TALLER DE HERRAMIENTAS DE PROGRAMACIÓN

Eclipce (entorno de desarrollo)

Una nueva clase es un nuevo programa

Cuando en la consola pongo Entre barras son comentarios //

Si quiero poner un comentario largo tengo q poner /* y cerrar con */

Variables es un “caja” que guardamos información

Tipo de datos se pueden dividir en

Numérica = int(entero), float(numero grande con comas), doublé (para comas con número muy grande arriba del millon)

alfanuméricas: string , char

Cuando yo tengo un solo = me guarda la variable que coloco

Constante en mayúscula , variable en minuscula

&& es el conector lógico Y

System.out.println = es la sentencia que se usa para que el usuario vea algo , lo q esta entre parentecis (es lo q va a mostrar) , para ponerlo automáticamente pones Syso(+control +espacio) y se pone todo

Cada instrucción se termina un ;

Para publicar el método main si nos olvidamos es main +control +espacio

Si yo en un println quiero hacer una suma directa tengo q poner en paréntesis

Ej System.out.println(“el resultado de la multiplicación es “ + (valorUno+valorDos))

Para una constante tenemos q poner final (si yo veo esto quiere decir q es una constante) y después el tipo de constante y para no confundirlas las nombramos todas en mayúsculas, cuando defino esto , uso todo el programa ese valor

Ej final int DIAS = 7;.

Import java.util.Scanner = es la que nos permite interactuar con el teclado , todo lo q es entrada de teclado

Static final Scanner input = new Scanner (System.in)

El objeto input(o como lo llamemos) tiene un metodo que es nextLine() toma lo q ingresaron por teclado y lo deja dentro de la variable cuando el usuario toca enter

Para ingresar algún tipo char por teclado , tiene que ir con comillas simple ‘ ‘

Siempre que uso un scanner siempre lo tengo q cerrar input(o nombre que puse).close , va a final de todo

Después del nombre de la variable

Si es el ingreso de teclado es para tipo string solamente coloco la palabra `teclado.nextLine();` después del nombre de la variable

Para numero enteros tengo q poner `Integer.parseInt(input.nextLine());` es sirve para convertir lo q viene en el teclado en un tipo de dato entero para asignar dentro de una variable entera , esto cambia si uso doublé , `Double.parseDouble(input.nextLine());` para un variable de ese estilo (ambos casos el interger y el doublé va con la primera letra mayúscula)

Variable de tipo **boolean**(valores lógicos) tienen dos valores nada mas true y false

El if lo q haces es evaluarnos una condición entre verdadero y falso , la condición va entre paréntesis lo q esta entre llaves solamente toma el parámetro q le di a if si no es verdadero salta a otro paso si yo no indico otra cosa

El else significa si esto no ocurio hago esto otro (se usa cuando lo q pusieron por teclado no pasa por el if o quiero mostrar otra cosa)

Las constante se definen antes del método main y empieza con static(todo lo q esta a nivel de la clase)

! es para negar

!= Es distinto

toUpperCase es para pasar todo a mayúscula (para que si el usuario pone en minúscula y yo lo necesito en mayúscula)

toLowerCase = para pasar todo a minúscula

```
switch(genero.toUpperCase()){
```

Colocamos la variable (género com arriba)

En el swicht (es un selector)

equalsIgnoreCase = es para ignorar como puso si en mayúscula o en minuscula

Tengo q colocar un case: con una variable o una constante y adentro su sentencia, para colocar otro case o salir del swicht tengo q poner un break (no va dentro de otro lado que sea un switch

Puedo colocar un default para decir que el lo que ingreso es invalido

si yo imprimir algo sin el enter coloco system.out.print() sin el ln

num++ (suma uno) es lo mismo que num= num+1 o num +=3 (el numero que esta después del igual es la cantidad de veces que se va a incrementar)

num-- (resta uno) es lo mismo que num= num-1 o num -=3

también se puede usar para multiplicar 3 veces ese numero como por ejemplo

num*= 3 o num=num*3

Ciclos (for , while y do while)

Las variables que se crean en los ciclos tiene solamente alcanse dentro de ellas

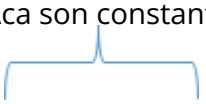
For

Si se cuántas veces estoy en el ciclo , uso el ciclo de 1 A N y es el ciclo for

Y en java es

Sintaxis , lleva tres parametos

Aca son constante que habíamos definido pero se puede reemplazar por otra cosa



```
for (int i = INICIO; i <= FIN; i++)
```

1 2 3

1) lleva una variable , que lleva un valor inicial es el que usamos para contar la cantidad de interacciones o de ciclos, se define dentro de los parentesis

2) condición; determina hasta cuando se ejecuta el ciclo

3)la usamos para definir si incrementa o discrementa , se coloca la variable ++ o –

While

Si no se cuantas veces estoy en el ciclo uso el ciclo 0 a N y es el ciclo while (no entro en el ciclo), puedo mostrar el error y solamente se entra si es error , EL PRIMER INGRESO ES FUERA DEL WHILE

Do while

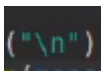
Si no se cuántas veces voy a entrar en el ciclo, pero se que tengo q entrar o usarlo, uso ciclo de 1-n y uso el do while

Siempre q ingreso un mayor , tengo q poner el MIN_VALUE;

Si yo voy a usar una variable acumulada q solamente sirve para un proceso (como sea mostrar por separor el total recaudado para cada autopista) la variable inicializada va dentro del ciclo

```
while (!nombre.isEmpty())
```

isEmpty () para ver si no esta en blanco



es es fin de línea , se usa para validad que no coloque solamente enter

MODULARIZACIÓN

Return variable; retoma lo que ingreso cuando se lo piden

FUNCIONES

Las funciones pueden o no devolvernos un valor , y tendrían que ser reutilizables

Las q no devuelven valor se llaman void

Se colocan abajo del main (este tiene que ser un coordinador ,el cual toma las funciones y solamente tiene que quedar la estructura del programa, el como se realiza estará en la funciones)

La firma del método es como se compone , no solamente en cuanto a los atributos , si no también tenemos el tipo de datos q devuelve esa función , si no devuelve es de tipo **void**

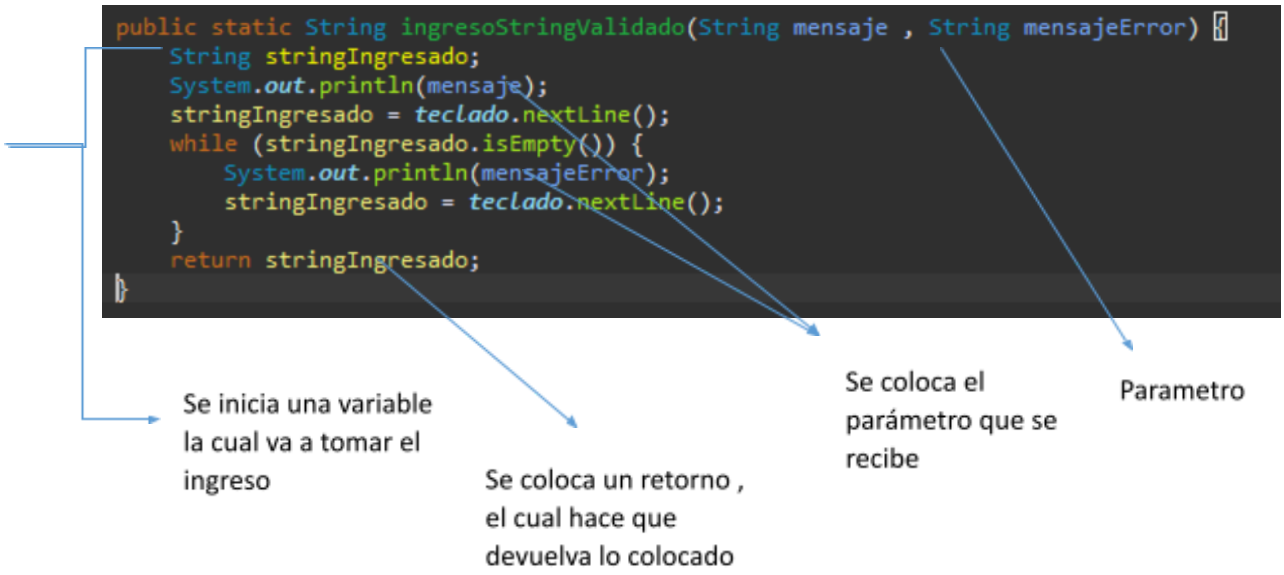
(ejemplo de funcio/modulo)

Las función es genérica , no hablamos de nombre o de algo especifico , se va encargar de recibir un tipo variable (sea string o int o el q sea)

Si yo tengo un mensaje dentro de esa función , no tiene q ser fijo para algo especifico , se resuelve con parámetros , se colocan dentro del parámetro

Las variables de cada función son locales , solamente sirven para es metodo

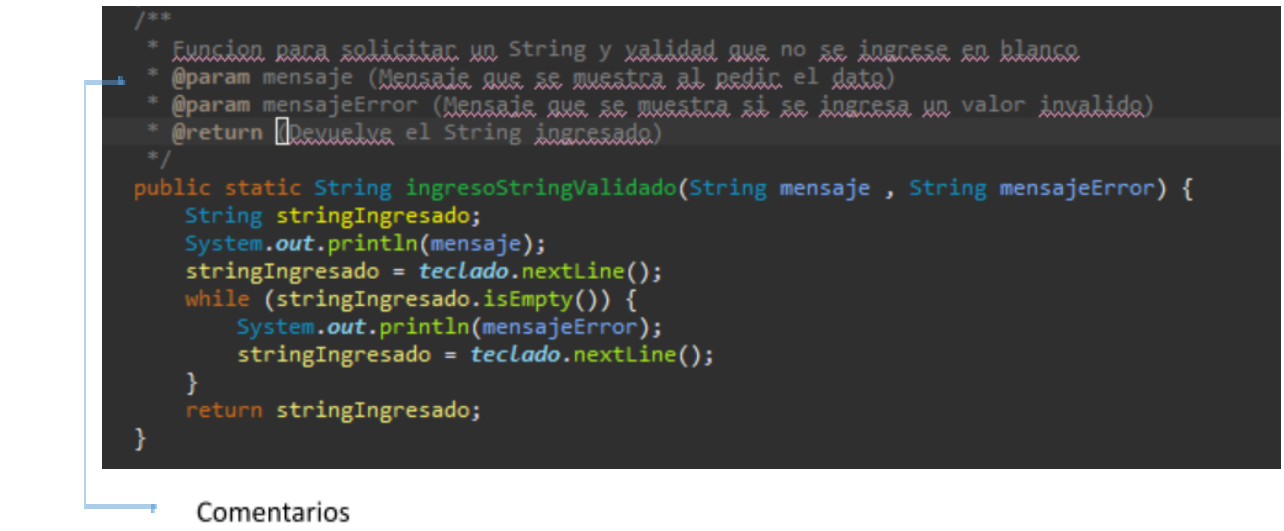
(ejemplo de método terminado, en este caso de tipo String)



Return(variable); retoma lo que ingreso cuando se lo piden

El método o función tiene que quedar documentado, esto se hace colocando arriba del método o función un /** (barra , doble asterisco) , se llama javaDoc

Ejemplo:



En el método main se remplaza el sistema.out.printl() y el teclado.nextLine por el nombre de la función.

Ahora si yo tengo una función se coloca o asigna , la variable, un igual (=) , el nombre de la función , y los parámetros que va a recibir

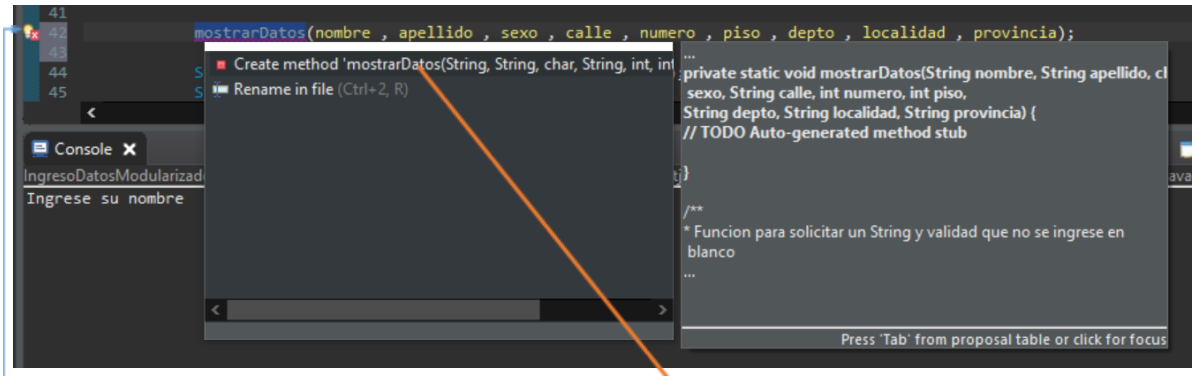
```
nombre= ingresoStringValidado("Ingrese su nombre", " Nombre es obligatorio , ingrese nuevamente");
apellido= ingresoStringValidado("Ingrese Apellido", " Apellido es obligatorio ingrese nuevamente");
calle= ingresoStringValidado("Ingrese calle", " calle es obligatorio ingrese nuevamente");
```

Variable

Nombre de la función

Parametros

Cuando yo en el método main coloco un método/ función y todavía no lo cree me aparece una advertencia , si todo sobre la advertencia , me da una opción de crear automática mente el método (o la primera parte)



advertencia

OPCION PARA CREAR EL METO AUTOMATICO

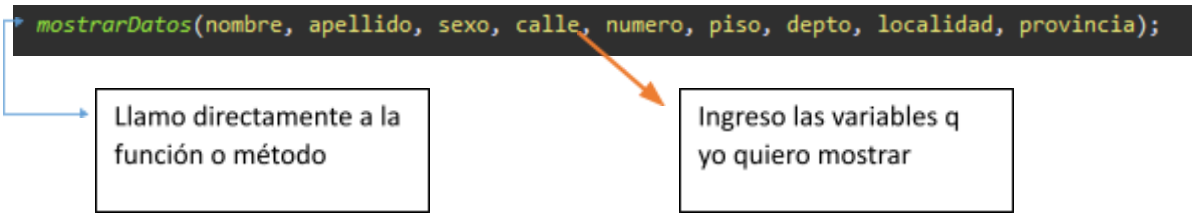
Funcion void

La funcion void no devuelve valor , generamente se usa para mostrar mensajes.

```
 * MUESTRA LOS RESULTADOS
public static void mostrarDatos(String nombre, String apellido, char sexo, String calle, int numero, int piso,
        String depto, String localidad, String provincia) {
    System.out.println("Confirme los datos ingresados ");
    System.out.println("Nombre " + nombre);
    System.out.println("Apellido " + apellido);
    System.out.println("Sexo " + sexo);
    System.out.println("Calle " + calle);
    System.out.println("Numero " + numero);
    System.out.println("Piso " + piso);
    System.out.println("depto " + depto);
    System.out.println("Localidad " + localidad);
    System.out.println("Provincia " + provincia);
```

Es una función void no devuelve ningún valor , si no devuelve un mensaje

Cuando yo llamo a una funcion void , no asigno ninguna variable delante , si no llamo directamente al metodo y entre los parentesis coloco las variables q se van a usar



OBJETOS

Un Objeto es la representación de una entidad o cosa de la vida real, sea concreta o conceptual (reloj, automóvil, avión, paloma, nota, calendario, superhéroe, etc.)

No todo puede ser considerado como un objeto, algunas cosas son simplemente características o atributos de los objetos: *color, velocidad, nombre*.

Qué es una Clase:

Es la forma de agrupar un conjunto de datos (**atributos**) y de funcionalidad (**comportamiento**) en un mismo bloque de código que engloba la definición de un objeto.
La **clase** a la que pertenece el objeto puede considerarse como un nuevo **tipo de dato**, tal como los que ya conocemos hasta ahora**.**

Es el "molde" que utilizaremos para crear nuestros objetos durante el programa.
Cada objeto, entonces, es un ejemplar construido a partir de la definición utilizada como "molde".
A través del objeto podemos acceder a sus propios atributos y a los métodos definidos en la clase.
Este, luego, puede ser referenciado desde otras partes del programa.

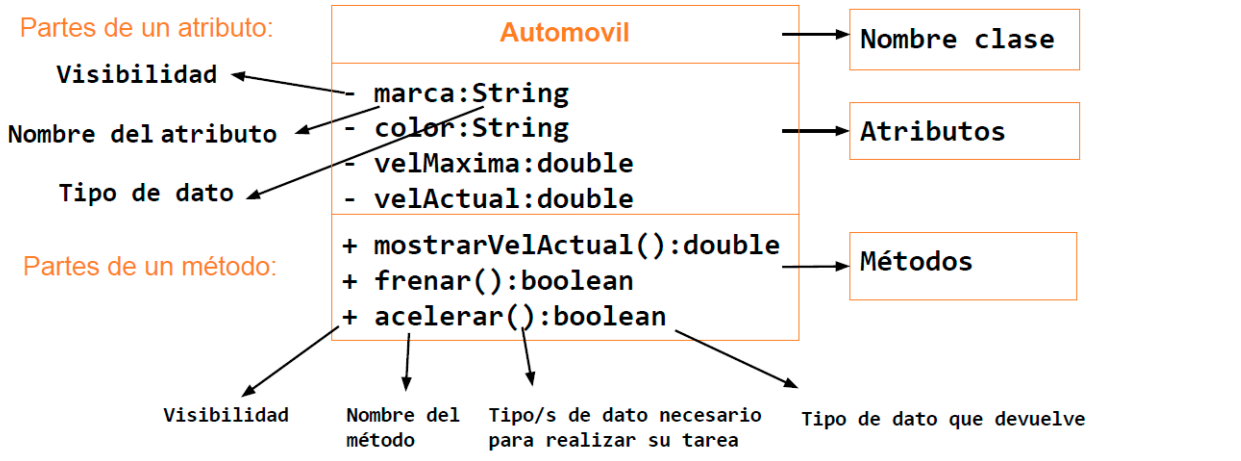
Clases vs. Objetos

Una Clase es una entidad abstracta con una clasificación de datos y define su comportamiento.

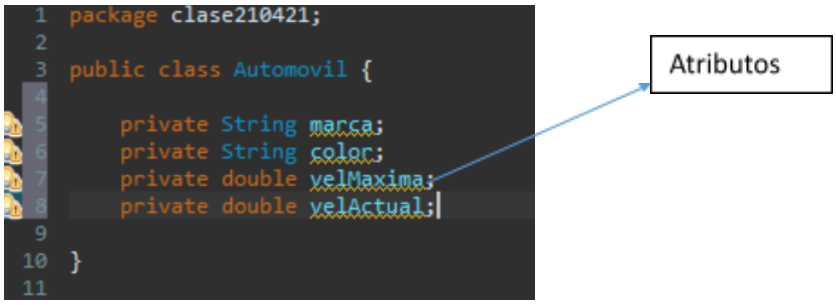
Un objeto es una instancia particular de la clase. Son automóviles "fabricados" a partir de la clase.

Modelando la clase Automovil

Una Clase es una entidad abstracta con una clasificación de datos y define su comportamiento.

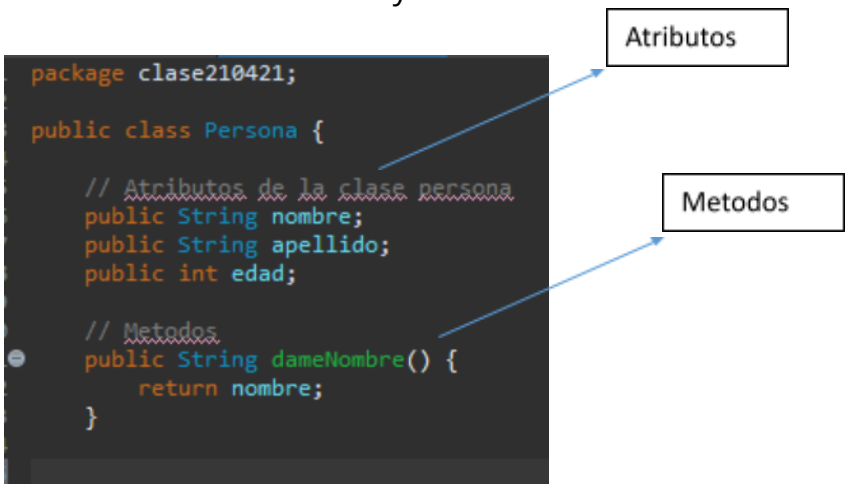


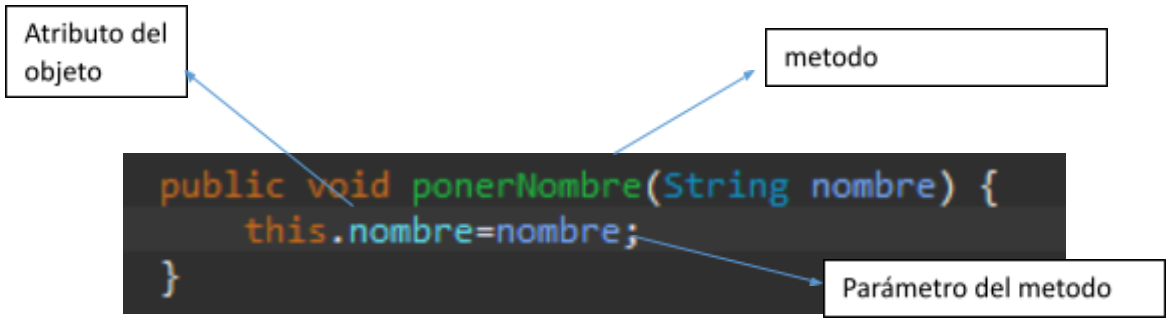
Para empezar a crear un clase en java , creo normalmente una clase pero sin tildar el método main , vamos a generar una estructura para después crear objetos de tipo automóvil .
Los atributos dentro de una clase son variables , los atributos siempre son privados(visibilidad) , tiene un nombre y un tipo de dato .



Es el mismo objeto que tiene q tener la responsabilidad de ser quien actualiza sus atributos , podemos tener determinada validaciones (reglas de negocios , etc) q tiene q esta presente de a la hora de actualizar los atributos

Toda clase tiene atributos y metodos

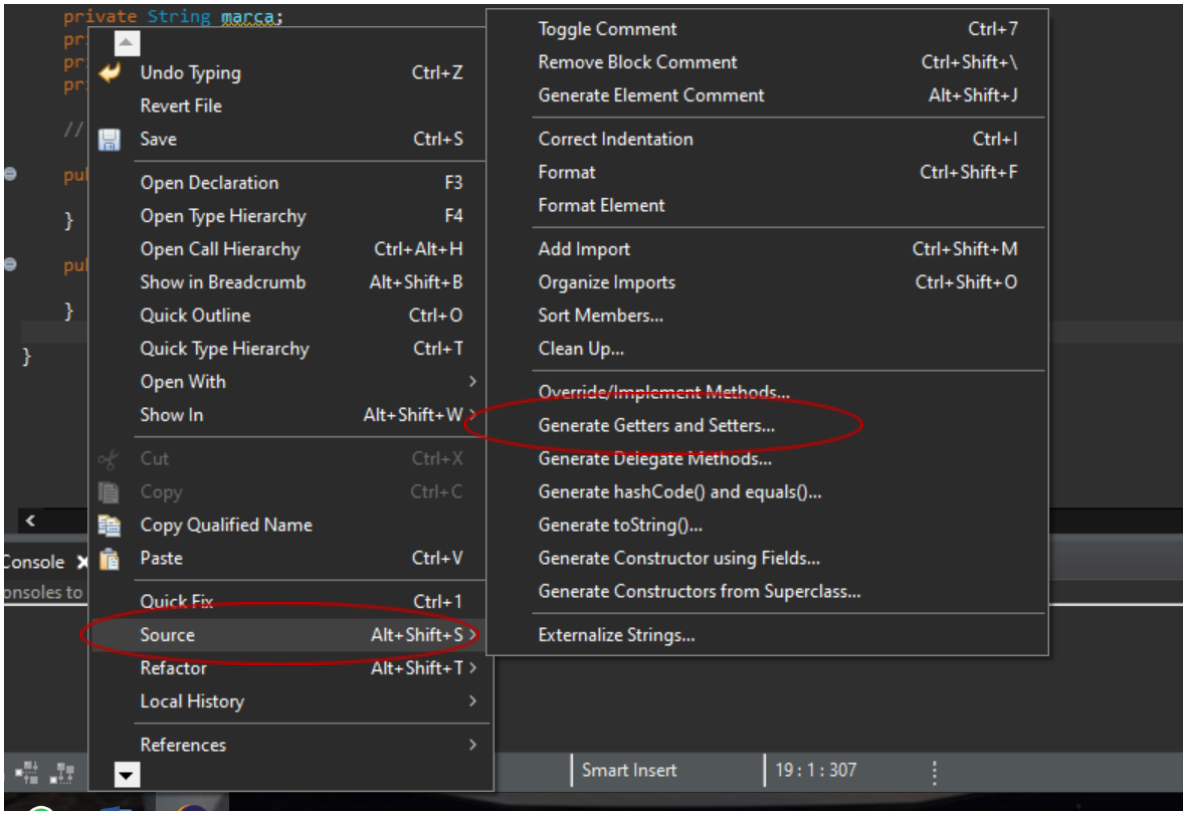




el this. sirve para eliminar la ambigüedad entre el parámetro y el atributo q se llaman igual .

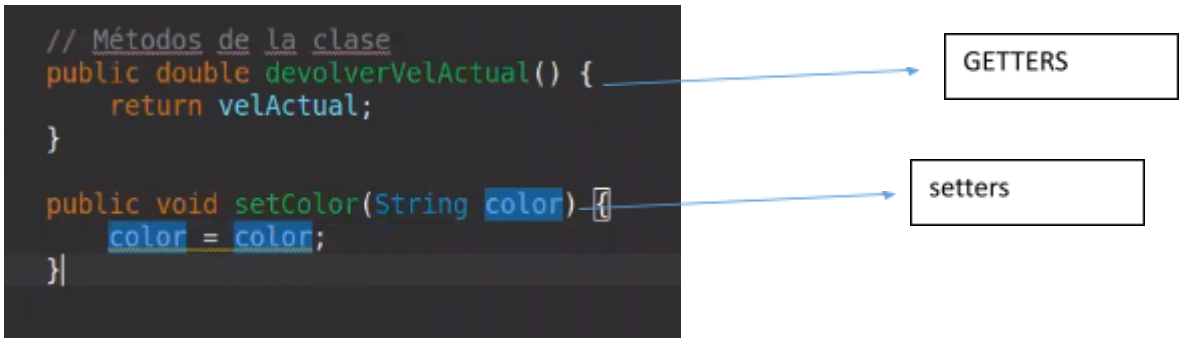
los Métodos tienen acceso a los atributos , Los métodos que permiten acceder al valor de un atributo se denominan "**getters**" (del verbo inglés "get", obtener) y y los que fijan el valor de un atributo se denominan "**setters**" (del verbo inglés "set", fijar).

Estos getters , se pueden crear automáticamente en eclipse con clip derecho sobre el renglón de escritura , vas a source y generate getters and setters

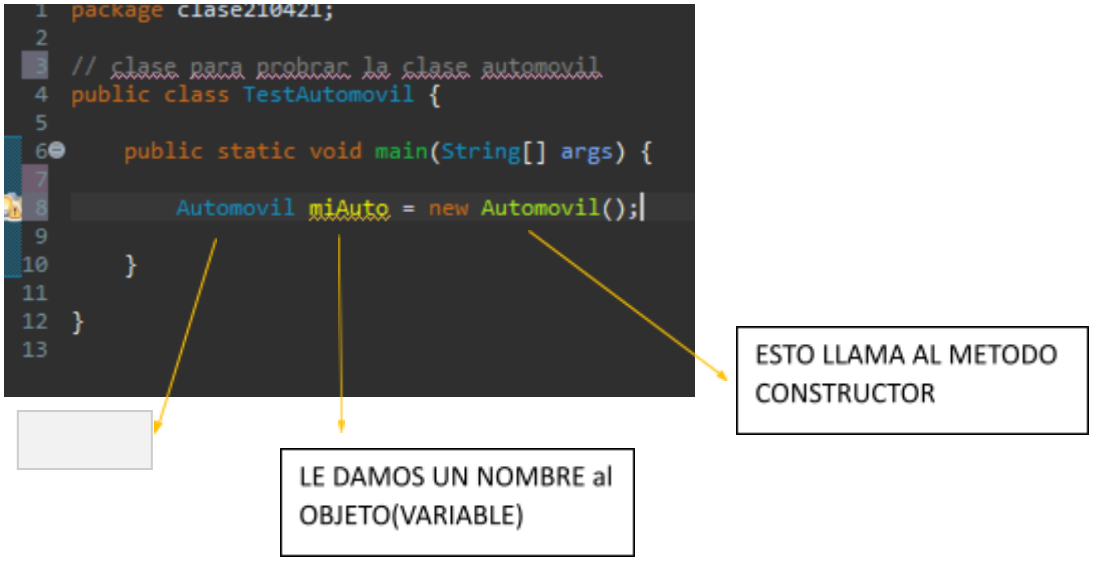


El getters esta echo para cuando yo le pido al objeto saber el estado de uno de sus atributos , me devuelve ese valor.

En cambio el setters recibe un parámetro que es el dato que se le coloca al atributo , por eso se pone el this.



Como creo/defino un objeto de tipo automóvil(clase) esto se realiza en el programa que yo creer , llamando a la clase o objeto que cree previamente

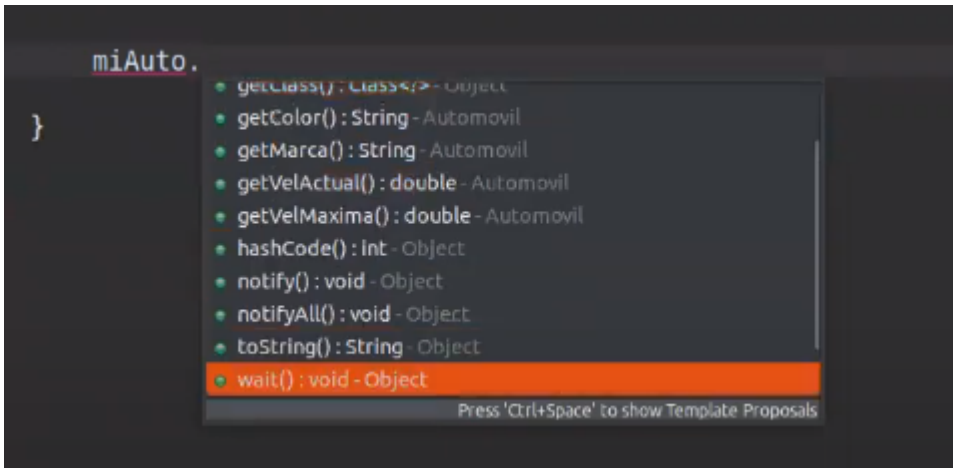


método constructor se ejecuta una sola vez en el momento que crea el objeto , ese objeto puede tener varios constructores , pero uno solo se ejecuta al momento de crearlos

cuando creamos el objeto , lo que decimos es q a partir de ese momento , se crea ese objeto en la memoria , se asigna un dirección en memoria , esa dirección de memoria se guarda en esa variable y en esa dirección de memoria lo q tendre guardado es los atributos de esa clase y los métodos que yo cree de esa clase.

Una vez que yo tengo creado el automóvil(clase) , para acceder a todos sus atributos o métodos públicos , es el nombre del objeto y un . y te despliega

una lista con lo que esta accesible del objeto y los que creaste dentro de ese objeto , ya sea métodos o getters o setters.
Ejemplo(esto esta dentro de programa que creo , no dentro de la clase)



Los constructores se crean dentro de la clase , puedo tener sobrecarga de constructores (dos métodos constructores o mas) , pero según el parámetro que recibe el programa se da cuenta cual tiene que ejecutar, por cada atributo que tengo puedo agregarle un parámetro , tengo q ver si tengo el getters para devolverlo , tengo que tener cuidado al colocar los parámetros , tengo q respetar el orden y el tipo
Esto se realiza dentro de la clase automóvil que yo realice

```java
public class Automovil {

    private String marca;
    private String color;
    private double velMaxima;
    private double velActual;

    //Constructor que recibe un parametro
            public Automovil() {
                this.marca= "Ort Cars";
            }

    //Constructor que recibe un parametro
        public Automovil(String marca) {
            this.marca= marca;
        }
```

Sobrecarga de consturctores

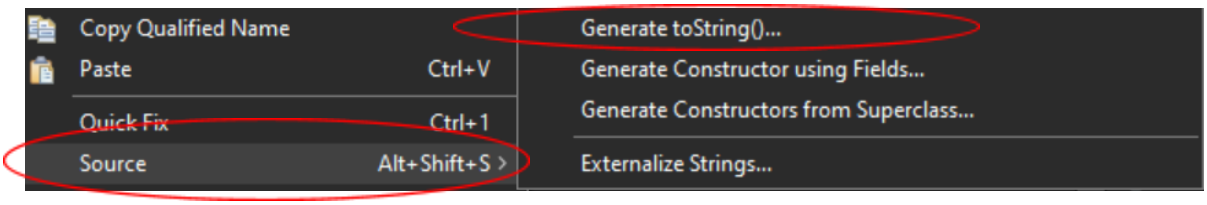
Uno con parámetros y otro sin , solamente nombrando dentro del constructor

El parámetro del constructor se coloca cuando creo el objeto

Creo el objeto automóvil en el programa , y le doy al consturctor automóvil(verde) un parámetro (lo q esta entre paréntesis)

```
Automovil miAuto2 = new Automovil("Ford");
System.out.println("\nMarca: " + miAuto2.getMarca());
System.out.println("Color: " + miAuto2.getColor());
System.out.println("Velocidad Maxima: " + miAuto2.getVelMaxima());
System.out.println("Velocidad Actual: " + miAuto2.getVelActual());
```

Para mostrar en un formato string el contenido del objeto para eso podemos usar toString , este se genera en source , generar toString , esto se crea en la clase que realice yo (ej automóvil)



Asi queda(yo eso puedo modificarlo a mi gusto o crearlo manuelmente)

```
@Override
public String toString() {
    return "Automovil [marca=" + marca + ", color=" + color + ", velMaxima=" + velMaxima + ", velActual="
            + velActual + "]";
}
```

Setters

Los setters no son solamente void , pueden ser boolean , string , etc
Puede tener un retorno también

Dentro de los setters podemos tener validaciones para que éstos actualicen los atributos , por ejemplo un if

```
public void setColor(String color) {
    if(this.marca.equalsIgnoreCase("Ferrari") && this.color.equalsIgnoreCase("Rojo")){
        this.color=color;
    }else if(!this.marca.equalsIgnoreCase("Ferrari")) {
        this.color=color;
    }


}
```

Ejemplo de setters tipo booleans

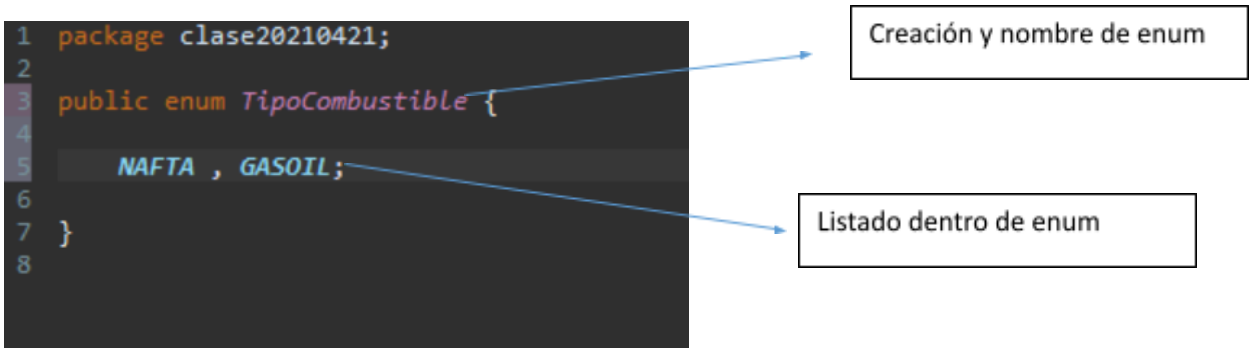
```java
public boolean setColor(String color) {
    boolean cambioColor = false;
    if (this.marca.equalsIgnoreCase("Ferrari") && color.equalsIgnoreCase("Rojo")) {
        this.color = color;
        cambioColor = true;
    } else if (!this.marca.equalsIgnoreCase("Ferrari")) {
        this.color = color;
        cambioColor = true;
    }
    return cambioColor;

}
```

Si nosotros tenemos setters que actualizan atributos con alguna validación , en el constructor que creemos tenemos q usar el setters de color que realizamos , para q no nos puedan actualizar sin validar ese setters .

Enumeradores

Un **enumerado** (o **Enum**) es una clase "especial" (tanto en **Java** como en otros lenguajes) que limitan la creación de objetos a los especificados explícitamente en la implementación de la clase.



DEBUGGER

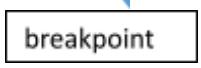
Su objetivo es encontrar errores que pueden impedir que los códigos funcionen de forma adecuada. Con este, es posible determinar lo que está ocurriendo dentro del código fuente y obtener sugerencias de acciones para mejoras.

Breakpoint(punto de tensión)
Se va a frenera en ese punto, podemos tener varios .

```java
public static void main(String[] args) {

int numero;
int mayorIngresado=Integer.MIN_VALUE;
int valoresIngresados=0;

System.out.println("Ingrese un valor o 0(cero) para salir");
numero=Integer.parseInt(input.nextLine());
while(numero!=0) {
    valoresIngresados++;
    if(numero>mayorIngresado) {
        mayorIngresado=numero;
    }
    System.out.println("Ingrese un valor o 0(cero) para salir");
    numero=Integer.parseInt(input.nextLine());
}
```

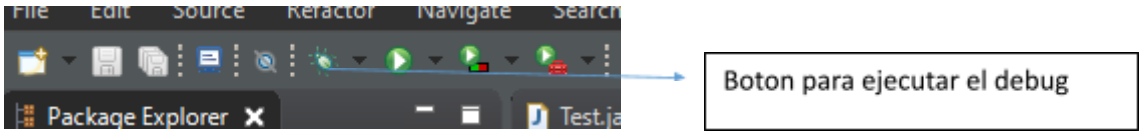


Varios breakpoint en un mismo código

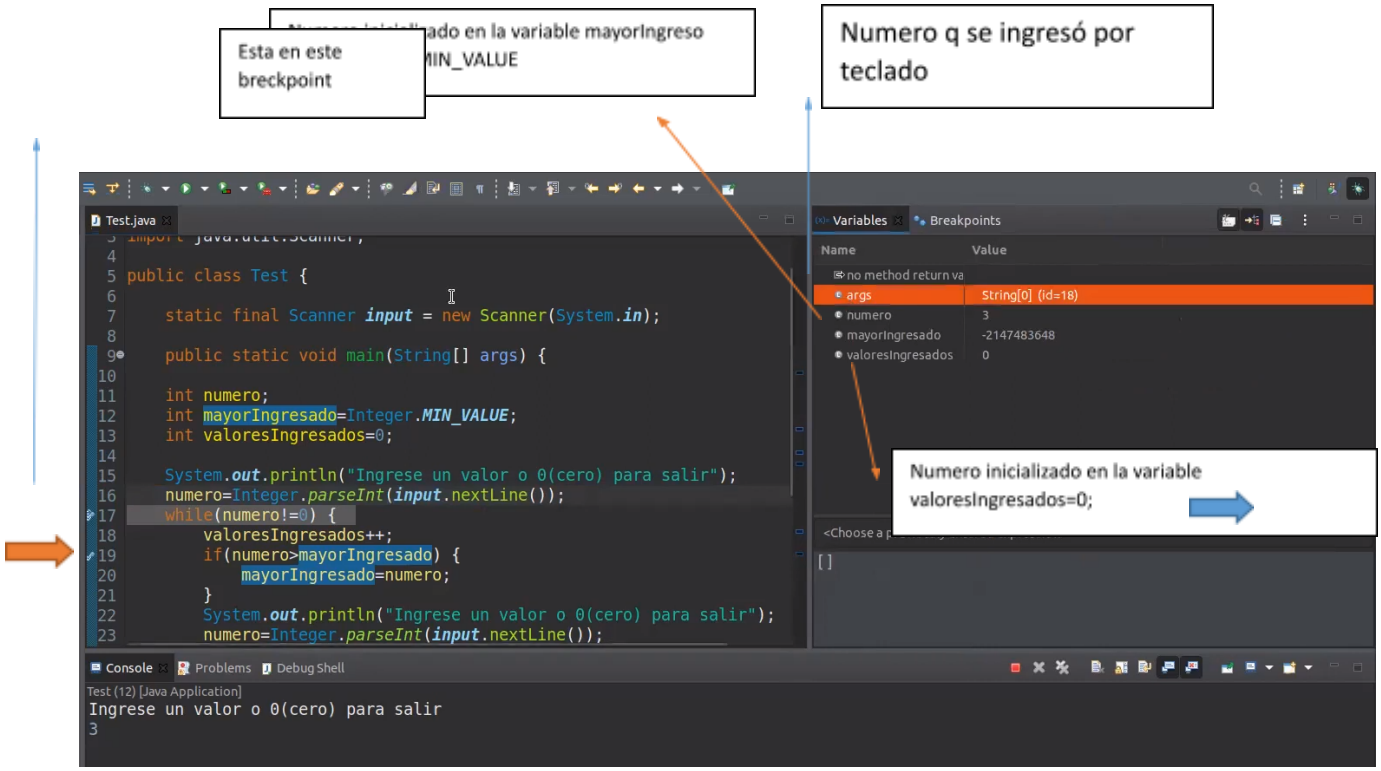
```java
while(numero!=0) {
    valoresIngresados++;
    if(numero>mayorIngresado) {
        mayorIngresado=numero;
    }
    System.out.println("Ingrese un valor o 0(cero) para salir");
    numero=Integer.parseInt(input.nextLine());
}

if(valoresIngresados>0) {
    System.out.println("Se ingresaron " + valoresIngresados + " datos ");
    System.out.println("El mayor valor fue: " + mayorIngresado);
} else {
    System.out.println("No ingreso valores");
}
```

Para activar o ejecutar el método Debug



Corriendo el debugger



Cuando le doy a continuar ,  o F8 , pasa al siguiente breakpoint , que es la flecha naranja (if) , en ese momento ya paso por la variable valor ingresado y pasara de 0 a 1 (por q ya ingresamos algo)

ATRIBUTOS DE LA CLASE (STATIC)

Una variable estática (**static**) en **Java** es una variable que pertenece a la clase en que fue declarada y se inicializa solo una vez al inicio de la ejecución del programa, la

característica principal de este tipo de variables es que se puede acceder directamente con el nombre de la clase sin necesidad de crear un objeto.

```
//atributos de instancia(los que tendran los objetos creados
// a partir de esta clase
private int nroFactura;
private int codCliente;
private double importe;

// atributos de la clase(static)
private static double alicIva;
private static double alicIvaReducida;
private static int ultimaFactGenerada;

Factura(int codCliente, double importe) {
    this.nroFactura = Factura.ultimaFactGenerada+1;
    this.setCodCliente(codCliente);
    this.importe = importe;
    Factura.ultimaFactGenerada = this.nroFactura;

}
```

Podemos tener métodos staticos propios de la clase
Estos métodos son solos de la clase. No de la instancia .

```
public static void actualizarIva(double iva) {
    Factura.alicIva = iva;
}

public static void actualizarIvaReducido(double iva) {
    Factura.alicIvaReducida = iva;
}
```

En el método main , asi le damos valores a los atributos de la clase

```
// actualizo atributo de la clase
Factura.actualizarIva(21);
Factura.actualizarIvaReducido(10.5);
```

ArrayList(Colecciones de Objetos)

En Java, la forma más simple de trabajar con colecciones de objetos es utilizando la clase ***ArrayList*** que permite guardar en forma dinámica y secuencial un elemento detrás del otro, acceder a los mismos y eliminarlos cuando lo necesitemos.

Algunos de los métodos de los arraylist son

- **add**(Elemento) = se utiliza para agregar un elemento al final de la colección
- add(n, elemento) = Permite agregar un elemento en la posición señalada por n. Si esta posición está fuera del rango aceptable (determinado por la cantidad de elementos de la colección) se producirá un error de ejecución.
- **get** (n) = Permite obtener el elemento n de la colección
- **remove** (n) = permite eliminar un elemento de la colección
- remove(elemento) = Permite extraer el elemento recibido de la colección. A diferencia del otro remove que recibe la posición del elemento a extraer, como ya tiene el elemento devuelve verdadero o falso dependiendo de si se pudo o no extraer el elemento (devuelve falso cuando el elemento indicado no estaba insertado en la colección, cuando era ajeno a ella).
- **size** ()= nos devuelve el largo del array
- **isEmpty()** = Devuelve un valor booleano que indica si la colección está vacía. Es equivalente a preguntar si count() == 0.
- **contains()** = Devuelve true si esta lista sobre la que se invoca contiene todos los elementos de la colección pasada por parámetro. se busca el objeto en sí dentro del arraylist

Para utiliza arraylist lo primero que hay q hacer es importar la clase arraylist , esta se hace con el método main

```
package clase20210519;

import java.util.ArrayList;

public class TestArraylist {

    public static void main(String[] args) {

        ArrayList<String> textos = new ArrayList<String>();
```

Definimos un objeto de tipo arrayslist que dentro de el contendrá tipos String , le declaramos una variable (texto) y invocamos al constructor de ese método

Agregamos elementos al array con el método add() , se coloca entre los parentecis

```
textos.add("Hola");
textos.add("Mundo");
```

```
System.out.println(textos.get(0) + " " + textos.get(1));
```

Imprimimos por pantalla con el método get() , el cual nos trae el elemento según el numero de parámetro que le coloquemos

```
ArrayList<Integer> enteros = new ArrayList<Integer>();
```

Aca son de tipos enteros , va Interger no int

Recorrer Arraylist

Con for común

```
for (int i = 0; i < enteros.size(); i++) {

    System.out.println("Posicion " + i + " valor " + enteros.get(i));
}
```

Con el método for podemos recorrer el arraylist usando el método size para definir hasta donde vamos a ir , esto tiene q ser siempre < que el size

Con for each

```
//recorro el array con for each
System.out.println(" \nRecoro el array con for each");
for ( int valorDelArray:enteros) {
    System.out.println("valor " + valorDelArray);
}
```

Defino la variable q va a guardar lo q lee de ese alrrays y luego se coloca el array que tiene esos objetos que yo quiero ver

Remover elemento de la Arraylist

```
// borro elemento del arrays con .remove(posicion) o .remove(objeto)
System.out.println(" \nBorro el 3er elemento de la lista");
enteros.remove(3);
```

Con el método remove , podemos borrar un objeto de la colección , por parámetro le podemos pasar un numero en donde se encuentra ese objeto a borrar o podemos borrarlo pasando ese objeto .

Para borrar todos los elementos uso el

```
System.out.println(" \nBorro todos los elemento de la lista");
enteros.clear();
```

Ejemplos de Arraylist

```
private ArrayList<Persona> listasPersonas; // defino atributo tipo arraylist
```

Persona (es el tipo de clase)

Nombre de la variable a utilizar

Inicializar una nueva array lista de la clase persona vacía

```
new ArrayList<Persona>();
```

Persona (es el tipo de clase)

```
laCadena.isEmpty()
```

Se usa para saber si un string esta vacio