# Java 3

## Ejercicio 17

### Crea un array de enteros y otro array de strings y rellénalo de datos.

int[] aEnteros={5,6,7,8,6,9,4,2,8,9};

String[] aString={"En","un","lugar","de","la","mancha","de","cuyo","nombre","no","quiero","acordarme"};

### Saca una posición aleatoria del array de strings y muestra el resultado por pantalla. Hazlo 10 veces.

public static void sacarPosN(String[] cadenas, int veces ) {

String txt="";

for(int i=0;i<veces;i++) {

int pos= (int)(Math.random()\*(cadenas.length-2));

if(i<=(veces-1)) {

txt+=cadenas[pos]+" ";

}else{

txt+=cadenas[pos]+".";

}

}

System.out.println(txt);

}

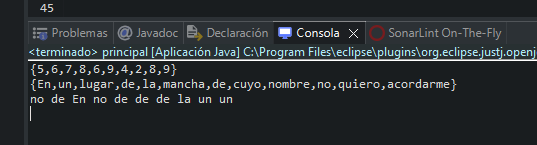


Ilustración 1 array de string aleatorio

### Muestra hasta una determinada posición, los valores guardados en el array de enteros :

public static void impArray(int[] enteros,int pos) {

String txt="{";

if(pos<=enteros.length) {

for(int i=0;i<pos;i++) {

if(i!=pos-1) {

txt+=enteros[i]+",";

}else {

txt+=enteros[i]+"}";

}

}

}else {

txt="Posicion erronea, fuera del array, debe ser menor de "+enteros.length;

}

System.out.println(txt);

}

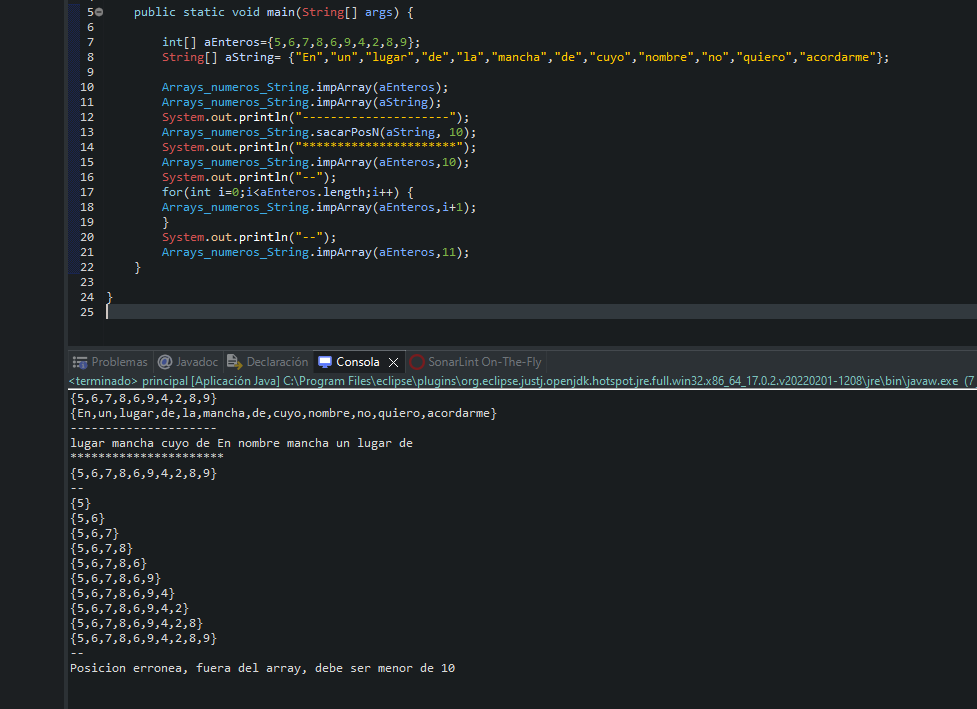


Ilustración 2 ejecución del programa

## Ejercicio 18 Crea un array de enteros y cuenta cuantas veces se repite un determinado valor.

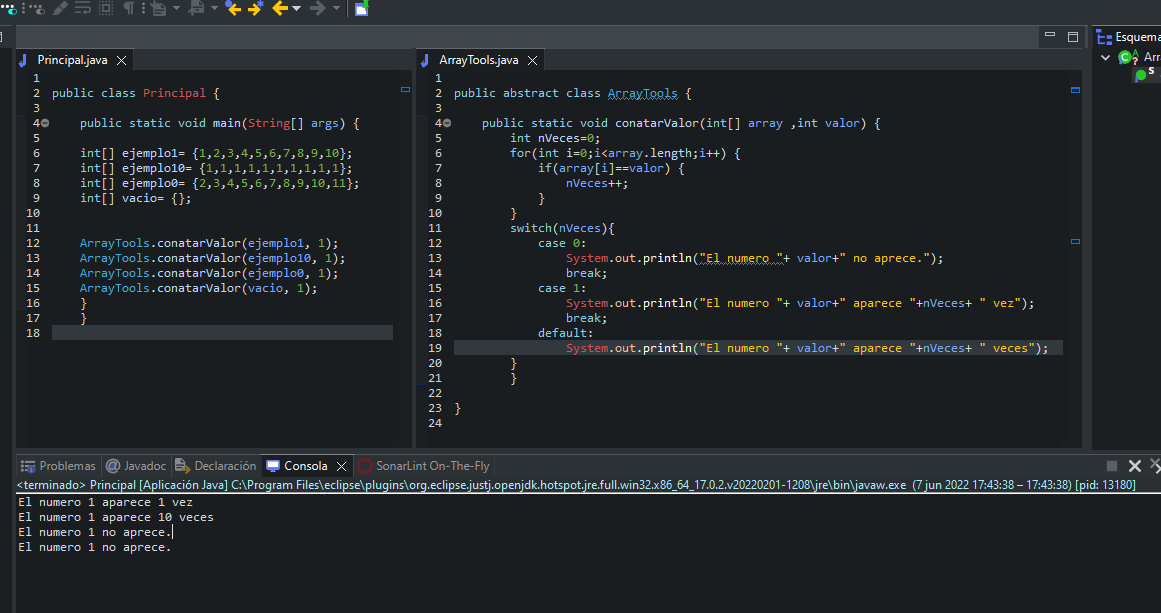


Ilustración 3 contar veces que aparece un número

## Ejercicio 19 Arrays multidimensionales

### Crea un array bidimensional y recórrelo mostrando los datos por pantalla.

public abstract class ArrayDimensiones {

public static void imprimirArray(int[][] a2D) {

String txt="{";

for(int i=0 ;i<a2D.length;i++) {

for(int j=0;j<a2D[i].length;j++) {

if(j==0) {

txt+="{"+a2D[i][j]+",";;

}else if(j==a2D[i].length-1) {

txt+=a2D[i][j]+"}";

}else {

txt+=a2D[i][j]+",";

}

}

if(i<a2D.length-1) {

txt+=",";

}else{

txt+="}";

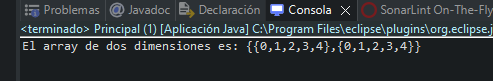
}

}

System.out.println("El array de dos dimensiones es: "+txt);

}

}



### Crea un array tridimensional de enteros y recórrelo sumando todos sus valores almacenados

public static void imprimirArray(int[][][] a3D) {

String txt="{\n";

int sum=0;

for(int i=0 ;i<a3D.length;i++) {

txt+=" {\n";

for(int j=0 ;j<a3D[i].length;j++) {

for(int k=0;k<a3D[i][j].length;k++){

sum+=a3D[i][j][k];

if(k==0) {

txt+=" {"+a3D[i][j][k]+",";;

}else if(k==a3D[i][j].length-1) {

txt+=a3D[i][j][k]+"}";

}else {

txt+=a3D[i][j][k]+",";

}

}

if(j<a3D[i].length-1) {

txt+=",\n";

}else{

txt+="}\n";

}

}

txt+="}";

}

System.out.println(txt+"=> Donde la Suma es :"+sum);;//30+35+108x2=281

}

}

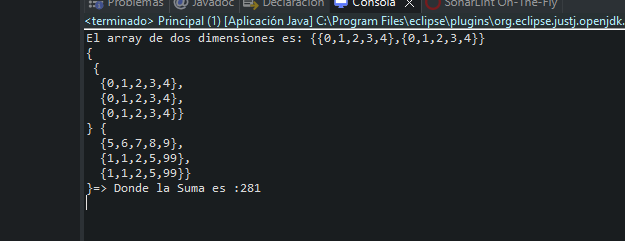


Ilustración 4 array tridimensional

## Ejercicio 20 Listas.

### Crea una lista de personas (la clase que creaste en el ejercicio 14) y muestra por pantalla los detalles de cada una de ellas .

Las dos listas más usadas son [ArrayList](https://www.w3schools.com/java/java_arraylist.asp) y LinkedList

Import java.util.List; Import java.util.ArrayList; Import java.util.LinkedList;

#### Java ArrayList.

La ArrayList clase es una matriz de tamaño variable, que se puede encontrar en el paquete java.util. La diferencia entre una matriz o array integrado es que el tamaño de este no se puede modificar (si se desea crear o eliminar elementos de una matriz esta se debe eliminar y crear una nueva) .En un ArrayList los elementos se puden agragar y eliminar en cualquier momento.La sintaxis también es ligeramente diferente:

List <tipoDato> nombrelista=new ArrayList<tipoDato>();

Los elementos de un ArrayList son realmente Objetos, todos los ejemplos los vemos con String pues es un objeto no un tipo primitivo. Para usar tipos primitivos debes especificar la clase contenedora equivalente: Integer,Boolean,Character,Double etc..

##### Añadir Elementos Método Add:

**Array**List<String> palabras=new ArrayList<String>();//Creaccion

palabras.add("Hola");

palabras.add("buenos");

palabras.add("dias");

##### Acceder a Elementos Método get(pos):

nombreLista.add(pos);

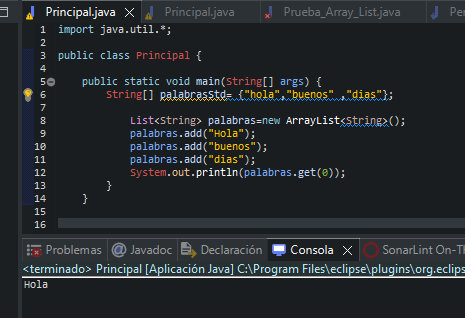


Ilustración 5 acceder a Elementos

##### Cambiar Elementos Método Set(pos,valor):

Para modificar un elemento, se utiliza el método set y la posición a modificar

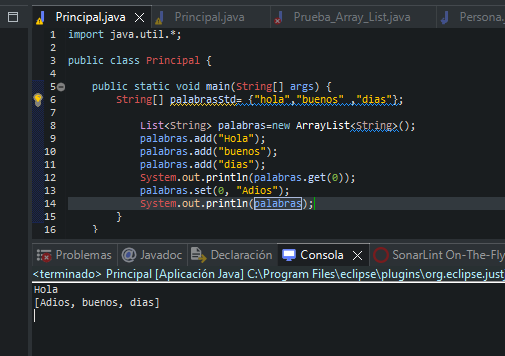


Ilustración 6 cambiar valor

##### Quitar Elementos 0 borrar todo:

Para eliminar un elemento, utilizamos el método remove(indice) y el número de índice .Da error si sales de la longitud.Reposiciona los índices de los elementos restantes.

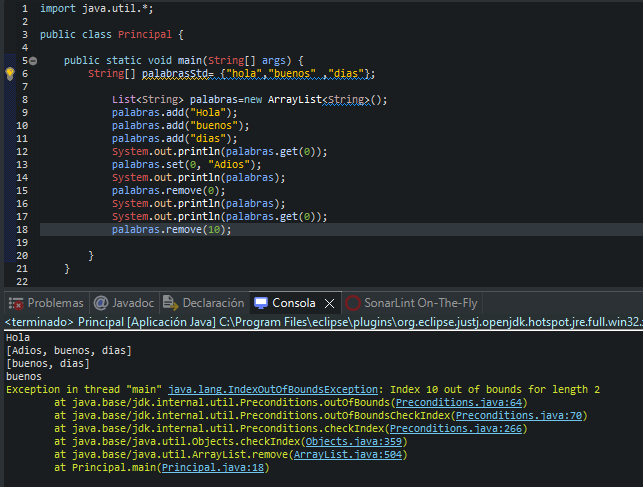


Ilustración 7 nombreArrayList.remove(pos)

Para eliminar todos los elementos del ArryList se utiliza el método clear().

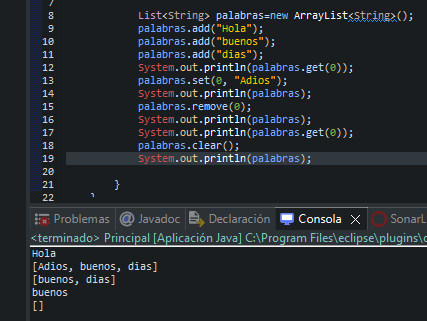


Ilustración 8 nombreArrayList.clear()

##### Tamaño del ArrayList.

Utilizamos el método nombreArrayList.size();

##### Bucles para recorrer ArrayList.For y Foreach

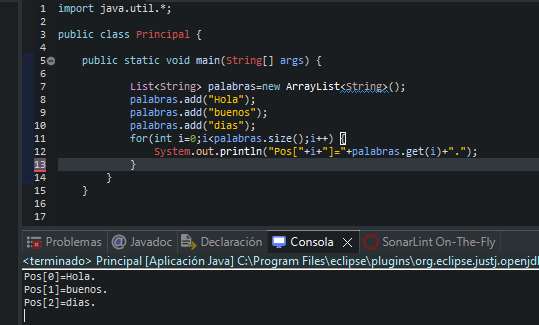


Ilustración 9 recorrer ArrayList Bucle For

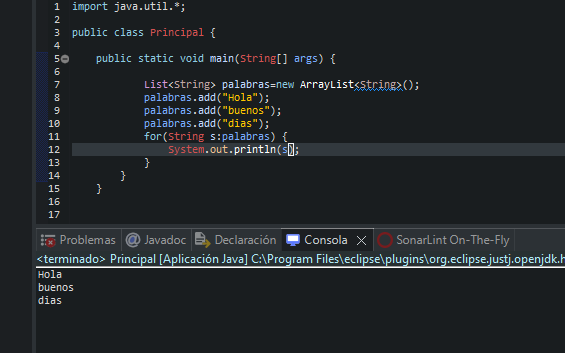


Ilustración 10 bucle forEach

##### Ordenar un ArrayList.

Otra clase útil en java.útil es la clase Collections , que incluye sort() método que ordena listas alfabéticamente o numéricamente:

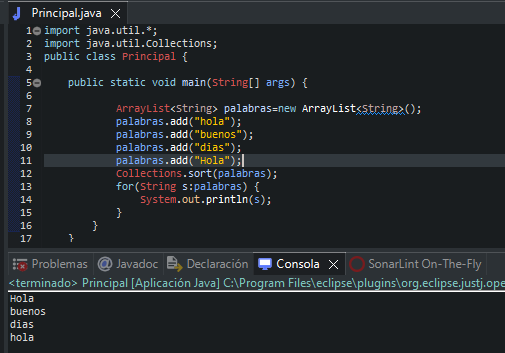


Ilustración 11 Ordenar Collections.sort()

Es sensible a Mayúsculas. Fíjate en el Hola.

Ordenamos un ArrayList de Enteros mediante la clase contenedora Integer y sort de Collection.

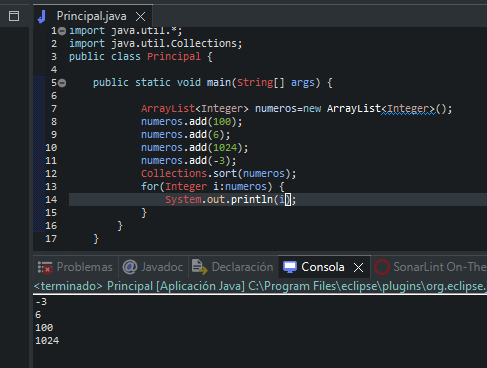


Ilustración 12 sort de una clase Integer

#### Java LinkedList.

Puede contener muchos objetos del mismo tipo como el ArrayList. La LinkedList tiene todos los mismos métodos que la clase ArrayList porque ambos métodos implementan la interfaz List .Esto quiere decir que todo lo anterior es válido, borrar, acceder etc...

Se puede usar de la misma manera pero se construyen de manera muy diferente.

**Como funciona ArrayList.**

La clase ArrayList tiene una matriz regular dentro de ella. Cuando se agrega un elemento, se coloca en la matriz .Si la matriz no es lo suficientemente grande, se crea una nueva matriz más grande para remplazar la anterior y se elimina la anterior.

**Como funciona LinkedList.**

El LinkedList almacena sus artículos en “contenedores”. La lista tiene un enlace al primer contendor y cada contenedor tiene un enlace al siguiente contendor de la lista .Para agregar u elemento a la lista, el elemento se coloca en un nuevo contenedor y ese contendor se vincula a uno de los contendores de la lista.

Como regla Use ArrayList para almacenar y acceder a datos, y LinkedList para manipular datos.

Los métodos de LinkedList:

* addFirst()
* addLst()
* removeFirst();
* getFirst();
* getLast();

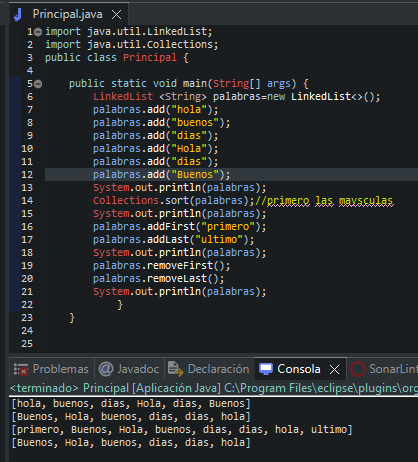
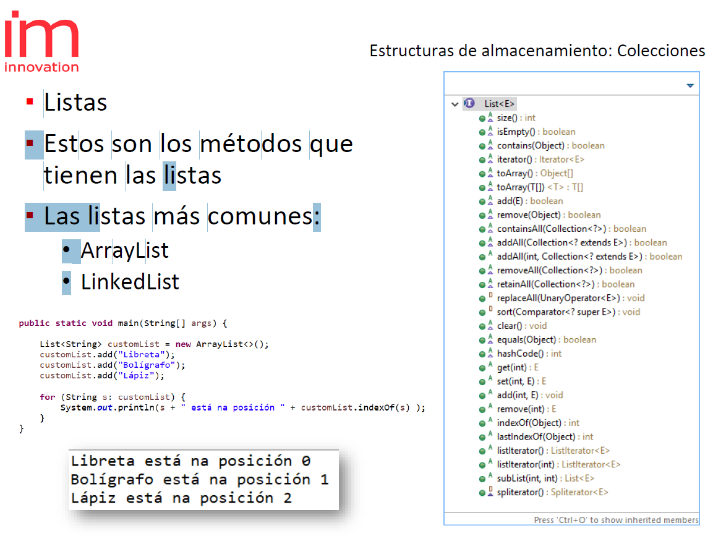


Ilustración 13 métodos LinkedList.



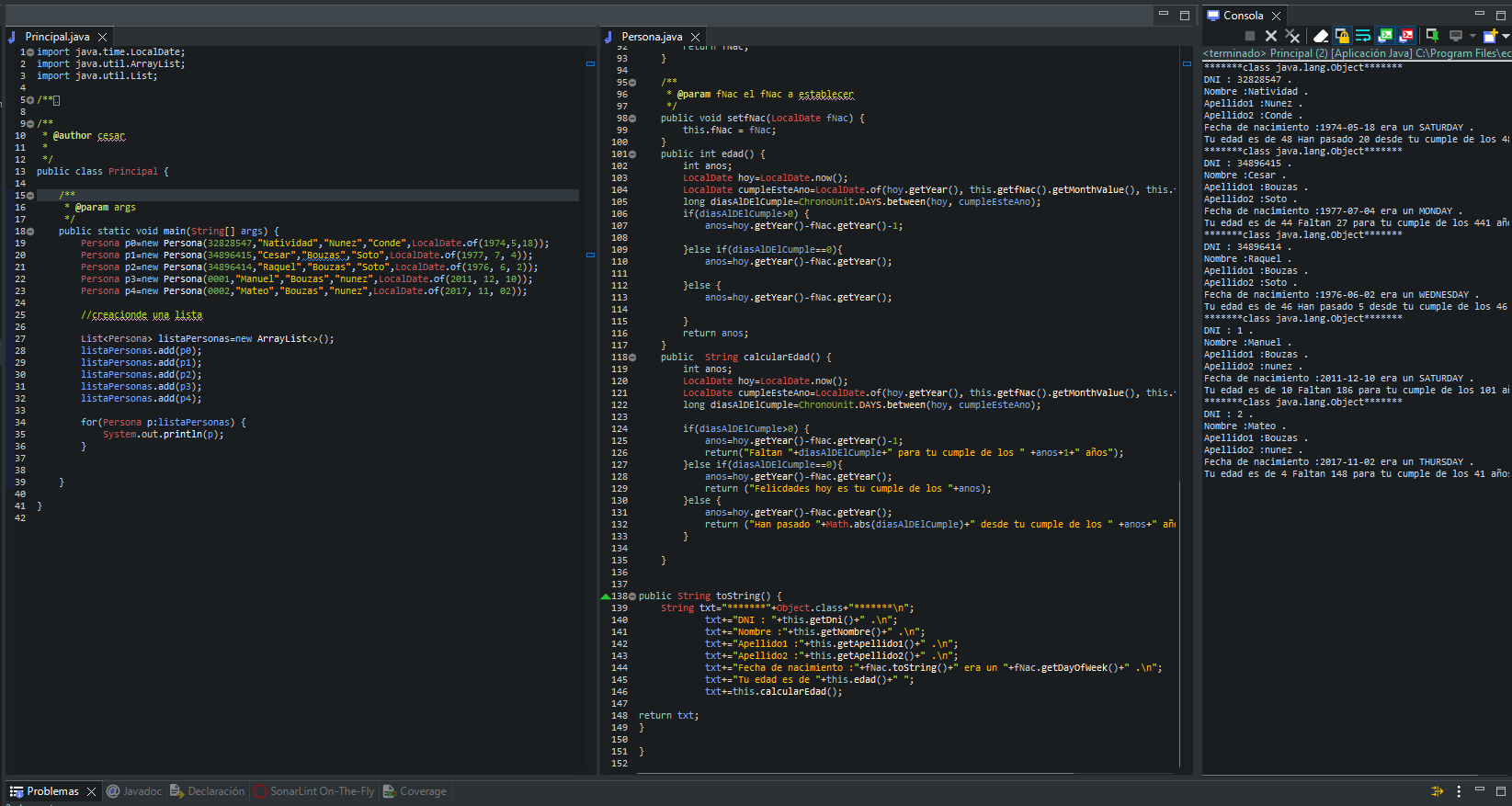


Ilustración 14 ArrayList personas

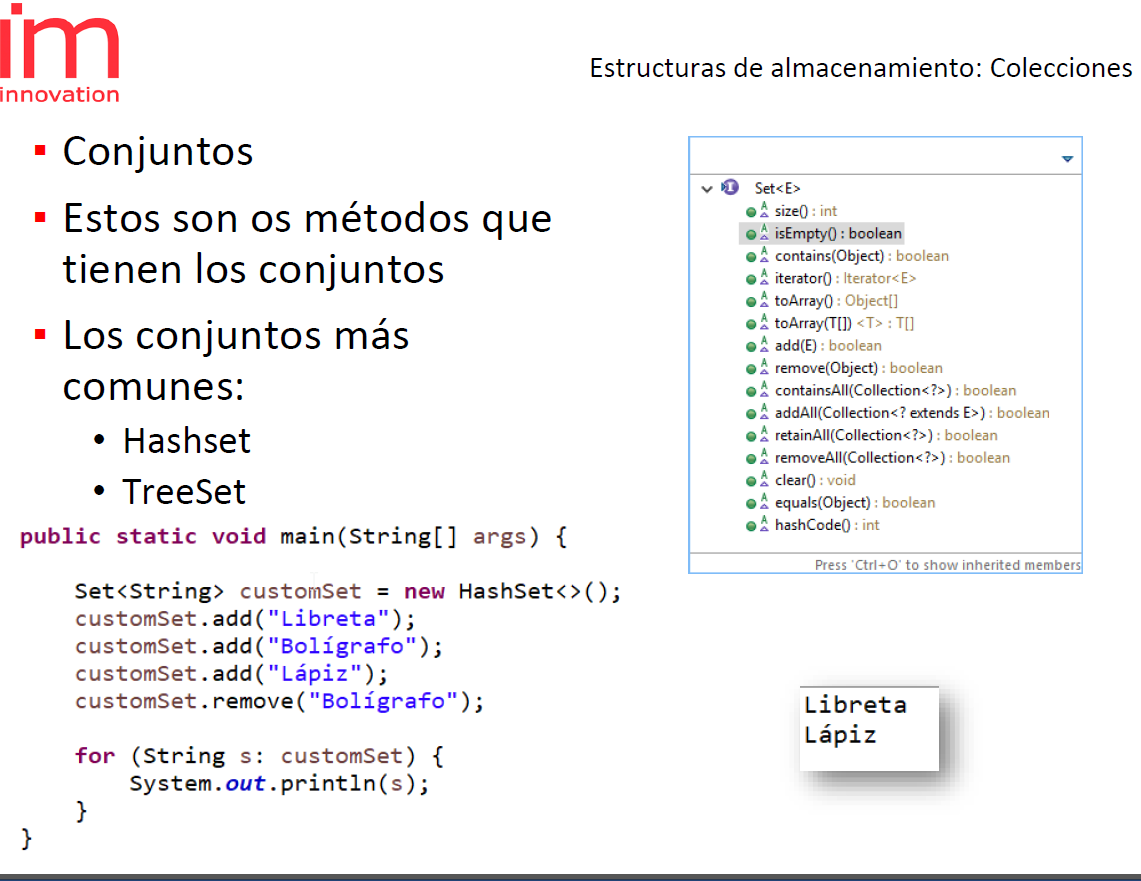
## Ejercicio 21:

### Crea un conjunto de strings (añade y elimina los que quieras), mete duplicados y muestra por pantalla lo que te queda del conjunto.

Los conjuntos no admiten como las listas elementos repetidos. Los conjuntos más comunes son Hashset y TreeSet.

La interfaz Set define una colección que no puede contener los elementos duplicados. Se encarga de gestionar conjuntos. Esta interfaz contiene, únicamente los métodos heredados de Collection añadiendo la restricción del almacenamiento de elementos duplicados.

HasSet



#### Conjuntos HashSet.

Un HashSet es una colección de elementos donde cada elemento es único y se encuentra en java.util.

* Almacena los elementos en una tabla hash
* No garantiza ningún orden a la hora de realizar iteraciones
* Se debe definir el tamaño inicial.
* Es la implentacion de Set con mejor rendimiento.

##### Creación de un objeto

HashSet<String> nombres=new HashSet <String>();

##### Añadir elementos add().

Los HashSet tienen muchos métodos y de ellos es add() para agregar elementos a la colección.

##### Remover Elemento remove(elemento).

Para eliminar use remove() , por ejemplo palabras.remove(“cesar”);

##### Eliminar todas los elementos método clear().

palabras.remove()

##### Tamaño del conjunto de HashSet. Método size();

Palabras.size();

##### Recorrer un HashSet.

for(String s:palabras){

System.out.println(s);

}

##### Otros tipos .

Como en el caso anterior los tipos primitivos deben utilizar la clase contenedora ,Integer, Boolean,Character ,Double etc..

##### Realizo el ejercicio.

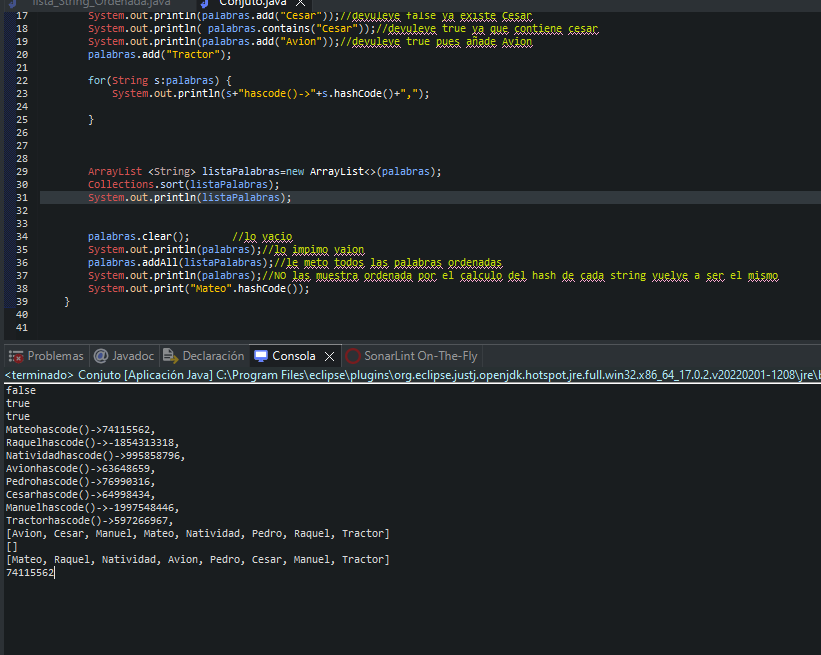


Ilustración 15 uso de HashSet e intento de ordenación

NO LO ORDENA!!! Fila 22 a 28. Por el cálculo del hashcode del stream siempre es el mismo.

#### TreeSet.

* Almacena los elementos ordenándolos en función de sus valores.
* Los elementos almacenados deben implementar la interfaz comparable
* Es la implementación con un rendimiento más lento.

## Ejercicio 22: Crea una lista de strings y muestra por pantalla el elemento y la posición en la que se encuentra.

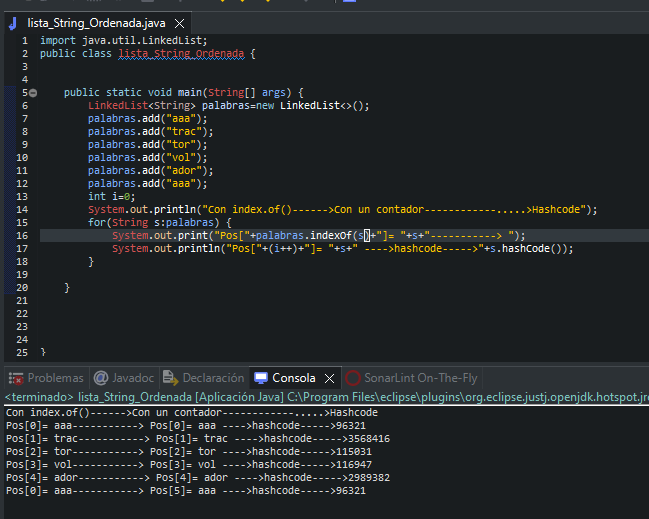


Ilustración 16 lista de string con contador

POSICION OTRA MANERA sin contador????

## Ejercicio 23: Crear una dupla clave-valor de Personas (la clase que creaste en el ejercicio 14), añade y quita elementos y muestra por pantalla los elementos que queden.

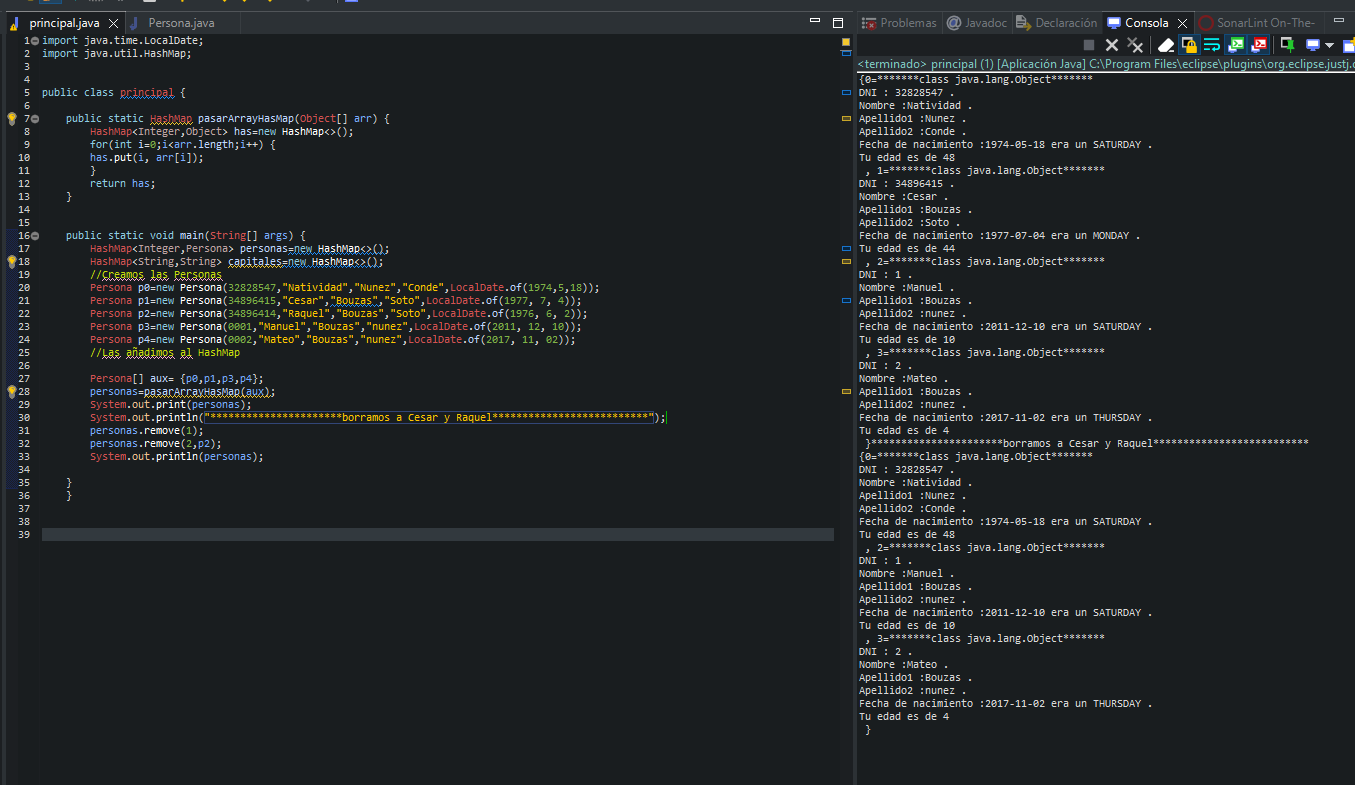
La clase ma permite definir colecciones d elementos que posee pares de datos Clave-Valor.Cada Elemento tiene asociado una clave y un valor. La clave es utilizada para acceder de forma muy rápida a un elemento.

Java permite que la clave sea cualquier tipo de objeto pero se sule utilizar Integer o String.

Los mapas no permiten insertar objetos nulos ya que povocarían excepciones de tipo NullPointerException

### Declaración.

HashMap<Integer,Persona> personas=new HashMap<>();



### Métodos destacados del HasHMap:

* Int size();
* Boolean isEmpty();
* Void put(k,v) :inserta el elemento v con la clave k;
* V get(K clave):Devuelve el valor de la clave que se pasa como parámetro , null si la clave no existe.
* V remove(k clave):borrar el par clave valor;
* Boolean containsKey(k clave):Devuelve true si existe la clave en el map
* Boolean containsValue(v valor): Devuelve true si es map hay un Valor V;
* Collection<v>values: Devuelve una Collection con los valores del objeto Map.
* Set<k> keySet():Devuelve un set con las claves del objeto Map.

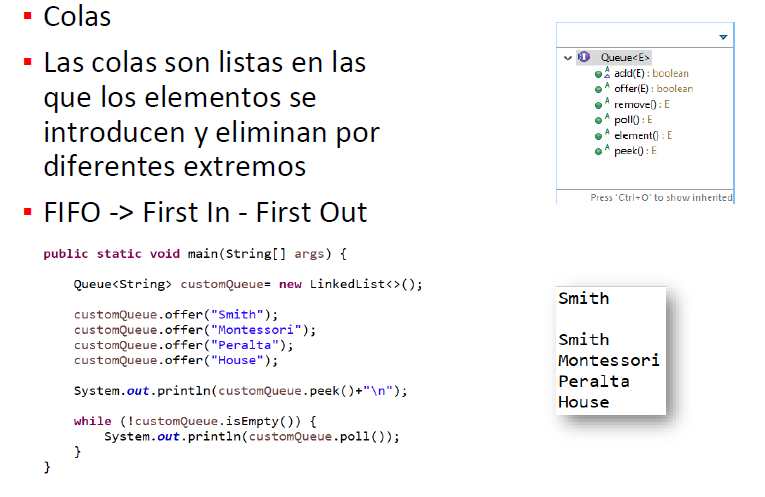
### Condiciones del Map(k,v)

El conjunto de claves no puede repetir la clave es decir una estructura de tipo Set por lo que para detectar si la clave esta repetida deben definir correctamente hashCode() y equals()

## Ejercicio 24: Colas.

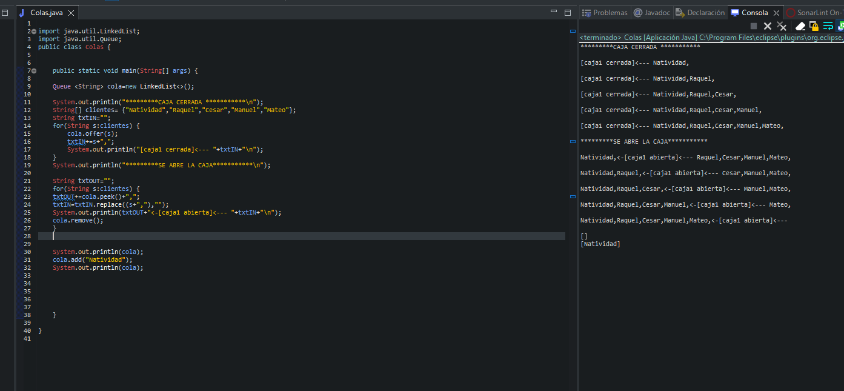
Las colas son listas en la que los elementos se introduce y eliminan por diferentes extremos .

### FIFO->Cola



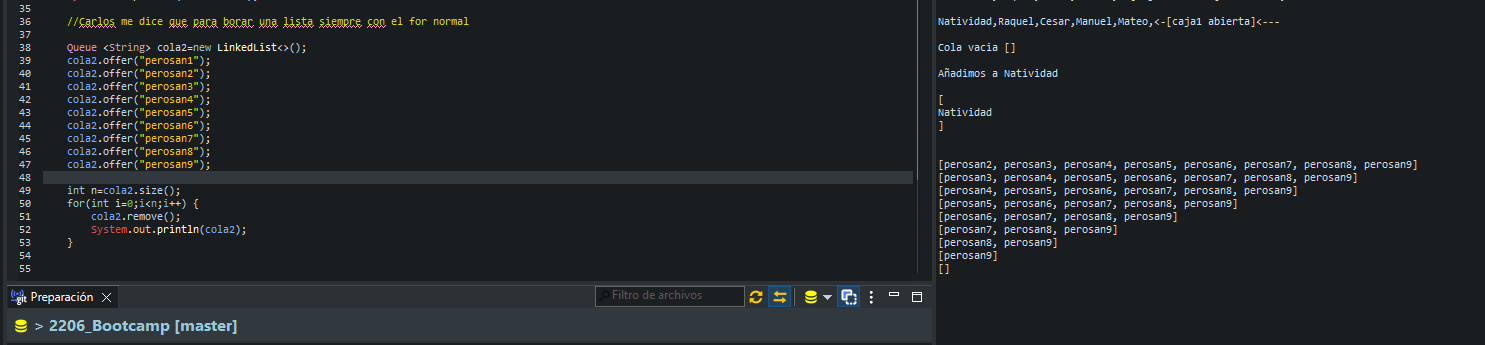
* Add() añade un elemento a la lista (posibles errores si tenemos limitación de tamaño)
* Offer() añade un elemento al final de la lista si es posible , preferible al uso de add.
* Remove() Borrar el primer elemento de la cola, que lleva mas tiempo esperando.

### Crea una cola, añade/quita elementos y muestra lo que quede por pantalla .



### Recorre la cola y elimina todos sus elementos.

Mejor recorrer con el for normal, el foreach se vuelve loco.



Se puede hacer sin sacar fuera el size() con un for???

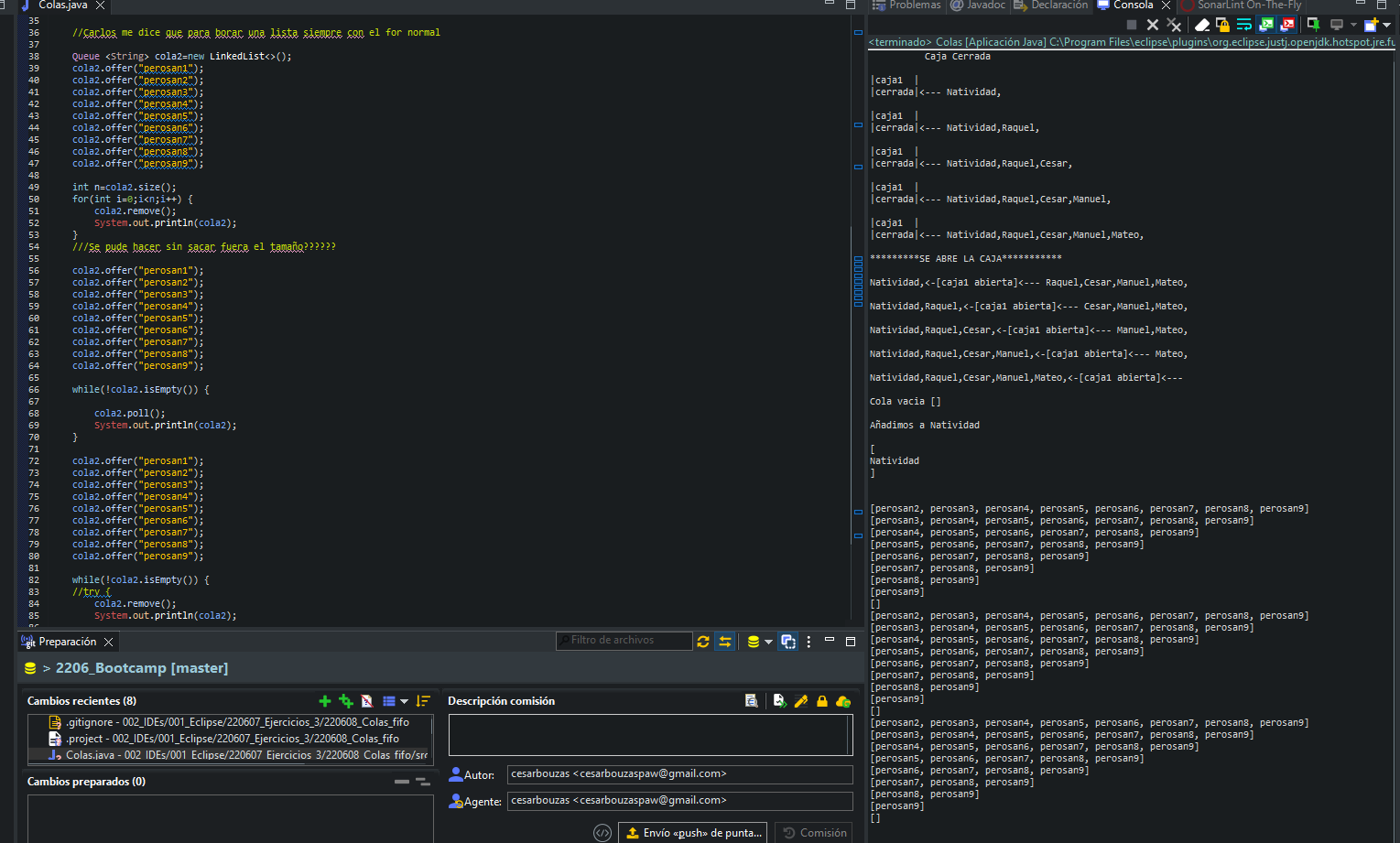


Ilustración 17 modos de borrar una cola con while

## Ejercicio 25: Pilas

### Crea una pila, añade/quita elementos y muestra lo que quede por pantalla .

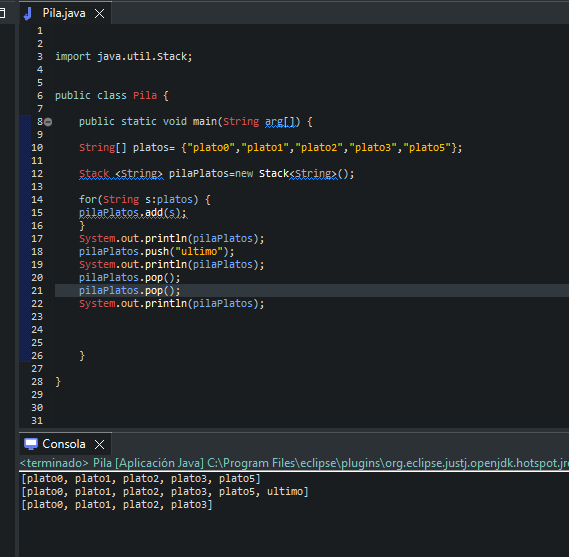
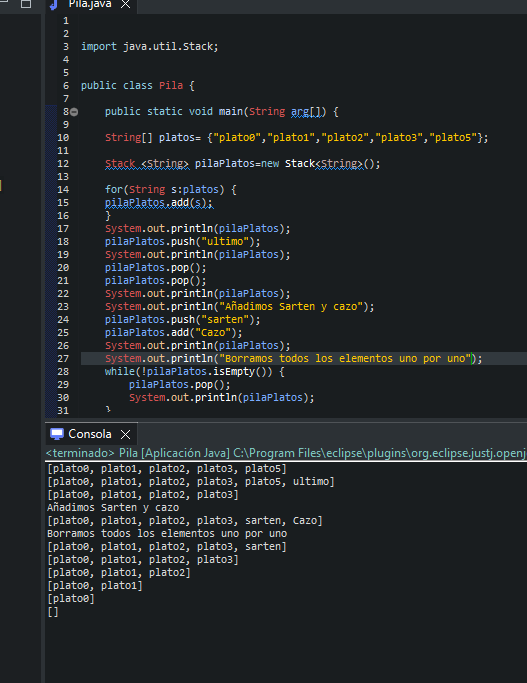


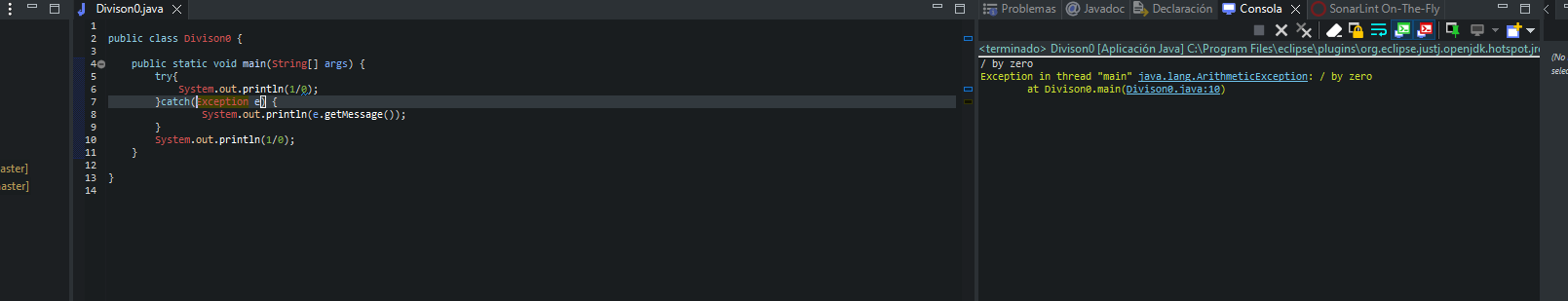
Ilustración Uso de Pila

### Recorre la pila y elimina todos sus elementos.

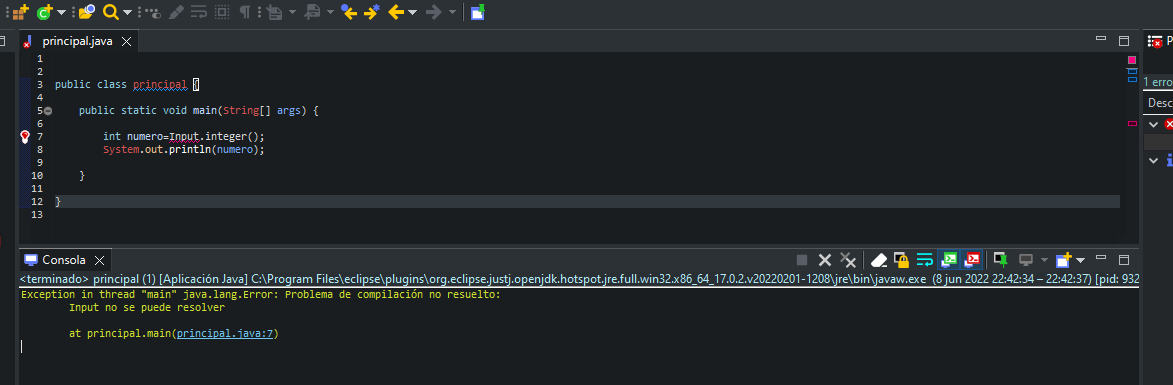


## Ejercicio 26  Crea un xml de los componentes de un coche, indicando la cantidad de cada uno de ellos  Crea un json de los componentes de un coche, indicando la cantidad de cada uno de ellos Ejercicio

## Intenta hacer una división por 0 y mira lo que pasa .



## Solicita por pantalla un número (con la clase Input) y mira qué pasa si lo que introduces es un string.



No funciona el ejemplo , Clase Input??? De deonde la saco??