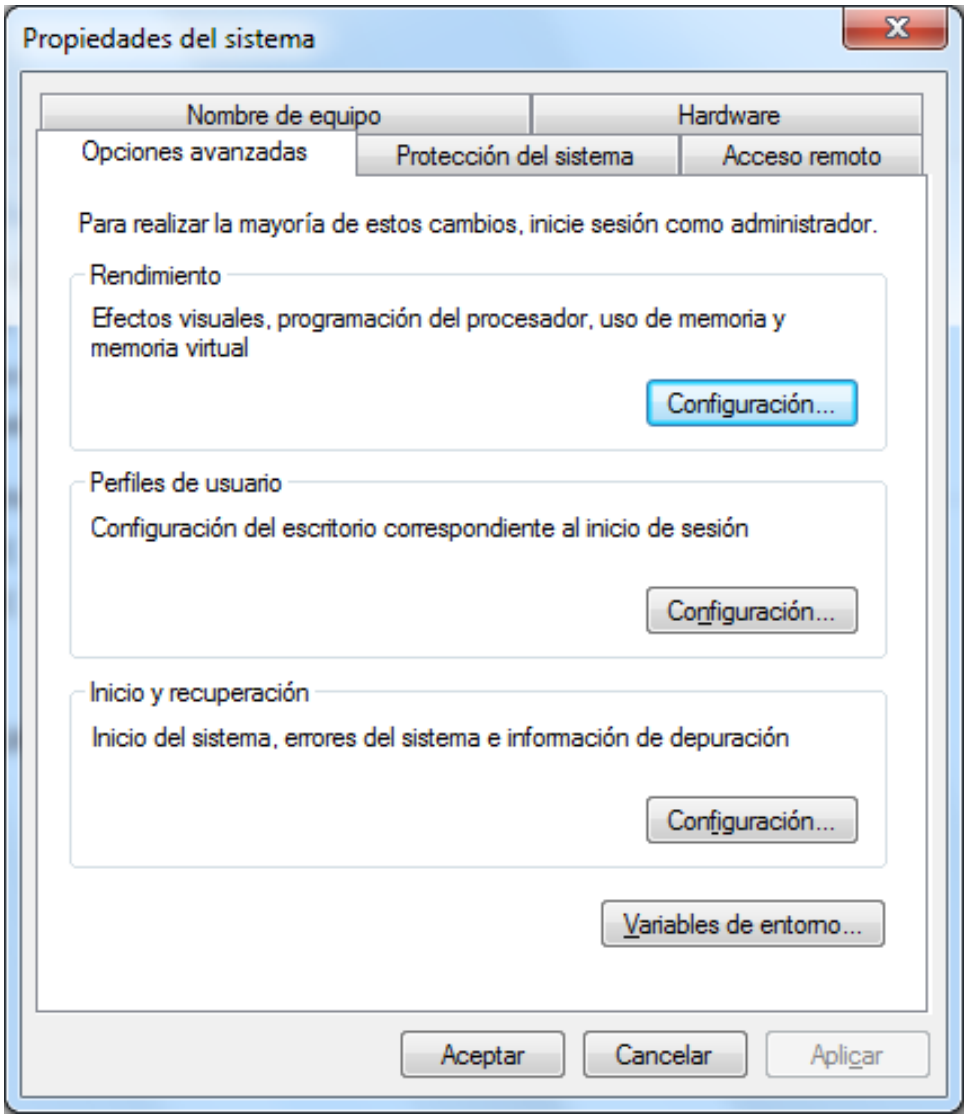
Descarga e Instalación de Java

Java es actualmente de propiedad de Oracle, desde su sitio web www.oracle.com, se puede realizar la descarga del aplicativo en forma gratuita, asegúrese de descargar la versión correcta para su pc y sistema operativo, algunas versiones no instalan en sistemas operativos con XP en el caso de Windows, después de seleccionar la versión correcta puede elegir entre una versión que solo contenga el lenguaje de programación o una versión que contenga el IDE Netbeans, ambiente creado inicialmente por Sun MicroSystem, definido estas condiciones proceda a descargar e instalar, se recomienda que al descargar la versión, descargue en forma adicional la documentación de Java en la distribución que requiere, esta documentación nos provee de ayudas, funciones, métodos, clases del sistema, además del API Application Programming Interface (Interfaz de Programación de Aplicaciones), esta documentación se ubica en la carpeta de instalación JDK.

Variables de entorno

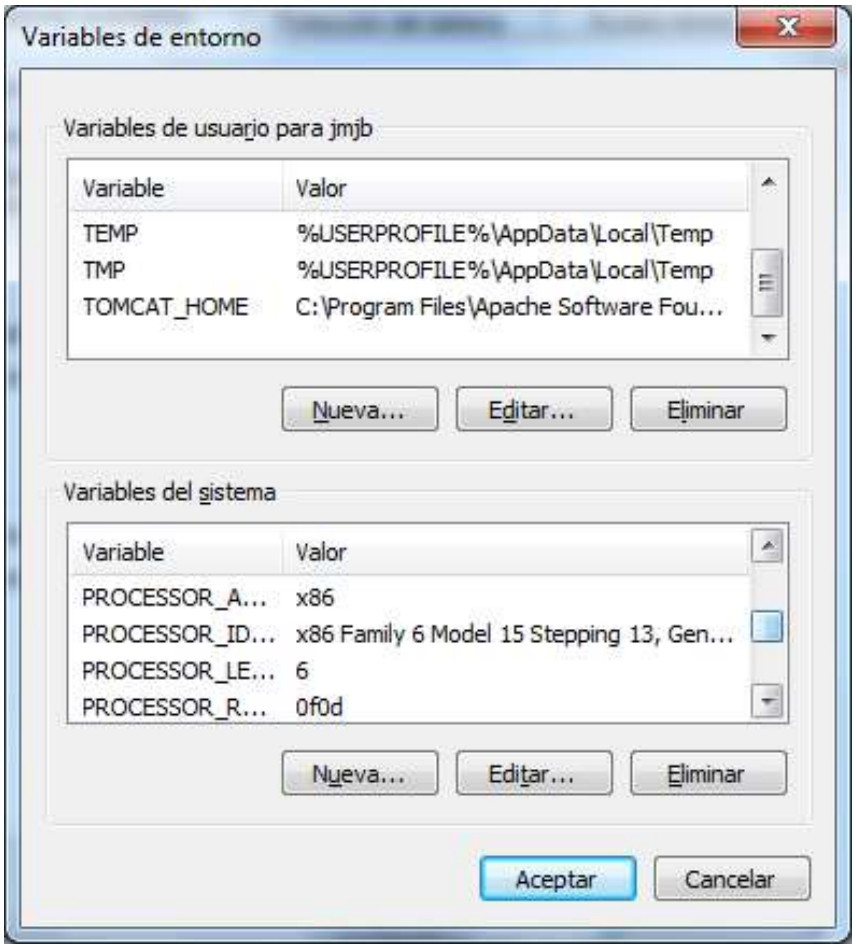
Si posee un IDE, no va a requerir la configuración de las variables de entorno, estas variables muy utilizadas para compilación y ejecuciones de aplicaciones en ambiente texto, nos darán la opción de crear un procesos más rápido y así evitar la búsqueda de una serie de rutas que complican el procesamiento de los datos.

Ubicados en Equipo, seleccione la opción Propiedades del sistema, y luego seleccione configuración avanzada del sistema,



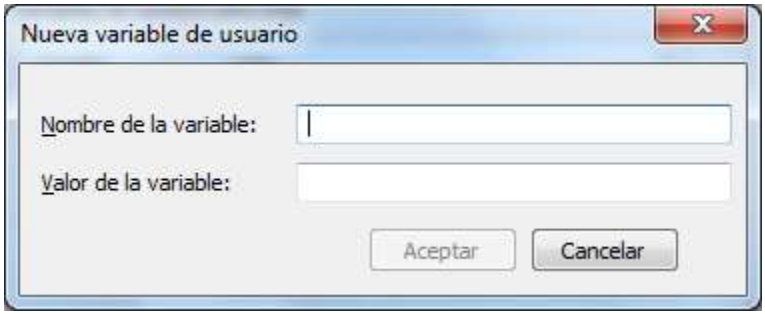
Elaborado por: César A. Jaramillo H.

En esta ventana encuenta el botón Variables de Entorno y dentro de esta aparecerá



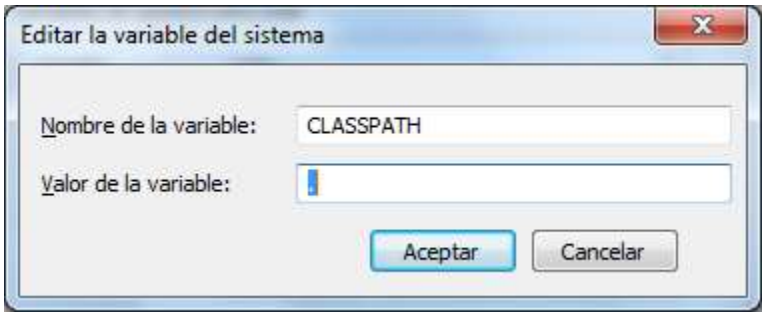
Elaborado por: César A. Jaramillo H.

Esta ventana tiene dos areas, la primera contiene las variables de usuario y la segunda variables de sistema, ubicados en la primera (variables de usuario), busquen la variable de entorno PATH, esta variable contendrá la ruta de ubicación de JDK que vamos a utilizar, si dicha variable no existe selección el botón Nueva



Elaborado por: César A. Jaramillo H.

Digite el nombre de la variable PATH y en valor de la variable coloque C:\Program Files\Java\jdk1.6.0_23\bin, que equivaldria a la version que estén manipulando de java, en este caso aparece el jdk en la versión 6 actualización 23, según su descarga o actualización variara la última parte del nombre, en caso de la variable ya exista en su equipo presiona doble click sobre ella, adicione un punto y coma (;) al final y coloque la ruta de ubicación del JDK (;c:\Program Files\Java\jdk1.6.0_23\bin), esta ruta nos permitirá la ubicación desde cualquier lugar del equipo de los archivos principales que contiene el java, sean los archivos de compilación (javac), de ejecución (java) o los de documentación (javadoc), lo propio sucede con las variables del sistema, en esta se crea o se busca la variable CLASSPATH, si esta variable no existe, sea crearan presionando el botón Nueva.



Elaborado por: César A. Jaramillo H.

Esta ventana contendra el nomber y el valor de la variable, a la que se le colocara como valor punto (.), para que tome la ruta de ubicación como propia y permita la compilación y ejecución de los archivos.

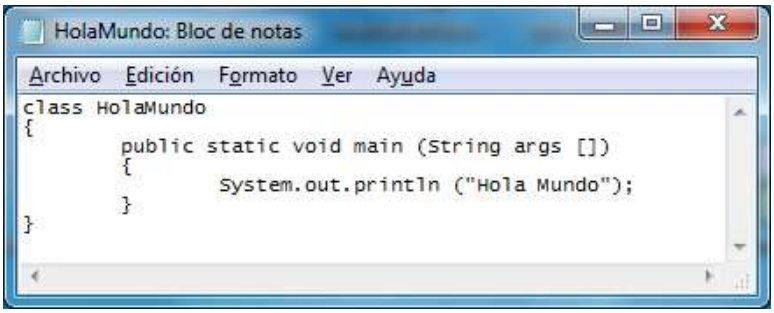
Ambiente de Trabajo

El ambiente de trabajo utilizado en esta etapa inicial será el más básico, utilizaremos el block de notas (notepad) como nuestro editor inicial,

Cuando se crea un primer aplicativo en java debemos recordar los pasos, los procesos lógicos previos.

Los archivos que se crean en java van a tener la extensión .java, esta lenguaje de programación permite identificar los archivos fuentes y objetos fácilmente, cuando el proceso está listo se procede a compilar, la compilación genera un archivo .class, es conocido como un bytecode que es un intérprete entre el programa y la máquina que será reconocido por JRE, que a su vez contiene JVM, recordemos que la compilación se encarga de verificar la sintaxis del programa mas no la lógica que se le aplico a este.

Primer Programa en Java



Elaborado por: César A. Jaramillo H.

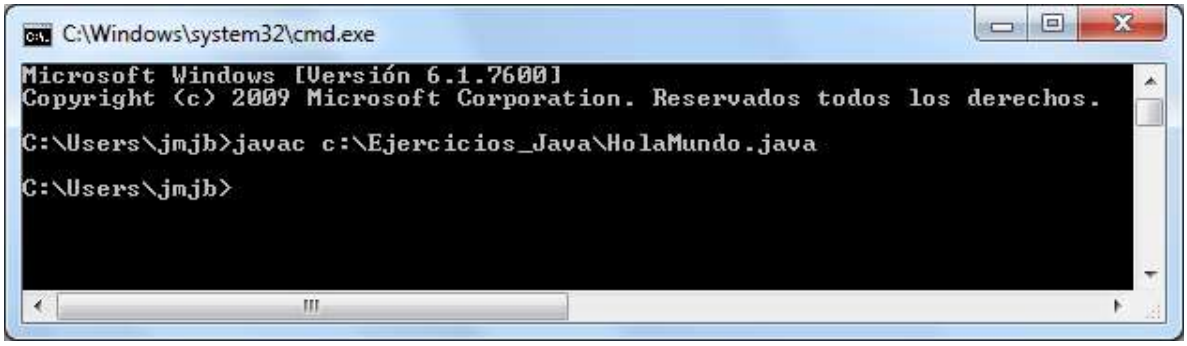
Características de este programa en java

El nombre de la clase debe ser el mismo nombre del archivo, conservando la escritura, hay que tener presente que es ambiente que distingue mayúsculas y minúsculas, utiliza las llaves como inicio y fin de los procesos, y una serie de comandos y funciones que deben de ser respetadas en su escritura.

Compilación

La compilación desde el ambiente texto (D.O.S), es muy simple.

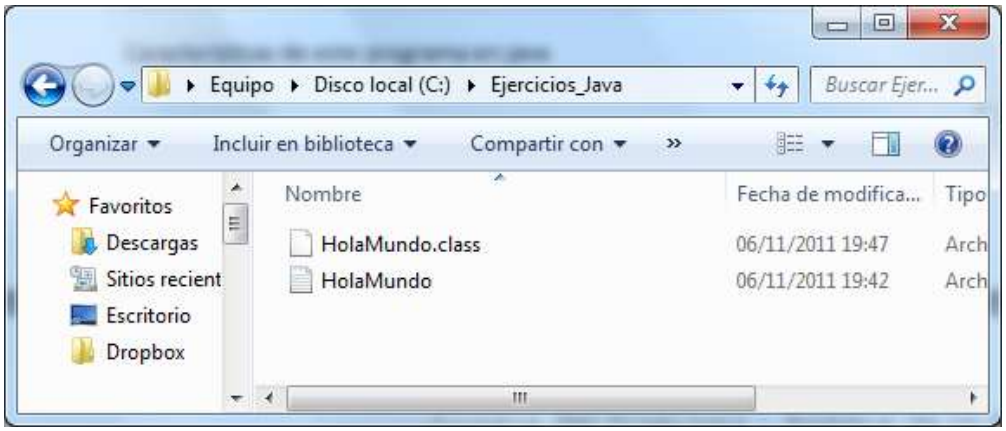
Ubicados él, (Inicio, Cmd)



Elaborado por: César A. Jaramillo H.

Se digita javac acompañado de la ruta del archivo (conservando la escritura de él), con su respectiva extensión.

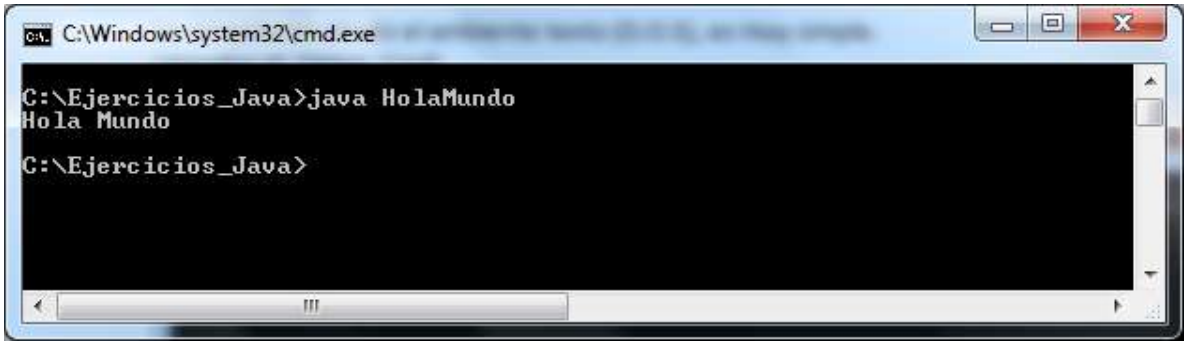
Si tuviéramos algún error nos mostraría inmediatamente debajo de esta línea.



Elaborado por: César A. Jaramillo H.

La compilación nos genera un archivo con el mismo nombre y extensión .class (bytecode) y nos permitirá la ejecución y visualización de los datos.

La ejecución se realiza con el comando java acompañado del nombre del archivo sin extensión.



Elaborado por: César A. Jaramillo H.

4.1.1 EJERCICIO

- Crear un programa que muestre su presentación personal, su nombre, edad, profesión, estado civil

> La configuración de java es fundamental para aprovechar las herramientas y las características del lenguaje. (http://java.com/es/download/help/index_configuration.xml?user_os=Windows 7)

4.2 TEMA 2 TIPOS DE DATOS, OPERADORES ARITMÉTICOS, RELACIONALES Y BOOLEANOS

Los tipos de datos en java y muchos otros lenguajes de programación corresponden a un eje principal del manejo de la información, mediante estos podemos clasificar los datos de manera apropiada y darle uso preciso a los recursos del equipo de cómputo, estos tipos de datos en java tienen una categoría adicional que otros lenguajes no tienen, Tipos Primitivos y Tipos Referencia.

Tipos de datos primitivos

TIPO	BYTES	RANGO
boolean		True – false
byte	1	-128 a 127
short	2	-32768 a 32767
int	4	-2147483648 a 2147483647
long	8	-9223372036854775808 a 9223372036854775807
char	2	0 a 65535
float	4	+/- 3.4028235E+38
double	8	+/-1.79769313486231570E+308

Dentro de sus características esta que se declaran en minúscula

TIPOS REFERENCIA

Son clases que utiliza java con distintos propósitos, entre otros podríamos realizar conversión de datos, mas no captura o declaración de datos directamente salvo el String.

String

Integer

Float

Double

Short

Byte

Cada una de estos tipos de referencia tiene relación con uno de los tipos primitivos, su condición básica es el inicio en mayúscula.

Operadores

Los operadores son componentes importantes dentro del lenguaje por su uso dentro de los procesos más básicos, dentro de ellos encontramos Aritméticos, Relacionales y Lógicos o booleanos.

Aritméticos

OPERADOR	OPERACIÓN
+	Suma
-	Resta
*	Multiplicación
/	División
%	Residuo

Relacionales

OPERADOR	OPERACIÓN
>	Mayor que

>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
!=	Diferente
==	Igualdad

Lógicos o Booleanos

OPERADOR	OPERACIÓN
&&	Y (And)
||	O(or)
!	No (Not)
^	XOR

Operadores de Asignación

OPERADOR	OPERACIÓN
++	Incremento unitario

--	Decremento unitario
=	Asignación
*=	Multiplicación por asignación
/=	División por asignación
+=	Suma por asignación
-=	Resta por asignación
%=	Modulo por asignación

PREFIJOS Y SUFIJOS

Dentro del manejo de los operadores, existen muchas formas de realizar asignaciones (véase cuadro anterior), la forma más tradicional es mediante el signo igual (=), pero el lenguaje permite que se apliquen de otras formas complementarias.

Las asignaciones en prefijo y sufijo son uno de los procesos más comunes en ambientes que tienen alguna herencia de C++.

Ejemplo de asignación simple

a=5

a=a+1

La primera asignación es un valor predeterminado, en el segundo caso es un incremento en 1

Ejemplo de asignación prefijo

a=5

++a

Este tipo de asignación tiene como resultado en mismo que el ejemplo anterior

Ejemplo de asignación sufijo

a=5

a++

Esta última en teoría realizaría la misma función pero no es así, la operación a++, equivale a dos funciones que son

a=a

a=a+1

Si se tomara cualquiera de estos procesos para imprimir un resultado el primer ejemplo y el segundo dan como resultado el mismo, pero el tercero no, es una situación lógica, muy común en su uso pero poco común en su análisis.

4.2.1 EJERCICIO

- Teniendo los valores X=3 y Y = X++ *10, cual es el resultado
- Teniendo los valores X=3 y Y = ++X *10, cual es el resultado

> Los operadores en java son elementos infaltables para cualquier tipo de operación o calculo.
> (http://www.desarrolloweb.com/articulos/1730.php)

4.3 TEMA 3 PALABRAS CLAVES, SECUENCIAS DE ESCAPE Y CONCATENACIÓN

Palabras claves
Son todos los comandos o funciones que son reservadas por el sistema y el usuario no puede utilizar para algo distinto a su función principal.

abstract	boolean	break	byte	case
catch	char	class	continue	default
do	double	else	extends	false
final	finally	float	for	if
implements	import	instanceof	int	interface
long	native	new	null	package
private	protected	public	return	short
static	super	switch	syncroniced	this
throw	throws	transient	true	try
void	volatile	while	var	rest
byvalue	cast	const	future	generic

goto	inner	operator	outer	

Elaborado por: César A. Jaramillo H.

Secuencias de escape

SECUENCIA DE ESCAPE	SIGNIFICADO
\b	Retroceso
\n	Salto de línea
\t	Tabulación
\\	Barra invertida
\’	Comilla simple
\”	Comilla doble

Concatenación

La concatenación es la forma más simple de unir varios procesos, habitualmente esta unión se da entre una variable tipo texto y una numérica o entre una variable tipo texto y otra texto, el signo que se utiliza para esto es el más (+).

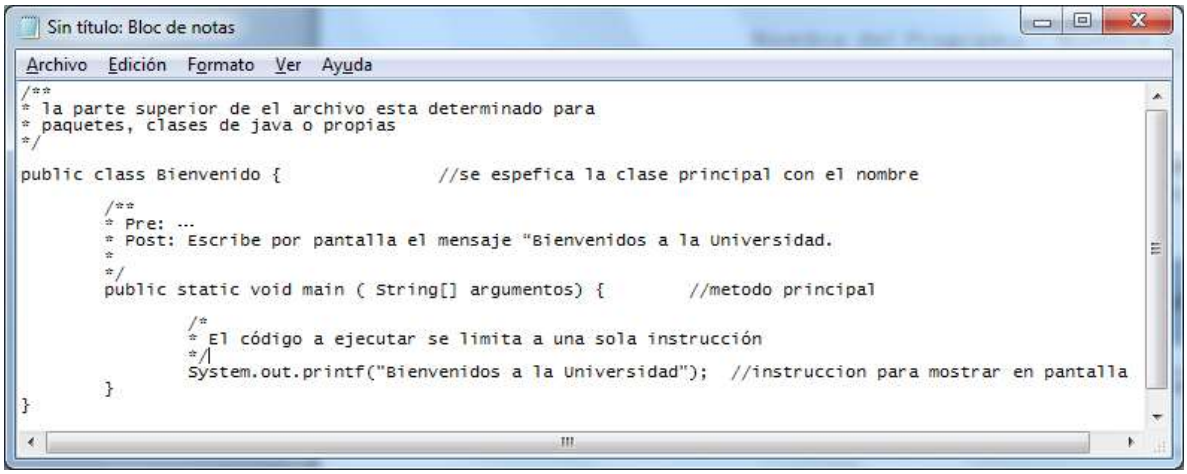
4.3.1 EJERCICIO

- Identifique las palabras reservadas y encuentre su significado para mayor comprensión de su utilidad.

> **Las palabras reservadas de java son herramientas o tareas prediseñadas con el fin de realizar operaciones de diferente índole (http://es.wikipedia.org/wiki/Java_(lenguaje_de_programa-ci%C3%B3n)).**

4.4 TEMA 4 MÉTODO PRINCIPAL, ESTRUCTURAS DE CONTROL Y OPERADOR TERNARIO

Estructura de un programa básico en java

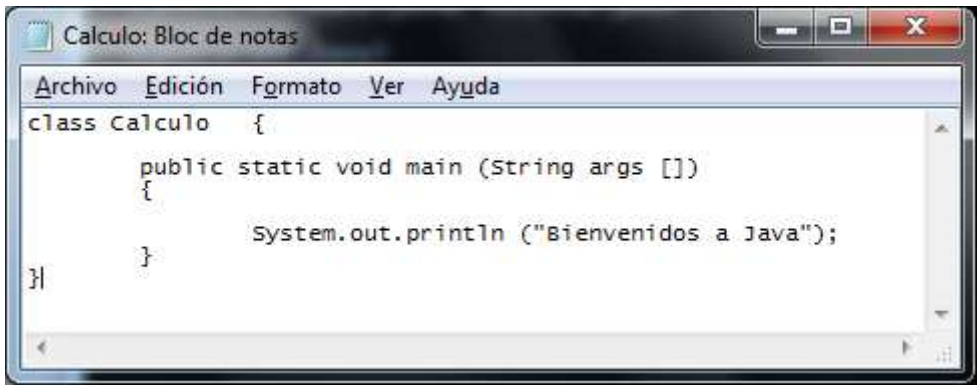


Elaborado por: César A. Jaramillo H.

Los archivos en java tienen una diferencia significativa con el lenguaje anterior, dentro de sus principales características está el uso de recursos propios del lenguaje acompañado de la lógica que se desea manipular en el aplicativo, en la parte superior puede ubicarse paquetes o clases que nos ayudan a ampliar las posibilidades del programa a realizar, para procesos simples de impresión y cálculos elementales de información el sistema carga de manera predeterminada la clase **lang** no se requiere colocar, pero si se ubicara no tendría ningún inconveniente.

Observemos un ejemplo con y sin la clase

Sin clase



Elaborado por: César A. Jaramillo H.

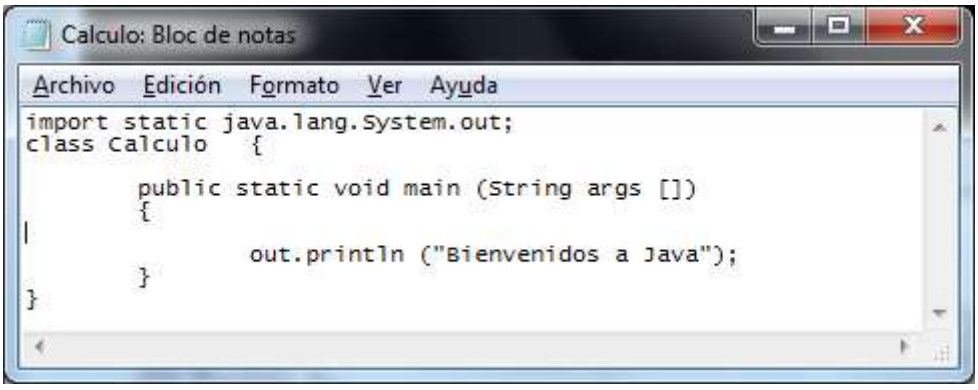
Compilación

Elaborado por: César A. Jaramillo H.

Ejecución

Elaborado por: César A. Jaramillo H.

Con Clase



Elaborado por: César A. Jaramillo H.

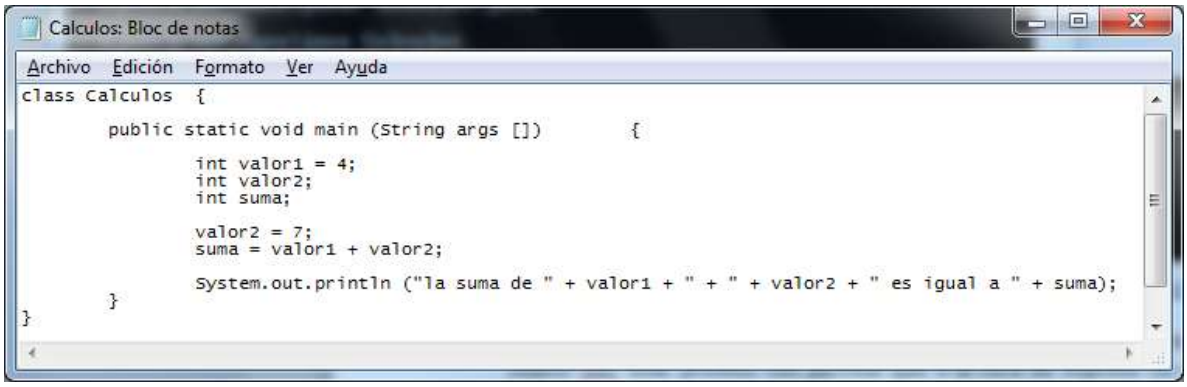
Compilación

Elaborado por: César A. Jaramillo H.

Ejecución

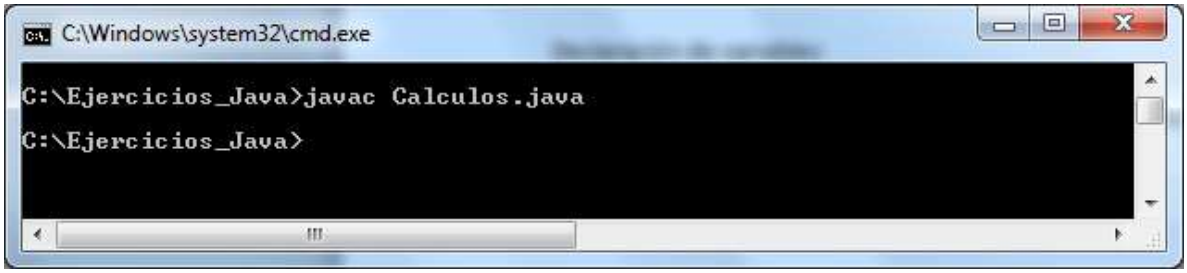
Elaborado por: César A. Jaramillo H.

La Clase **lang** se importa y se está especificando que traiga por defecto el método System con el objeto out, este proceso nos permite que a la hora de imprimir no se utilice nuevamente le comando como en el ejemplo anterior. Pero existen otras clases que son fundamentales para realizar otros procesos, estas se verán más adelante.
Declaración de variables
Crear un programa que lea valor 1 y valor 2, calcular la suma de estos valores e imprimirlos



Elaborado por: César A. Jaramillo H.

Compilación



Elaborado por: César A. Jaramillo H.

Ejecución

Elaborado por: César A. Jaramillo H.

Observemos el programa con detenimiento y nos encontramos el nombre de la clase Cálculos, este nombre de la clase es el mismo nombre del archivo, debe conservar la misma escritura, sin signos, sin espacios, si se tuviera más de una palabra como CalculosMatematicos, cada palabra quela conforme debe de ir en mayúscula inicial, no quiere decir que si se coloca en minúscula la clase y el archivo no funciones, es solo la norma que tiene el lenguaje de programación.

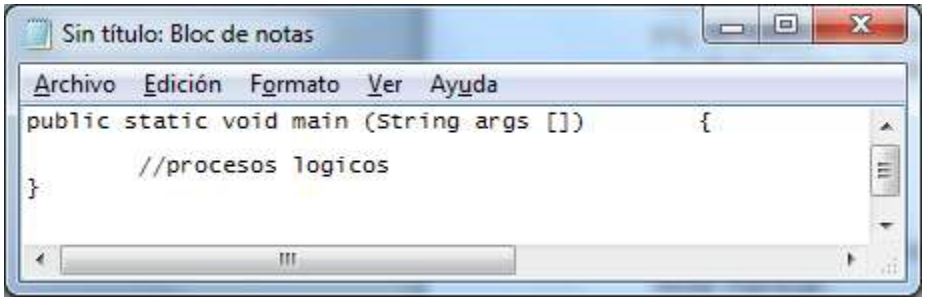
En la declaración de las variables es recomendado que cada una de ellas este en línea independiente, algunos declararan las variables en la misma línea, no habrá problemas en cálculos ni en funciones, estas variables tienen en su forma de escritura minúscula, es lo recomendado, se tuviera más de una palabra se recomienda que la segunda este en mayúscula inicial, así, valorInicial.

Las variables se pueden iniciar inmediatamente se declaran como en el caso de valor1 o se pueden iniciar después de declarar como en el caso valor2.
Para el cálculo matemático se utilizó la variable suma acompañada de un operador de asignación (=), el valor1 + el valor2, la instrucción terminada en punto y coma (;) y luego se imprime el resultado acompañado de una serie de concatenaciones.

MÉTODO PRINCIPAL

Ya conociendo la estructura de un programa básico en java, veamos el método principal que este debe manejar.

Hay que tener presente que toda aplicaciones debe estar formado por lo menos una clase, en algunas de esas clases debe estar declarado el modificador de acceso *public,* y debe existir un método estático llamado main().



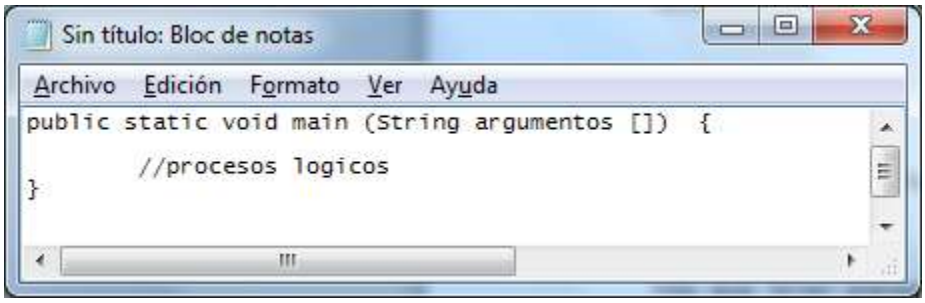
Elaborado por: César A. Jaramillo H.

El método main() debe de cumplir las siguientes características.

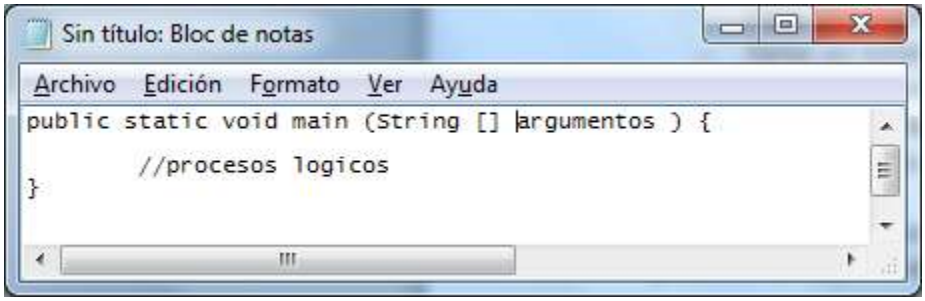
- Debe ser publico
- Debe ser estático
- No puede devolver ningún resultado por esto la sentencia *void*
- Se debe declarar un arreglo de cadena de caracteres en la lista de parámetros, en este caso tenemos String args []

Este método main() es el inicio de un programa en java, cuando se ejecuta el programa el archivo java.exe la máquina virtual de java (JVM), busca la clase indicada en el método main(), .

La sentencia String contiene un parámetro llamado args, este puede ser cualquier nombre, solo que por convención se utilizan estas cuatro letras, igual los símbolos [] pueden ir antes o después del parámetro.



Elaborado por: César A. Jaramillo H.



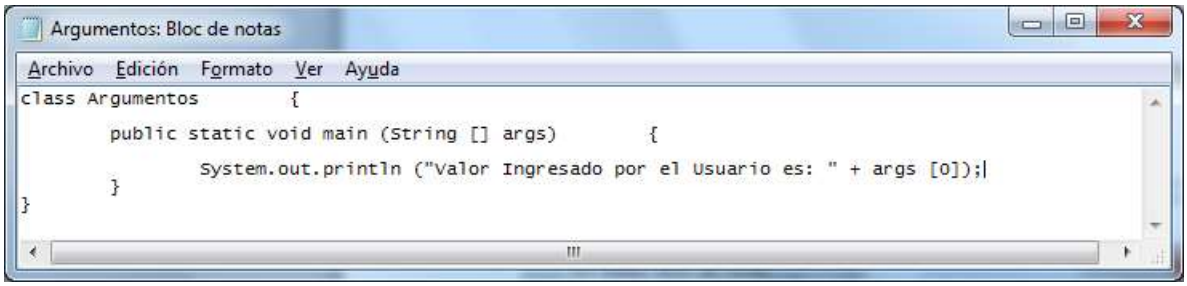
Elaborado por: César A. Jaramillo H.

Captura de Información

La captura de información es vital para el usuario programador y mucho más para el usuario operativo, quien tendrá que realizar procesos esenciales y muchos de estos procesos dependen de la información con la que alimente el programa.

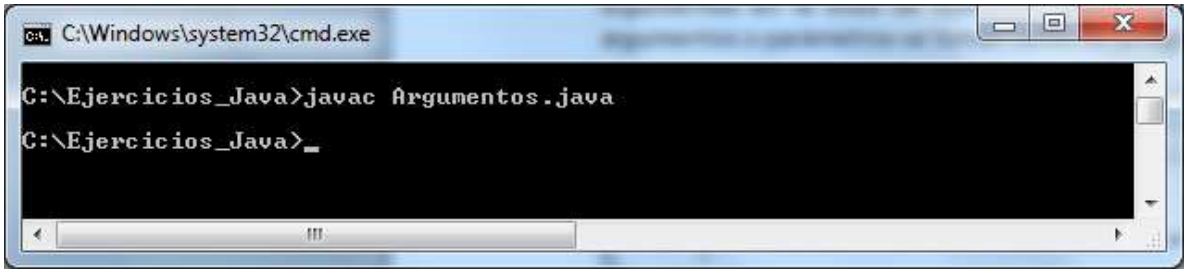
Captura de argumentos

La forma más simple que posee java para la captura de información es mediante el paso de argumentos en la línea de ejecución utilizando las características del método main(), estos argumentos o parámetros se toman como un arreglo, observemos un ejemplo.



Elaborado por: César A. Jaramillo H.

Compilación



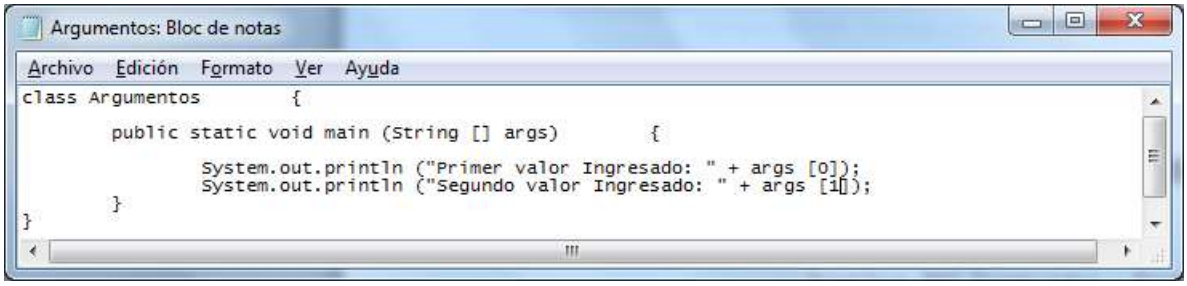
Elaborado por: César A. Jaramillo H.

Ejecución

Elaborado por: César A. Jaramillo H.

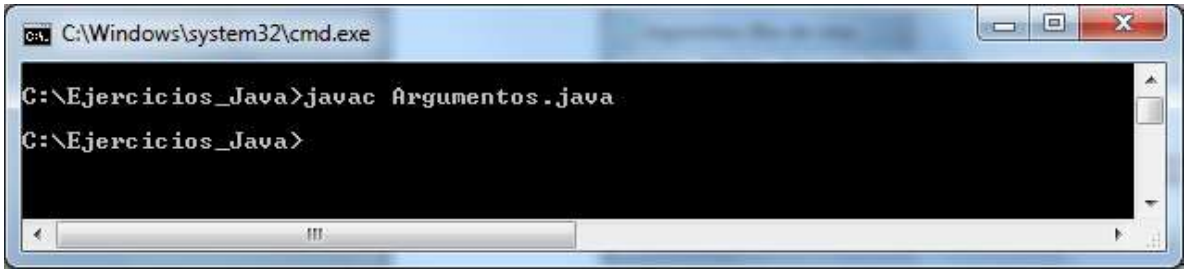
Observemos que en el momento de ejecutar se especifica java (para ejecutar), acompañado del nombre del archivo y el parámetro que deseamos mostrar o procesar, puedo colocar cualquier cantidad de argumentos pero me implicaría mayor cantidad de procesos o la utilización de un ciclo que me identifique cuantos ingrese.

Observemos el mismo ejemplo con 2 parámetros



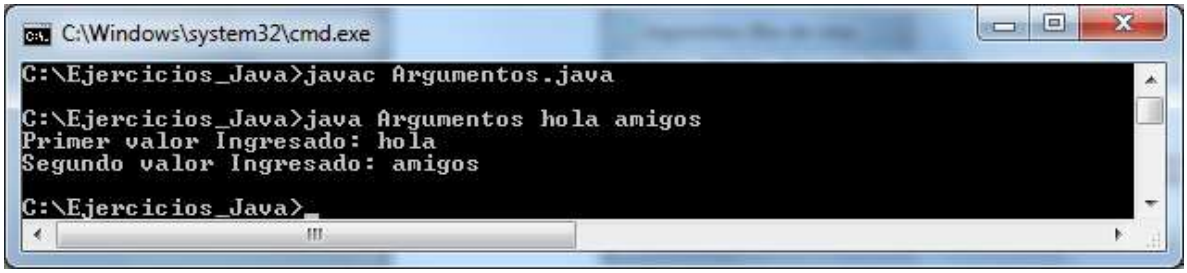
Elaborado por: César A. Jaramillo H.

Compilación



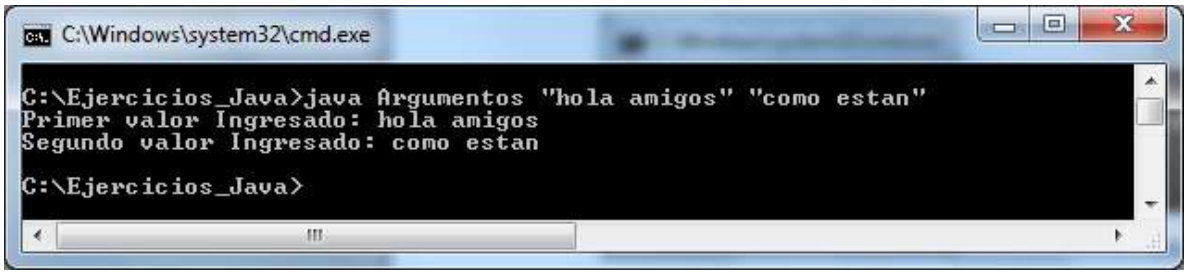
Elaborado por: César A. Jaramillo H.

Ejecución



Elaborado por: César A. Jaramillo H.

Cada parámetro o argumento se especifica colocando solo un espacio, si la palabra o frase es compuesta, debe de ir entre comillas dobles para que el sistema lo tome como uno solo.



Elaborado por: César A. Jaramillo H.

Captura con Scanner

El método Scanner es uno de los métodos más utilizados para captura de información más no el más moderno de estos, Observemos en ejemplo.

Se desea leer nombre, Edad y estatura, mostrar la información en pantalla.

CapturaScanner: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
import java.util.Scanner;

class CapturaScanner    {

        public static void main (String args [])        {

                Scanner lectura = new Scanner (System.in);

                String nom;
                int ed;
                float est;

                System.out.print ("Digita un Nombre: ");
                nom = lectura.nextLine ();
                System.out.print ("Digita una Edad: ");
                ed = lectura.nextInt();
                System.out.print ("Digita una Estatura: ");
                est = lectura.nextFloat();

                System.out.println ("Nombre " + nom + " Edad " + ed + " Estatura " + est);
        }
}
```

Elaborado por: César A. Jaramillo H.

Veamos que para este proceso utilizamos un import (forma de llamar una clase predeterminada en java para la captura de información, entre otras funciones), la clase útil y el método Scanner, separado por punto (.) en cada opción, estas cadenas de procesos pueden ser más largas dependiendo de la complejidad de las tareas que se puedan procesar, algunos muy prácticos pueden utilizar import java. Útil. *; con el fin de utilizar mayor cantidad de procesos de la clase útil, pero para nuestro método de trabajo utilizaremos solo las que requerimos.

Después del método main (), se realiza el proceso de instanciar una variable que realice las tareas, esta variable es únicamente para este propósito y la colocamos “lectura”, el nombre puede ser otro si lo prefieren, esta línea consta de:

Scanner lectura = new Scanner (System.in)

Scanner como clase, lectura como variable, new permite instanciar el Scanner y por último se especifica System.in que se encarga del ingreso de los datos, recuérdese otros procesos con el System. Out que nos permite o mostrar la información, esta última función va acompañada de println que imprime dejando una línea, en este ejemplo utilizaremos el print para que el sistema no parte líneas, sino que procese en la misma.

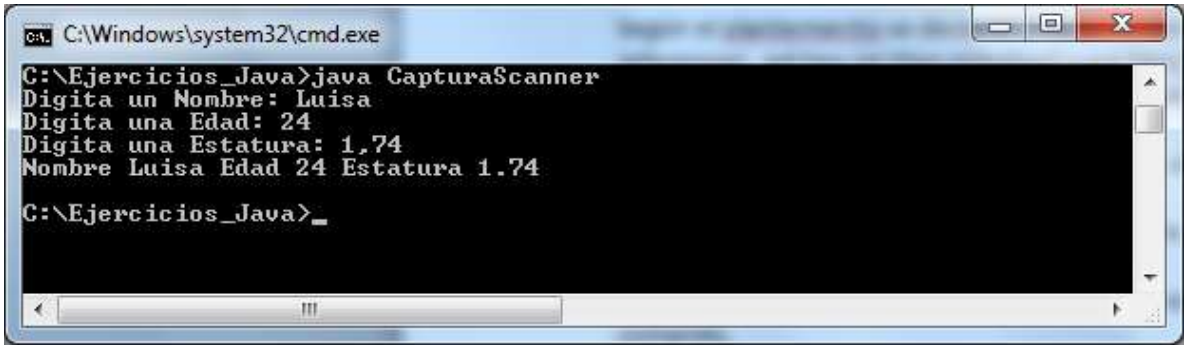
Según el planteamiento se declaran 3 variables con 3 tipos de datos distintos, nom tipo String (tipo referencia), ed tipo int (tipo primitivo) y est tipo float (tipo primitivo), en la captura de los valores se debe tener presente mediante este grafico que según el tipo de datos se cambia el método de lectura.
Nom = nextLine, esta es la forma de captura de una variable tipo texto
Ed = nextInt, forma de captura tipo entero
Est = nextFloat, forma de captura de datos reales

Todas acompañadas de la variable lectura, recuerde respetar la forma de escriturо de cada comando.

Compilación

Elaborado por: César A. Jaramillo H.

Ejecución

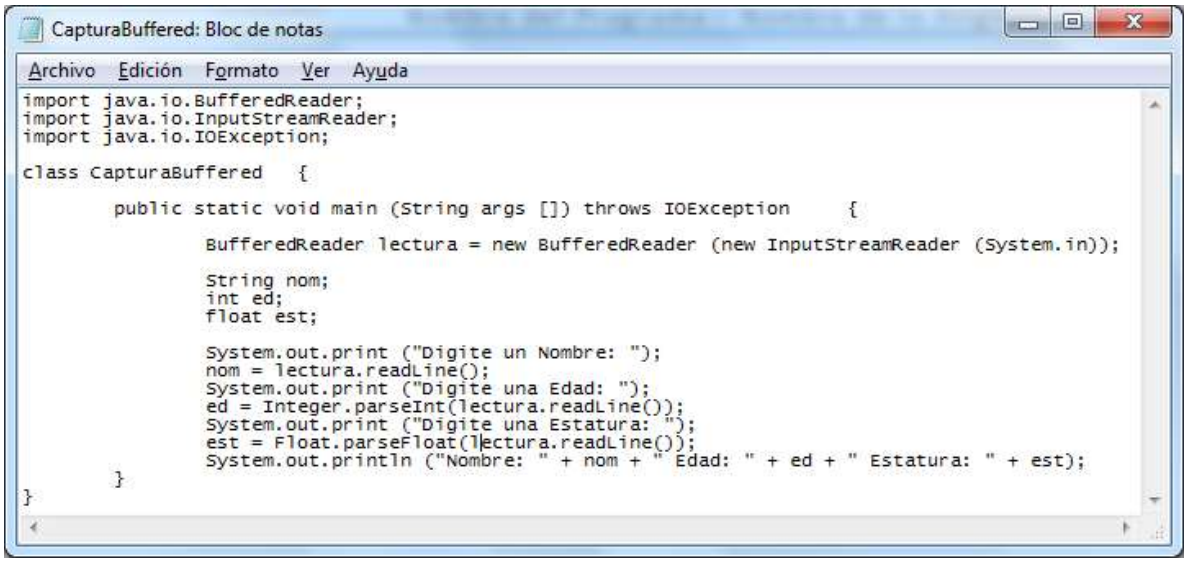


Elaborado por: César A. Jaramillo H.

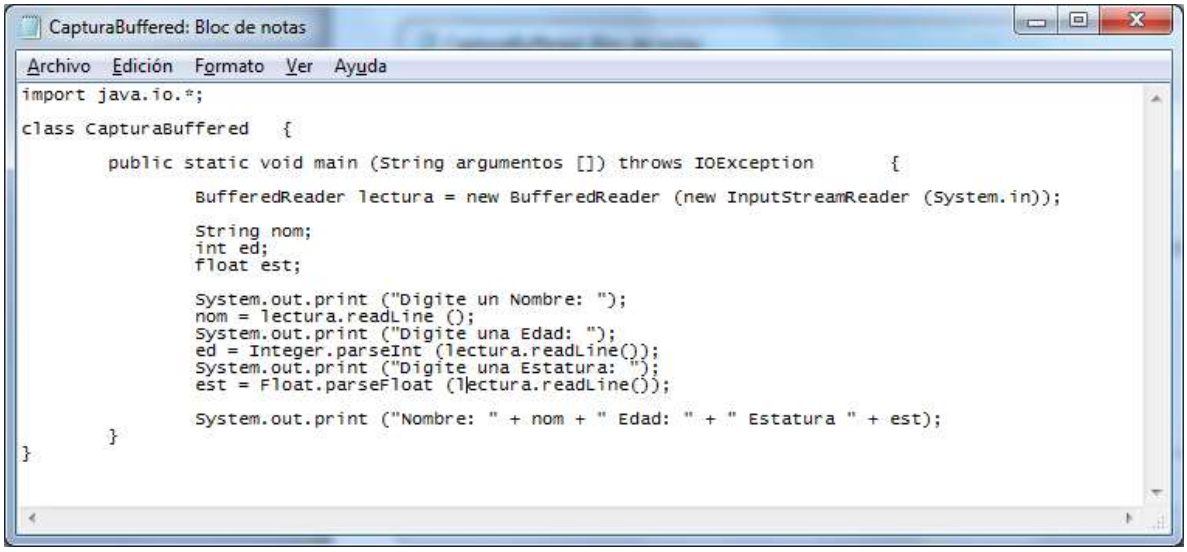
Captura con BufferedReader

Este método de captura de información es más utilizado, aunque sea más extenso en su creación, tiene más alternativas en los procesos.

Creemos el ejemplo anterior con este nuevo método de trabajo.



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.

Observemos que estos dos ejemplos tienen la misma finalidad, solo cambian el encabezado, el primero especifica la clase io (entrada y salida) y los métodos que se requieren para este ejercicio, el segundo toma solo la clase io y asterisco (*) para representa cualquier método que este reconozca.

Los métodos utilizados en esta clase son BufferedReder, InputStremReader y requerimos de uno adiciona IOException que nos permitirá más adelante la captura de excepciones que el sistema pueda arrojar.

La línea más importante para este comando es

BufferedReader lectura = new BufferedReader (new InputStreamReader (System.in));

En esta línea especificamos la variable que vamos a utilizar llamada lectura, recuerden que es una variable y puede tener un nombre distinto, instanciamos con la sentencia new y al final encontramos System.in que nos permite ingresar información.

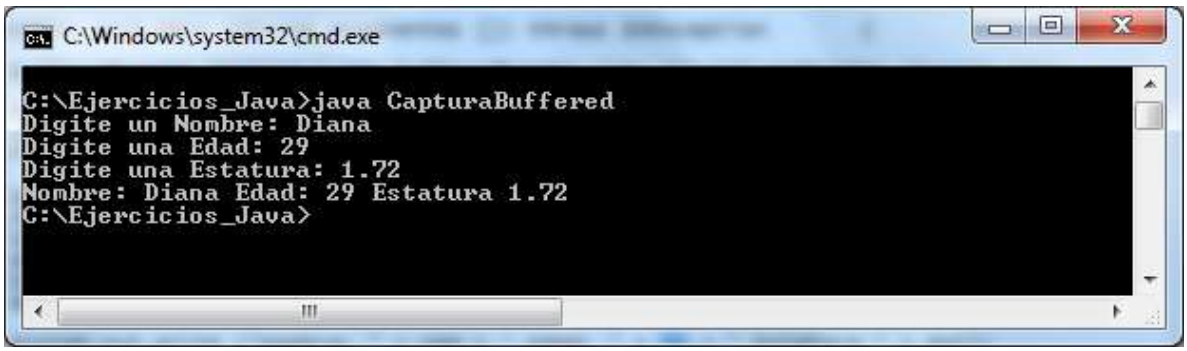
La forma de captura de las variables según su tipo de declaración

Los String se capturan con readLine
Los int se capturan con readLine y previo a esto debe de ir un tipo referencia Integer.parsiInt, este tipo de datos captura la información tipo texto y hay que convertirla para procesarla adecuadamente.
Los float se capturan con readLine y previo a esto va la conversión con Float.parseFloat

Compilación

Elaborado por: César A. Jaramillo H.

Ejecución



Elaborado por: César A. Jaramillo H.

Estructuras Lógicas

Las estructuras lógicas se encargan del procesamiento de la información y la evaluación de los procesos que lo requieren, los más comunes con las preguntas, ciclos y selectores múltiples.

Preposiciones lógicas simples – preposiciones lógicas compuestas.

Las preposiciones se refieren a preguntas o formas de evaluar tareas comunes de nuestro aplicativo, para este caso tomaremos como ejemplo el último método de captura de información.

La sintaxis de las preposiciones es

```
if (condición) {
Procesos
}
else {
Procesos
}
```

Téngase en cuenta que el else o la negación de una condiciones es opcional, además se puede utilizar la sentencia else if como una forma de ahorrar sentencias y líneas de código

```
if (condicion) {
        procesos
}
else {
        if (condicion) {
                procesos
        }
        else {
                if (condicion) {
                        Procesos
                }
                else    {
                        Procesos
                }
        }
}
```

Observemos la cantidad de líneas que una condición anidada puede tener, si aplicamos else if quedaría así

```
if (condicion) {
        Procesos
}
```

```
else if (condicion) {
        procesos
}
else if (condición) {
        Procesos
}
Else    {
        Procesos
}
```

Edad: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

class Edad      {

        public static void main (String argumentos []) throws IOException       {

                BufferedReader lectura = new BufferedReader (new InputStreamReader (System.in));

                int ed;

                System.out.print ("Digite una Edad: ");
                ed = Integer.parseInt (lectura.readLine());

                if (ed >= 18)   {
                        System.out.print ("Es Mayor de Edad");
                }
                else    {
                        System.out.print ("Es Menor de Edad");

                }

        }
}
```

Elaborado por: César A. Jaramillo H.

Ejecución

C:\Windows\system32\cmd.exe

```
C:\Ejercicios_Java>java Edad
Digite una Edad: 34
Es Mayor de Edad
C:\Ejercicios_Java>
```

Elaborado por: César A. Jaramillo H.

Elaborado por: César A. Jaramillo H.

Ciclo para

El ciclo para tiene por característica principal su uso para cantidad de tareas conocidas de forma automática.

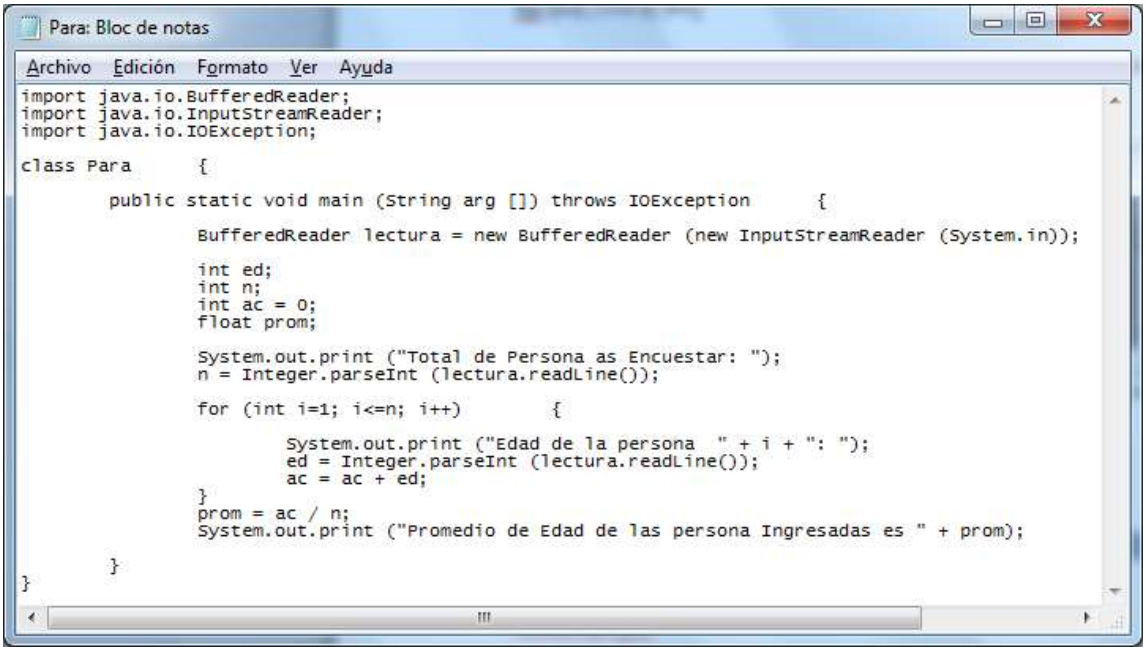
La sintaxis es

Para Índice = inicio; fin, incremento

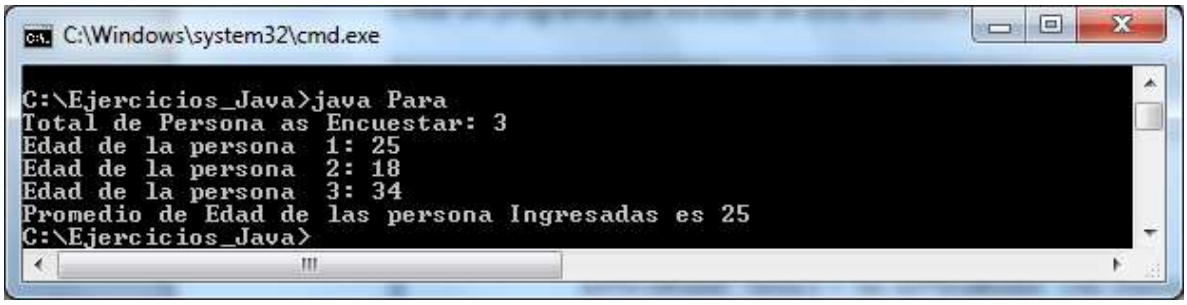
for (i=1 ; i <= n; i++)

Se divide en 3 áreas, la primera el inicio de la variable índice, la segunda indica hasta dónde va el índice y la tercera el incremento, que puede ir i++ o i = i + 1.

Crear un programa que lea edad de una cantidad conocida de personas y calcule su promedio.



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.

Ciclo Mientras que

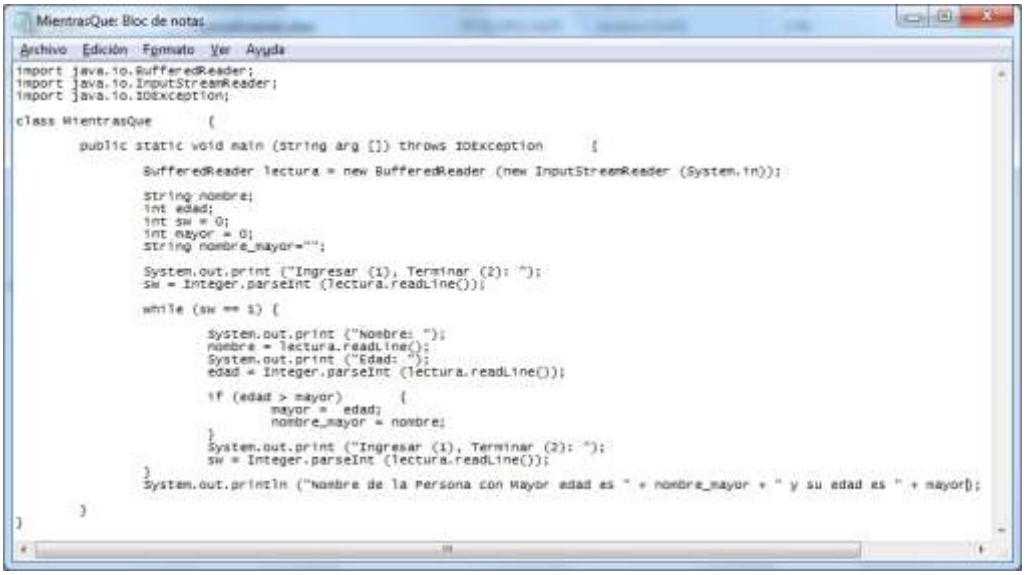
Este ciclo es el más común y más utilizado por su potencia, su característica de permitir tamaños conocidos de datos o no conocidos le da un gran alcance en el manejo de la información.

Leer para una cantidad desconocida de datos, Nombre y Edad, calcular la edad mayor de los ingresados y a quien corresponde.

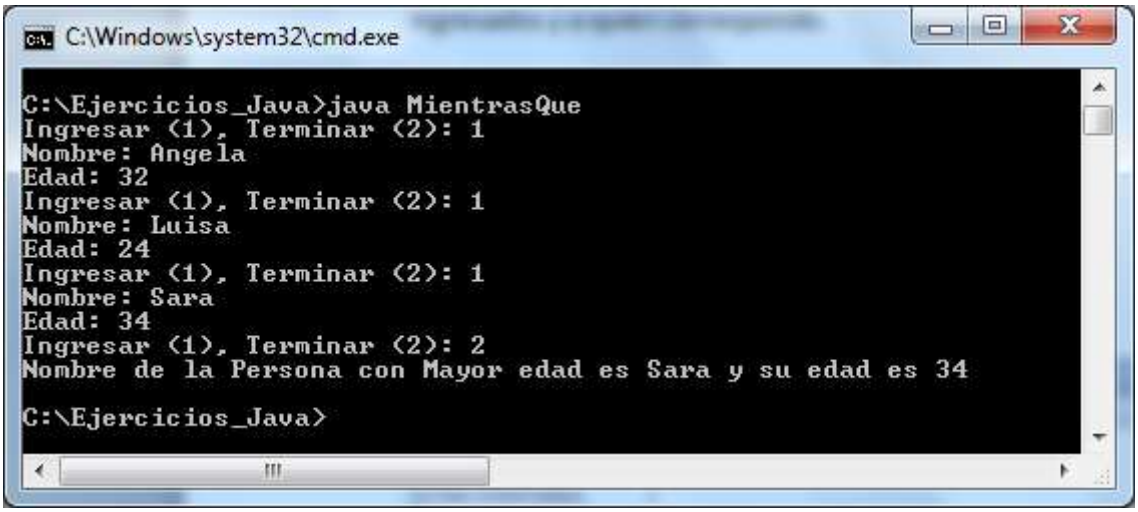
Su sintaxis es

Mientras que (condición)

```
while (condición) {
        Procesos
}
```



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.

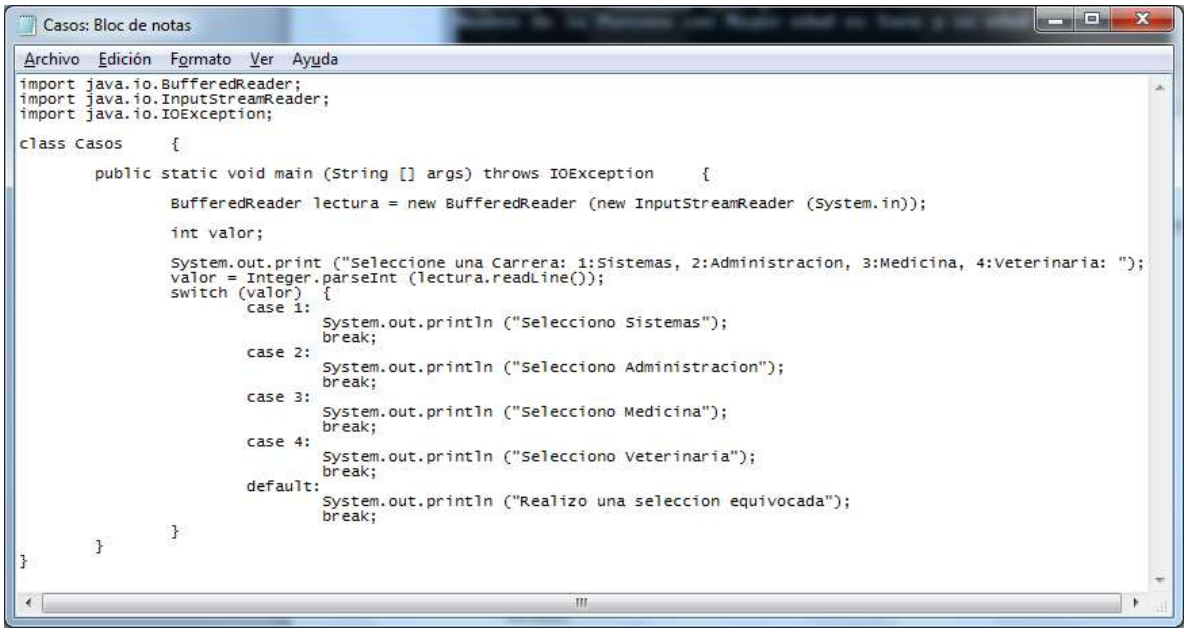
Selector Múltiple

El selector múltiple es una forma simple de evaluar múltiples opciones, a diferencia del condicional if que evalúa una y arroja 2 posibles respuestas.

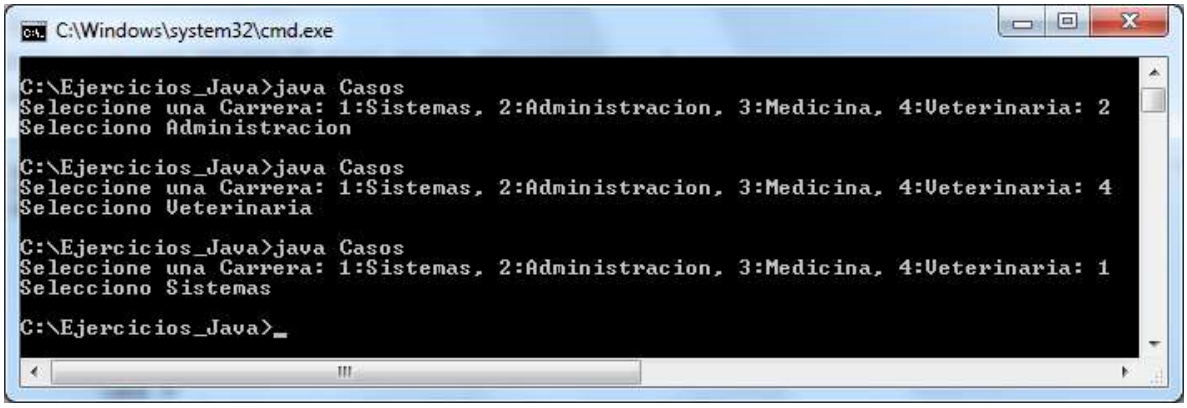
Su sintaxis es

```
switch (variable) {
        case 1:
                proceso;
        breark;
        case 2:
                proceso;
        breark;

        case 3:
                proceso;
        breark;
        default:

        break;
}
```

El selector múltiple recibe un valor, este es evaluado según las condiciones que se establezcan, cada opción se evalúa con case y debe finalizar con la palabra break, si el valor digitado no cumple ninguna condición se utiliza la sentencia default que permite procesar los datos que no estén dentro de las opciones válidas.



Elaborado por: César A. Jaramillo H.



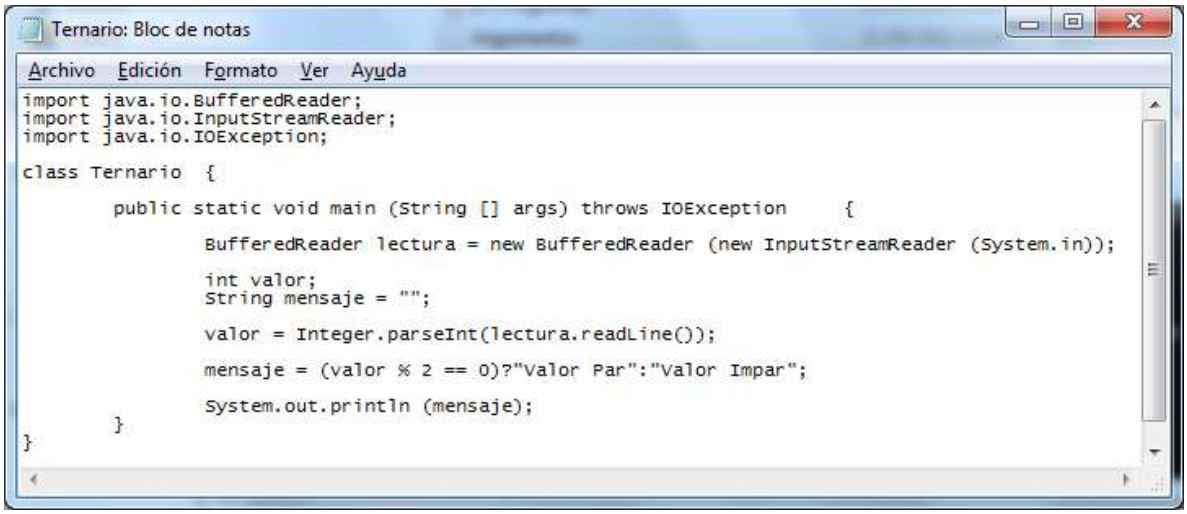
Elaborado por: César A. Jaramillo H.

Operador ternario

El operador ternario es una forma simple de evaluar una condición en una sola línea.

La sintaxis es

Variable receptora = (condición)?”Verdad”: “falso”;



Elaborado por: César A. Jaramillo H.

Elaborado por: César A. Jaramillo H.

4.4.1 EJERCICIO

- Crear la seria siguiente para una cantidad conocida de datos (2 5 11 23 47…=
- Leer para cantidad desconocida de datos, nombre, edad, carrera (1 sistemas, 2 administración, 3, derecho, 4 medicina), calcula porcentaje más alto de alumnos según la carrera.

> Las estructuras de control son herramientas base en todo el proceso lógico, por esto hacemos referencia constante porque nos permite ampliar la cantidad de opciones dentro de la programación (http://zarza.usal.es/~fgarcia/doc/tuto2/II_4.htm).

4.5 TEMA 5 NORMAS BÁSICAS DE DESARROLLO

a menudo los procesos de desarrollo se realizan como el programador cree que debería realizarse esta actividad, existen una seria de normas lógicas, una serie de normas de los lenguajes, pero lo que muchos desconocen son las normas que la empresa creadora del software establece para darle a sus desarrollares un perfil de uso, de aprovechamiento de los espacios, de modo que cualquier programador pueda observar el código de los demás y se dé una idea de que se conocen normas y características del lenguaje.

Sun MicroSystem, creo a finales de los años 90's, una serie de normas para los desarrolladores de java y Oracle su actual propietario las conserva, consta de alrededor de 120 normas, de las cuales se han utilizado algunas que iremos mencionado.

Nombre de los archivos: se recomienda que sea en mayúscula inicial, sin espacios ni símbolos, cada palabra que la conforme debe de ir con estas normas.

Clases: debe de contener el mismo nombre del archivo, en escritura y en su forma de escritura, además se recomienda que la llave que da apertura al programa este al final del nombre de dicha clase, no debajo como se acostumbra en la mayoría de los lenguajes.

Métodos: deben de iniciar en minúscula, si se compone de más de una palabra a partir de la segunda va en mayúscula inicial, no lleva símbolos, ni espacios.

Variables: cumple la misma norma de los métodos, inician en minúscula, si la complementa con varias palabras a partir de la segunda va en mayúscula inicial.

Utilizar sangría para darle una distribución correcta al código.

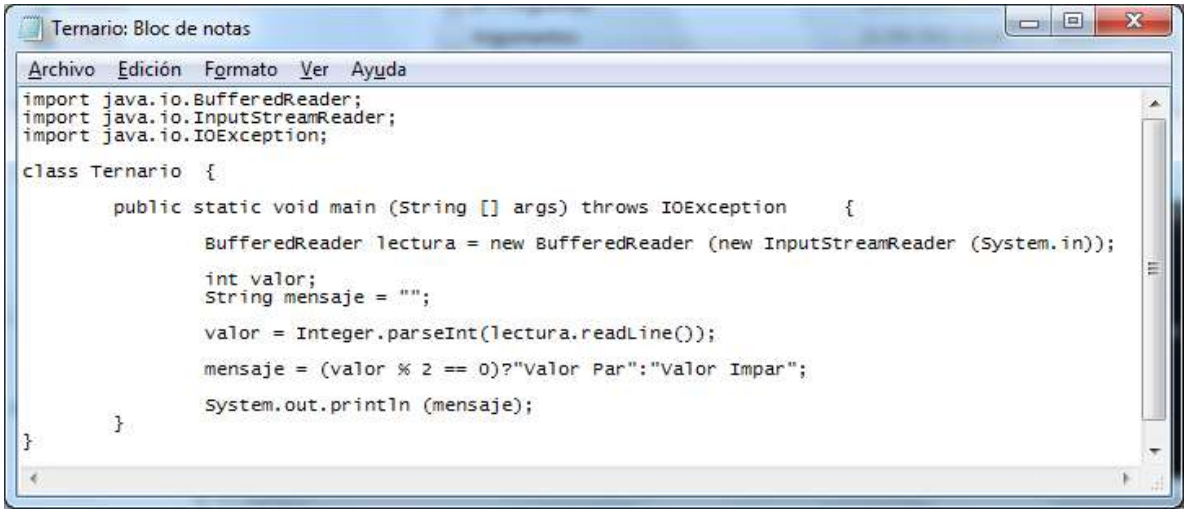
Cerrado de las llaves: estas se ubican en la línea de inicio de la clase, método, función, etc, no debajo de la apertura de llave.

Se recomienda que entre la clase y el método siguiente exista un espacio (enter), esto mismo sucede con la declaración de las variables, terminada la declaración, dejar un espacio para las asignación, lecturas y procesos, también aplica para todos los procesos que generen una llave de apertura.

La sentencia return, debe de ir con un espacio antes de él y un espacio posterior a él.

Longitud de las líneas: es recomendado que las líneas no sean excesivamente largas, es recomendable máximo 80 caracteres y se realizara una continuación de ser necesario.

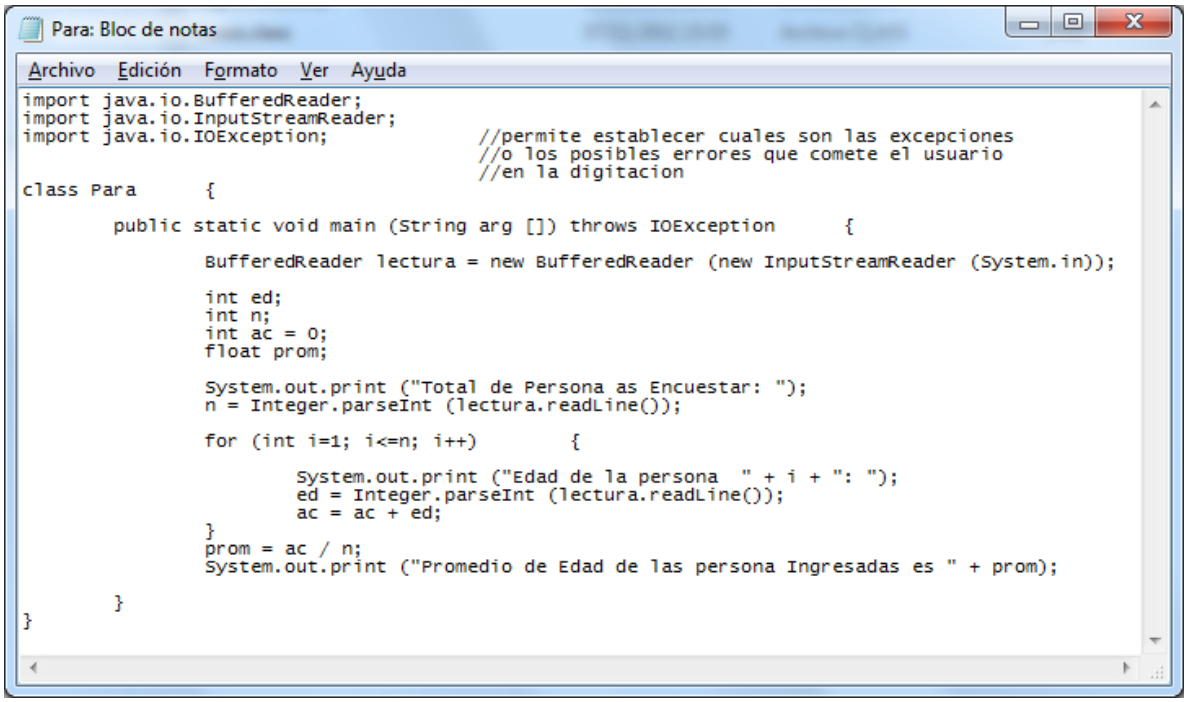
Con estas normas básicas tenemos un programa adecuado según su constructor Oracle.



Elaborado por: César A. Jaramillo H.

Comentarios: los comentarios en los lenguajes de programación se utilizan para varias tareas, en primera instancia permiten identificar o explicar comandos, funciones, líneas de código, como segundo propósito está el de anular una o varias líneas de código con fines de prueba, veamos un ejemplo de esto.

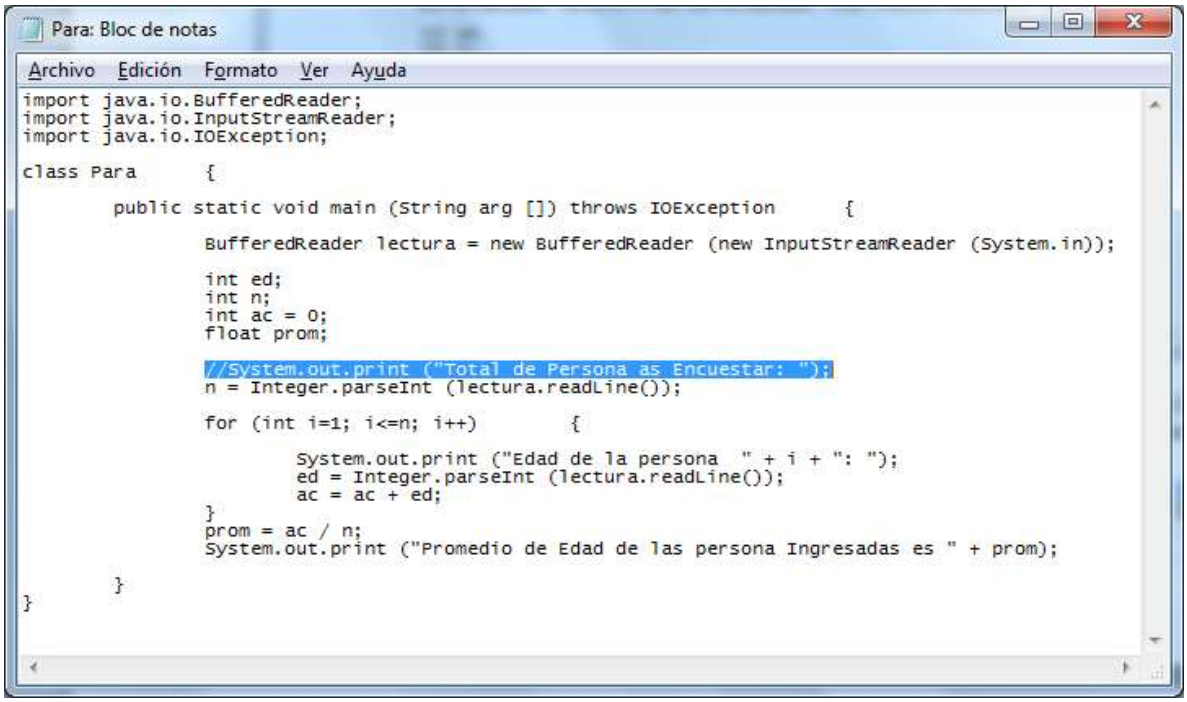
Comentarios de una línea con //



Elaborado por: César A. Jaramillo H.

En el ejemplo anterior observamos al frente del import java.io.IOException un par de slash (//), estos símbolos permiten que se ubique un comentario, pero solo tiene valides en la línea que se ubica, si observamos en la línea siguiente aparece otro par de slash, para otro comentario o una continuación, estos slash, se pueden colocar al inicio de una sentencia, al frente o al final.

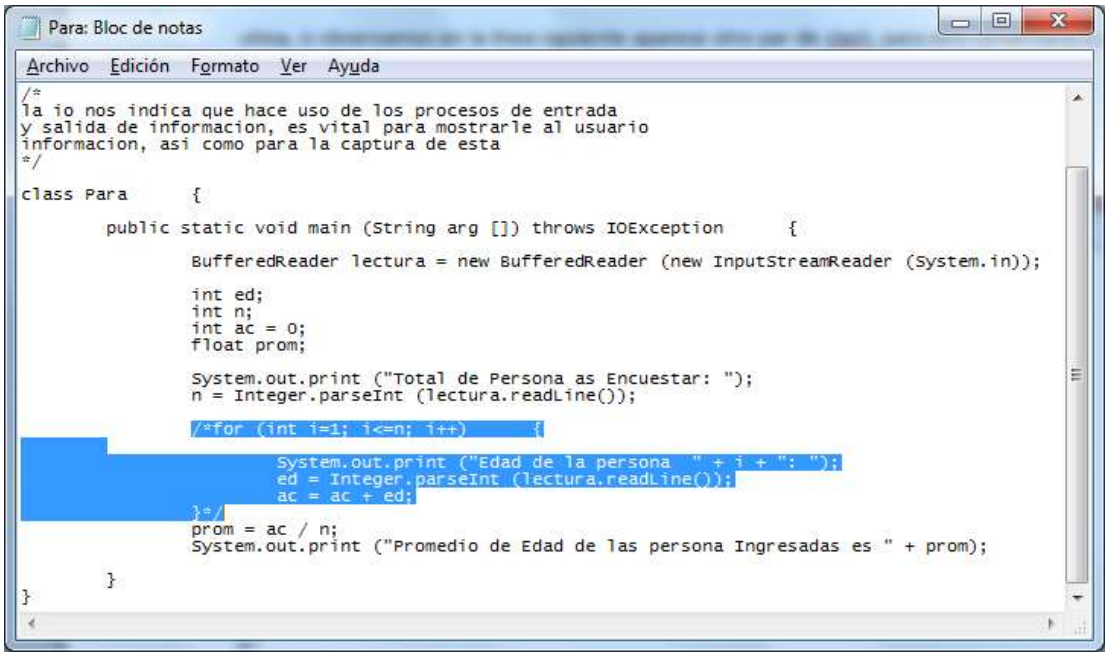
Comentarios como anulación de una línea de código



Elaborado por: César A. Jaramillo H.

Observemos la línea subrayada, inicia con doble slash, esto no es un comentario, es la anulación de una línea de código del programa, hay que tener presente que cuando se colocan los comentarios estas líneas no se tienen en cuenta a la hora de compilar o ejecución del programa.

Comentario o anulación de código de varias líneas



Elaborado por: César A. Jaramillo H.

Obsérvese que en este ejemplo utilizamos una forma distinta de procesamiento de la información, utilizamos los símbolos slash y asterisco para iniciar un comentario de varias línea y asterisco slash para cerrarlo, el primero lo encontramos antes de la clase y el segundo lo encontramos en una anulación del ciclo para.

Documentación:

La documentación es una herramienta provista por java para facilitar los procesos finales y posibles continuaciones futuras a manos de personal diferente al que creo el aplicativo, esta documentación se va desarrollando a la par con el aplicativo, y permitirá que sea generada y manipulada de una forma más ligera y con menos contratiempos de los procesos de otros lenguajes, en otros lenguajes esta no se ve, hay personal que desarrolla y personal que documenta, explica la codificación que se está creando, con la posibilidad de errores y falta de conceptos que java previene de una manera muy alta.

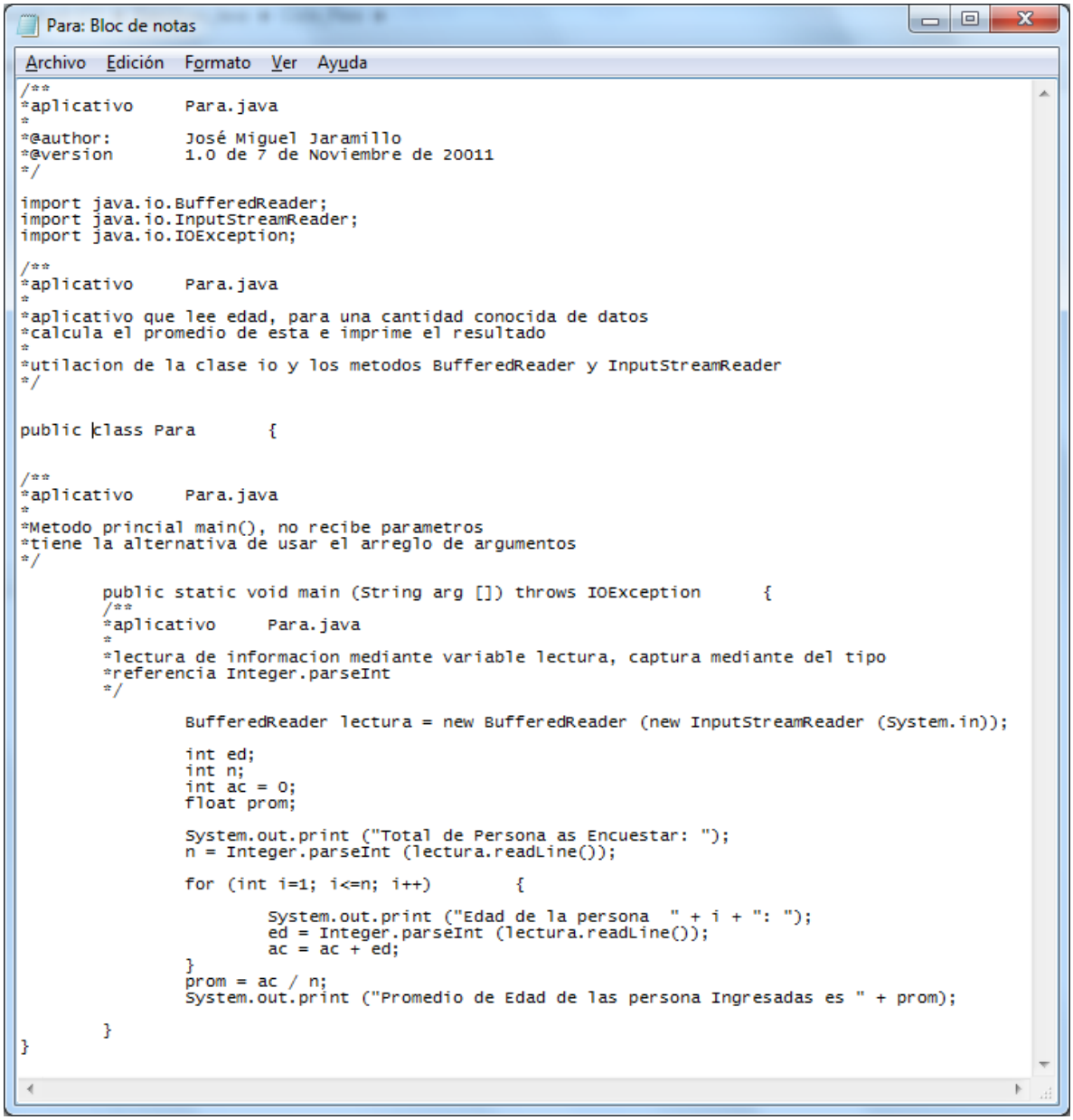
Creación de documentación. Ejemplo

Elaborado por: César A. Jaramillo H.

La documentación va entre los símbolos
/**

*/

Similar a los comentarios previamente vistos, esta documentación permite que se identifiquen características del programa que se está desarrollando como, el nombre del aplicativo, el autor, la versión, las excepciones, los parámetros, entre otras opciones.




```
/**
*aplicativo     Para.java
*
*@author:       José Miguel Jaramillo
*@version       1.0 de 7 de Noviembre de 20011
*/

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

/**
*aplicativo     Para.java
*
*aplicativo que lee edad, para una cantidad conocida de datos
*calcula el promedio de esta e imprime el resultado
*
*utilacion de la clase io y los metodos BufferedReader y InputStreamReader
*/


public class Para       {


/**
*aplicativo     Para.java
*
*Metodo princial main(), no recibe parametros
*tiene la alternativa de usar el arreglo de argumentos
*/

        public static void main (String arg []) throws IOException      {
        /**
        *aplicativo     Para.java
        *
        *lectura de informacion mediante variable lectura, captura mediante del tipo
        *referencia Integer.parseInt
        */

                BufferedReader lectura = new BufferedReader (new InputStreamReader (System.in));

                int ed;
                int n;
                int ac = 0;
                float prom;

                System.out.print ("Total de Persona as Encuestar: ");
                n = Integer.parseInt (lectura.readLine());

                for (int i=1; i<=n; i++)        {

                        System.out.print ("Edad de la persona  " + i + ": ");
                        ed = Integer.parseInt (lectura.readLine());
                        ac = ac + ed;
                }
                prom = ac / n;
                System.out.print ("Promedio de Edad de las persona Ingresadas es " + prom);

        }
}
```

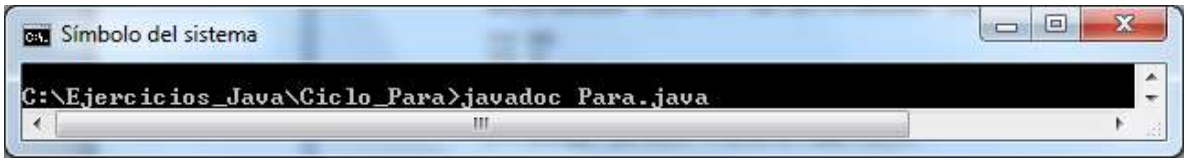
Elaborado por: César A. Jaramillo H.

Esta documentación no afecta la compilación ni la ejecución del programa que se esta desarrollando.

Modo de creación de la documentación.

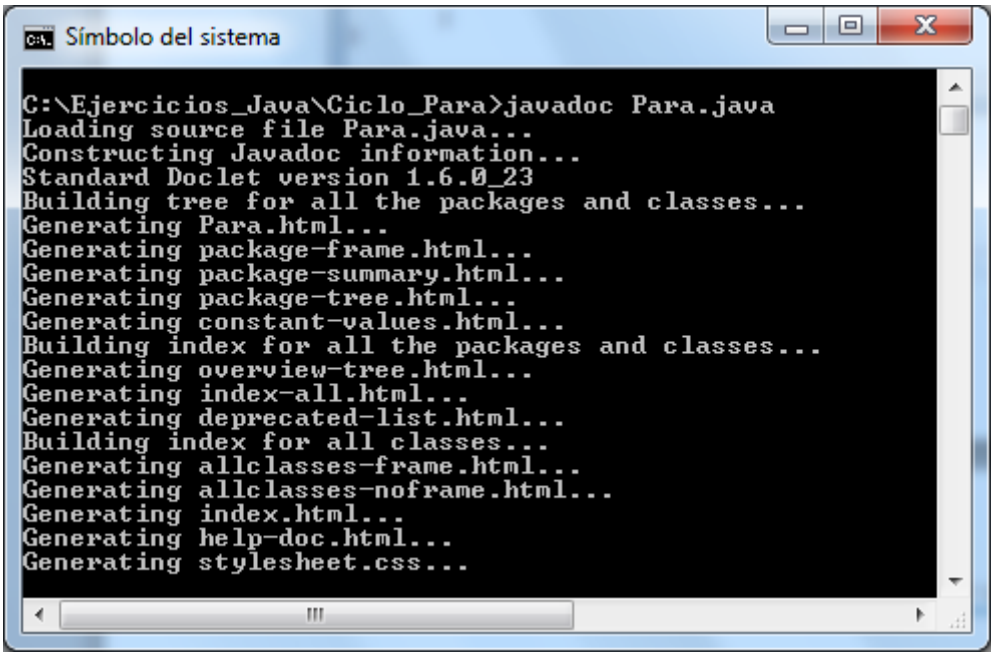
Existe una sola condición para esto, la clase debe ser pública (public)

Para crear la documentación se utiliza le comando javadoc, acompañado del nombre y la extensión del archivo.



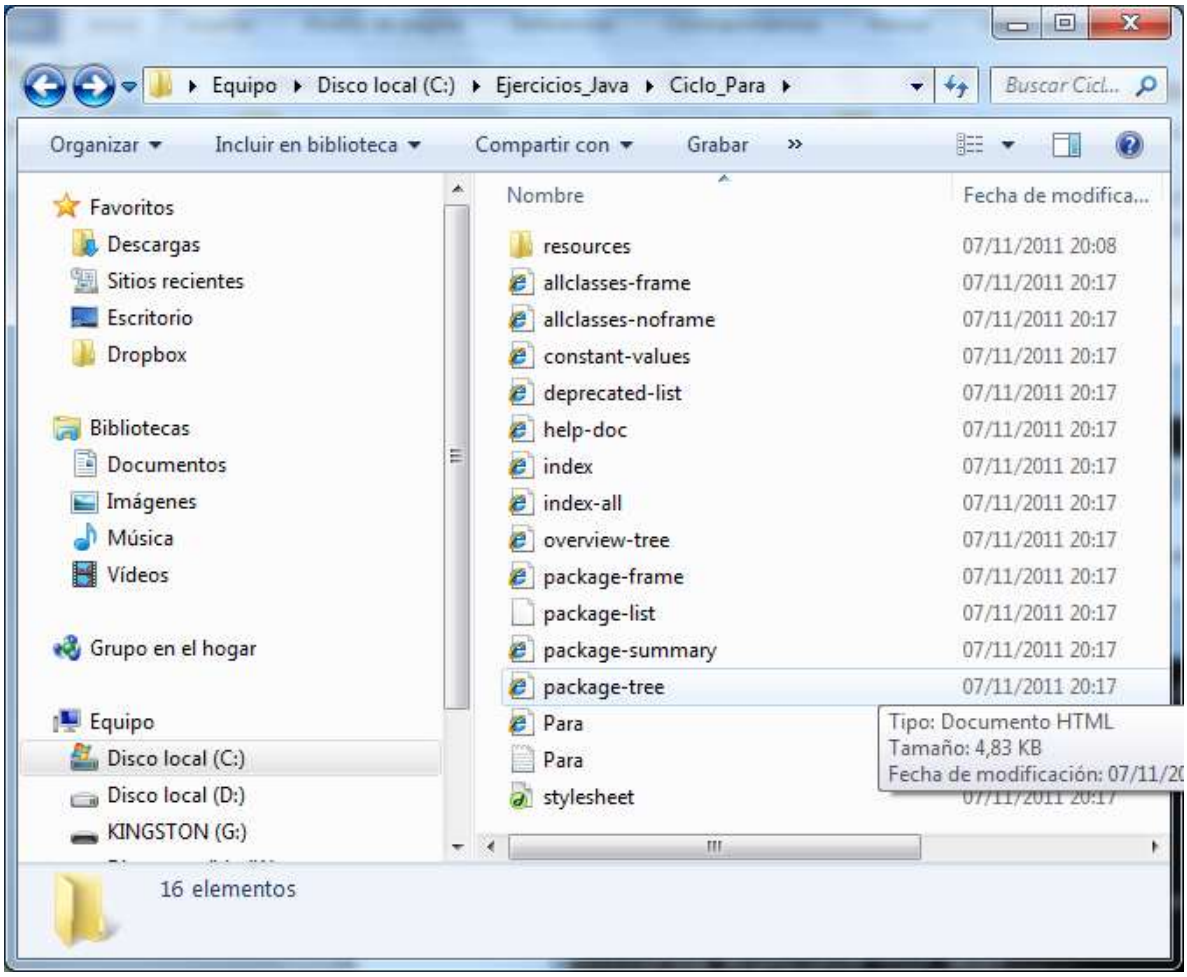
Elaborado por: César A. Jaramillo H.

Y se genera un grupo de archivos en formato .html que contiene la documentación.

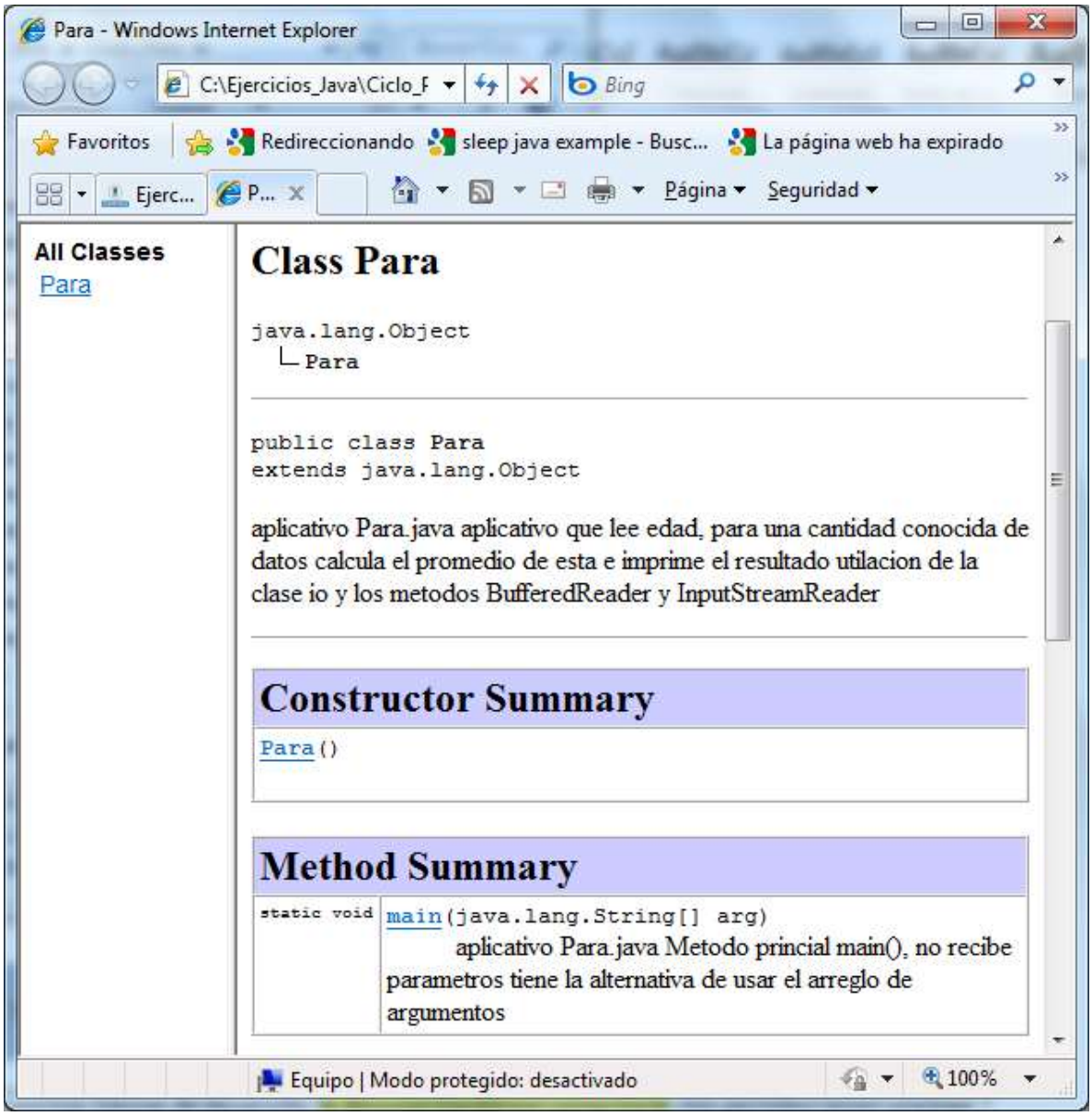


Elaborado por: César A. Jaramillo H.

En la ubicación del archivo con extensión .java, se creara este grupo de archivos, del cual ejecutaremos, index.htm



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.

Obsérvese que muestra parte de lo que se ubicó en el programa, se puede hacer un recorrido por todos los vínculos y veremos la información creada por nosotros.

4.5.1 EJERCICIO

- Buscar las normas complementarias que ofrece Oracle y aplicarlas a los nuevos proyectos de desarrollo.

- Crear un programa que lea para una cantidad conocida de datos y genere la seria (1 22 333 4444 55555 666666….)

> Las normas básicas de desarrollo, la documentación y el comentario, nos permiten tener una base muy firme a la hora de crear una aplicativo (http://gpd.sip.ucm.es/rafa/docencia/programacion/tema1/java-doc.html)

4.6 TEMA 6 CLASES COMUNES

Dentro del lenguaje de programación java, existe un número muy amplio de clases, estas están diseñadas para propósitos específicos, algunas para crear ambientes gráficos, otros para procesos matemáticos, otros para el ingreso y salida de información, etc., cada área que se desee vincular encontrara una clase para dicha tareas. Esto facilita al programador de tal modo que no tenga que colocar clases de manera innecesaria, sino que colocara solo las que requiera, evitando así, desperdician de espacio en la memoria.

Clases comunes.

Para nuestro curso, las clases comunes utilizadas son:

Lang: clase por defecto que permite la manipulación de datos, variables e impresión de información.

Io: hace referencia a entrada y salida de información

Scanner: permite la captura de información mediante el objeto del mismo nombre.

Existen otras que más adelante se trabajaran como son

Swing: permite trabajar con un ambiente grafico muy profesional y un conjunto de controles muy amplios.

Applet: permite la creación de páginas livianas, que se pueden mezclar con sitios web

Awt: ambiente gráfico, fue el primero que creo java, aún sigue vigente.

Math: funciones matemáticas, adicionales a la aritmética tradicional.

Sql: interprete de sentencias sql, asociado a la conexión de un motor de bd.

4.6.1 EJERCICIOS

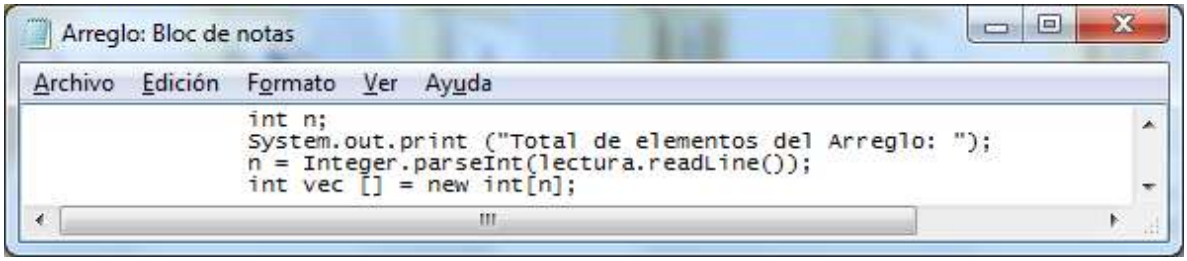
- Buscar las clases antes mencionadas en el API de java y verificar funciones alternas o complementarias que este pueda tener.

> El API de java es un conjunto muy amplio de ayudas y de información relacionada con la programación en java, es recomendado siempre tenerlo presente y a la mano para conocer las características, escritura y funcionamiento (http://download.oracle.com/javase/6/docs/api/).

4.7 TEMA 7 INTRODUCCIÓN A LOS ARREGLOS

Los arreglos dentro de programación son estructuras estáticas, que permiten que dentro de una misma variable existan N cantidad de elementos, en posiciones sucesivas, a diferencia de las variables tradicionales que permiten un valor al tiempo, en caso de un nuevo valor al anterior desaparece, estas estructuras aportan un valor significativo a la lógica de programación, y permite múltiples operaciones en paralelo con la manipulación de los datos.

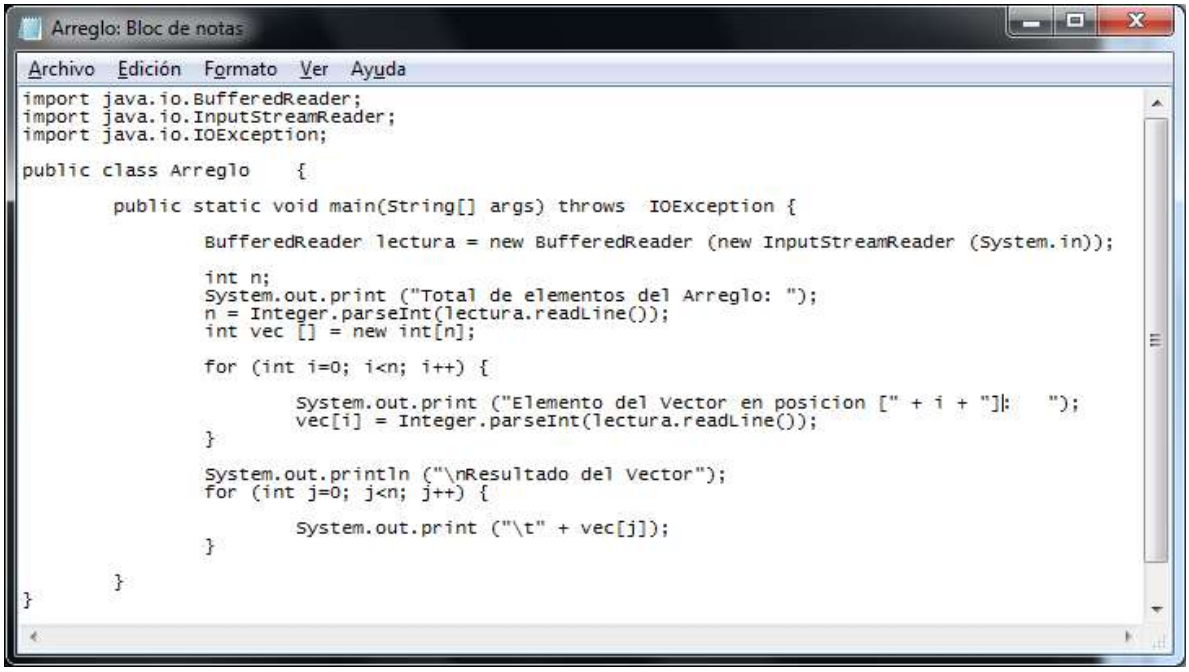
Para la creación de un arreglo unidimensional (hay que tener presente que existen de varias dimensiones), se realiza el siguiente proceso



Elaborado por: César A. Jaramillo H.

Obsérvese que se declara una variable n, esta se lee y posteriormente se declara el vector, la forma de hacer esto es int vec [] = new int [n], esta línea nos indica que el vector es de tipo entero, y se instancia tipo int (tipo primitivo) con un tamaño de n elementos que fue lo que se leyó previamente, también se puede establecer un valor predeterminado en lugar de n, más adelante veremos esto.

Ejemplo
Crear un vector de tamaño conocido e imprimirlo

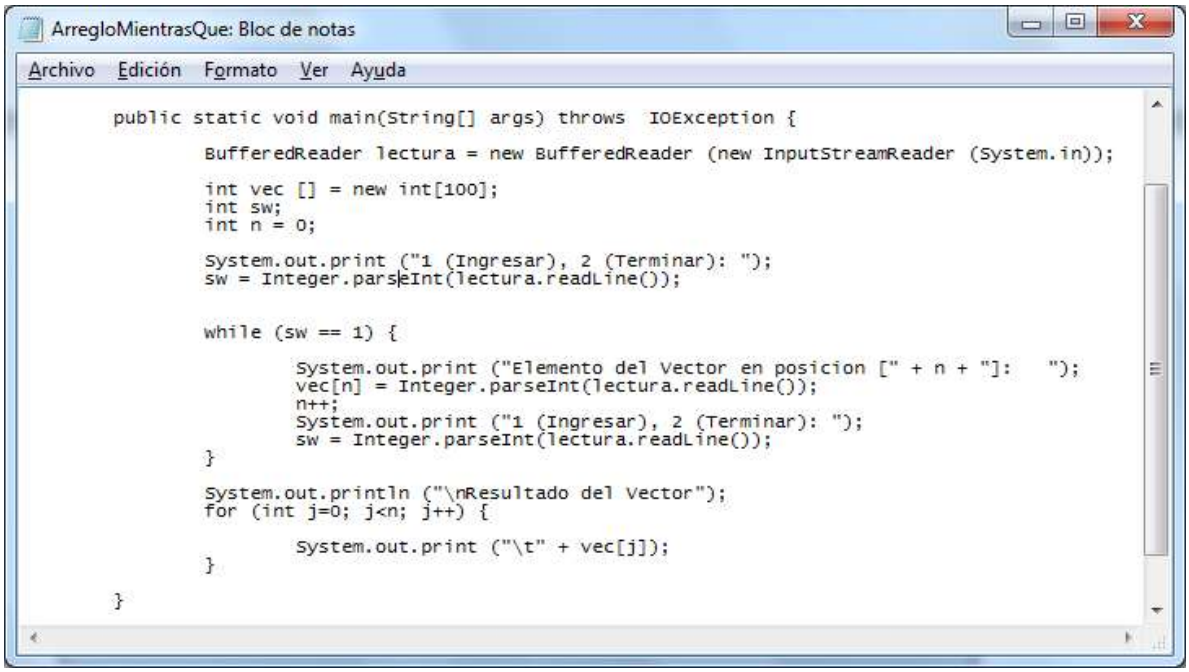


Elaborado por: César A. Jaramillo H.

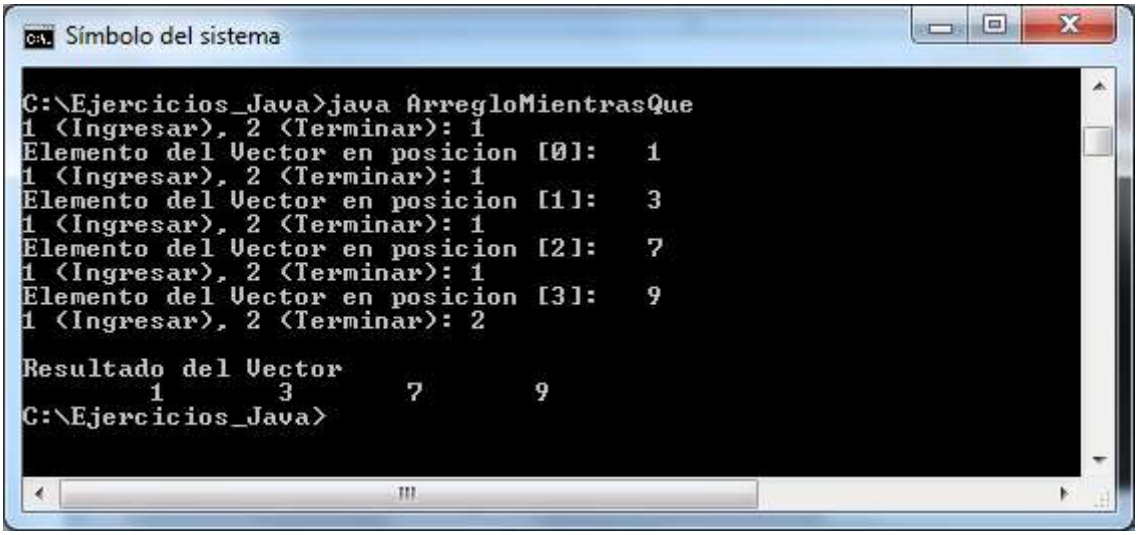
Elaborado por: César A. Jaramillo H.

Dentro de esta creación encontramos la declaración de las variables principales, pero no las variables i y j, estas las estamos declarando dentro del mismo ciclo, cualquiera que se la declaración es válida, después de la línea de captura o dentro del ciclo, también encontramos que el ciclo en su índice inicia en cero (0), todos los arreglos, en cualquier lenguaje siempre inician en cero (0) y terminan en n – 1.

Arreglos de tamaño desconocido



Elaborado por: César A. Jaramillo H.



Elaborado por: César A. Jaramillo H.

En esta creación encontramos que el vector esta predeterminado a 100 posiciones, esta es otra forma de definirlos de manera predeterminada.

4.7.1 EJERCICIO

- Crear un vector de tamaño desconocido que solo permita números pares
- Crear un vector que lea N elementos, elimine los repetidos e imprima el vector resultante
- Crear un vector de N elementos, invertir los elementos del vector, el primero para la última posición, el ultimo para la primera, el segundo para la penúltima y el penúltimo para la segunda, así sucesivamente, no utilizar vectores adicionales ni imprimir de atrás hacia adelante.

> **Los arreglos son herramientas lógicas de gran valor por sus características unidimensionales**
>
> **(http://codigoprogramacion.com/java/96-arreglos-en-java.html).**

5 PISTAS DE APRENDIZAJE

Recuerde que: java es un lenguaje orientado a objetos y requiere de una compresión de términos y condiciones que permitan el aprovechamiento del lenguaje.

Tenga en cuenta: que java es un ambiente de desarrollo que puede apoyarse en IDEs que le permitan aprovechar los recursos y el tiempo de desarrollo.

Traiga a la memoria: los tipos de variables primitivas y de referencia, le serán de mucha ayuda a la hora de procesar la información.

6 GLOSARIO

JDK	Java Development Kid (conjunto de herramientas de desarrollo de Java)
Java ME	Java Micro Edition (ambiente para desarrollo de dispositivos móviles)
Java SE	java Standard Edition (Edición de propósito general de Java)
Java EE	java Enterprise Edition (Edición empresarial de Java)
IDE	Entorno de Desarrollo Integrado
JVM	Java Virtual Machine (Máquina Virtual de Java)
JRE	Java Runtime Environment (ambiente de ejecución de Java)
API	Application Programming Interface (Interfaz de Programación de Aplicaciones)
Compilación	proceso que permite determina que no existan errores de sintaxis en un aplicativo
Ejecución	proceso que permite ver en funcionamiento un aplicativo
Comando	instrucción que se aplica directamente a un proceso
Función	instrucción de envía y recibe parámetros
Arreglo	estructura estática de memoria, permite almacenar múltiples datos en la misma variable
Asignación	forma de llevar información o procesos a otros
Bytecode	interprete entre el archivo objeto y la máquina que lo va a ejecutar.
Multiplataforma	posibilidad de trabajar sobre distintos sistemas operativos.

7 BIBLIOGRAFÍA

Fuentes bibliográficas

HOLZNER, Steven. La biblia de java 2. Anaya multimedia 2000

CEBALLOS, Francisco Javier. Java 2 Interfaces gráficas y aplicaciones para internet. Alfaomega Ra-Ma. 2006

JOYANES AGUILAR, Luis; FERNANDEZ AZUELA, Matilde. Java 2 Manual del programador. Ra-Ma. 2001

CEBALLOS, Francisco Javier. Java 2 Curso de Programación. Ra-Ma. 2005

ECKEL, Bruce. Pensando en Java. Prentice-Hall. 2000

VILLALOBOS, Jorge; CASALLAS, Ruby. Fundamentos de Programación. Prentice Hall. 2006

DEAN, John; DEAN, Raymond. Introducción a la programación en java. McGraw Hill

FERNANDEZ, Carmen. Java Básico. Starbook.

Fuentes digitales o electrónicas

www.oracle.com

http://profesores.fi-b.unam.mx

www.java.com

upi2.uniandes.edu.co

www.programacion.com/java

www.monografias.com

http://www.javahispano.org/