**Sleep en un for, delay en un for**

**Thread** es una clase del api de JAVA que se usa para implementar diferentes “hilos” de ejecución de un programa. En este caso lo que hacemos es usar el **método** **sleep** que **detiene** la **ejecución** del **hilo** (**programa**) durante el nro indicado de milisegundos.

Este tipo de instrucciones se usan por ejemplo cuando hay un bucle en ejecución y se quiere hacer un calculo sin sobresaturar al procesador.

public static void main(String[] vector) throws InterruptedException{

int num=1;

while(num<10){

System.out.println(num);

Thread.sleep(1000);

num++;

}

**Cuando se utiliza el length-1**

Por ejemplo en un for el -1 se utiliza cuando como condición booleana utilizamos un <=. Ya que el for arranca en el cero y termina en una posición previa.

Ahora si en la condición booleana tenemos un < ahí si utilizamos solo el length ya que con eso será necesario para llegar al ultimo valor.

Ej:

int [] oldArray = {1, 5, 6, 10, 25, 17};

System.out.println("Primer loop");

for(int i = 0; i <= oldArray.length-1; i++){

System.out.println(oldArray[i]);

}

System.out.println("Segundo loop");

for(int i = 0; i < oldArray.length; i++){

System.out.println(oldArray[i]);

}

Ahora por ejemplo si quisiésemos sacar por consola el ultimo valor de un vector utilizando la propiedad length, debemos utilizar la siguiente expresión:

System.out.println("Ultima Posicion: "+ oldArray[oldArray.length - 1]);

Claro porque el vector oldArray[5] = 17 que es lo que buscamos. Por lo tanto tenemos que utilizar oldArray[oldArray.length-1]=17.

**Proceso de instalación del plugin de Maven.**

**Update the Maven version repository:**

Download the Apache Maven binary that includes the default https addresses ([Apache Maven 3.6.3 binary](https://www-eu.apache.org/dist/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.zip)). And open the *Options=> dialog window* in tools of NetBeans menu bar ([Java Maven Dialog View](https://i.stack.imgur.com/bjcF4.png)). And select browse option in Maven Home List Box ([Maven Home List Box View](https://i.stack.imgur.com/Q9k32.jpg)). After adding the Apache Maven newly downloaded version ([Updated Maven Home List Box View](https://i.stack.imgur.com/d7eTG.png)), the project builds and runs successfully.

**Fuente:** <https://stackoverflow.com/questions/59763531/maven-dependencies-are-failing-with-a-501-error>

**Sintaxis de JAVA – Parte II**

**Ambito variables**

boolean llueve = true;

if(llueve){

int x = 10;

}

System.out.println(x);

Por que el codigo anterior da un error de compilación?.

**Laboratorio**

Realizar del Laboratorio #1 el Ejercicio #1.

Imprimir la suma totalPositivos y totalNegativos.



Alternativa al ejercicio (no utilización de variable auxiliar, a través de un método que haga la suma o resta) – **Ejercicio para hacer entre todos.**

Ej2 Imprimir los valores 1000 2000 3000 pasados como argumentos del programa principal. (Properties del Project, agregamos los argumentos en “Run”).

main

for (int i = 0; i < args.length; i++) {

System.out.println(args[i]);

}

**Opciones de Netbeans (IDE, pestañas, beneficios)**

Algunos de los beneficios de Netbeans son:

* Atajos del teclado. Por ejemplo:

sout + Tabulador: System.out.println(“”);

for + Tabulador: for (int i = 0; i < 10; i++) {}

 Control + Barra Espaciadora: Obtener sugerencias acerca del código que estamos escribiendo.

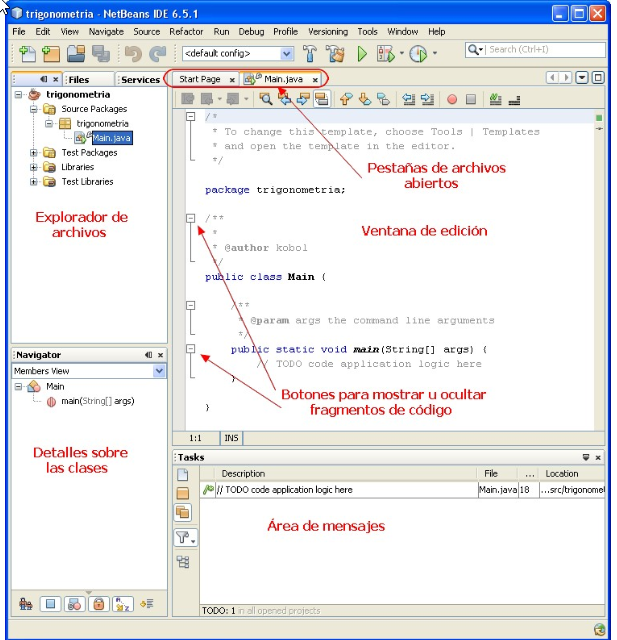
Control + A: Seleccionar todo el código.

Control + Shift + C: Comentar linea seleccionada.

* Es de codigo abierto.
* Fue desarrollado por la compañía Sun, la misma que creo Java, lo cual permite ser descargado e instalarse en el mismo paquete JDK, lo cual simplifica mucho la instalación para los alumnos y desarrolladores. Actualmente pertenece a Oracle.

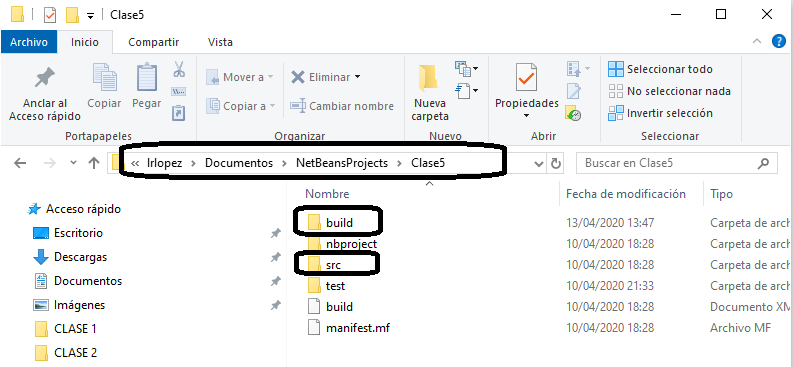
***Crear un proyecto nuevo.***

Vemos los componentes principales del IDE:



* Netbeans complila automáticamente los archivos en el momento en que se graban (icono del diskette). Como también cuando se ejecuta la flechita verde de “Run”.
* Vemos que se han creado las siguientes carpetas en el directorio de trabajo:

C:\Users\lrlopez\Documents\NetBeansProjects\Clase5

****

La carpeta **src** se encuentran los archivos fuente de nuestro programa, mientras que en **build** es donde NetBeans guarda los archivos compilados (.class).

* A diferencia de los entornos anteriormente estudiados, en NetBeans puedes ver las clases en la parte izquierda agrupadas por el paquete al que pertenecen
* La clase que contiene el método principal main está destacada con una flecha verde en su icono
* La parte inferior izquierda es como un diagrama de clase: incluye toda la información sobre atributos y métodos de la clase que estemos editando
* NetBeans también puede mostrar números de línea, mediante el menú View > Show line numbers.
* Otra característica, incorpora autocompletado del codigo:



***Puntos fuertes de NetBeans***

* Sin duda, el más completo, estable y fiable de los tres
* Si un alumno necesita programar en su vida profesional y ha aprendido con NetBeans, podrá enfrentarse con confianza a cualquier entorno presente en la empresa, ya que todos son muy parecidos entre sí
* La gestión de paquetes y sus avanzadas detecciones de errores (incluso antes de compilar) resultan más cómodas e intuitivas que en los otros entornos

***Puntos débiles de NetBeans***

* Su consumo de recursos es significativamente mayor que el de las otras alternativas

**Eclipse** es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida a través de plug-ins. No tiene en mente un lenguaje específico, sino que es un IDE genérico, aunque tiene mucha popularidad entre la comunidad de desarrolladores del lenguaje **Java** usando el plug-in **JDT** que viene incluido en la distribución estándar del IDE.

**Principales características**

**Perspectivas, editores y vistas**: pre configuración de ventanas y editores nos permiten trabajar de forma óptima.

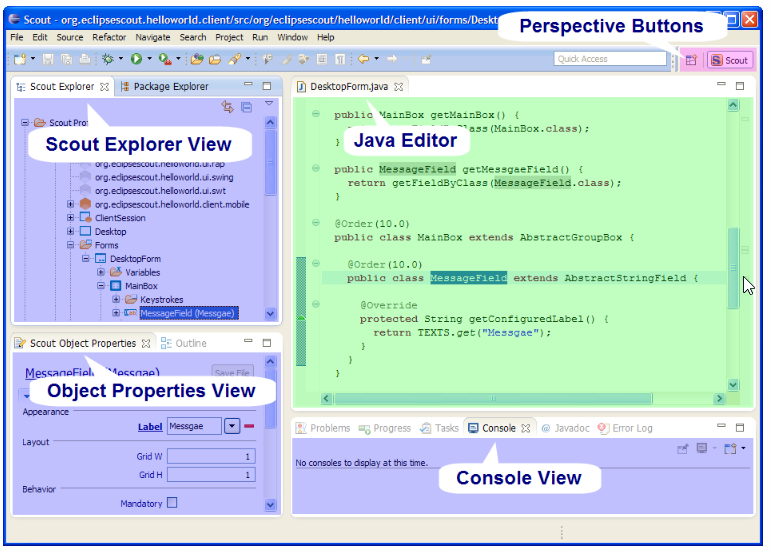
**Gestión de proyectos**: los proyectos son el conjunto de recursos relacionados entre sí, como puede ser el código fuente, documentación, ficheros configuración, árbol de directorios. El IDE nos proporcionará asistentes y ayudas para la creación de proyectos.

**Depurador de código**: se incluye un potente depurador, de uso fácil e intuitivo, y que visualmente nos ayuda a mejorar nuestro código, logrando la **perspectiva de depuracion**.

**Extensa colección de *plug-ins***. JDT es el plugin para JAVA.

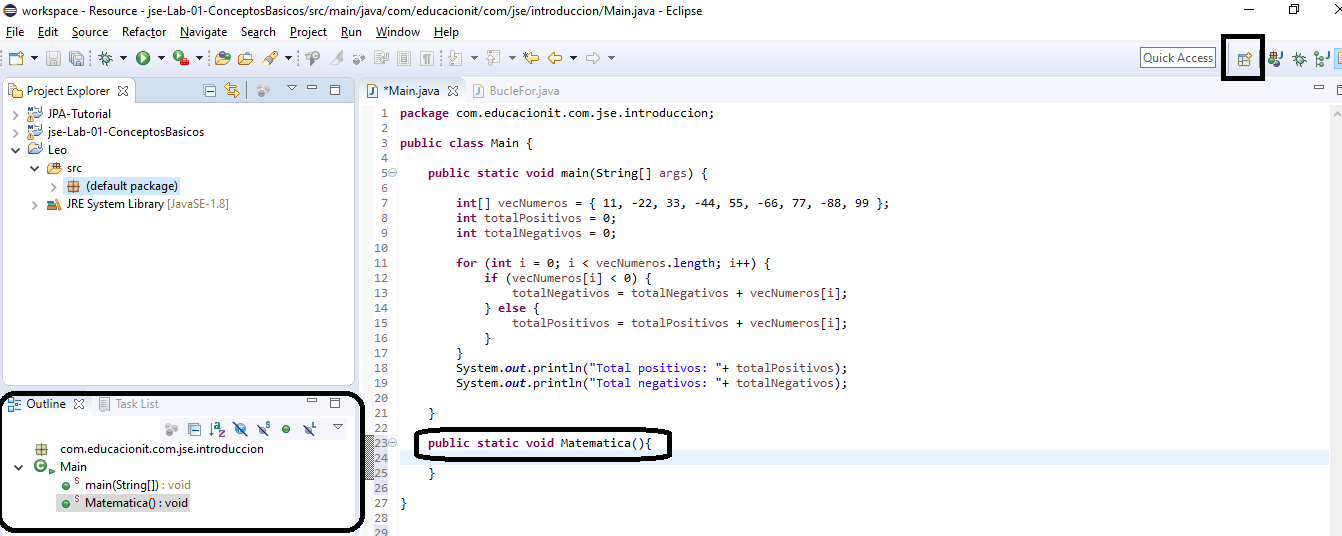
**Ventajas: JAVA 06-06**  
\*Dispone de un editor de texto con resaltador de sintaxis.  
\*La compilación es en tiempo real  
\*Tiene pruebas unitarias con (JUnit).  
\*Control de versiones con (CVS).  
\*Integración con (Ant), Ant es una aplicación que permite dividir los codigos fuentes de los compilados. Es decir me ayuda a organizar el codigo.  
\*Asistentes (wizards) para creación de proyectos  
\*Emplea plug-ins.

\*Este mecanismo de módulos le permite extenderse usando otros lenguajes de programación como son C/C++ y Python, LaTeX, aplicaciones en red como Telnet y Bases de Datos.  
  
**Desventajas:**  
\*Su principalmente inconveniente, común a otros IDEs en mayor o menor medida, en el consumo de recursos del sistema.



OPEN PERSPECTIVE => RESOURCE

Se puede ver la sección donde hace referencia a los objetos de la clase.



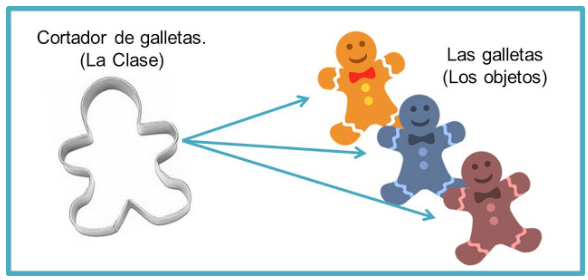
**Introducción a Objetos – Parte I**

**¿Que es una clase?**

Es una agrupación de reglas de negocio o representación de un concepto de la vida real. **Es una plantilla, con reglas definidas para armar un objeto.**

Esta formada por atributos que definen un estado y métodos que definen su comportamiento.

Ej: Una clase es un molde para fabricar objetos. Ejemplo un molde de cocina donde el molde de cocina es la clase y cada comida que surge de dicho molde es el objeto.



Creamos una clase Persona en Netbeans. Las clases siempre empiezan en mayuscula con la lógica CamelCase.

**Objeto**: Es una instancia de la clase. Puede crearse multiples veces. Palabra reservada **new.**

*Proceso de Instanciacion de un objeto:*

Main

Persona p = new Persona();

//referencia = creacion y reserva

//de persona en memoria del objeto

// p apunta a este lugar en memoria.

p. // muestra los métodos de la clase object. Son los métodos que JAVA guarda para los objetos.

A diferencia de la asignación de memoria para una variable donde tenemos:

int a = 19; //en donde se asigna directamente a “a” el valor “19” en memoria. Para un objeto es necesaria la palabra reservada **new.**

**¿que es un atributo?**

Debe comenzar con letras minúsculas.

Los atributos son las características de un objeto. Describen a mi objeto.

Los valores de los atributos me determinan el estado del objeto.

Los datos de los atributos pueden ser primitivos u objetos de cualquier clase.

Los atributos van a depender del negocio.

Se agregan dentro de la clase en particular.

package clase5;

public class Persona {

String nombre;

String apellido;

int edad;

}

*Crear una clase Domicilio(calle, numero). Luego en el main crear un objeto de tipo persona. Asignarle valores a sus atributos. Y asignarle el atributo de tipo Domicilio.* ***Ejercicio para hacer entre todos (b).***

**Que es un método?**

Los métodos determinan el comportamiento y la responsabilidad que tendrán las clases.

Representan como se van a utilizar las clases.

Debe comenzar con letras minúsculas.

Ej: Creo un método dentro de la clase persona // Definicion del metodo

public String dameTuNombreCompleto(){

return nombre + ", " + apellido;

}

Invocacion del método (desde el main)

System.out.println(p.dameTuNombreCompleto());

La **diferencia** entre un método y un atributo es que el método se observa como finaliza entre paréntesis y el **atributo** no.

Ej Metodo que recibe un parámetro.

public String saluda(String saludo){

return "Hola "+ saludo;

}

**Constructor:** Metodos que se utilizan para construir o instanciar una clase. Son llamados cuando se utiliza la palabra reservada **new.**

**Armar clase auto con atributos marca (String), color (String), velocidad (int) con método acelerar(), que incrementa la velocidad en 10**

**Armar clase programa**

Main es el punto de entrada de un programa como ya mencionamos.

Recordamos el concepto de Objeto.

**Objeto**: Es una instancia de la clase. Puede crearse multiples veces. Palabra reservada **new.**

**Instanciar un objeto del tipo auto dentro del main()**

public class Auto {

String marca;

String color;

int velocidad;

public void acelerar(){

velocidad+=10;

}

}

Le asignamos valores a los atributos

main

Auto a1 = new Auto(); //instanciamos.

a1.marca = "Ford";

a1.color = "Azul";

a1.velocidad = 100;

Invocar el método acelerar()

main

System.out.println("Velocidad Actual: "+a1.velocidad);

a1.acelerar();

System.out.println("Velocidad Modificada: "+a1.velocidad);

**Codificar el método mostrar Info() en Clase Auto.**

Deberá informar marca, color y velocidad junto con sus respectivos valores

Auto

public void informar(){

System.out.println("Marca: "+marca);

System.out.println("Color: "+color);

System.out.println("Velocidad: "+velocidad);

}

Continuar con la clase Programa invocar al mostrar Info

Main

a1.informar();

**Armar el método frenar() en la clase Auto.**

Este método debe bajar la velocidad en 80, pero tiene como requisito que no puede quedar negativo. Informar "frenando..." cada vez que es invocado.

Auto

public void frenar(){

velocidad-=80;

if(velocidad<0)

velocidad=0;

}

Creamos un Constructor para practicar: Crear un constructor de Auto que tome los dos atributos marca y velocidad y que pueda actualizar valores en dichos atributos. **Ejercicio para hacer entre todos (c)**

**Main**

//Constructores

Auto a2 = new Auto("Dodge", 30);

a2.frenar();

a2.informar();//me devolvio null porque justamente no le asignamos un valor a Color.

a2.color = "Azul";

a2.informar();

**Crear una nueva clase llamada DatosAuto**

Con un atributo string chasisNro y un método que informe el nro de chasis

Crear un atributo en Auto que sea DatosAuto.

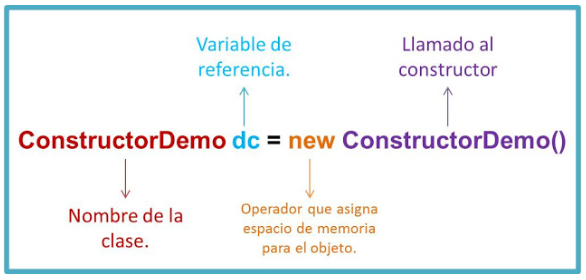
Crear una instancia de DatosAuto en el main, una instancia de auto a3 y llamar a los métodos de ambas clases, en forma separada.

**Diferencia entre Clase y Objeto:**

Una clase se genera cuando se codifica la misma. En cambio un objeto es creado al momento en que se ejecuta el programa, específicamente cuando se llega a la línea de codigo que contiene **new.** En ese momento se llama al constructor de la clase.

La diferencia mas importante entre una Clase y objeto es que este ultimo normalmente tiene un estado. Este se utiliza para diferenciarse de otros objetos, a través de los diferentes valores que toman sus atributos.

Otro punto importante para mencionar en la creación de un objeto es la llamada al Constructor. Vemos que ocurre cuando creamos un objeto.



Otra diferencia es

La **Clase** es una **abstracción** que contiene código, principalmente atributos y métodos que operan sobre ellos. Por otro lado el **objeto** es una cosa **real** que operan sobre ellos.

**Destructor Que es?**

El destructor se utiliza para destruir una instancia de una clase y liberar memoria. En Java no hay destructores, ya que la liberación de memoria es llevada acabo por el Garbage Collector cuando las instancias de los objetos quedan desreferenciadas.  
El método dispose() de cada objeto se llama previo a ser “recolectado”.

Es decir, la aplicación de los destructores quedan bajo la orbita del Garbage collector el cual libera recursos de la heap que es el lugar de memoria donde residen los objetos.

Forzar la ejecución del Garbage collector para destruir objetos no seria del todo adecuado ya que el garbage collector realiza varias tareas previas a la destrucción de un objeto (ej: liberar conexiones a base de datos, liberar conexiones a redes, liberar referencias).