JAVA

MÓDULO Nro.3

1000 PROGRAMADORES

Ministerio de Economía y Servicios Públicos
Gobierno de Salta

Ministerio de Educación Cultura, Ciencia y Tecnología
Gobierno de Salta

Universidad Nacional de Salta

Temas:

- Entrada y Salida por consola
- Variables de texto. Leer/escribir textos.

Clase nro. 2

- **Listas.**
- Colas.
- Pilas.
- Hash Map
- Recursión

MÓDULO 3.

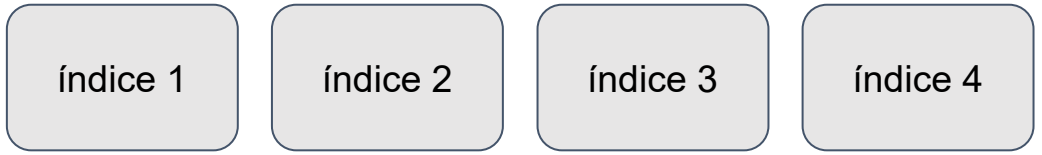
1000 PROGRAMADORES

Soria Juan Pablo

Listas:

1000 PROGRAMADORES

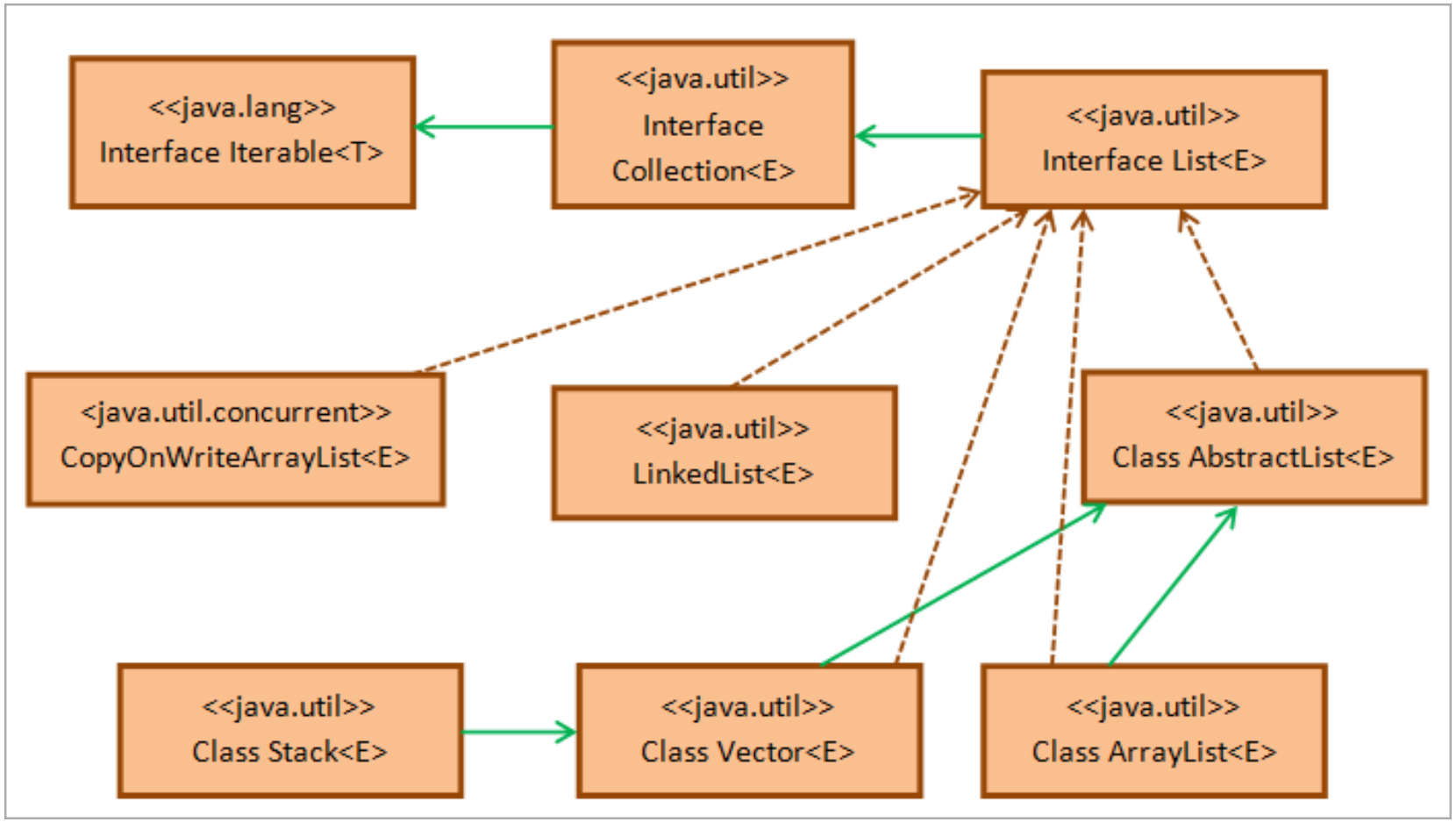
Las listas son un tipo de colección que hereda de la interface collection, son una estructura de datos que respeta el orden en el cual fueron agregados los elementos, también permiten registros repetidos.



- El número de elementos de la lista no suele estar fijado
- Se tiene un control absoluto y preciso del lugar en el que se quiere insertar.
- Es posible acceder a sus elementos a través de su índice
- Maneja grandes volúmenes de datos
- List se encuentra en el paquete java.util
- Algunas clases que implementa son:
  - ArrayList
  - LinkedList
  - Stack
  - Vector

Soria Juan Pablo

Listas:


<<java.lang>>
Interface Iterable<T>
<<java.util>>
Interface
Collection<E>
<<java.util>>
Interface List<E>
<java.util.concurrent>>
CopyOnWriteArrayList<E>
<<java.util>>
LinkedList<E>
<<java.util>>
Class AbstractList<E>
<<java.util>>
Class Stack<E>
<<java.util>>
Class Vector<E>
<<java.util>>
Class ArrayList<E>

MÓDULO 3.

1000 PRO GRA MA DORES

Listas:

Crear y declarar una lista

Hemos indicado que List es una interfaz y está implementada por clases como:

- ArrayList
- Stack
- Vector
- LinkedList.

Por lo tanto, puede declarar y crear instancias de la lista de cualquiera de las siguientes maneras:

```
List linkedlist = new LinkedList();
List arraylist = new ArrayList();
List vec_list = new Vector();
List stck_list = new Stack();
```

el orden de los elementos cambiará dependiendo de la clase utilizada para crear una instancia de la lista.

por ejemplo, para una lista con clase stack, el orden es Last In, First Out (LIFO).

Listas:

Métodos:

- add()
- addAll()
- clear()
- contains()
- containsAll()
- copyOf()
- equals()
- get()
- hashCode()
- indexOf()
- isEmpty()
- iterator()
- lastIndexOf()
- listIterator()
- of()
- remove()
- removeAll()
- replaceAll()
- retainAll()
- set()
- size()
- sort()
- spliterator()
- subList()
- toArray()

1000 PRO GRA MA DORES

MÓDULO 3.

Listas:

Ejemplo

```java
p.java
App.java > ficheros
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

class ficheros {


    Run | Debug
    public static void main(String[] args) throws Exception {
        List<String> list = new ArrayList<String>();

        // Añadimos elementos
        list.add(e: "Luis");
        list.add(e: "Marta");
        list.add(e: "Julio");

        // Obtenemos un Iterador y recorremos la lista.
        //Iterator<String> lista = new list.iterator();
        Iterator<String> iter = list.iterator();
        while (iter.hasNext())
            System.out.println(iter.next());
    }

}
```

Definimos un ArrayList del tipo string.
Agregamos datos a la lista con add().

iterador es un objeto que nos permite recorrer una lista y presentar por pantalla todos sus elementos . Dispone de dos métodos clave para realizar esta operación hasNext() y next().

Listas:

Diferencias entre listas y arrays

Arrays

- estructura de datos para almacenar una secuencia de valores, **NO redimensionable**.
- Series de variables consecutivas en memoria.
- Se debe especificar su tamaño.
- Se debe definir el tipo de datos que manejara.
- Se maneja con índice.
- No se puede agregar o eliminar elementos

Listas

- estructura de datos para almacenar una secuencia de valores, que **ES redimensionable**.
- Se guardan en memoria de manera dinámica.
- No es necesario definir el tamaño.
- Puede almacenar distintos tipos de datos (no recomendado).
- Se maneja con puntero.
- Se puede agregar y eliminar elementos.

Temas:

- Entrada y Salida por consola
- Variables de texto. Leer/escribir textos.

Clase nro. 2

- Listas.
- **Colas.**
- **Pilas.**
- Hash Map
- Recursión

MÓDULO 3.

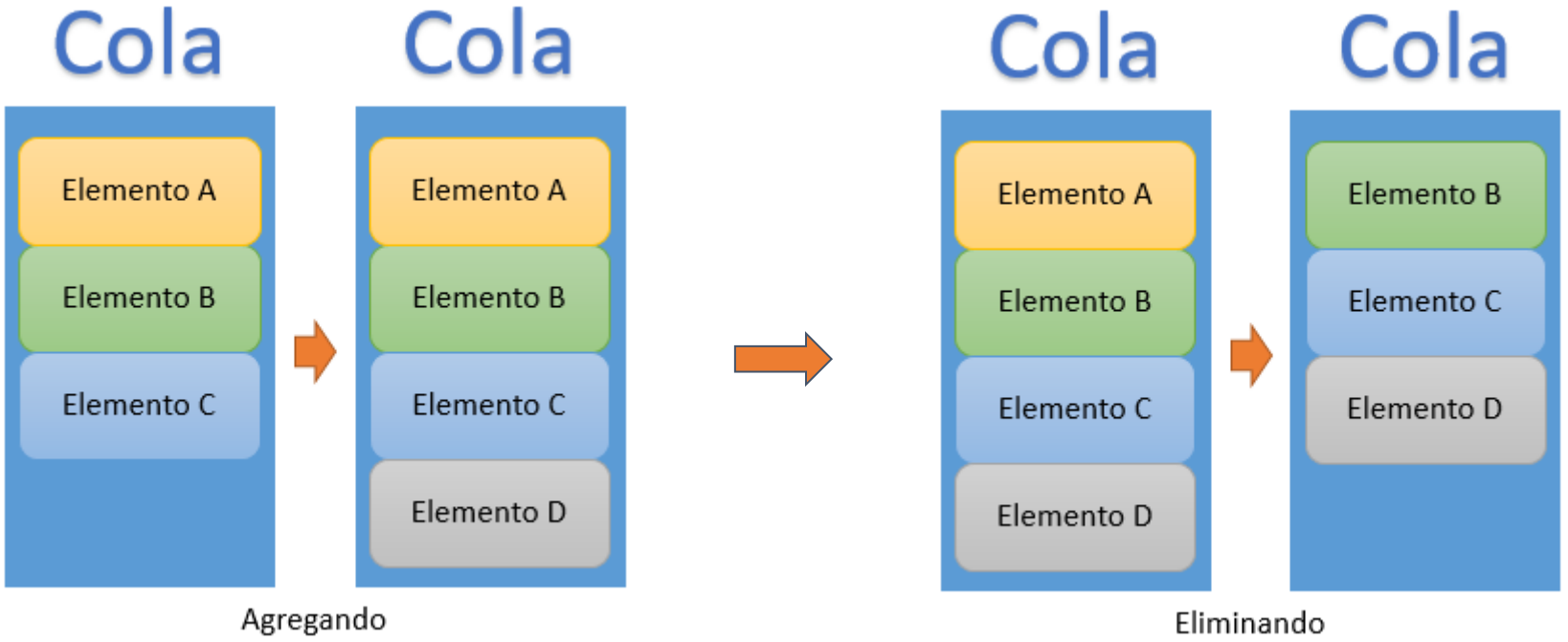
1000 PRO GRA MA DORES

Soria Juan Pablo

Colas:

La interfaz de cola se proporciona en el paquete java.util e implementa la interfaz de Colección.

- Son un tipo especial de listas
- Es un tipo de dato abstracto (TDA)
- La cola implementa FIFO (first in, first out), es decir, primero en entrar, primero en salir..



Soria Juan Pablo

Colas:

- En el mundo real podemos encontrar este ejemplo en las colas de un banco, la cadena de impresión de documentos, etc.

- En el caso de la cola en el banco, la primera persona en llegar es también la primera en irse (suponiendo una única ventanilla) y en los documentos a imprimir, la impresora imprime según el orden de llegada.

- Las colas se pueden implementar utilizando una estructura estática (arreglos), o una estructura dinámica (listas enlazadas, vectores, etc).

Colas:

Operaciones:

- add(e): Inserta el elemento e al final de la cola
- poll(): Elimina el elemento del frente de la cola y lo retorna. Si la cola está vacía se produce un error
- peek(): Retorna el elemento del frente de la cola. Si la cola está vacía se produce un error
- isEmpty(): Retorna verdadero si la cola está vacía.
- size(): Retorna la cantidad de elementos de la cola

Soria Juan Pablo

Colas:

Implementación:

- Se crea una colección llamada cola de tipo Queue
    - **Queue**: es una forma lineal especial, que sólo permite la eliminación en el extremo frontal de la lista, mientras que el extremo posterior de la operación de inserción de la lista

- Con una implementación de lista enlazada (LinkedList):
    - clase LinkedList implementa la interfaz de cola, por lo que puede usar listas enlazadas como una cola

- Se define excepciones para las condiciones de error

Soria Juan Pablo

Colas:

Ejemplo:

va ×

App.java > listas > cola()

```
public static void lista(){···

public static void cola(){
    Queue<Integer> cola = new LinkedList<>();
    System.out.println(x: "Agregando valores");
    for (int i = 0; i < 5; i++) {
        cola.add(i);
        System.out.println("Valor: " + i);
    }
    System.out.println("\nTamaño Inicial de la cola: " + cola.size());
    System.out.println(x: "\nRetirando valores");
    while (cola.peek()!=null) {
        System.out.println("Valor: " + cola.poll());
        System.out.println("Tamaño actual de la cola: " + cola.size());
    }
    System.out.println("Tamaño final de la cola: " + cola.size());
}
Run | Debug
public static void main(String[] args) throws Exception {
    //lista();
    cola();
}
```

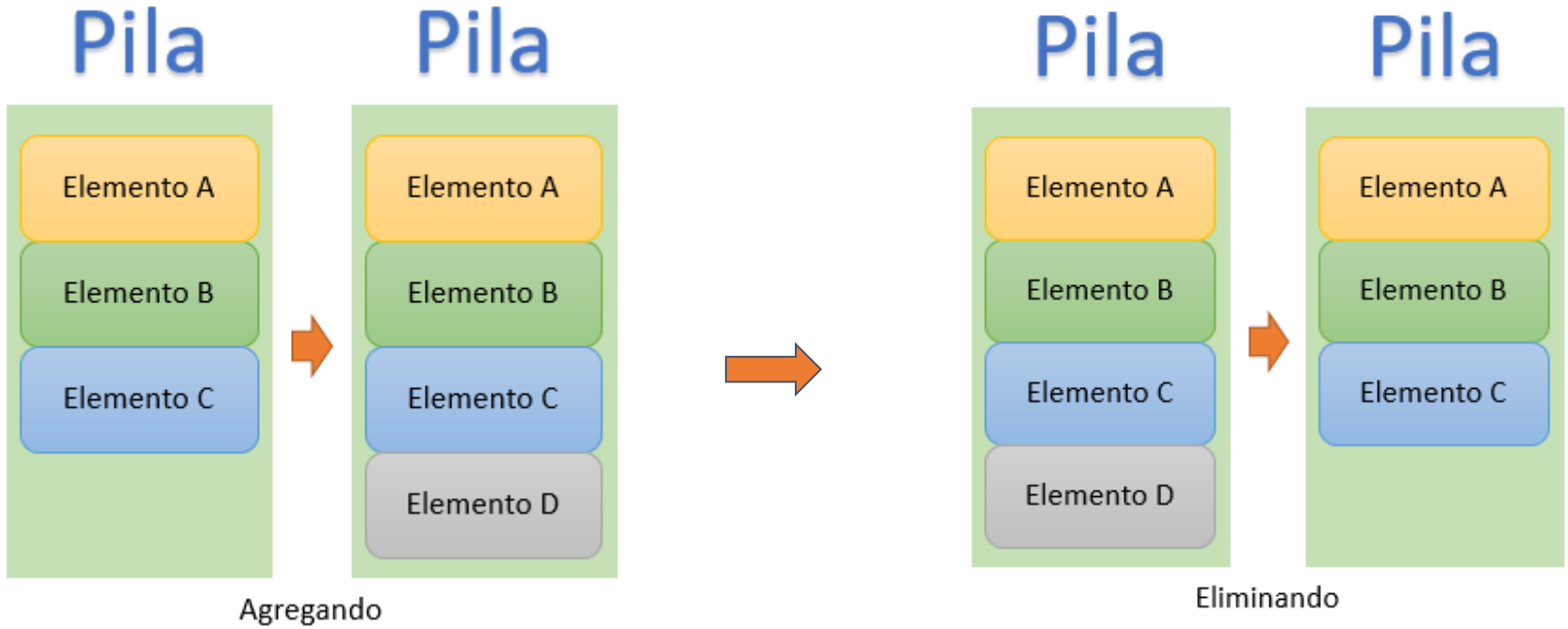
Definimos una cola del tipo integer.
Agregamos datos a la cola con **add()**.

iteramos hasta que la cola está vacía. **peek()** nos muestra el elemento en la cola. Y con **poll()** sacamos el elemento de la cola.

Pilas (stacks):

La interfaz de pila se proporciona en el paquete java.util e implementa la interfaz de Colección.

- Son un tipo especial de lista.
- Es un tipo de dato abstracto (TDA)
- La pila implementa LIFO (Last in, first out), es decir, último en entrar, primero en salir..



Soria Juan Pablo

Pilas:

1000 PROGRAMADORES

- En el mundo real podemos encontrar este ejemplo al apilar platos en un punto, cuando queremos mover los platos uno por uno se comienza retirando el último plato colocado.

- Las pilas se pueden implementar utilizando una estructura estática (arreglos), o una estructura dinámica (listas enlazadas, vectores, etc).

Soria Juan Pablo

Pilas:

Operaciones:

- push(e): Inserta el elemento e al tope de la pila.
- pop(): Elimina el elemento del tope de la pila y lo retorna. Si la pila está vacía se produce un error
- top(): Retorna el elemento del tope de la pila. Si la pila está vacía se produce un error
- isEmpty(): Retorna verdadero si la pila está vacía.
- size(): Retorna la cantidad de elementos de la pila

Soria Juan Pablo

Pilas:

Implementación:

- Se puede crear una pila de forma sencilla con la clase Stack que hereda de la clase Vector

- Se define excepciones para las condiciones de error

Soria Juan Pablo

MÓDULO 3.

Pilas:

Ejemplo:

```
pp.java
App.java > listas > pila()

    public static void cola(){···

    public static void pila(){
        Stack<Integer> pila = new Stack<>();
        System.out.println(x: "Agregando valores");
        for (int i = 0; i < 7; i++) {
            pila.push(i);
            System.out.println("Valor: " + i);
        }
        System.out.println("\nTamaño Inicial de la pila: " + pila.size());
        System.out.println(x: "\nRetirando valores");
        while (!pila.isEmpty()) {
            System.out.println("Valor: " + pila.pop());
            System.out.println("Tamaño actual de la pila: " + pila.size());
        }
        System.out.println("Tamaño final de la pila: " + pila.size());
    }
    Run | Debug
    public static void main(String[] args) throws Exception {
        //lista();
        //cola();
        pila();
    }
```

Definimos una pila del tipo integer. Agregamos datos a la cola con **push()**.

iteramos hasta que la pila está vacía. **isEmpty()** nos muestra el elemento en la cola. Y con **pop()** sacamos el elemento de la pila.

Gracias.

WEB: http://milprogramadores.unsa.edu.ar/

CANAL TELEGRAM: https://t.me/milprogramadoressaltenios

CENTRO DE AYUDA: http://ayudamilprogramadores.com/

1000 PROGRAMADORES