

COLECCIONES DINAMICAS

Las colecciones dinámicas son estructuras de datos que pueden cambiar de tamaño y permiten la insercción y eliminación de elementos de manera flexible.

CARACTERISTICAS:

- 1.-TAMAÑO VARIABLE :
- 2.- ESTRUCTURAS DE DATOS :
- 3.-ACCESO A ELEMENTOS:
- 4.-FLEXIBILIDAD:
- 5.-EFICIENCIA EN INSERCCIONES/ ELIMINACIONES:
- 6.-TIPOS DE DATOS:

DURANTE LA EJECUCION

- AGREGAR ELEMENTOS:
- Eliminar elementos
- Acceder elementos
- Recorrer colección
- Buscar elementos
- Ordenar
- Filtrar
- Combinar colecciones
- Clonar y copiar
- Transformaciones



TIPOS DE COLECCIONES

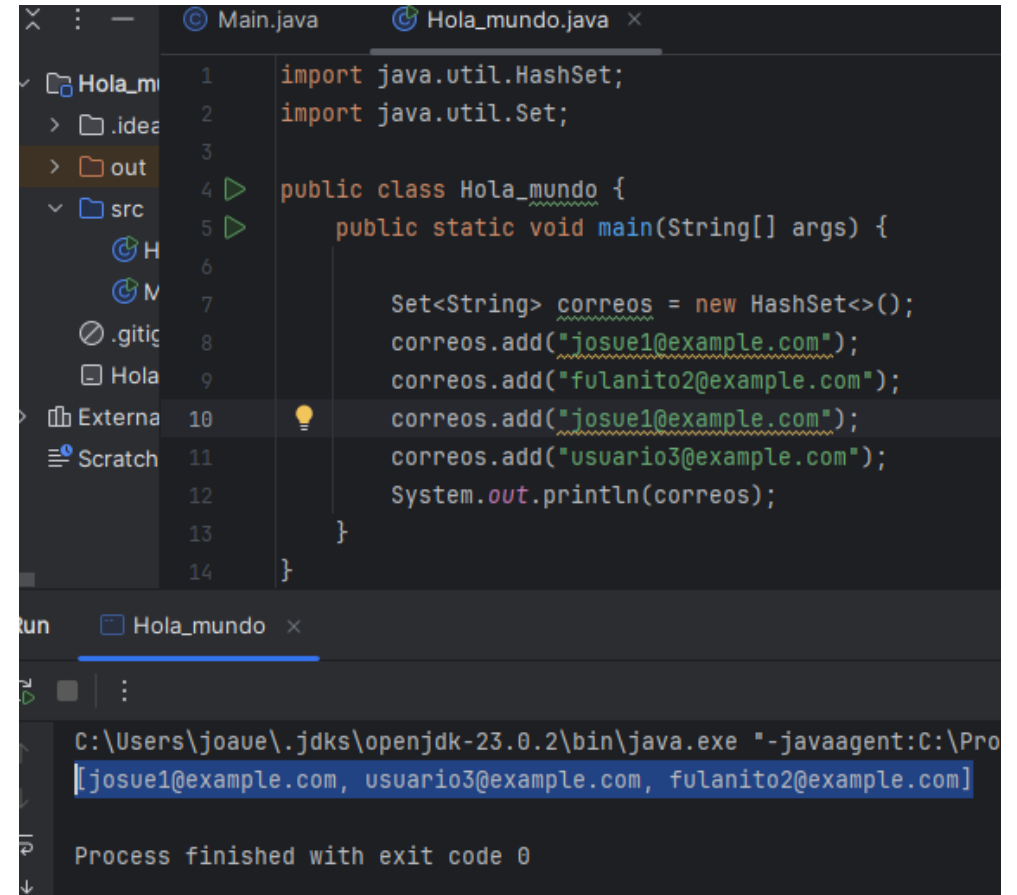
LISTAS =[]

- las listas son colecciones ordenadas , estas pueden contener elementos duplicados.
- Estas se pueden modificar lo que significa que puedes agregar u eliminar elementos después de que la lista ha sido creada.
- USO → es ideal para almacenar elementos en un orden específico y acceder a ellos
- Ejemplo de uso :

```
public static void main(String[] args) {  
  
    List<String> estudiantes = new ArrayList<>();  
    estudiantes.add("Ana");  
    estudiantes.add("Luis");  
    estudiantes.add("Pedro");  
    estudiantes.add("María");  
  
    estudiantes.add("Carlos");  
  
    System.out.println(estudiantes);  
}
```

CONJUNTOS(Set)

- Los conjuntos son colecciones no ordenadas de elementos únicos , esto significa que no pueden contener “DUPLICADOS” .
- Uso→ ideal para almacenar elementos cuando no se necesita mantener el orden y se quiere evitar DUPLICADOS
- Ejemplo de uso : almacenar email únicos



The screenshot shows an IDE with two tabs: 'Main.java' and 'Hola_mundo.java'. The 'Hola_mundo.java' tab is active, displaying the following Java code:

```
1 import java.util.HashSet;
2 import java.util.Set;
3
4 public class Hola_mundo {
5     public static void main(String[] args) {
6
7         Set<String> correos = new HashSet<>();
8         correos.add("josue1@example.com");
9         correos.add("fulanito2@example.com");
10        correos.add("josue1@example.com");
11        correos.add("usuario3@example.com");
12        System.out.println(correos);
13    }
14 }
```

Below the code editor, the 'run' tab is visible, showing the command executed and the output:

```
C:\Users\joave\.jdk\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Pro
[josue1@example.com, usuario3@example.com, fulanito2@example.com]
```

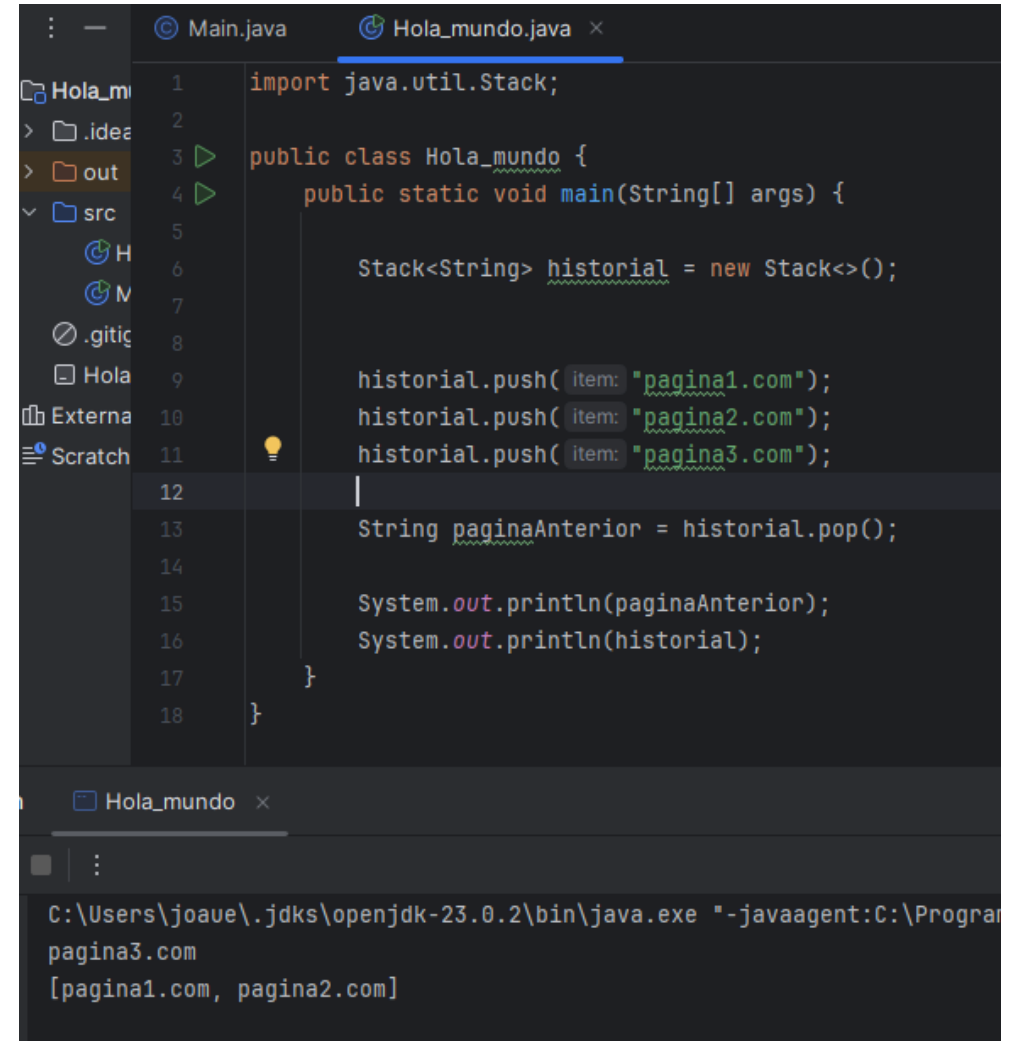
Process finished with exit code 0

Colas (Queue)

- Las colas (queue) son colecciones ordenadas que siguen el principio “FIFO” esto significa que el primer elemento en entrar es el primero en salir
- Uso → so útiles para manejar tareas que deben ser procesadas en el orden que llegaron como en sistemas de impresión
- Ejemplo:

Pilas (Stack)

- Las pilas son colecciones que siguen el principio LIFO es decir que el ultimo elemento en entrar es el primero en salir
- Uso→ son utilizadas en algoritmos que requieren un seguimiento del ultimo elemetno añadido como en la implementación de funciones recursivas o deshacer acciones en aplicaciones
- Ejemplo:



The screenshot shows an IDE with two tabs: 'Main.java' and 'Hola_mundo.java'. The 'Hola_mundo.java' tab is active, displaying the following Java code:

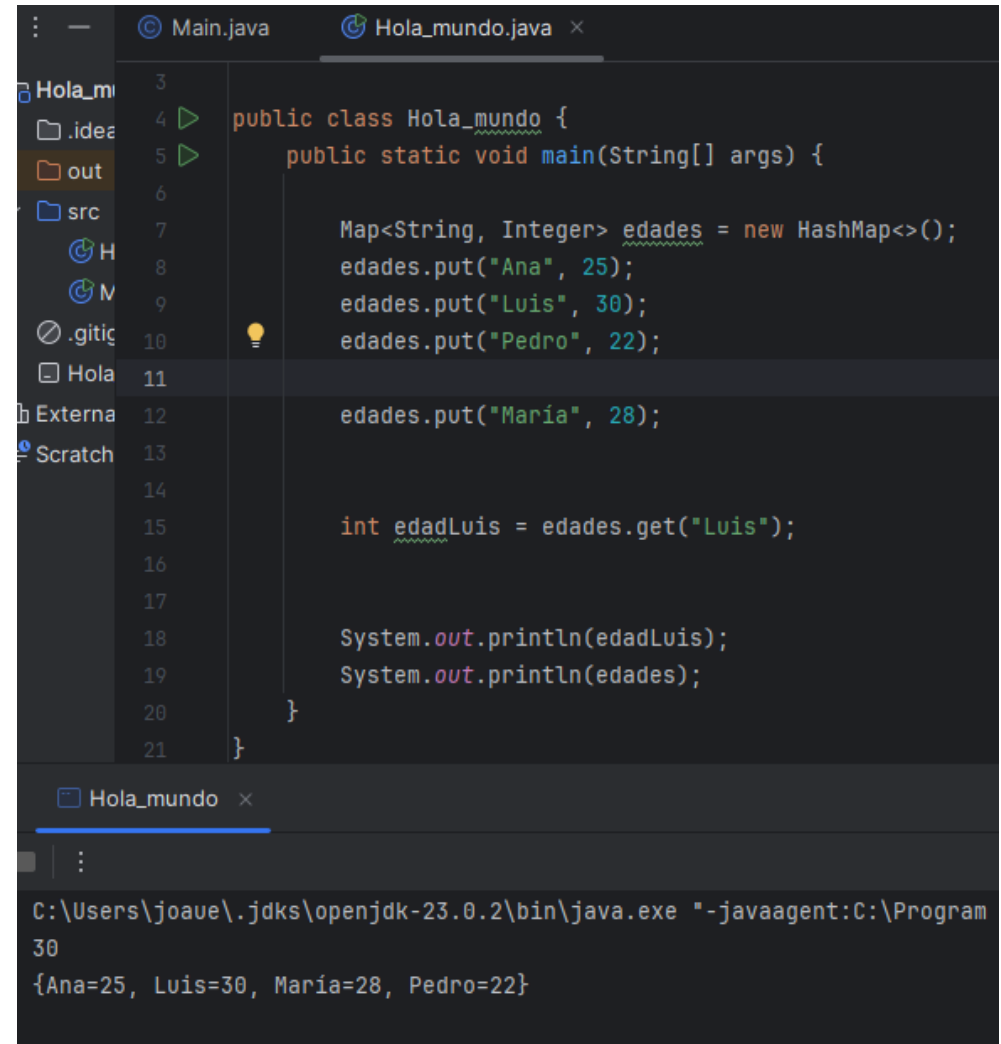
```
1 import java.util.Stack;
2
3 public class Hola_mundo {
4     public static void main(String[] args) {
5
6         Stack<String> historial = new Stack<>();
7
8         historial.push(item: "pagina1.com");
9         historial.push(item: "pagina2.com");
10        historial.push(item: "pagina3.com");
11
12
13        String paginaAnterior = historial.pop();
14
15        System.out.println(paginaAnterior);
16        System.out.println(historial);
17    }
18 }
```

Below the code editor, there is a console window titled 'Hola_mundo' showing the output of the program:

```
C:\Users\joave\.jdk\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program
pagina3.com
[pagina1.com, pagina2.com]
```


MAP(map)

- Los mapas son colecciones de pares clave-valor donde cada clave es única . Esto permite acceder rápidamente a un valor utilizando su clave
- Uso→ son muy útiles para almacenar datos que se pueden asociar con una clave ,como configuraciones ,datos de usuario o inventarios .
- Ejemplo:



The screenshot shows an IDE with two tabs: 'Main.java' and 'Hola_mundo.java'. The 'Hola_mundo.java' tab is active, displaying the following Java code:

```
3  
4 public class Hola_mundo {  
5     public static void main(String[] args) {  
6  
7         Map<String, Integer> edades = new HashMap<>();  
8         edades.put("Ana", 25);  
9         edades.put("Luis", 30);  
10        edades.put("Pedro", 22);  
11  
12        edades.put("María", 28);  
13  
14  
15        int edadLuis = edades.get("Luis");  
16  
17        System.out.println(edadLuis);  
18        System.out.println(edades);  
19    }  
20 }  
21 }
```

Below the code editor, the output of the program is shown in a terminal window:

```
C:\Users\joave\.jdk\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program  
30  
{Ana=25, Luis=30, María=28, Pedro=22}
```

¿Como recorrer la colección ?

“ITERATOR”

- List→ se usa Iterator , puedes recorrer los elementos de una lista uno por uno .
- Set→ se usa Iterator, puede recorrer los elementos de un conjunto sin duplicados
- Colas queue→ se usa iterator, este permite recorrer los elementos de una cola en el orden en que fueron añadidos
- Pilas→ se usa iterator , permite acceder a los elementos de una pila desde el fondo hasta la cima
- Mapas→ se usa iterator , este te permite recorrer las entradas (pares clave- valor) de un mapa

OPERACIONES: colecciones

- Add
- Remove
- Get

Listas (list)

- add → agrega un elemento al final o en un lugar específico
- Remove → elimina un elemento por índice o valor
- Get → obtiene el elemento en un índice

Conjuntos (set)

- Add→ agrega un elemento único (no permite duplicado)
- Remove→ elimina un elemento
- Get→ no existe (no accede por índice)

Colas (queue)

- Add → agrega un elemento al final
- Remove → elimina el primer elemento (el que ha estado mas tiempo)
- Get → no existe (a traves de otros métodos)

Pilas (stack)

- Add → (push) agrega un elemento a la cima
- Remove → (pop) elimina el elemento de la cima
- Get → no existe (accede mediante pop)

Mapas (map)

- Add → (put) agrega un par clave-valor , actualiza si la clave existe
- Remove → elimina un par por clave
- Get → obtiene el valor asociado a una clave

Colecciones avanzadas

- Las **colecciones avanzadas** en Java se refieren a estructuras de datos más complejas que van más allá de las colecciones básicas (como listas, conjuntos y mapas) que se encuentran en la interfaz de colecciones estándar. Estas colecciones están diseñadas para resolver problemas específicos y optimizar ciertas operaciones



- **Ejemplos de Colecciones Avanzadas**

- 1. ConcurrentHashMap:**

- 1. **Uso:** Mapa que permite el acceso concurrente desde múltiples hilos.
 - 2. **Ventaja:** Mejora el rendimiento en aplicaciones multihilo.

- 2. CopyOnWriteArrayList:**

- 1. **Uso:** Lista que crea una copia del array interno en cada modificación.
 - 2. **Ventaja:** Ideal para escenarios de lectura frecuente y modificación poco frecuente.

- 3. BlockingQueue:**

- 1. **Uso:** Cola que permite la operación de bloqueo en la inserción y eliminación de elementos.
 - 2. **Ventaja:** Útil en la programación concurrente para controlar el flujo de datos entre hilos.

- 4. TreeSet:**

- 1. **Uso:** Conjunto que mantiene los elementos en orden natural o según un comparador.
 - 2. **Ventaja:** Permite búsquedas rápidas y ordenadas.

- 5. PriorityQueue:**

- 1. **Uso:** Cola que ordena los elementos según su prioridad.
 - 2. **Ventaja:** Útil para algoritmos que requieren un procesamiento basado en prioridades.