Unidad I

Fundamentos del lenguaje.

1.1 Entorno de desarrollo.

API Java, proporciona a los programadores un entorno de desarrollo completo, así como una infraestructura. JVM, la máquina virtual que ejecuta bytecode de Java.
Lenguaje de programación orientado a objetos desarrollado por Su Microsystems a principios de los 90Las aplicaciones son compiladas en un bytecode, aunque también es posible la compilación en código de máquina nativo. En tiempo de ejecución, el bytecode es normalmente interpretado

1.2 Configuración del entorno de desarrollo.

En esta sección, se proporciona información acerca del modo de configurar el entorno de desarrollo para desarrollar programas en lenguaje java. El entorno de desarrollo es el lugar donde los programadores y diseñadores crean programas para facilitarse la tarea manual. Los IDEs (Integrated Development Environment) son un conjunto de herramientas
parael programador, que sulen incluir en una misma suite, un buen editor de código, administrador de proyectos y archivos, enlace transparente a compiladores y debuggerse integración con sistemas controladores de versiones o repositorios. Cuando de Java se trata, son varias las opciones de IDEs, para utilizar. Gracias aEl CoDiGo K mencionamos algunos de los principales y más populares:

- BlueJ: desarrollado como un proyecto de investigación universitaria, es libre.
- Eclipse: desarrollado por la Fundación Eclipse, es libre y de código abierto,
- IntelliJ IDEA: desarrollado por JetBrains, es comercial.
- Jbuilder : desarrollado por Borland, es comercial pero también existe la versióngratuita.
- JCreator : desarrollado por Xinox, , es comercial pero también existe la versióngratuita.
- JDeveloper : desarrollado por Oracle Corporation, es gratuito.
- NetBeans – gratuito y de código abierto.

1.3 Palabras reservadas.

L as palabras reservadas, son palabras definidas en Java, que NO se pueden utilizar como nombres en variables, clases o métodos. True, false y null tampoco pueden ser utilizadas, y junto con la sintaxis de los operadores y separadores, forman la definición del lenguaje.

1.4 Comentarios.

Existen diferentes tipos de comentarios que pueden incluirse dentro de nuestro código Java.

Tenemos los comentarios de linea, que inician con los caracteres "//" y terminan en la misma linea en que se incluyen.

Los comentarios de bloque, que inician con los caracteres "/*" y terminan con los caracteres "*/", y pueden extenderse en una o varias lineas.

Y los comentarios para documentación, que inician con los caracteres "/**" y terminan con los caracteres "*/"

1.5 Tipos de datos.

En Java, contamos con 8 tipos de datos primitivos, cuatro son para valores enteros, 2 son para valores con punto decimal, uno es para almacenar un caracter, y uno mas para datos booleanos, que solo puede tener valor de verdadero o falso.

Tipo primitivo	Tamaño	Valor Mínimo	Valor Máximo	Clase	
byte	8 bits	-128	127	Byte	
short	16 bits	-32,768	32,767	Short	
int	32 bits	-2,147,483,648	2,147,483,647	Int	
long	64 bits	-9,223,372,036,854,775,808	9,223,372,036,854,775,807	Long	
float	32 bits	-3.4028235E + 38 to -1.4E – 45	1.4E–45 to 3.4028235E + 38	Float	
double	64 bits	-1.7976931348623157E + 308 to -4.9E–324	4.9E - 324 to 1.7976931348623157E + 308	Doble	
char	16 bits	Unicode 0	Unicode 2	Character	
boolean	-	false (no es valor mínimo)	true (no es valor máximo)	Boolean	

1.6 Variables.

Una variable, en Java, es un identificador que representa un espacio de memoria que contiene información. La información es del tipo con que se declaro la variable.
Hay 2 categorías de tipos para variables, los tipos primitivos (como int, short, byte, long, char, boolen y double) y las referencias a objetos (como String, Array y otros objetos).
Java tiene ademas, tres tipos de variables, que son:

- De instancia
- De clase
- Locales

Las variables de instancia, se usan para guardar los atributos de un objeto particular.
Las variables de clase, son variables que guardan el mismo valor, para todos los objetos de una clase determinada.
Las variables locales, se utilizan dentro de los métodos, en Java, las variables locales pueden declararse en el momento en que serán utilizadas, y una buena costumbre es inicializar las variables al momento de declararlas.

1.7 Constantes.

Una constante, es una variable que mantiene un valor inmutable durante la ejecución del programa, se define utilizando el modificador "final".

1.8 Operadores.

En Java, contamos con operadores aritméticos

+ suma
- resta
* multiplicación
/ división
% residuo
++var preincremento
--var predecremento
var++ postincremento
var-- postdecremento

Operadores de asignación
= asignación
+= asignación de suma
-= asignación de resta
*= asignación de multiplicación
/= asignación de división

%= asignación de residuo

Operadores relacionales
< menor que
<= menor o igual que
> mayor que
>= mayor o igual que
== igual a
!= no igual

Operadores lógicos
&& AND
|| OR
! NOT
^ OR exclusivo

1.9 Sentencias.

Para entender este capítulo el lector debe conocer la definición de la Sentencia en donde Alfredo de la Cruz Gamboa define como “La sentencia, se entiende como la resolución llevada a cabo por el órgano jurisdiccional que pone fin a un procedimiento judicial. La sentencia contiene una declaración de voluntad del juez o tribunal en la que se aplica el Derecho a un determinado caso concreto, es condenatoria o estimatoria cuando el juez o tribunal acoge la pretensión del demandante, es decir, cuando el dictamen del juez es favorable al demandante o denunciante. Por el contrario, la sentencia es absolutoria o desestimatoria cuando el órgano jurisdiccional da la razón al demandado o denunciado. Son sentencias firmes aquéllas que no admiten contra ellas la interposición de algún recurso ordinario o extraordinario. Se contraponen a las no firmes o recurribles o también llamadas definitivas que son aquellas contra las que cabe interponer recurso”.

1.10 Conversión de tipos de datos (cast).

En Java es posible transformar el tipo de una variable u objeto en otro diferente al original con el que fue declarado. Este proceso se denomina **"conversión", "moldeado" o "tipado"** y es algo que debemos manejar con cuidado pues un mal uso de la conversión de tipos es frecuente que dé lugar a errores.

Una forma de realizar conversiones consiste en colocar el tipo destino entre paréntesis, a la izquierda del valor que queremos convertir de la forma siguiente: Tipo VariableNueva = (NuevoTipo) VariableAntigua;

Por ejemplo: int miNumero = (int) ObjetoInteger; char c = (char)System.in.read();

En el primer ejemplo, extraemos como tipo primitivo int el valor entero contenido en un campo del objeto Integer. En el segundo caso, la función read devuelve un valor int, que se convierte en un char debido a la conversión (char), y el valor resultante se almacena en la variable de tipo carácter c.

El tamaño de los tipos que queremos convertir es muy importante. No todos los tipos se convertirán de forma segura. Por ejemplo, al convertir un long en un int, el compilador corta los 32 bits superiores del long (de 64 bits), de forma que encajen en los 32 bits del int, con lo que si contienen información útil, ésta se perderá. Este tipo de conversiones que suponen pérdida de información se denominan “conversiones no seguras” y en general se tratan de evitar, aunque de forma controlada pueden usarse puntualmente.

De forma general trataremos de atenernos a la norma de que **"en las conversiones debe evitarse la pérdida de información".**En la siguiente tabla vemos conversiones que son seguras por no suponer pérdida de información.

TIPO ORIGEN	TIPO DESTINO
byte	*double, float, long, int, char, short*
short	*double, float, long, int*
char	*double, float, long, int*
int	*double, float, long*
long	*double, float*
float	*Double*

1.11 Estructuras de control

En lenguajes de programación, las **estructuras de control** permiten modificar el flujo de ejecución de las instrucciones de un programa.

Con las estructuras de control se puede:

- De acuerdo a una condición, ejecutar un grupo u otro de sentencias (If-Then-Else)
- De acuerdo al valor de una variable, ejecutar un grupo u otro de sentencias (Select-Case)
- Ejecutar un grupo de sentencias **mientras** se cumpla una condición (Do-While)
- Ejecutar un grupo de sentencias **hasta** que se cumpla una condición (Do-Until)
- Ejecutar un grupo de sentencias un número determinado de veces (For-Next)

Todas las estructuras de control tienen un único punto de entrada y un único punto de salida. Las estructuras de control se puede clasificar en : secuenciales, iterativas y de control avanzadas. Esto es una de las cosas que permite que la programación se rija por los principios de la programación estructurada.

Los lenguajes de programación modernos tienen estructuras de control similares. Básicamente lo que varía entre las estructuras de control de los diferentes lenguajes es su sintaxis, cada lenguaje tiene una sintaxis propia para expresar la estructura.

Otros lenguajes ofrecen estructuras diferentes, como por ejemplo los comandos guardados.