# GUÍA DE ESTUDIO PARA CIBERNÉTICA Y COMPUTACIÓN I Y II

Colegio de Ciencias y Humanidades | UNAM

Profesor Gamar Zaid Joseph García Castillo
gamar@cch.unam.mx

# Contenido

# Cibernética y computación I

## Unidad I. La cibernética.

### Definición del concepto de cibernética.

La cibernética se ha definido de distintas maneras a lo largo del tiempo y el espacio. Algunas de las más importantes son:

- Andréi Nikoláyevich Kolmogórov: “La cibernética se ocupa de sistemas de cualquier naturaleza que son capaces de recibir, almacenar, y transformar información con fines de control.”

- Raymond Ruyer: “La ciencia del control por medio de información sean estas naturales, orgánicas o artificiales.”

- Norbert Wiener: “Ciencia que se ocupa de los sistemas de control y de comunicación en las personas y las máquinas, estudiando y aprovechando todos sus aspectos y mecanismos comunes.”

- Diccionario de la Lengua Española: “Estudio de las analogías entre los sistemas de control y comunicación de los seres vivos y los de las maquinas; y en particular, el de las aplicaciones de los mecanismos de regulación biológica a la tecnología.”

Con base en estas definiciones planteamos la siguiente que se adapta al curso de Cibernética y Computación:

"La cibernética “es la ciencia que estudia sistemas naturales, sociales y, en el caso de la materia, de su aplicación tecnológica con la finalidad de establecer relaciones entre sus elementos, proponer un modelo y establecer una metodología para implementar un sistema de cómputo capaz de recibir, almacenar y transformar la información.”

Para contextualizar esta ciencia te presentaremos los orígenes de la computación y su desarrollo. Se define qué es un sistema, los elementos que lo forman, su relación con los modelos y se identifican las similitudes y diferencias entre los sistemas naturales y artificiales.

### Antecedentes de la cibernética.

El origen principal de la cibernética se debe a la integración de estudios matemáticos, físicos, ingenieros, fisiólogos y técnicos para analizar los sistemas de control en las máquinas y los seres vivos.

Las piedras angulares de la cibernética son la teoría de la información, la teoría de los algoritmos y la teoría de los autómatas que estudia los métodos de construcción de los sistemas para el procesamiento de la información.

Cabe señalar que un factor decisivo en el proceso de creación de la nueva ciencia fue el crecimiento impetuoso de la automática electrónica y, especialmente, la aparición de las computadoras de acción rápida. Estas últimas abrieron posibilidades nunca vistas en el procesamiento de la información y en la simulación de los sistemas de dirección.

## Relación y aplicación de la cibernética con otras ciencias.

La participación de la cibernética con otras áreas no es fácil identificarlo, pero aquí presentamos algunos ejemplos que pueden ser de utilidad:

La **Biocibernética** surge como una disciplina auxiliar de la Cibernética. Es el producto de una "simbiosis" entre la Biología y la Cibernética Moderna. Algunos autores la denominan Fisiología Integrativa, término que ya prácticamente está en desuso. En la Biocibernética se puede distinguir dos grandes tendencias de desarrollo: Biocibernética Teórica y la Biocibernética Técnica.

En las **matemáticas administrativas**, se preocupa por crear modelos matemáticos capaces de simular situaciones reales en la empresa. La creación de estos se orienta, hacia la solución de problemas que se presentan en la toma de decisión.

La transdisciplinariedad ha permitido el beneficio de la aplicación del conocimiento científico y tecnológico al fortalecimiento de disciplinas como la **medicina**. En la actualidad la tecnología informática tendrá el poder de intervenir en la reconstrucción de la ciencia médica y en el nuevo paradigma de la cirugía.

La cibernética en el **lenguaje natural** es el estudio de las estructuras que constituyen el lenguaje humano articulado y que hacen que éste cumpla con eficacia la función de la comunicación, imponiéndole cuando es necesario las transformaciones necesarias para su funcionamiento efectivo.

## Obra de distintos autores en trabajos científicos sobre la cibernética.

**Blaise Pascal (1623 - 1662)**

Blaise Pascal fue un físico y matemático brillante; de los más reconocidos de su época.

No tenía un interés marcado por la ciencia, pues se mostraba como alguien escéptico del alcance de ésta sobre nuestro conocimiento del mundo; sin embargo sus trabajos relacionados con problemas muy diversos sobre matemáticas y física incidieron decisivamente en el posterior desarrollo de la tecnología de su tiempo.

Pascal inventó una de las primeras calculadoras mecánicas (1642); el aparato llamado pascalina que permitía realizar las operaciones aritméticas de suma y resta.

La pascalina medía algo menos que una caja de zapatos y era de forma alargada. En su interior existían unas ruedas dentadas conectadas entre sí, formando una cadena de transmisión, de modo que cuando una rueda giraba completamente sobre su eje, hacía avanzar un grado a la siguiente. Las ruedas representaban el sistema decimal de numeración. Cada rueda constaba de diez pasos, para lo cual estaba convenientemente marcada con números del 9 al 0. Mediante una manivela se hacía girar las ruedas dentadas. Para sumar o restar no había más que accionar la manivela en el sentido apropiado, con lo que las ruedas recorrían los pasos necesarios.

**Charles Babbage (1791-1871)**

Matemático británico, que realizó estudios y experimentos para conseguir una máquina capaz de realizar con precisión tablas matemáticas. En 1833 terminó el diseño de su máquina diferencial concebida para construir tablas logarítmicas y de funciones trigonométricas y posteriormente se dedicó al diseño de una máquina analítica que tuviera la capacidad de realizar cualquier secuencia de instrucciones aritméticas.

Aunque no llegó a conseguir su propósito, debido a que el sistema de engranajes para su construcción presentaba problemas de esfuerzo y temperatura en su época, Charles Babbage sentó los principios básicos de las computadoras modernas, como el concepto de programa o instrucciones básicas, que se introducen en la máquina de manera independiente de los datos, el uso de la memoria para retener resultados y una unidad aritmética.

**George Boole (1815- 1864)**

George Boole fue un brillante matemático y filósofo británico de origen humilde, que ya en la adolescencia dominaba las lenguas del latín, griego, alemán, italiano y francés. Lenguas que le permitieron incursionar en la lectura de libros sobre teología cristiana y también sobre el estudio de las ciencias matemáticas.

Gracias sus investigaciones que le permitieron escribir una publicación en el que define los principios del álgebra que lleva su nombre en 1854, Boole es considerado como uno de los pioneros en el campo de las Ciencias de la computación.

El álgebra de Boole establece los fundamentos de la aritmética utilizada en la electrónica y cómputo moderno. Desarrolla un sistema de reglas basadas en procedimientos matemáticos que permiten expresar, manipular y simplificar problemas lógicos y filosóficos cuyos argumentos admiten solamente dos estados: verdadero o falso.

**Herman Hollerith (1860-1929)**

Herman Hollerith fue un estadístico que inventó el primer sistema de tabulación para el tratamiento de información. Revolucionando de esta forma, el manejo de información a gran escala mediante la automatización de procesos. Fue el fundador de la empresa Tabulating Machine Company dedicada a fabricar y vender sus máquinas; empresa que posteriormente al fusionarse con otras dos originaría a la International Business Machines (IBM).

El origen de la máquina tabuladora se remonta a 1879, año en que Hollerith, entra a trabajar en la Oficina de Censos de EEUU. Allí tuvo ocasión de trabajar en la realización del censo de 1880, lo que le permitió comprobar la ineficiencia del método utilizado para la captura de los datos, dado que éste era completamente manual.

Durante su trabajo en el censo, Hollerith conoció a John Shaw Billings, quien en una cena llego a comentarle que el proceso que se realizaba en el censo era tan mecánico que podría llevarse a cabo por maquinas. Este comentario le hizo imaginar y desarrollar su primer diseño que consistía en un sistema de almacenamiento basado en una cinta de papel que después cambiaría por tarjetas perforadas.

La codificación de la información se basó en la lógica de Boole, pues muchas respuestas podían codificarse en forma de la ausencia o presencia de un agujero en la tarjeta que podía ser leída por métodos eléctricos.

La máquina tabuladora, que se componía de un lector de tarjetas, un contador, un clasificador y un aparato tabular.

**Alan Mathison Turing (1912-1954)**

Turing fue un matemático, filósofo, criptógrafo, lógico y teórico de la computación que nació en Paddington, Londres.

Cuando era estudiante de postgrado en la universidad de Princeton en 1937, publicó el artículo “On computable numbers”, en el que definió una maquina teórica de capacidad infinita. Esta máquina que después fue llamada "Máquina de Turing" se operaba basándose en una serie de instrucciones lógicas que eran leídas de una cinta de papel perforada y que posteriormente ejecutaba dichas las operaciones expresadas en un lenguaje formal determinado.

Demostró que dicha máquina era capaz de implementar cualquier problema matemático que pudiera representarse mediante un algoritmo. Este artículo también fijó los límites de las ciencias de la computación al demostrar que existen problemas que ningún tipo de computadora podrá resolver.

Turing incursionó además en muchos otros problemas de la época en que vivió, por ejemplo, cuando trabajó para el gobierno británico su habilidad matemática le permitía realizar una actividad que consistía en descifrar códigos de comunicación secretos utilizados por los nazis en la segunda guerra. También se adentraría en las primeras disertaciones sobre la posibilidad de que las máquinas pudieran llegar a tener pensamiento y trabajar de manera autónoma; lo que hoy se conoce como inteligencia artificial.

**Norbert Wiener (1894-1964)**

Fue un matemático estadounidense que a la edad de 18 años ya contaba con un doctorado realizando estudios basados en lógica matemática. Tenía múltiples intereses de los que se ocuparía viajando a diferentes universidades y realizando investigaciones sobre la teoría de los

espacios vectoriales, estudios sobre series y transformadas de Fourier y sobre teoría de números junto con otros notables investigadores de la época.

En los años cuarenta, trabajaría en los principios de la Cibernética, un término que aportaría en su célebre publicación de nombre "Cibernética o control y comunicación en el animal y la máquina", desarrollada en 1948. La cibernética es una ciencia interdisciplinaria que se dedica a estudiar y comprender la interrelación que existe entre las máquinas y el ser humano.

Por esta razón es conocido como el padre de la cibernética; sin embargo, él siempre otorgo el reconocimiento correspondiente a los colegas que le ayudaron a desarrollar el término. Algunos de estos investigadores son el Dr. Stafford Beer, el matemático John Von Neumann y el investigador fisiólogo mexicano Arturo Rosenblueth.

**Arturo Rosenblueth Stearns (1900 - 1970)**

Científico y médico mexicano nacido en Guerrero, Chihuahua; realizó sus estudios de doctorado en medicina en la universidad de París gracias la ayuda económica que le fue otorgada por su hermano Emilio Rosenblueth, un próspero constructor de aquella época.

A su regreso a México en el año de 1972, se desempeñó como ayudante y poco después como profesor de fisiología de la Escuela Nacional de Medicina de la UNAM.

En el año 1930 obtuvo una beca de la Fundación Guggenheim y se fue a la Universidad de Harvard, al Departamento de Fisiología donde realizó investigaciones colaborando con Walter Cannon, acerca de los problemas de la transmisión química entre los elementos nerviosos.

Rosenblueth publicó una gran cantidad de artículos en diversas revistas científicas, del total de estos trabajos, 21 los realizó con Cannon y Wiener.

A partir principios de los años 40, cuando Estados Unidos de Norteamérica entró de lleno a la guerra, el gobierno intensifico el reclutamiento de científicos y su integración a proyectos bélicos y de seguridad nacional. Dentro de la serie de restricciones impuestas a investigadores extranjeros se exigió a Rosenblueth que obtuviera la nacionalidad estadounidense para que pudiera continuar su labor en la Universidad de Illinois, donde lo habían invitado a colaborar; sin embargo, ante esta exigencia, Rosenblueth se niega totalmente y decide regresar a México aceptando la oferta del Dr. Chávez para hacerse cargo del departamento de fisiología del nuevo Instituto Nacional de Cardiología. Este fue el punto de partida del florecimiento de la investigación científica en México en las ciencias fisiológicas.

Asociado con el Dr. Norbert Wiener y Bigelow escribió el ensayo denominado "Behavior, Purpose and Teology" que sirvió de base para la creación, por parte del Dr. Norbert Wiener, de la nueva ciencia llamada "Cibernética". Por esta razón también a él se le considera un pionero de esta ciencia.

**John Von Neumann (1903 -1957)**

Eminente genialidad del siglo XX cuya curiosidad parecía no tener fin. Von Neumann incursionó en diferentes ramas de la ciencia durante toda su vida, desde la Teoría de conjuntos y mecánica cuántica, pasando por Geometría continua, el diseño de computadoras y la teoría de autómatas hasta la teoría de juegos, la astrofísica y la meteorología.

Estudió en la universidad de Budapest donde obtuvo un doctorado en Matemáticas y posteriormente realizó estudios de Ingeniería Química en la Escuela Técnica Superior de Zurich.

Von Neumann escribió un ensayo que influiría de manera definitiva en la construcción y diseño de las computadoras modernas. El articulo describía la estructura de una computadora; dividiendo el diseño en una unidad de procesamiento, una unidad de control, memoria interna y dispositivos de entrada y de salida. La principal contribución era que dentro de la memoria de la computadora se podían almacenar programas que podrían leerse y posteriormente ejecutarse mediante las instrucciones, sin necesidad de tener que volver a escribirlas.

Von Neumann le dio su nombre a la arquitectura utilizada en casi todas las computadoras actuales, debido precisamente a su publicación del concepto; a pesar de que muchos piensan que este nombramiento ignora la contribución de J. Presper Eckert y John William Mauchly, quienes aportaron muchas ideas al concepto durante la construcción de ENIAC, la primer computadora electrónica de propósito general.

El problema al que más importancia dedico al final de su vida tenía que ver con del concepto de auto reproducción; él partía del siguiente cuestionamiento: "¿Puede ser una máquina artificial capaz de producir una copia de ella misma, que pudiera también, ser capaz de crear más copias?". En sus ponencias para la Universidad de Yale "The Computer and the Brain" afirmaba que las computadoras y los seres humanos son diferentes clases de autómatas.

**Claude Elwood Shannon (1916-2001)**

Realizó aportaciones teóricas, e impulsó la digitalización al emplear la lógica booleana. En 1948 fundó la Teoría de la Información, su trabajo hizo posible definir la información en términos matemáticos y operacionalmente precisos, lo que permitía medir su cantidad en bits.

**Ejercicio 1.**

Investiga el concepto de cibernética y trata de dar una definición con tus propias palabras.

_____
_____
_____
_____
_____

**Ejercicio 2.**

Escribe en la columna de la izquierda el nombre del precursor que corresponda a la descripción que se menciona en la columna derecha.

| | |
|---|---|
| | Conocido como el padre de la cibernética, por su publicación “Cibernética o Control y comunicación en el animal y la maquina”. |
| | Desarrolló un sistema de reglas basadas en procedimientos matemáticos que permiten expresar, manipular y simplificar problemas lógicos cuyos argumentos admiten solamente dos estados: verdadero o falso. |
| | Definió una maquina teórica de capacidad infinita que se operaba por una serie de instrucciones expresadas en un lenguaje formal determinado, leídas de una cinta de papel que posteriormente ejecutaba dichas operaciones. |
| | Asociado con otros investigadores, escribió el ensayo "Behavior, Purpose and Teology" que sirvió de base para la concepción de la nueva ciencia llamada "Cibernética". |
| | Escribió un ensayo cuya principal contribución era que dentro de la memoria de la computadora se podían almacenar programas que podrían leerse y posteriormente ejecutarse las veces que quisiéramos, sin necesidad de tener que volver a escribir las instrucciones. |
| | Fundó la Teoría de la Información, puesto que su trabajo hizo posible definir la información en términos matemáticos y operacionalmente precisos, lo que permitía medir su cantidad en bits. |

## Sistemas.

De acuerdo con O´Brien un sistema es “un grupo de componentes interrelacionados que trabajan juntos hacia un fin común, aceptando inputs y produciendo outputs en un proceso de transformación organizado.” [O’Br 93]. Partiendo de esta definición podemos decir que un sistema puede describirse como un conjunto de elementos (objetos, entidades o conceptos), interrelacionados de algún modo a fin de lograr un objetivo común.

**Elementos de un sistema**

Un sistema de cualquier tipo que este sea consta de por lo menos los siguientes elementos:

- **Entradas**: Recursos del medio que funcionan como insumos para el proceso.
- **Proceso**: Transforma o procesa una entrada en salida; es una actividad o fenómeno que modifica las entradas para producir un resultado.
- **Salida**(s): Son los resultados que se obtienen de procesar las entradas.

**Ambiente de un sistema**

El ambiente de un sistema se define como el conjunto de elementos que sin formar parte del sistema poseen propiedades relevantes que tiene un efecto sobre el sistema. Es decir está formado por todas las variables que pueden afectar el estado del sistema desde el exterior.

**Clasificación de los sistemas**

**Cerrado**. Aquel que no tiene ambiente ni contexto. No presentan intercambio con el medio ambiente que los rodea, son herméticos a cualquier influencia ambiental. No reciben ningún recurso externo y nada producen que sea enviado hacia fuera. También se aplica el término a los sistemas completamente estructurados, donde los elementos y relaciones se combinan de una manera peculiar y rígida produciendo una salida invariable, como las máquinas. En rigor, no existen sistemas cerrados. Se da el nombre de sistema cerrado a aquellos sistemas cuyo comportamiento es determinístico y programado y que opera con muy pequeño intercambio de energía y materia con el ambiente.

**Abierto**. Aquel que tiene medio ambiente o si interactúa con su medio ambiente. Los sistemas abiertos presentan intercambio con el ambiente, a través de entradas y salidas. Intercambian energía y materia con el ambiente. Son adaptativos para sobrevivir. Su estructura es óptima cuando el conjunto de elementos del sistema se organiza, aproximándose a una operación adaptativa.

**Natural**. Nacen en respuesta a fenómenos físicos, químicos y biológicos y que se crean por la naturaleza.

**Artificial**. Son aquellos que fueron logrados por la intervención directa de la actividad humana. Un ser humano participó de manera activa en su diseño, manejo, control y ejecución.

## Sistemas de control.

Un sistema de control es un arreglo de componentes físicos conectados de tal manera que el arreglo se puede comandar, dirigir o regular a sí mismo o a otro sistema.

La cibernética estudia la teoría de los sistemas de control basada en la comunicación (transferencia de información) entre sistemas y medio ambiente o internamente en el sistema, y en el control (retroalimentación) del funcionamiento del sistema.

**Sistema de control de lazo cerrado**

Son aquellos en los que la señal de salida tiene efecto directo sobre la acción de control, esto es, los sistemas de control de lazo cerrado son sistemas de control retroalimentados. A diferencia del control de lazo abierto, en el de lazo cerrado sí se mide la salida del proceso para verificar si está dentro del valor deseado al compararlo con éste. Un ejemplo lo constituye el control de un sistema térmico.

**Sistema de control de lazo abierto**

Son sistemas de control en los que la salida o resultado del proceso no tiene ningún efecto sobre la acción de control, es decir, en un sistema de control de lazo abierto la salida no se mide (no se retroalimenta) para comparar con lo que deseamos obtener y así verificar qué tanto nos estamos desviando de ello.

**Ejercicio 3.**

Escribe en la columna de la derecha el nombre del tipo de sistema de control al que se refiere cada diagrama.

| Diagrama | Tipo de sistema de control |
|---|---|
|  | |
|  | |

## Modelos

Los modelos nos sirven para simplificar estructuras y procesos complejos, de manera que podamos representarlas, estudiarlas y comprenderlas. Un modelo se basa en las propiedades más importantes y básicas de lo que queremos representar. Tomando esto en cuenta entonces podemos definir a un modelo como:

"Una simplificación que imita los fenómenos del mundo real, de modo que se puedan comprender las situaciones complejas y podamos hacer predicciones. Los modelos son muy útiles para describir, explicar o comprender mejor la realidad, cuando es imposible trabajar directamente en la realidad en sí."

## Elementos para modelar un sistema.

Un buen modelo permite predecir situaciones futuras porque como imita la realidad da la posibilidad de adelantarse al presente y situarse en lo que vendrá.

Otra ventaja de los modelos es que permiten hacer "experimentos" que nunca serían posibles en la realidad. Por ejemplo, si se dispone de un buen modelo del funcionamiento de la atmósfera se podrá predecir qué pasaría si se aumenta la concentración de un gas, por ejemplo del $CO_2$, y ver cómo variará la temperatura.

La limitación obvia es que un modelo imita, pero no es, la realidad. Por muy bueno que sea siempre está lejos de la complejidad del proceso natural. Así se ha comprobado que la complejidad y la aleatoriedad de los procesos climáticos es tan grande que, a pesar de haberse empleado las mayores supercomputadoras y los más sofisticados modelos computacionales, no se ha logrado predecir el tiempo con fiabilidad para periodos mayores de 24 horas.

**Ejercicio 4.**

Escribe en el espacio la frase que complete correctamente cada enunciado.

1. La "entrada" el "proceso" y la salida son las partes que constituyen un _____
2. Un sistema _____ interactúa con su medio ambiente.
3. Un sistema _____ no recibe entradas ni recibe salidas.
4. La _____ es una característica de los sistemas de control de lazo cerrado.
5. Un _____ es un arreglo de componentes físicos conectados de tal manera que el arreglo se puede dirigir o regular a sí mismo o a otro sistema.
6. 6. Un reloj es un ejemplo de un sistema _____
7. 7. La computadora, el fax, impresora son ejemplos de sistemas _____

# Unidad II. Circuitos lógicos.

## Sistemas de numeración.

Un sistema de numeración puede definirse como un conjunto de signos, relaciones, convenios y normas destinados a expresar de modo gráfico y verbal el valor de los números y las cantidades numéricas.

En la actualidad, se usan predominantemente sistemas de numeración de carácter posicional, donde cada numeral o guarismo representa un valor distinto según la posición que ocupa en la cadena numérica (por ejemplo, el numeral 1 significa unidad en la cantidad 1, pero es decena en 13, centena en 148, etcétera).

En un sistema de numeración se contemplan varios elementos fundamentales:

- La base del sistema, que se define como un convenio de agrupación de sus unidades. Por ejemplo, la base 10 o decimal agrupa diez unidades, mientras que la binaria únicamente agrupa dos.
- Los numerales del sistema, o cifras elementales que se utilizan, según la base. En el sistema decimal, se usan los numerales 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9. En cambio, en el sistema binario tan sólo se emplean el 0 y el 1.
- Las normas de combinación de los numerales para formar los números. Según ello, a cada cifra se le asocian dos propiedades: su valor absoluto intrínseco y su valor posicional o relativo, que depende de la posición que ocupa en la cantidad numérica.

## Aritmética del sistema de numeración binario.

**Conversiones entre sistemas de numeración**

- Binario a decimal

Un numero decimal lo representamos como la suma de los valores multiplicados por la potencia de 10 a la posición-1 que ocupa. Si aplicamos esto a un número binario cualquiera, con su correspondiente base, tendremos lo siguiente:

| 1 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|
| $1 \cdot 2^5 = 32$ | $1 \cdot 2^4 = 0$ | $1 \cdot 2^3 = 0$ | $1 \cdot 2^2 = 4$ | $1 \cdot 2^1 = 2$ | $1 \cdot 2^0 = 0$ |

El valor en decimal de 100110 es 38.

- Decimal a binario

Ahora lo que tendremos que hacer es dividir la cifra decimal entre la base del sistema al que lo queremos convertir, en este caso el 2. Realizaremos este procedimiento hasta que ya no sea posible efectuar ninguna división más.

| Número | 38 | 19 | 9 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|
| División | ÷2=19 | ÷2=9 | ÷2=4 | ÷2=2 | ÷2=1 | – |
| Resto | 0 | 1 | 1 | 0 | 0 | 1 |

Así tenemos el siguiente resultado: 100110

- Binario a octal

La conversión entre ambos sistemas de numeración es muy sencilla debido a que la base del sistema octal es la misma que en el sistema binario pero elevado a la potencia de 3, $23=8$. Entonces, en base a esto, lo que vamos a hacer es agrupar los términos binarios en grupos de tres empezando desde la derecha hacia a izquierda y hacer directamente la conversión a un número decimal. Veamos el ejemplo con el número 100110:

| 1 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|
| 100 | | | 110 | | |
| $0\cdot2^2=4$ | $0\cdot2^1=0$ | $1\cdot2^0=0$ | $1\cdot2^2=4$ | $1\cdot2^1=2$ | $0\cdot2^0=0$ |
| 4 | | | 6 | | |

Agrupamos cada tres cifras y hacemos la conversión a decimal. El resultado final será que $100110=46$

- Octal a binario

Pues el procedimiento es tan simple como hacer lo contrario, es decir pasar de binario a decimal en grupos de 3. Veámoslo con el número 115:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Valor | 1 | | | 1 | | | 5 | | |
| División | ÷2=0 | 0 | 0 | ÷2=0 | 0 | 0 | ÷2=2 | ÷2=1 | – |
| Resto | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| Grupo | 001 | | | 001 | | | 101 | | |

- Decimal a octal

Siguiendo el procedimiento del método decimal-binario vamos a llevarlo a cabo con el ejemplo de 238.32:

Parte entera. Dividimos por la base, que es 8:

| | | | |
|---|---|---|---|
| Numero | 238 | 29 | 3 |
| División | ÷8=29 | ÷8=3 | – |
| Resto | 6 | 5 | 3 |

Parte decimal, multiplicamos por la base, que es 8:

| | | | | | | |
|---|---|---|---|---|---|---|
| Numero | 0,32 | 0,56 | 0,48 | 0,84 | 0,72 | ... |
| Multiplicación | *8=2,56 | *8=4,48 | *8=3,84 | *8=6,72 | *8=5,76 | ... |
| Parte entera | 2 | 4 | 3 | 6 | 5 | ... |

El resultado obtenido es el siguiente: 238.32 = 356.24365

- Octal a decimal

Pues bien, hagamos entonces el proceso contrario. Vamos a pasar el número octal 356.243 a decimal:

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 5 | 6 | , | 2 | 4 | 3 |
| $3 \cdot 8^2 = 192$ | $5 \cdot 8^1 = 40$ | $6 \cdot 2^0 = 6$ | | $2 \cdot 8^{-1} = 0{,}25$ | $4 \cdot 8^{-2} = 0{,}0625$ | $3 \cdot 8^{-3} = 0{,}005893$ |

El resultado es: 192 + 40 + 6 , 0,25 + 0,0625 + 0,005893 = 238.318

- Decimal a hexadecimal

Siguiendo el procedimiento del método decimal-binario y decimal-octal vamos a llevarlo a cabo con el ejemplo de 238.32:

Parte entera. Dividimos por la base, que es 16:

| | | |
|---|---|---|
| Numero | 238 | 14 |
| División | ÷16= 14 | – |
| Resto | E | E |

Parte decimal, multiplicamos por la base, que es 16:

| | | | | | | |
|---|---|---|---|---|---|---|
| Numero | 0,32 | 0,12 | 0,92 | 0,72 | 0,52 | ... |
| Multiplicación | *16=5,12 | *16=1,92 | *16=14,72 | *16=11,52 | *16=8,32 | ... |
| Parte entera | 5 | 1 | E | B | 8 | ... |

- Hexadecimal a decimal

Pues bien, hagamos entonces el proceso contrario. Vamos a pasar el número hexadecimal EE,51E a decimal:

| | | | | | |
|---|---|---|---|---|---|
| E | E | , | 5 | 1 | E |
| $E \cdot 16^{1}=224$ | $E \cdot 16^{0}=14$ | | $5 \cdot 16^{-1}=0{,}3125$ | $1 \cdot 16^{-2}=0{,}003906$ | $E \cdot 16^{-3}=0{,}00341$ |

El resultado es: 224 + 14 , 0.3125 + 0.003906 + 0.00341 = 238.3198

**Ejercicio 5.**

Realiza las siguientes conversiones entre sistemas de numeración:

| DECIMAL | BINARIO | OCTAL | HEXADECIMAL |
|---|---|---|---|
| 0 | | | |
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |
| 10 | | | |
| 11 | | | |
| 12 | | | |
| 13 | | | |
| 14 | | | |
| 15 | | | |
| 16 | | | |

**Operaciones con diferentes sistemas de numeración**

- **Suma en binarios**

La tabla de sumar, en binario, es mucho más sencilla que en decimal. Sólo hay que recordar cuatro combinaciones posibles:

| + | **0** | **1** |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 | 0 + 1 |

Las sumas 0 + 0, 0 + 1 y 1 + 0 son evidentes:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

Pero la suma de 1+1, que sabemos que es 2 en el sistema decimal, debe escribirse en binario con dos cifras (10) y, por tanto 1+1 es 0 y se arrastra una unidad, que se suma a la posición siguiente a la izquierda. Veamos algunos ejemplos:

**$010 + 101 = 111 = 2_{10} + 5_{10} = 7_{10}$**

**$001101 + 100101 = 110010 = 13_{10} + 37_{10} = 50_{10}$**

**$1011011 + 1011010 = 10110101 = 91_{10} + 90_{10} = 181_{10}$**

**$110111011 + 100111011 = 1011110110 = 443_{10} + 315_{10} = 758_{10}$**

- **Resta en binarios**

La técnica de la resta en binario es, nuevamente, igual que la misma operación en el sistema decimal. Pero conviene repasar la operación de restar en decimal para comprender la operación binaria, que es más sencilla. Los términos que intervienen en la resta se llaman minuendo, sustraendo y diferencia.

| - | 0 | 1 |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 + 1 | 0 |

La resta 0 - 1 se resuelve, igual que en el sistema decimal, tomando una unidad prestada de la posición siguiente: 10 - 1, es decir, $2_{10} - 1_{10} = 1$. Esa unidad prestada debe devolverse, sumándola, a la posición siguiente. Veamos algunos ejemplos:

**$111 - 101 = 010 = 7_{10} - 5_{10} = 2_{10}$**

**$10001 - 01010 = 00111 = 17_{10} - 10_{10} = 7_{10}$**

**$11011001 - 10101011 = 00101110 = 217_{10} - 171_{10} = 46_{10}$**

**$111101001 - 101101101 = 001111100 = 489_{10} - 365_{10} = 124_{10}$**

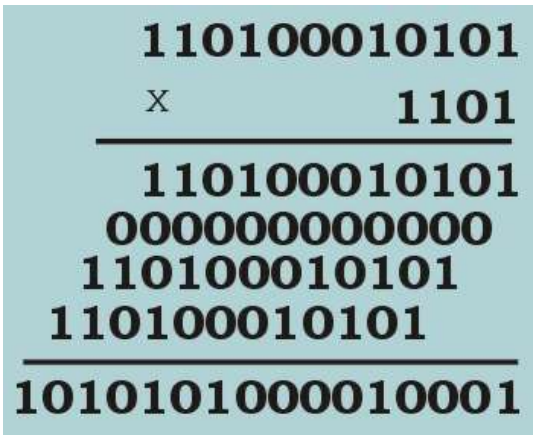
- **Multiplicación en binarios**

La multiplicación en binario es más fácil que en cualquier otro sistema de numeración. Como los factores de la multiplicación sólo pueden ser CEROS o UNOS, el producto sólo puede ser CERO o UNO. En otras palabras, las tablas de multiplicar del cero y del uno son muy fáciles de aprender:

| x | 0 | 1 |
|---|---|---|
| **0** | 0 | 0 |
| **1** | 0 | 1 |

La operación de multiplicar se realiza mediante sumas repetidas. Eso crea algunos problemas en la programación porque cada suma de dos UNOS origina un arrastre, que se resuelven contando el número de UNOS y de arrastres en cada columna. Si el número de UNOS es par, la suma es un CERO y si es impar, un UNO. Luego, para determinar los arrastres a la posición superior, se cuentan las parejas de UNOS.

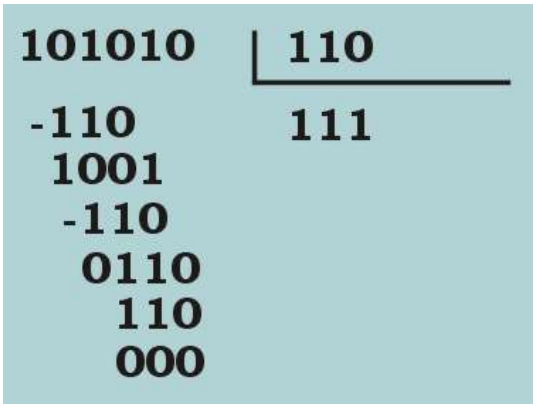
Veamos, por ejemplo, una multiplicación:




- **División en binarios**

Igual que en el producto, la división es muy fácil de realizar, porque no son posibles en el cociente otras cifras que UNOS y CEROS.

Consideremos el siguiente ejemplo, 42 : 6 = 7, en binario:




Se intenta dividir el dividendo por el divisor, empezando por tomar en ambos el mismo número de cifras (100 entre 110, en el ejemplo). Si no puede dividirse, se intenta la división tomando un dígito más (1001 entre 100).

Si la división es posible, entonces, el divisor sólo podrá estar contenido una vez en el dividendo, es decir, la primera cifra del cociente es un UNO. En ese caso, el resultado de multiplicar el divisor por 1 es el propio divisor. Restamos las cifras del dividendo del divisor y bajamos la cifra siguiente.

**Ejercicio 6.**

Realiza las siguientes operaciones binarias:

1) 111011 + 110
2) 111110111 + 111001

3) 111011 - 110
4) 111110111 – 111001

5) 10110101000101 x 1011
6) 10100001111011 x 10011

7) 10110101000101 : 1011
8) 10100001111011 : 10011

## Elementos del álgebra de Boole.

El álgebra booleana es una estructura algebraica que esquematiza las operaciones lógicas Y, O, NO (AND, OR, NOT), así como el conjunto de operaciones unión, intersección y complemento.

El álgebra booleana es un sistema matemático deductivo centrado en los valores cero y uno (falso y verdadero). Un operador binario " º " definido en este juego de valores acepta un par de entradas y produce un solo valor booleano, por ejemplo, el operador booleano AND acepta dos entradas booleanas y produce una sola salida booleana. Para cualquier sistema algebraico existen una serie de postulados iniciales, de aquí se pueden deducir reglas adicionales, teoremas y otras propiedades del sistema. Es posible probar todos los teoremas del álgebra booleana utilizando estos postulados, además es buena idea familiarizarse con algunos de los teoremas más importantes de los cuales podemos mencionar los siguientes:

- Teorema 1: A + A = A
- Teorema 2: A · A = A
- Teorema 3: A + 0 = A
- Teorema 4: A · 1 = A
- Teorema 5: A · 0 = 0
- Teorema 6: A + 1 = 1
- Teorema 7: (A + B)' = A' · B'
- Teorema 8: (A · B)' = A' + B'
- Teorema 9: A + A · B = A
- Teorema 10: A · (A + B) = A
- Teorema 11: A + A'B = A + B
- Teorema 12: A' · (A + B') = A'B'
- Teorema 13: AB + AB' = A
- Teorema 14: (A' + B') · (A' + B) = A'
- Teorema 15: A + A' = 1
- Teorema 16: A · A' = 0

Los teoremas siete y ocho son conocidos como **Teoremas de DeMorgan** en honor al matemático que los descubrió.

### Función booleana.

Una función booleana es una de A x A x A x....A en A, siendo A un conjunto cuyos elementos son 0 y 1 y tiene estructura de álgebra de Boole. Supongamos que cuatro amigos deciden ir al cine si lo quiere la mayoría. Cada uno puede votar sí o no. Representemos el voto de cada uno por xi. La función devolverá sí (1) cuando el número de votos afirmativos sea 3 y en caso contrario devolverá 0. Si x1 vota 1, x2 vota 0, x3 vota 0 y x4 vota 1 la función booleana devolverá 0. Producto mínimo (es el número posible de casos) es un producto en el que aparecen todas las variables o sus negaciones. El número posible de casos es 2n.

Las funciones booleanas se pueden representar como la suma de productos mínimos (minterms) iguales a 1.

## Compuertas lógicas y tablas de verdad.

**Compuerta AND**

Esta compuerta es representada por una multiplicación en el Algebra de Boole. Indica que es necesario que en todas sus entradas se tenga un estado binario 1 para que la salida otorgue un 1 binario. En caso contrario de que falte alguna de sus entradas con este estado o no tenga si quiera una accionada, la salida no podrá cambiar de estado y permanecerá en 0. Esta puede ser simbolizada por dos o más interruptores en serie de los cuales todos deben estar activos para que esta permita el flujo de la corriente.

| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$$Q = A * B$$

**Compuerta OR**

En el Algebra de Boole esta es una suma. Esta compuerta permite que con cualquiera de sus entradas que este en estado binario 1, su salida pasara a un estado 1 también. No es necesario que todas sus entradas estén accionadas para conseguir un estado 1 a la salida pero tampoco causa algún inconveniente. Para lograr un estado 0 a la salida, todas sus entradas deben estar en el mismo valor de 0. Se puede interpretar como dos interruptores en paralelo, que sin importar cual se accione, será posible el paso de la corriente.

| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

$$Q = A + B$$

**Compuerta NOT**

En este caso esta compuerta solo tiene una entrada y una salida y esta actúa como un inversor. Para esta situación en la entrada se colocará un 1 y en la salida otorgara un 0 y en el caso contrario esta recibirá un 0 y mostrara un 1. Por lo cual todo lo que llegue a su entrada, será inverso en su salida.

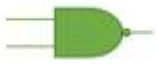
| Q | Q' |
|---|---|
| 0 | 1 |
| 1 | 0 |

$$Q = \bar{Q}$$

**Compuerta NAND**

También denominada como AND negada, esta compuerta trabaja al contrario de una AND ya que al no tener entradas en 1 o solamente alguna de ellas, esta concede un 1 en su salida, pero si esta tiene todas sus entradas en 1 la salida se presenta con un 0.

| A | B | Q |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



$$Q = \overline{A * B}$$

**Compuerta NOR**

Así como vimos anteriormente, la compuerta OR también tiene su versión inversa. Esta compuerta cuando tiene sus entradas en estado 0 su salida estará en 1, pero si alguna de sus entradas pasa a un estado 1 sin importar en qué posición, su salida será un estado 0.

| A | B | Q |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

$$Q = \overline{A + B}$$

**Compuerta XOR**

También llamada OR exclusiva, esta actúa como una suma binaria de un digito cada uno y el resultado de la suma seria la salida. Otra manera de verlo es que con valores de entrada igual el estado de salida es 0 y con valores de entrada diferente, la salida será 1.

| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$$Q = A * \bar{B} + \bar{A} * B$$

**Compuerta XNOR**

Esta es todo lo contrario a la compuerta XOR, ya que cuando las entradas sean iguales se presentará una salida en estado 1 y si son diferentes la salida será un estado 0.

| A | B | Q |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$$Q = A * B + \overline{A * B}$$

**Ejercicio 7.**

Responde las siguientes preguntas:

¿Qué es una variable lógica?

_____

_____

¿Cuáles son los operadores lógicos básicos?

_____

_____

¿Qué es una compuerta lógica?

_____

_____

¿Qué es un circuito lógico?

_____

_____

¿Qué es un circuito integrado?

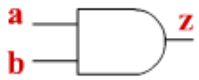
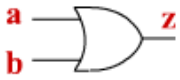
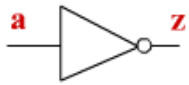
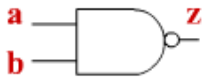
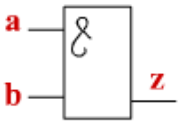
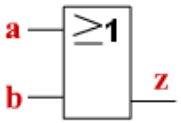
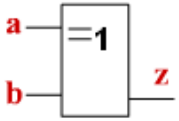
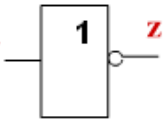
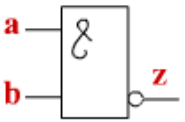
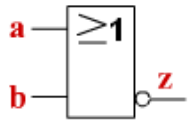
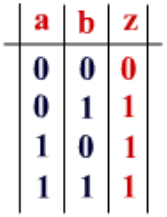
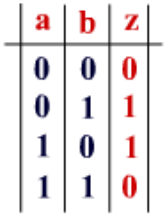
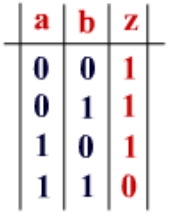
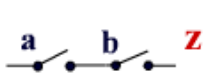
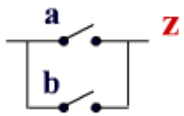
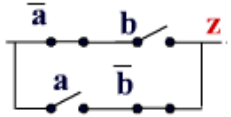
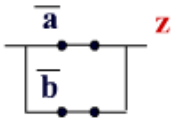
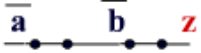
_____

_____

¿Qué es el álgebra booleana?

_____

_____

| NOMRE | AND - Y | OR - O | XOR<br>O-exclusiva | NOT<br>Inversor | NAND | NOR |
|---|---|---|---|---|---|---|
| SÍMBOLO |  |  |  |  |  |  |
| SÍMBOLO |  |  |  |  |  |  |
| TABLA DE VERDAD |  |  |  |  |  |  |
| EQUIVALENTE EN CONTACTOS |  |  |  |  |  |  |
| AXIOMA | $z = a \cdot b$ | $z = a + b$ | $z = \bar{a} \cdot b + a \cdot \bar{b}$ | $z = \bar{a}$ | $z = \overline{a \cdot b}$ | $z = \overline{a + b}$ |

## Simplificación de funciones booleanas.

Algunos ejemplos de simplificación de funciones booleanas son las siguientes:

1) $A+AB=A$

Solución:

$A+AB=AI+AB=A(I+B)=A$

2) $A(A+B)=A$

Solución:

Aplicando la ley distributiva y la ley de absorción, tenemos:

$A(A+B)=AA+AB=A+AB=A(I+B)=A$

3) $AB+AB'=A$

Solución:

$AB+AB'=A(B+B')=AI=A$

4) $A(A'+B)=AB$

Solución:

$A(A'+B)=AA'+AB=AB$

**Ejercicio 8.**

Completa las tablas de verdad de las siguientes compuertas lógicas:

| AND | | |
|---|---|---|
| A | B | Z |
| | | |
| | | |
| | | |
| | | |

| OR | | |
|---|---|---|
| A | B | Z |
| | | |
| | | |
| | | |
| | | |

| XOR | | |
|---|---|---|
| A | B | Z |
| | | |
| | | |
| | | |
| | | |

| NAND | | |
|---|---|---|
| A | B | Z |
| | | |
| | | |
| | | |
| | | |

| NOR | | |
|---|---|---|
| A | B | Z |
| | | |
| | | |
| | | |
| | | |

| NOT | |
|---|---|
| A | Z |
| | |
| | |

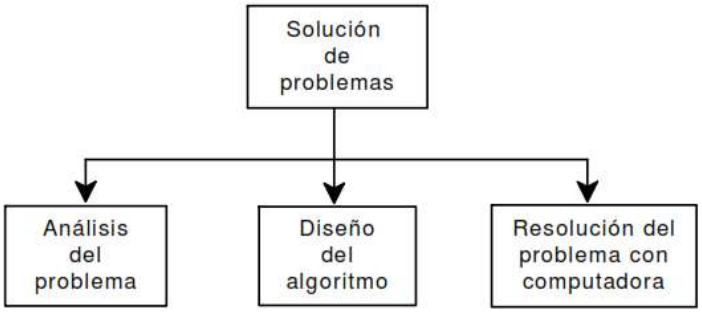
# Unidad III. Metodología de solución de problemas e introducción al lenguaje de programación Java.

## Definiciones y conceptos generales de un problema.

La solución de problemas consiste en un proceso que abarca diversas actividades estas pueden ser cortas o extensas, la solución de problemas es un factor determinante dentro de empresas y organizaciones, que principalmente es un riesgo tomado por los administradores. De esta necesidad surge el conocer los pasos principales para la identificación y la buena aplicación de técnicas, métodos y modelos que ayuden a la solución de problemas.

## Etapas de la metodología de solución de problemas.

Para resolver problemas con la computadora y las herramientas de programación, es conveniente que te apoyes de la metodología de la solución de problemas que se propone en seguida:




**Análisis del problema**. Se debe comprender el problema definirlo, estructurarlo y analizar las fallas si el problema resulta muy complejo es recomendable dividirlo en segmentos y realizar una breve descripción de cada una de las partes.

Se debe verificar la comprensión del problema esto resulta ser más fácil si se comparte con otras personas para asegurar de que la comprensión sea la adecuada.

**Algoritmo y características**. Es un conjunto de pasos lógicos que permiten la solución de problemas. El algoritmo es preciso (indica el orden de realización en cada paso), definido (al seguir el algoritmo varias veces se obtiene el mismo resultado) y finito (tiene un fin, número determinado de pasos).

**Diseño del algoritmo**. Es el proceso que convierte los resultados del análisis en un diseño modular (descendente) con refinamiento sucesivo (divide al problema en cada etapa y expresa a cada paso en forma más detallada), que permitan una posterior traducción a un lenguaje.

**Diagrama de flujo**. En esta herramienta se utilizan figuras geométricas que tiene su significado en el diagrama, con el fin de solucionar problemas o mejorar algún sistema o proceso.

**Pseudocódigo**. Es un lenguaje para la especificación de algoritmos, cuyas instrucciones se representan mediante palabras en el lenguaje natural, el español o inglés, para facilitar tanto la lectura como escritura de programas.

**Prueba de escritorio**. La prueba de escritorio no es más que efectuar un proceso de simulación con el algoritmo desarrollado (ver que haría la computadora). Este trabajo se realiza en base a una tabla cuyos encabezados son las variables que se usan en el algoritmo y debajo de cada una de ellas se van colocando los valores que van tomando, paso a paso y siguiendo el flujo indicado por el algoritmo, hasta llegar al final.

**Codificación**. Es el proceso que consiste en la traducción de las instrucciones del diagrama de flujo o pseudocódigo al lenguaje de programación.

**Compilación**. El proceso de compilación acepta como entrada el programa fuente y checa la sintaxis de las instrucciones de este, de no existir errores de sintaxis, se genera el programa objeto escrito en el lenguaje de máquina, listo para su ejecución, en caso contrario dicho programa no es generado hasta que quede exento de errores.

**Ejecución**. El proceso de ejecución acepta como entrada el programa objeto que se generó en el proceso de compilación y realiza la ejecución de este.

**Comprobación**. Proceso que consiste en verificar que los resultados obtenidos en la corrida sean correctos.

**Documentación**. Proceso que consiste en la descripción de los pasos a dar en la solución del problema. La documentación puede ser interna y externa. La interna es mediante cometarios al programa para clarificarlos y la externa consiste en el manual de usuario.

## Concepto de algoritmo, diagrama de flujo y pseudocódigo.

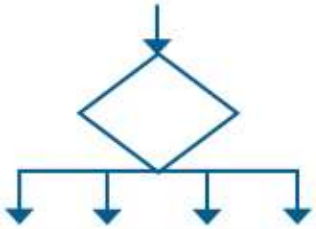
**Algoritmo**

Para implementar la solución de un problema mediante el uso de una computadora es necesario establecer una serie de pasos que permitan resolver el problema, a este conjunto de pasos se le denomina algoritmo, el cual debe tener como característica final la posibilidad de transcribirlo fácilmente a un lenguaje de programación, para esto se utilizan herramientas de programación, las cuales son métodos que permiten la elaboración de algoritmos escritos en un lenguaje entendible.

**Diagramas de flujo**

Los diagramas de flujo son una herramienta que permite representar visualmente qué operaciones se requieren y en qué secuencia se deben efectuar para solucionar un problema dado. Por consiguiente, un diagrama de flujo es la representación gráfica mediante símbolos especiales, de los pasos o procedimientos de manera secuencial y lógica que se deben realizar para solucionar un problema dado.

Los diagramas de flujo facilitan la comunicación entre los programadores y los usuarios, además de que permiten de una manera más rápida detectar los posibles errores de lógica que se presenten al implementar el algoritmo.

| Símbolo | Significado |
|---|---|
|  | Terminal /Inicio. |
|  | Entrada de datos. |
|  | Proceso. |
|  | Decisión. |
|  | Decisión múltiple. |
|  | Imprimir resultados. |
|  | Flujo de datos. |
|  | Conectores. |

**Operadores**

| Símbolo | Operación |
|---|---|
| + | Suma |
| - | Resta |
| * | Multiplicación |
| / | División |
| ^ | Exponenciación |
| > | Mayor que |
| < | Menor que |
| >= | Mayor o igual que |
| <= | Menor o igual que |
| <> | Diferente que |
| = | Igual que |

## Elaboración de algoritmos secuenciales.

En este tipo de estructura las instrucciones se realizan o se ejecutan una después de la otra y, por lo general, se espera que se proporcione uno o varios datos, los cuales son asignados a variables para que con ellos se produzcan los resultados que representen la solución del problema que se planteó. Los algoritmos tienen como fin actuar sobre los datos proporcionados por el usuario, a los que se les aplican procesos con el fin de generar la información o un resultado.

## Algoritmos secuenciales a través de diagramas de flujo y pseudocódigo.

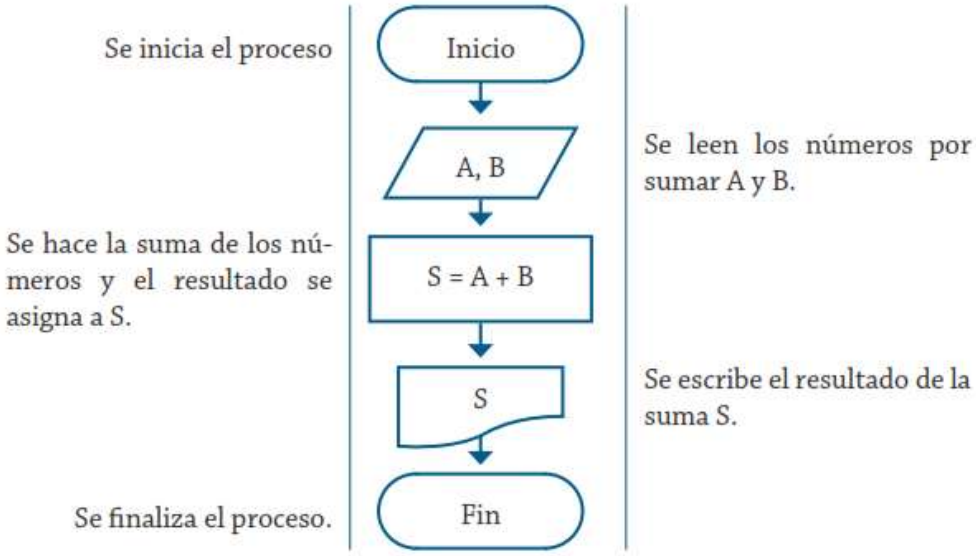
Para resolver un problema mediante la utilización de cualquier herramienta es necesario entender y establecer con qué datos se cuenta, los procesos que se deben realizar y la secuencia apropiada para obtener la solución que se desea.

Se desea implementar un algoritmo para obtener la suma de dos números cualesquiera. Se debe partir de que para poder obtener la suma es necesario contar con dos números, pues el proceso que debemos realizar es precisamente la suma de éstos, la cual se asigna a una variable que se reporta como resultado del proceso.

El pseudocódigo es:

1. Inicio
2. Leer A, B
3. Hacer S = A + B
4. Escribir S
5. Fin

Y el diagrama de flujo secuencial es:




## Lenguaje de programación Java.

**Historia del lenguaje**

- 1991 - se crea una herramienta de programación para el proyecto Green Project en Sun Microsystems.
- Green Team tardó 18 meses en crearlo.
- James Gosling, Arthur Van Hoff, Andy Bechtolsheim .
- Objetivo: implementar una máquina virtual y un lenguaje similar a C++ .
- 1994 - se reorienta hacia la web.
- Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.
- Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere").
- Java es propiedad de Oracle.
- Su sintaxis se deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos.

- Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora.

**El JDK y el JRE**

- **JDK** (Java Development Kit) – Kit de desarrollo de Java

  ✓ Software que provee herramientas de desarrollo para la creación de programas en Java.
  ✓ javac.exe - compilador
  ✓ java.exe – intérprete

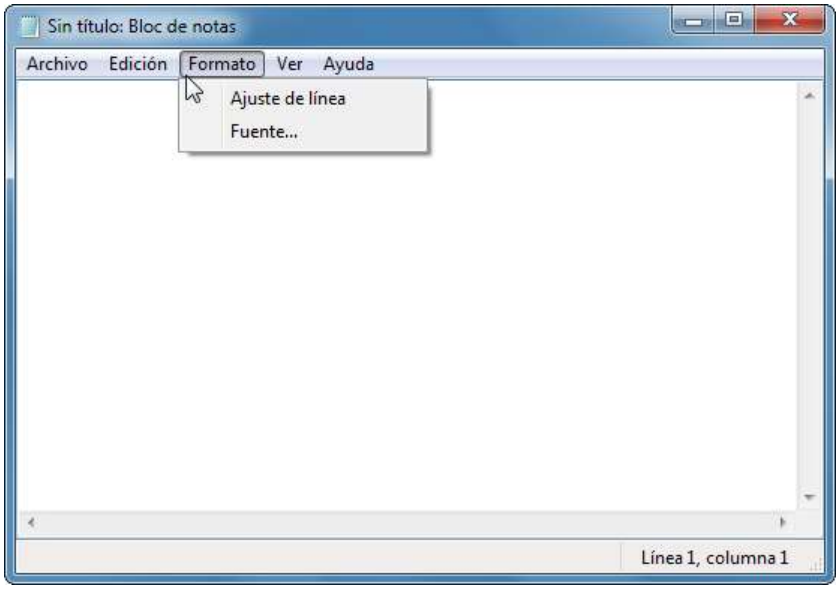
- **JRE** Java Runtime Enviroment – Entorno de ejecución de Java

  ✓ JRE está formado por Java Virtual Machine (Máquina Virtual de Java) y esencialmente sirve para ejecutar programas o aplicaciones creadas en lenguaje Java.
  ✓ Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo.

## Entornos de desarrollo para el lenguaje Java.

Los entornos de desarrollo son programas que facilitan la escritura del código del lenguaje Java para el usuario, ya que tienen predefinidas muchas funciones. Los ideales para usar dentro del Colegio de Ciencias y Humanidades son los siguientes:

- Bloc de notas




- BlueJ

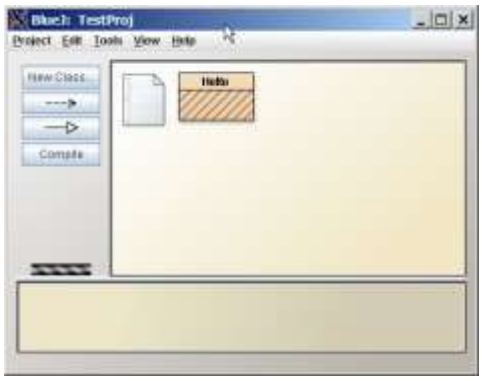
- ✓ BlueJ es un entorno integrado de desarrollo muy sencillo de uso, pensado para aprender a programar en Java.

- ✓ Es recomendable para estudiantes por el diseño de interfaz gráfica.

- ✓ Las principales ventajas de BlueJ son:
- ✓ Es gratuito
- ✓ Es fácil de usar
- ✓ Es ligero (no requiere una máquina muy potente)
- ✓ Puede ser portable.

**Instalación:**

1. Ingresar a http://www.bluej.org/
2. Descargar el ejecutable para Windows
3. Vaya a la sección “download and install”
4. Descargue la última versión oficial
5. Ejecute el archivo que ha descargado y siga las instrucciones de instalación
6. Puede encontrar instrucciones detalladas de instalación en http://www.bluej.org/tutorial/tutorial-spanish-201.pdf

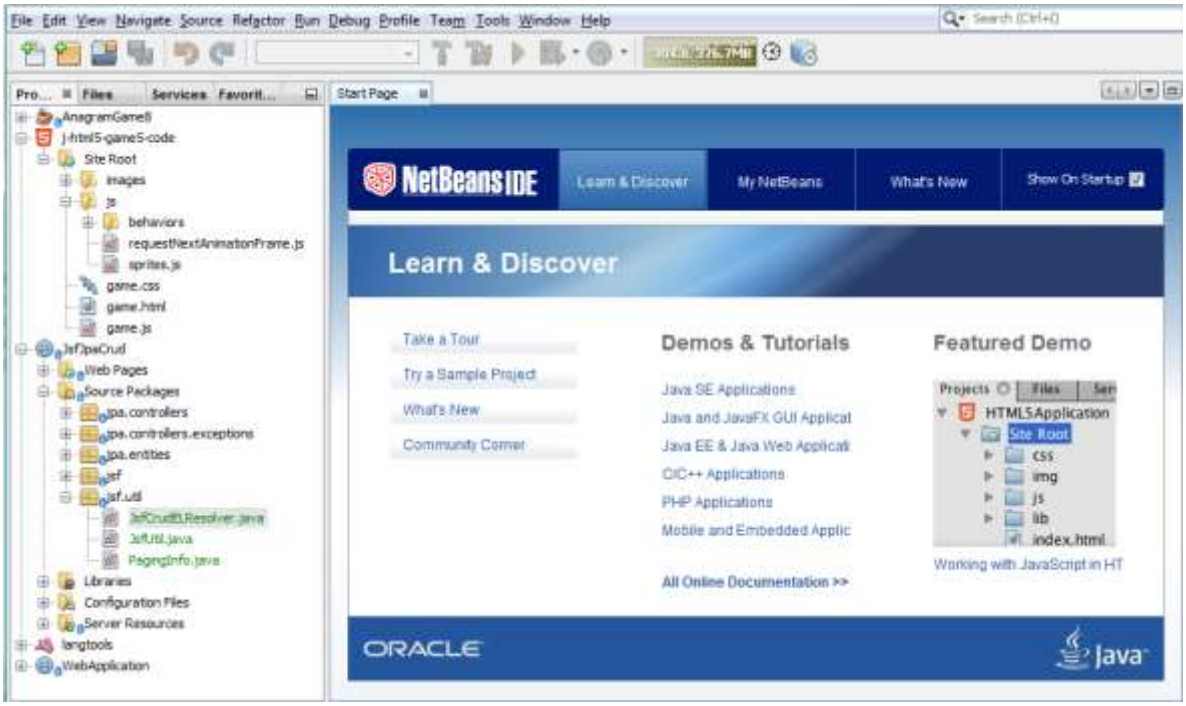



- NetBeans

Es un entorno de desarrollo, es decir, es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. NetBeans es un producto libre y gratuito sin restricciones de uso.

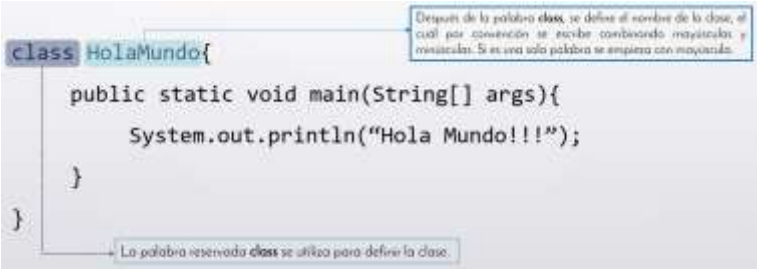
Ingresar a la página: https://netbeans.org/downloads/index.html
Ejecute el archivo descargado y sigas las instrucciones de instalación.

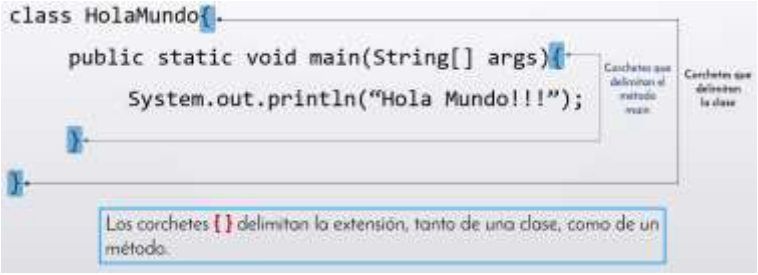

NetBeans IDE 8.2 Download
Email address (optional):
Subscribe to newsletters:
Monthly
Weekly
NetBeans can contact me at this address
IDE Language:
English
Platform:
Windows
NetBeans IDE Download Bundles
Supported technologies *
Java SE
Java EE
HTML5/JavaScript
PHP
C/C++
All
NetBeans Platform SDK
Java SE
Java FX
Java EE
Java ME
HTML5/JavaScript
PHP
C/C++
Groovy
Java Card™ 3 Connected
Bundled servers
GlassFish Server Open Source Edition 4.1.1
Apache Tomcat 8.0.27
Download
Download x86
Download x64
Free, 95 MB
Free, 197 MB
Free, 105 - 112 MB
Free, 105 - 112 MB
Free, 107 - 110 MB
Free, 221 MB


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Start Page
NetBeans IDE
Learn & Discover
My NetBeans
What's New
Show On Startup
Learn & Discover
Take a Tour
Try a Sample Project
What's New
Community Corner
Demos & Tutorials
Java SE Applications
Java and JavaFX GUI Applicat
Java EE & Java Web Applicati
C/C++ Applications
PHP Applications
Mobile and Embedded Applic
All Online Documentation >>
Featured Demo
Projects
Files
HTML5Application
Site Root
css
img
js
lib
index.html
Working with JavaScript in HT
ORACLE
Java

## Pasos para implementar un programa con el lenguaje de programación Java.

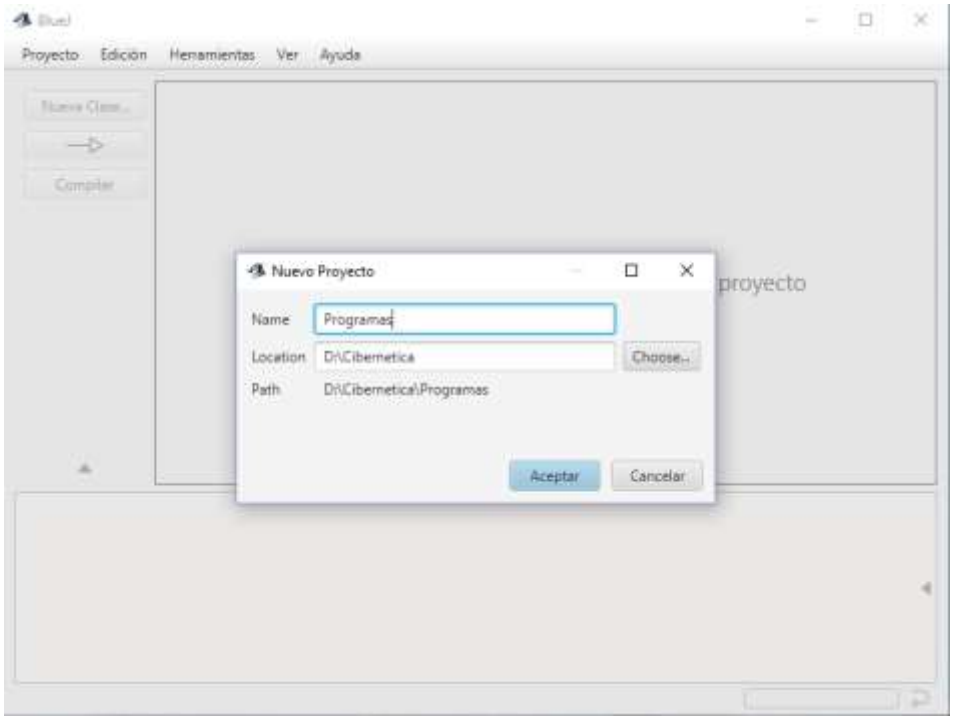
A continuación se muestran los elementos básicos de un programa escrito en Java:



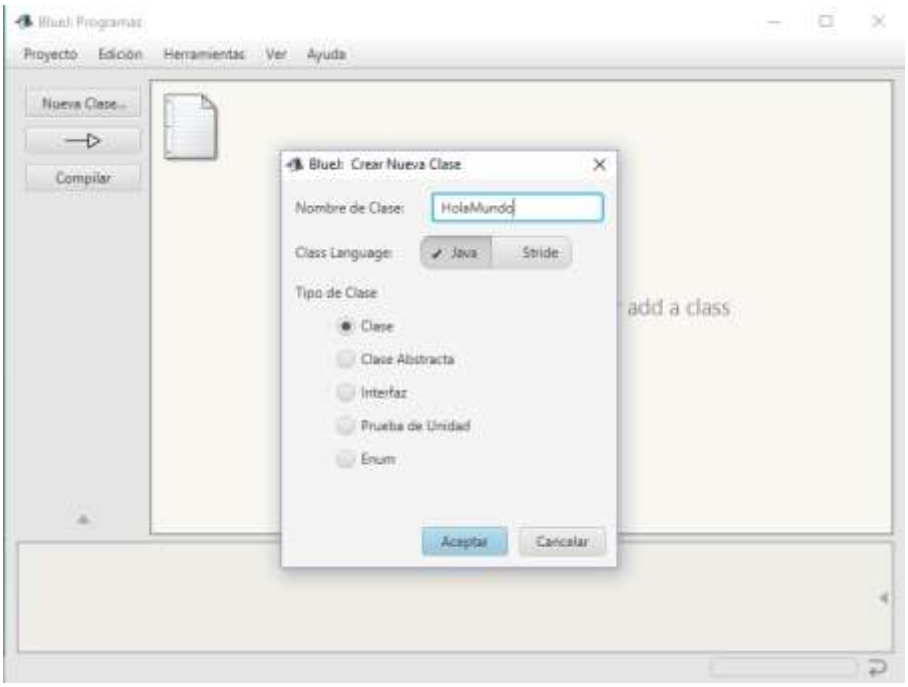




**Ahora los vamos a implementar con BlueJ**

**Paso 1**. Abre BlueJ y en la pestaña Proyecto, selecciona Nuevo Proyecto y debes escribir un nombre para tu archivo y también debes seleccionar la ruta en donde se guardará. Al final da clic en el botón aceptar.

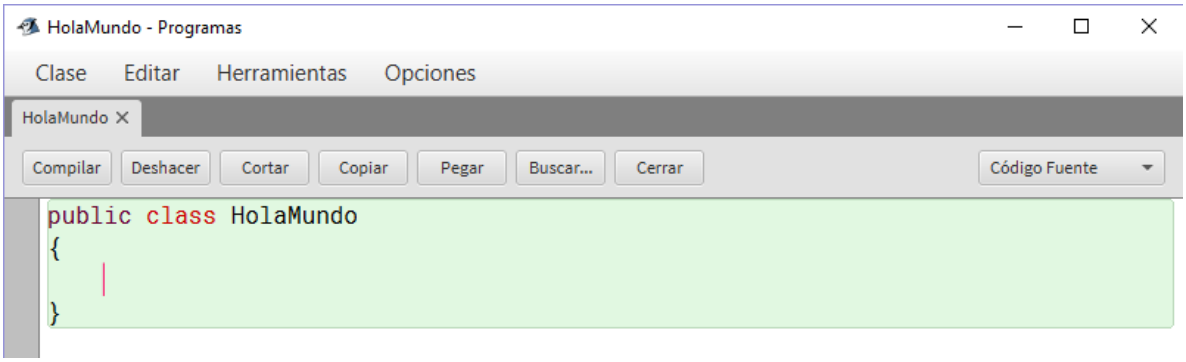


**Paso 2**. Da clic en el botón Nueva Clase… escribe el nombre de la clase, en este caso "HolaMundo" sin espacios y alternando mayúsculas y minúsculas. Al fina da clic en el botón aceptar.

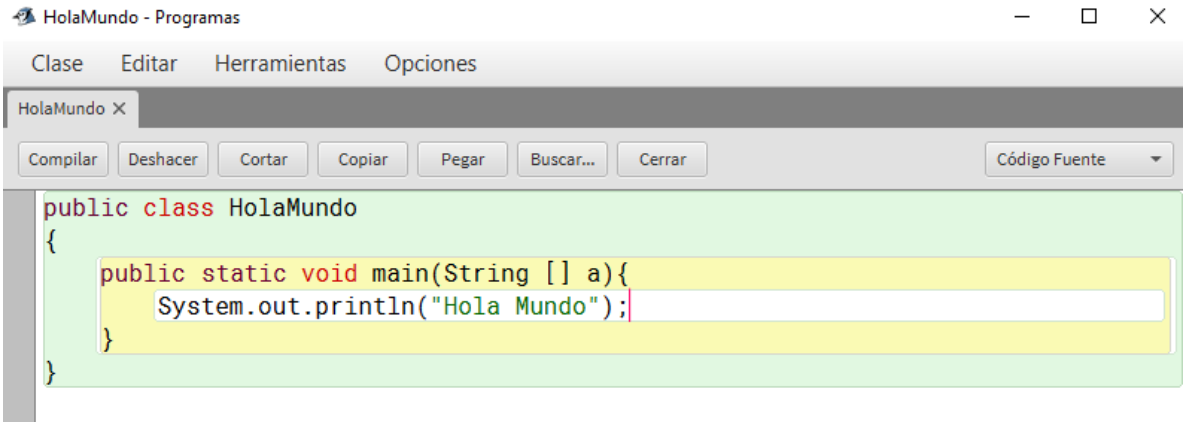



**Paso 3**. Aparecerá un recuadro anaranjado que representa la Clase que acabas de crear. Da doble clic sobre ese recuadro y te llevará al código del programa para editarlo.


**Paso 4**. BlueJ pone mucho código predeterminado pero tú lo puede borrar y sólo dejar lo necesario, observa qué código es el que dejaremos:



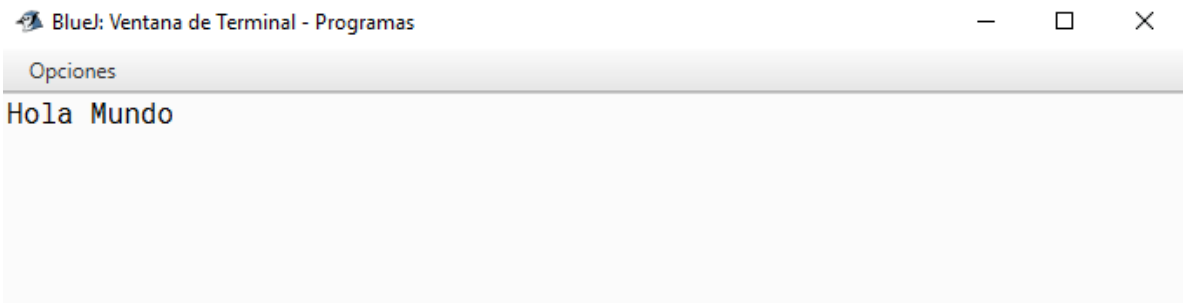
**Paso 5.** Prácticamente eliminamos todo el código y sólo dejamos el nombre de la clase y sus llaves. Dentro de las llaves escribiremos el método main y el mensaje “Hola Mundo” de la siguiente forma:




**Paso 6**. Da clic en el botón compilar, no debe marcar ningún error. Posteriormente da clic en el botón cerrar y te regresará a la pantalla con el recuadro anaranjado:


**Paso 7**. Ahora hay que correr el programa, da clic derecho sobre el recuadro anaranjado y selecciona la opción **void main(String [] a)**. Aparecerá la siguiente ventana, debes dar clic en el botón aceptar:


**Paso 8**. Aparecerá la siguiente pantalla de salida del programa:




## Introducción de datos desde el teclado.

Para introducir datos desde el teclado en un programa en Java, es necesario hacer uso de la clase Scanner.

- La clase Scanner permite leer datos a través del teclado.
- Para hacer uso de esta clase, es necesario importarla desde el paquete: java.util
- Su sintaxis sería la siguiente:
  import java.util.Scanner;

**El Objeto System.out**

- El objeto System.out nos sirve tener un flujo de salida hacia el monitor. Este objeto cuenta con dos métodos:

  System.out.print
  System.out.println

- El primero imprime en la consola el valor del argumento que le pasamos. El segundo hace lo mismo, pero agrega un salto de línea al final.

```
import java.util.Scanner;

class PideDatos{
      public static void main(String []args){
            Scanner teclado = new Scanner(System.in);
            System.out.print("Ingresa tu nombre: ");
            String nombre;
            nombre = teclado.nextLine();
            System.out.println("Hola " + nombre);
      }
}
```

**El método next()**

Los métodos next() sirven para la lectura de datos a través del teclado y puede tener las siguientes formas:

- next() solo lee hasta donde encuentra un espacio (hasta un espacio).
- nextLine() lee todo incluyendo espacios (hasta un enter).
- nextInt() lee un número entero.
- nextDouble() lee un número de tipo double.
- nextFloat() lee un número de tipo float.
- next().charAt(0) lee un carácter.

**Declaración de variables**

Podemos definir variables en cualquier parte del código simplemente indicando el tipo de datos y el nombre de la variable (identificador).

Identificadores válidos son:

- ✓ fecha
- ✓ iFecha
- ✓ fechaNacimiento
- ✓ fecha_nacimiento
- ✓ fecha3
- ✓ _fecha

Identificadores NO válidos son:

- ✓ 3fecha

- ✓ fecha-nacimiento
- ✓ fecha+nacimiento
- ✓ -fecha

**Constantes**

En Java, se utiliza la palabra clave final para indicar que una variable debe comportarse como si fuese constante, significando con esto que no se permite su modificación una vez que haya sido declarada e inicializada.

Por ejemplo:

```
final float PI = 3.14159;
```

**Comentarios**

En Java hay tres tipos de comentarios:

// comentarios para una sola línea

/* comentarios de una o más líneas */

/** comentario de documentación, de una o más líneas*/

**Tipos de datos**

| Tipo | Valor por default | Longitud |
|---|---|---|
| byte | 0 | 8 bits |
| char | \u000 | 16 bits |
| short | 0 | 16 bits |
| int | 0 | 32 bits |
| long | 0 | 64 bits |
| float | 0.0 | 32 bits |
| double | 0.0 | 64 bits |
| boolean | false | 1 bit |
| String | null | |

**Palabras reservadas**

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

**Operadores aritméticos**

| Operador | Operación |
|---|---|
| + | Suma |
| - | Resta |
| * | Multiplicación |
| / | División |
| % | Resto |
| = | Asignación |

**Operadores relacionales**

| Operador | Significado | Ejemplo |
|---|---|---|
| == | Igual a | a == b |
| != | No igual a | a != b |
| > | Mayor que | a > b |
| < | Menor que | a < b |
| >= | Mayor o igual que | a >= b |
| <= | Menor o igual que | a <= b |

**Operadores lógicos**

| Operador | Significado |
|---|---|
| ! | Negación |
| \|\| | O lógica |
| && | Y lógica |

# Cibernética y computación II

## Unidad I. Lenguaje de programación orientada a objetos con Java.

### Lenguaje de programación orientado a objetos.

Java es un lenguaje de programación de propósitos generales. Podemos usar Java para desarrollar el mismo tipo de aplicaciones que programamos con otros lenguajes como C o Pascal.

Habitualmente, tendemos a asociar el término “Java” al desarrollo de páginas de Internet. La gente cree que “Java es un lenguaje para programar páginas Web”, pero esto es totalmente falso. La confusión surge porque Java permite “incrustar” programas dentro de las páginas Web para que sean ejecutados en el navegador del usuario. Estos son los famosos Applets, que fueron muy promocionados durante los años noventa pero que hoy en día son obsoletos y, prácticamente, quedaron en desuso.

Tampoco debemos confundir Java con JavaScript. El primero es el lenguaje de programación que estudiaremos en este libro. El segundo es un lenguaje de scripting que permite agregar cierta funcionalidad dinámica en las páginas Web. Nuevamente, la similitud de los nombres puede aportar confusión por eso, vale la pena aclarar que JavaScript no tiene nada que ver con Java. Son dos cosas totalmente diferentes.

No obstante, podemos utilizar Java para desarrollar páginas Web. La tecnología Java que permite construir este tipo de aplicaciones está basada en el desarrollo de Servlets, pero esto es parte de lo que se conoce como JEE (Java Enterprise Edition) y excede el alcance de este libro. Solo mencionaremos que JEE es un conjunto de bibliotecas que permiten desarrollar "aplicaciones empresariales" con Java. Es decir que para programar con JEE primero debemos conocer el lenguaje de programación Java.

Java, como lenguaje de programación, se caracteriza por dos puntos bien definidos:

- Es totalmente orientado a objetos.
- La sintaxis del lenguaje es casi idéntica a la del lenguaje C++.

## Clases.

Una clase es código que especifica tanto los atributos, como los métodos de un objeto en particular.

Una analogía serían los moldes de pastel.



## Métodos.

Son funciones que ejecutan operaciones sobre los atributos del objeto.

**Métodos constructores**

- Los constructores son métodos especiales que sirven para inicializar un objeto de una determinada clase al mismo tiempo que se declara.

- Tienen el mismo nombre que la clase a la que pertenecen.

- No tienen tipo de retorno, por lo tanto no retornan ningún valor, ni siquiera void.

- Deben ser públicos (no tendría ningún sentido declarar un constructor como privado, ya que siempre se usan desde el exterior de la clase).

**Métodos de clase**

Son aquellos en los que no hace falta instanciar un objeto de la clase para utilizarlos. Son métodos estáticos, por lo tanto se usa la palabra reservada **static** para definirlos.

```
nombreClase.variable;
nombreClase.metodo();

Ejemplo:
public class Persona{
      public static int obtenerAltura() {
            //Aquí va la definición del método.
       }
}
```

**Métodos de instancia**

Son aquellos en los que se necesita crear un objeto de la clase para poder utilizarlos. Son métodos que no usan la palabra reservada static para definirlos.

```
objeto.metodo();

Ejemplo:
public class Persona{
      public void comer (int cantidadDeAlimento){
            //Aquí va la definición del método.
       }
}
```

**Métodos void**

Los métodos pueden devolver algo, por ejemplo, un método que suma dos números devuelve el resultado de la suma; pero hay métodos que no devuelven nada y que sólo ejecutan acciones. Dichos métodos se declaran con la palabra reservada “void” (vacío) como tipo de retorno.

**Métodos Get y Set**

Los métodos get y set, son simples métodos que usamos en las clases para mostrar (get) o modificar (set) el valor de un atributo.

El nombre del método siempre será get o set y a continuación el nombre del atributo, su modificador siempre es public ya que queremos mostrar o modificar desde fuera la clase. Por ejemplo, getNombre o setNombre.

```
public tipo_dato_atributo getAtributo(){
     return atributo;
}

public void setAtributo(tipo_dato_atributo variable){
    this.atributo = variable;
}
```

**Método toString()**

El método toString() devuelve un String que representa al objeto. El contenido de este String depende del objeto sobre el que se esté aplicando.

El método toString() se invoca de forma automática cuando se muestra un objeto mediante la instrucción System.out.println o System.out.print

## Objetos.

- Un objeto es una entidad de software que contiene tanto información (atributos) como procedimientos (métodos).
- Atributos: características del objeto.
- Métodos: Son funciones que ejecutan operaciones sobre los atributos del objeto.

**Instanciar**

La palabra instanciar significa crear objetos de una clase. La creación de objetos nos permitirá acceder a los atributos y métodos de una clase, es decir, si no se crea un objeto es imposible acceder a las características y a las acciones de una clase. Para crear un objeto se utiliza la palabra reservada new.

Ejemplo:

```
Persona persona1 = new Persona();
```

## La Clase Scanner.

Para introducir datos desde el teclado en un programa en Java, es necesario hacer uso de la clase Scanner.

- La clase Scanner permite leer datos a través del teclado.
- Para hacer uso de esta clase, es necesario importarla desde el paquete: java.util
- Su sintaxis sería la siguiente:
  import java.util.Scanner;

**El Objeto System.out**

- El objeto System.out nos sirve tener un flujo de salida hacia el monitor. Este objeto cuenta con dos métodos:

  System.out.print
  System.out.println

- El primero imprime en la consola el valor del argumento que le pasamos. El segundo hace lo mismo, pero agrega un salto de línea al final.

```
import java.util.Scanner;

class PideDatos{
      public static void main(String []args){
            Scanner teclado = new Scanner(System.in);
            System.out.print("Ingresa tu nombre: ");
            String nombre;
            nombre = teclado.nextLine();
            System.out.println("Hola " + nombre);
      }
}
```

**El método next()**

Los métodos next() sirven para la lectura de datos a través del teclado y puede tener las siguientes formas:

- next() solo lee hasta donde encuentra un espacio (hasta un espacio).
- nextLine() lee todo incluyendo espacios (hasta un enter).
- nextInt() lee un número entero.
- nextDouble() lee un número de tipo double.
- nextFloat() lee un número de tipo float.

- next().charAt(0) lee un carácter.

# Unidad II. Estructuras de control de secuencia en Java

## Estructuras condicionales.

**if … else**

Permite tomar una decisión dependiendo de un resultado (verdadero o falso).

```
if(condición){
      //Código 1;
}else if(condición 2){
      //Código 2;
}else{
      //Código 3;
}
```

Ejemplo:

```
import java.util.*;
public class MenuCalculos {

    public static void main(String[] args) {

        Scanner entrada = new Scanner(System.in);
        int opcion;

        System.out.println("Menú de opciones");
        System.out.println("------------------------");
        System.out.println("1. Calcular el área de un Cuadrado");
        System.out.println("2. Calcular el área de un Triángulo");
        System.out.println("3. Calcular el área de un Circulo");
        System.out.println("4. Finalizar");

        System.out.print("Elija una opción: ");
        opcion = entrada.nextInt();

        if (opcion == 1) {
```

```
            System.out.println("Ha seleccionado calcular el área de un cuadrado...");
        } else if (opcion == 2) {
            System.out.println("Ha seleccionado calcular el área de un triángulo...");
        } else if (opcion == 3) {
            System.out.println("Ha seleccionado calcular el área de un circulo...");
        } else {
            System.out.println("Ha seleccionado terminar");
        }
    }
}
```

## Estructura condicional múltiple.

**Switch**

Permite ejecutar una de varias opciones dependiendo del valor que tenga cierta expresión

```
switch (expresión){
      case x:
            //Código para x;
            break;
      case z:
            //Código para z;
      break;
      default:
            //Código para default;
      break;
}
```

Donde x y z son expresiones y si coincide con expresión entra a ese caso.

Ejemplo:

```
public class Test
{
    public static void main(String[] args)
    {
        int day = 5;
        String dayString;

        // instrucción switch con tipo de datos int
        switch (day)
        {
            case 1:  dayString = "Lunes";
                     break;
            case 2:  dayString = "Martes";
                     break;
```

```
            case 3:  dayString = "Miercoles";
                     break;
            case 4:  dayString = "Jueves";
                     break;
            case 5:  dayString = "Viernes";
                     break;
            case 6:  dayString = "Sabado";
                     break;
            case 7:  dayString = "Domingo";
                     break;
            default: dayString = "Dia inválido";
                     break;
        }
        System.out.println(dayString);
    }
}
```

## Estructura repetitiva for.

**For**

El bucle for proporciona una forma concisa de escribir la estructura de bucle. A diferencia de un ciclo while, una sentencia for consume la inicialización, la condición y el incremento/decremento en una línea, proporcionando así una estructura de bucle más corta y fácil de depurar.

1. **Condición de inicialización:** Aquí, inicializamos la variable en uso. Marca el inicio de un ciclo for. Se puede usar una variable ya declarada o se puede declarar una variable, solo local para el bucle.
2. **Condición de prueba**: se usa para probar la condición de salida de un bucle. Debe devolver un valor booleano. También es un bucle de control de entrada cuando se verifica la condición antes de la ejecución de las instrucciones de bucle.
3. **Ejecución de instrucción**: una vez que la condición se evalúa como verdadera, se ejecutan las instrucciones en el cuerpo del bucle.
4. **Incremento/Decremento**: se usa para actualizar la variable para la siguiente iteración.
5. **Terminación de bucle**: cuando la condición se vuelve falsa, el bucle termina marcando el final de su ciclo de vida.

```
for (condición de inicialización, condición de prueba; incremento / decremento)
{
    declaracion(es)
}
```

Ejemplo:

```
class forLoopDemo
{
    public static void main(String args[])
    {
        // bucle for comienza cuando x=2
        // y corre hasta x <=4
```

```
        for (int x = 2; x <= 4; x++)
            System.out.println("Valor de x: " + x);
    }
}
```

## Estructura repetitiva while.

Un bucle while es una sentencia de control de flujo que permite que el código se ejecute repetidamente en función de una condición booleana dada. El bucle while se puede considerar como una instrucción if repetitiva.

```
while (condición booleana)
{
   declaraciones del bucle ...
}
Ejemplo:

class whileLoopDemo
{
    public static void main(String args[])
    {
        int x = 1;

        // Salir cuando x llega a ser mayor que 4
        while (x <= 4)
        {
            System.out.println("Valor de x: " + x);

            //incrementa el valor de x para la siguiente iteración
            x++;
        }
    }
}
```

## Estructura repetitiva do–while.

El bucle do while es similar al while con la única diferencia de que comprueba la condición después de ejecutar las instrucciones, y por lo tanto es un ejemplo de Exit Control Loop (Salir del bloque de control).

1. El bucle do while comienza con la ejecución de la(s) declaración(es). No hay verificación de ninguna condición la primera vez.
2. Después de la ejecución de los enunciados, y la actualización del valor de la variable, la condición se verifica para el valor verdadero o falso. Si se evalúa como verdadero, comienza la siguiente iteración del ciclo.
3. Cuando la condición se vuelve falsa, el ciclo finaliza y marca el final de su ciclo de vida.
4. Es importante tener en cuenta que el bucle do-while ejecutará sus declaraciones al menos una vez antes de que se verifique cualquier condición, y por lo tanto es un ejemplo de bucle de control de salida.

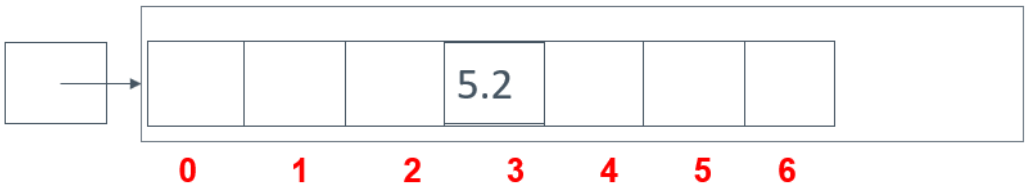
```
Ejemplo:
```

```
class dowhileloopDemo
{
    public static void main(String args[])
    {
        int x = 21;
        do
        {
            //El código dentro del do se imprime incluso
            //si la condición es falsa
            System.out.println("Valor de x :" + x);
            x++;
        }
        while (x < 20);
    }
}
```

## Arreglos unidimensionales.

Un arreglo es un objeto que almacena datos del mismo tipo.

Los más comunes son los arreglos de una dimensión (vectores) y de dos dimensiones (matrices).




La estructura de declaración de un arreglo es la siguiente:

```
tipo_de_dato[] nombre_variable;

Ejemplos:
char[] arreglo;
int[] arreglo;
double[] arreglo;
String[] arreglo;
```

**Asignar tamaño a un arreglo**

Al ser un objeto en el que se guardan varios valores de un tipo de dato en específico, a los arreglos se les debe asignar un tamaño y se hace de la siguiente forma:

```
arreglo = new tipo_de_dato[tamaño];

Ejemplos:
arreglo = new char[5];
```

```
arreglo = new int[10];
arreglo = new double[4];
arreglo = new String[20];
```

**Inicializar un arreglo**

Se puede hacer por separado:

```
int[] arreglo = new int[3];
      arreglo[0] = 17;
      arreglo[1] = 20;
      arreglo[2] = 18;
```

En una sola línea:

```
int[] arreglo = {17,20,18};
```

**Acceder a elementos de un arreglo**

Se puede acceder a cada elemento de un arreglo de forma separada:

```
//Imprime el elemento del arreglo ubicado en la posición 1
System.out.println(arreglo[1]);
```

Imprimir todos los elementos del arreglo de forma dinámica:

```
for(int i = 0; i < 5; i++){
     System.out.println(arreglo[i]);
}
```

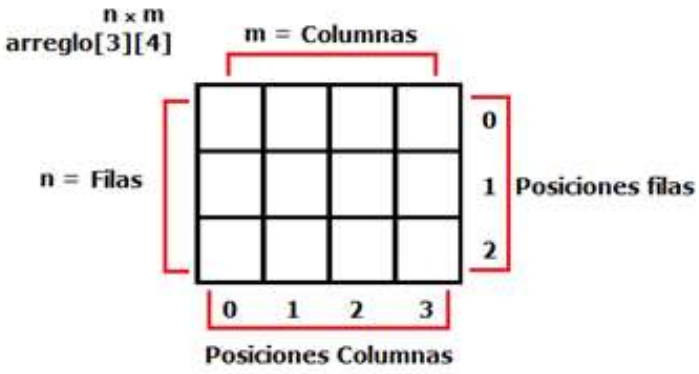
**Tamaño de un arreglo**

Para obtener el tamaño de un arreglo se usa el atributo length

```
arreglo.length
```

## Arreglos bidimensionales.

Este tipo de arreglos son conocidos como matrices y corresponden a una estructura de datos que puede almacenar muchos más datos que los arreglos unidimensionales, ya que se componen de n filas por m columnas.




**Declaración e inicialización**

La estructura de declaración de un arreglo bidimensional es la siguiente:

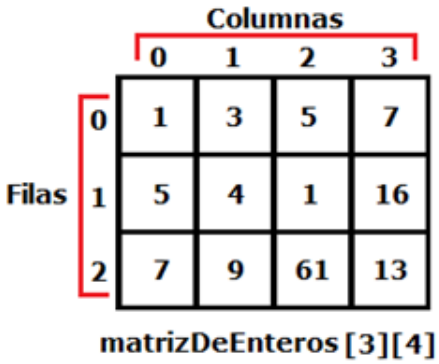
```
tipo_de_dato[][] nombre_variable;
nombre_variable = new tipo_de_dato[filas][columnas];

Ejemplo:
int[][] matrizDeEnteros;
matrizDeEnteros = new int[3] [4];
```

**Asignar valores:**

**Llenado fila 0 columnas del 0 al 3**
matrizDeEnteros[0][0]=1;
matrizDeEnteros[0][1]=3;
matrizDeEnteros[0][2]=5;
matrizDeEnteros[0][3]=7;

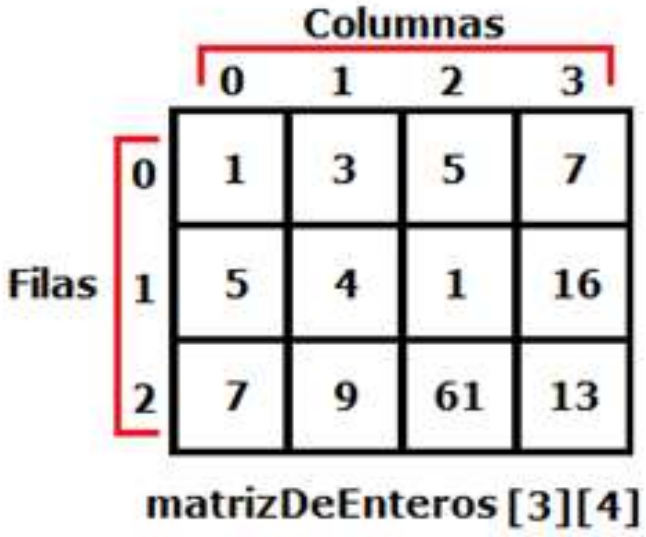
**Llenado fila 1 columnas del 0 al 3**
matrizDeEnteros[1][0]=5;
matrizDeEnteros[1][1]=4;
matrizDeEnteros[1 ][2]=1;
matrizDeEnteros[1][3]=16;

**Llenado fila 2 columnas del 0 al 3**
matrizDeEnteros[2][0]=7;
matrizDeEnteros[2][1]=9;
matrizDeEnteros[2][2]=61;
matrizDeEnteros[2][3]=13;




**Otra forma de asignar valores:**

```
matrizDeEnteros = { {1, 3, 5, 7} , {5, 4, 1, 16} , {7, 9, 61, 13} };
```




**Matriz de cadenas:**

```
14  public static void main(String[] args) {
15
16      String estaciones[][]=new String[2][2];
17
18      //llenado de la matriz
19      estaciones[0][0]="Otoño";
20      estaciones[0][1]="Verano";
21      estaciones[1][0]="Invierno";
22      estaciones[1][1]="Primavera";
23
24      //Obteniendo información de la matriz
25      System.out.println("estación en la posición (0,0): "+estaciones[0][0]);
26      System.out.println("estación en la posición (0,1): "+estaciones[0][1]);
27      System.out.println("estación en la posición (1,0): "+estaciones[1][0]);
28      System.out.println("estación en la posición (1,1): "+estaciones[1][1]);
29  }
```

Lo anterior crea una matriz de tipo String y tamaño 2x2 con posiciones de 0 a 1 en filas y 0 a 1 en columnas

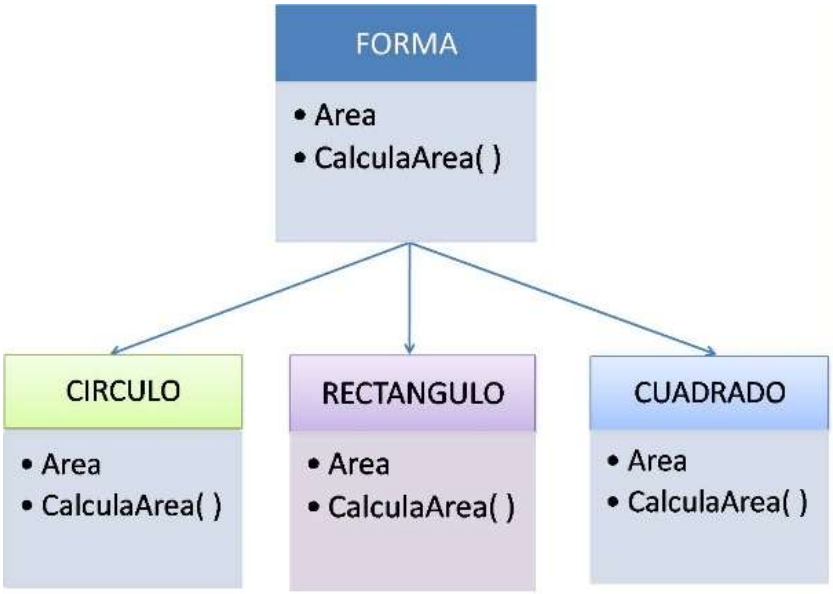
**estaciones =**

| | | |
|---|---|---|
| "Otoño" | "Verano" | 0 |
| "Invierno" | "Primavera" | 1 |
| 0 | 1 | |

## Unidad III. Polimorfismo, constructores, colaboración y herencia de clases.

## Concepto de polimorfismo.

Permite usar un nombre para varios propósitos relacionados, pero ligeramente diferentes. Los comportamientos pueden ser identificados bajo el mismo nombre pero procesan información de manera diferente de acuerdo con el objeto que lo contenga.




## Concepto de constructor.

- Los constructores son métodos especiales que sirven para inicializar un objeto de una determinada clase al mismo tiempo que se declara.
- Tienen el mismo nombre que la clase a la que pertenecen.
- No tienen tipo de retorno, por lo tanto no retornan ningún valor, ni siquiera void.
- Deben ser públicos (no tendría ningún sentido declarar un constructor como privado, ya que siempre se usan desde el exterior de la clase).

## Interacción entre clase y herencia.

La Herencia es uno de los 4 pilares de la programación orientada a objetos (POO) junto con la Abstracción, Encapsulación y Polimorfismo. Respecto a la herencia se han dado muchas definiciones como por ejemplo la siguiente: "La herencia es un mecanismo que permite la

definición de una clase a partir de la definición de otra ya existente. La herencia permite compartir automáticamente métodos y datos entre clases, subclases y objetos.".

**Encapsulamiento**

Esta propiedad permite el ocultamiento de la información, es decir, permite asegurar que el contenido de un objeto se pueda ocultar del mundo exterior dejándose ver lo que cada objeto necesite hacer público.

**Modificadores de acceso**

Permiten controlar la forma de acceder a los atributos y métodos encapsulados dentro de una clase.

**PÚBLICO**: Cualquier atributo o método Publico puede sea accedido desde fuera de la clase.

**PRIVADO**: Cualquier atributo o método Privado NO puede sea accedido desde fuera de la clase. Solo puede ser utilizado internamente en la clase.

**PROTEGIDO**: Cualquier atributo o método Protegido puede ser heredado por otra clase pero en esta última se convierten en elementos.

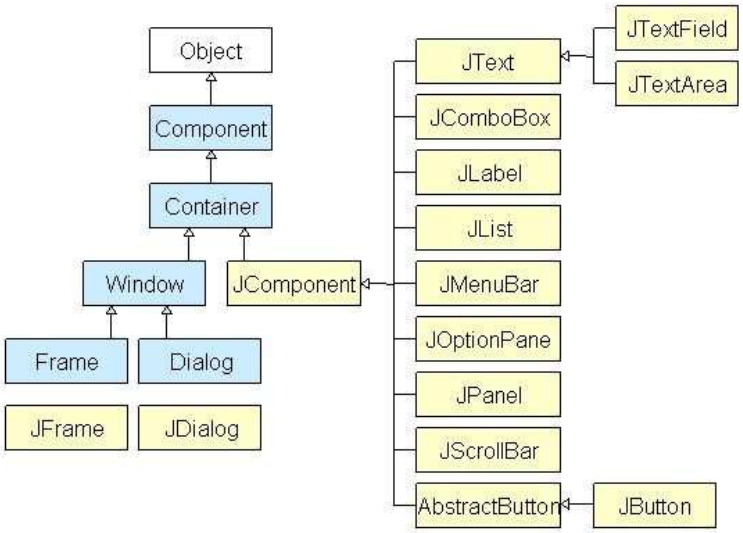
**Herencia – Super y This**

La palabra **super** se utiliza para hacer referencia al constructor de la clase padre. En Java no existe la “herencia múltiple”, por lo tanto, para cualquier clase siempre se tendrá un único padre. Además, se debe tener muy en cuenta que los constructores no se heredan.

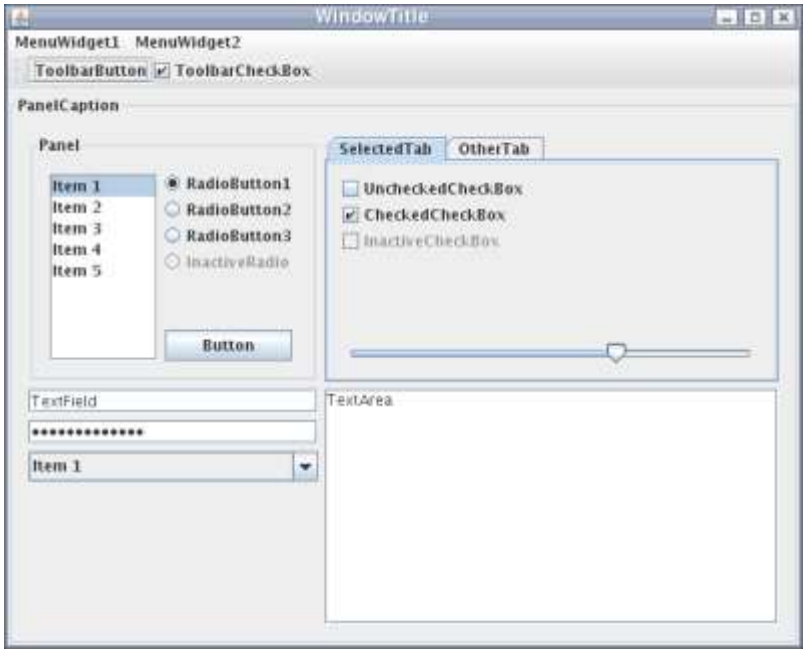
La palabra **this** se utiliza para hacer referencia a los constructores de la misma clase. Se pueden invocar tanto atributos como métodos.

# Unidad 4. Interfaz gráfica de usuario.

Es un paquete que hace parte de la Java Foundation Classes o más conocida como JFC, la cual provee herramientas o facilidades para la construcción de GUI's o interfaces Graficas de Usuario (Graphical User Interface).

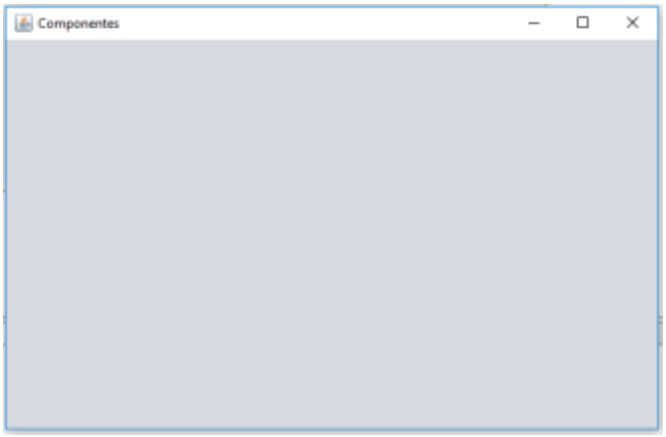



Swing es la evolución del AWT (Abstract Window Toolkit), la cual al igual que Swing es un conjunto de librerías enfocadas a la construcción de interfaces, solo que con esta se presentaron algunos problemas en cuanto a portabilidad principalmente cuando se desarrollaban aplicaciones para diferentes sistemas operativos, pues el comportamiento de los componentes gráficos en ocasiones podían variar.




## JFrame

El JFrame es ventana de la interfaz gráfica de una aplicación en Java.




## JPanel

El JPanel es el contenedor de todos los componentes que se agreguen a la interfaz gráfica de una aplicación en Java.


## JLabel

El JLabel sólo se usa para mostrar texto en una línea dentro de un contenedor.

Instanciar un JLabel
```
JLabel label = new JLabel();
```

Poner texto en el JLabel
```
label.setText("Hola");
```

Leer texto en el JLabel
```
String texto = label.getText();
```

Limpiar el JLabel
```
label.setText("");
```


## JButton

Un objeto de control JButton permite dibujar en el formulario un objeto que contiene un proceso a ejecutar.

**Propiedades más usadas:**

- Text: Contiene el valor o dato introducido en el cuadro de texto.
- Font: Permite establecer el tipo de letra del texto en la caja.
- Enabled: Para habilitar o inhabilitar el uso del objeto de control.

**Evento más usado:**

ActionPerformed: Este evento se lleva a cabo cuando el usuario da click sobre el objeto de control JButton.


## JTextField

El JTextField sólo admite y devuelve String, por lo que si queremos ingresar u obtener números, debemos hacer una conversión.

Poner números en el JTextField

```
int valor = 33;
textField.setText(Integer.toString(valor));
```

Leer números en el JTextField

```
int valor;
```

```
String texto = textField.getText();
valor = Integer.parseString(texto);
```

Instanciar un JTextField

```
JTextField textField = new JTextField();
```

Poner texto en el JTextField

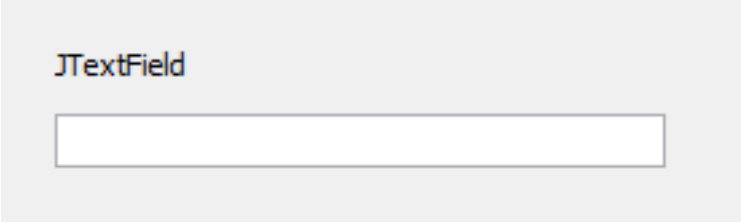
```
textField.setText("Hola");
```

Leer texto en el JTextField

```
String texto = textField.getText();
```

Limpiar el JTextField

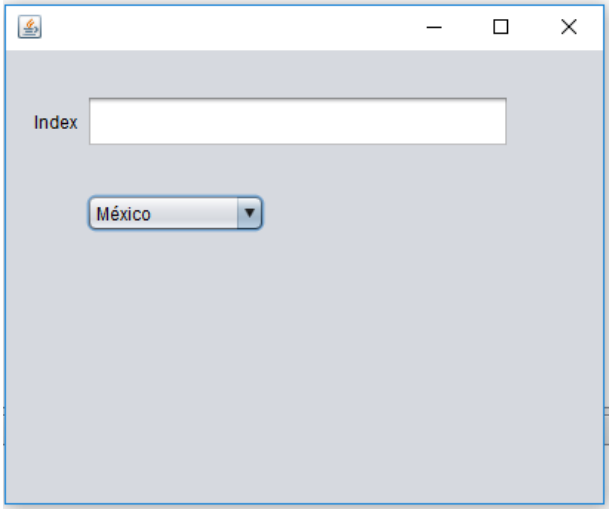
```
textField.setText("");
```




## JComboBox

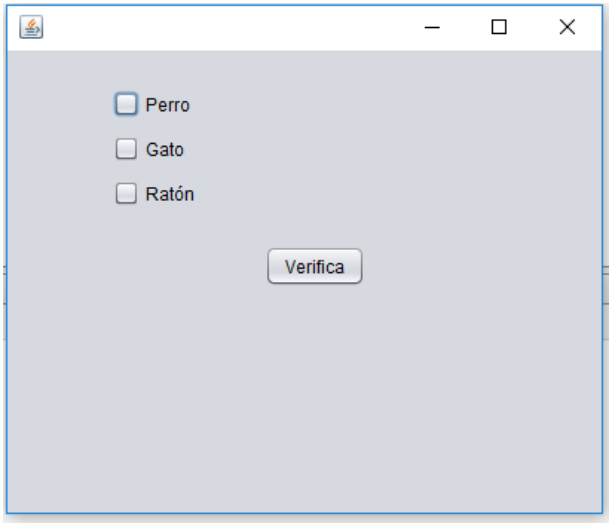
Los combos son listas desplegables donde se puede elegir una de las opciones propuestas.

A través del método getSelectedItem() se puede extraer la opción seleccionada o el texto escrito en el combo.




## JCheckBox

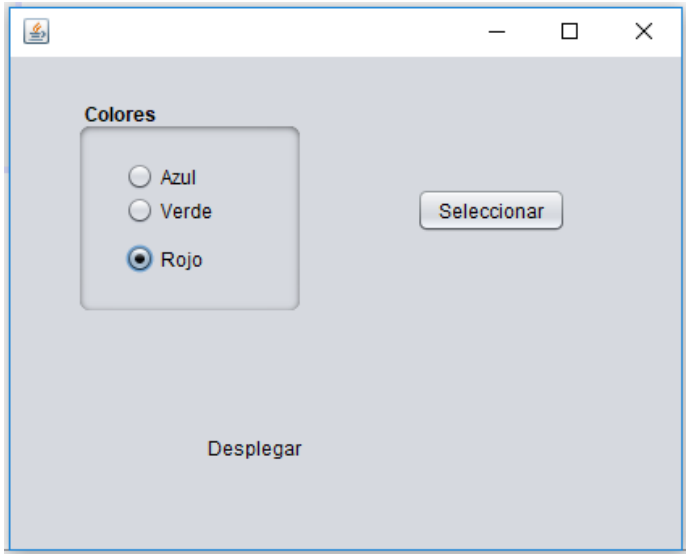
Las cajas de verificación se utilizan cuando se desea hacer una selección múltiple dentro de un listado.




## JRadioButton

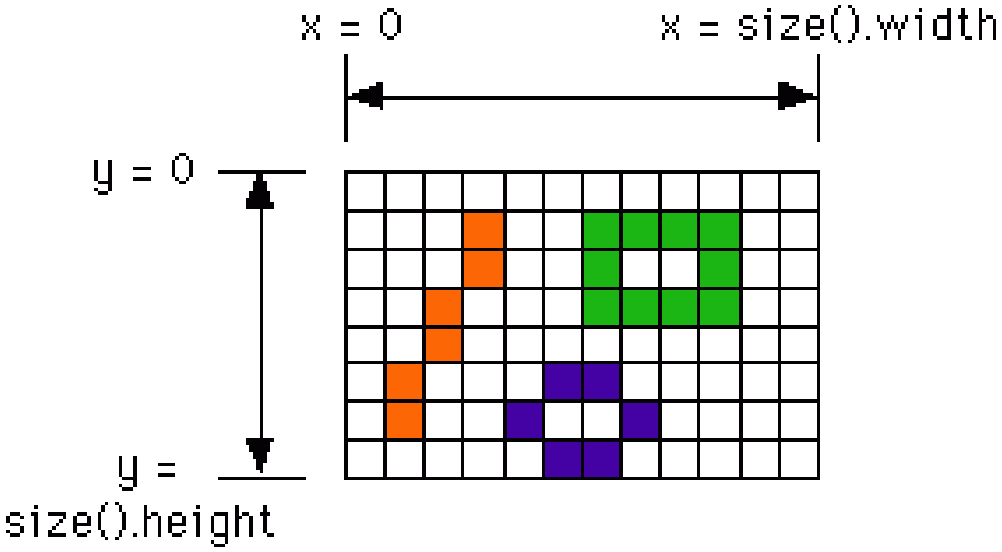
Los botones de opción, también llamados botones de radio (JRadioButton) se usan cuando quieres que el usuario pueda elegir una opción de entre varias.

Es totalmente necesario añadir un objeto del tipo ButtonGroup, y hacer que los botones de radio pertenezcan a dicho grupo. En caso contrario, será posible activar varios botones de opción a la vez.




## Clase Graphics

Casi todas las componentes y contenedores de Swing tienen un método paint() asociado que sirve para dibujarlos en pantalla. Java invoca este método automáticamente cuando tiene que mostrar, de forma estándar, el componente o contenedor en cuestión.




El método paint() se redefine cuando se quiere que estos elementos tengan un aspecto particular, por ejemplo, cuando se quiere dibujar algo específico sobre ellos.

El método paint() es de la forma:

```
public void paint(Graphics g) {
      ...
}
```

Donde g es un objeto de la clase abstracta Graphics. Todo contenedor o componente que se pueda dibujar en pantalla tiene asociado un objeto g de esta clase, con la información sobre el área de la pantalla que el contenedor o el componente cubre. Además, el objeto g dispone de métodos para hacer gráficos (dibujar círculos, rectángulos, líneas, etc.).

Cuando el método paint(g) se ejecuta es porque ha sido invocado por otros métodos, nunca invocado por nosotros, y el parámetro que usa corresponde a un objeto de la clase Graphics asociado al contenedor o componente que estemos manejando.

Cuando se redefine el método paint(g), siempre se comienza con una invocación super.paint(g) al método de la superclase, asegurando así que se dibuja la parte estándar del contenedor o componente que estemos manejando.

## Páginas consultadas

1. http://laciberneticaporbiaanigupo512.blogspot.com/2017/08/la-cibernetica-con-otras-ciencias.html

2. https://www.hiru.eus/es/matematicas/sistemas-de-numeracion

3. https://www.profesionalreview.com/2018/12/11/sistema-binario-decimal-octal-hexadecimal/

4. https://www.ecured.cu/%C3%81lgebra_Booleana

5. https://www.logicbus.com.mx/compuertas-logicas.php

6. https://sites.google.com/a/educacion.navarra.es/electronica-digital-basica/circuitos-integrados

7. https://www.matematicasypoesia.com.es/ProbBoolePropo/problema111.htm

8. https://www.gestiopolis.com/solucion-de-problemas-y-toma-de-decisiones-administrativas/

9. http://hustariz.cs.umss.edu.bo/algPruEscr.htm

10. http://platea.pntic.mec.es/~lgonzale/tic/binarios/aritmetica.html

11. https://portalacademico.cch.unam.mx/alumno/cibernetica1