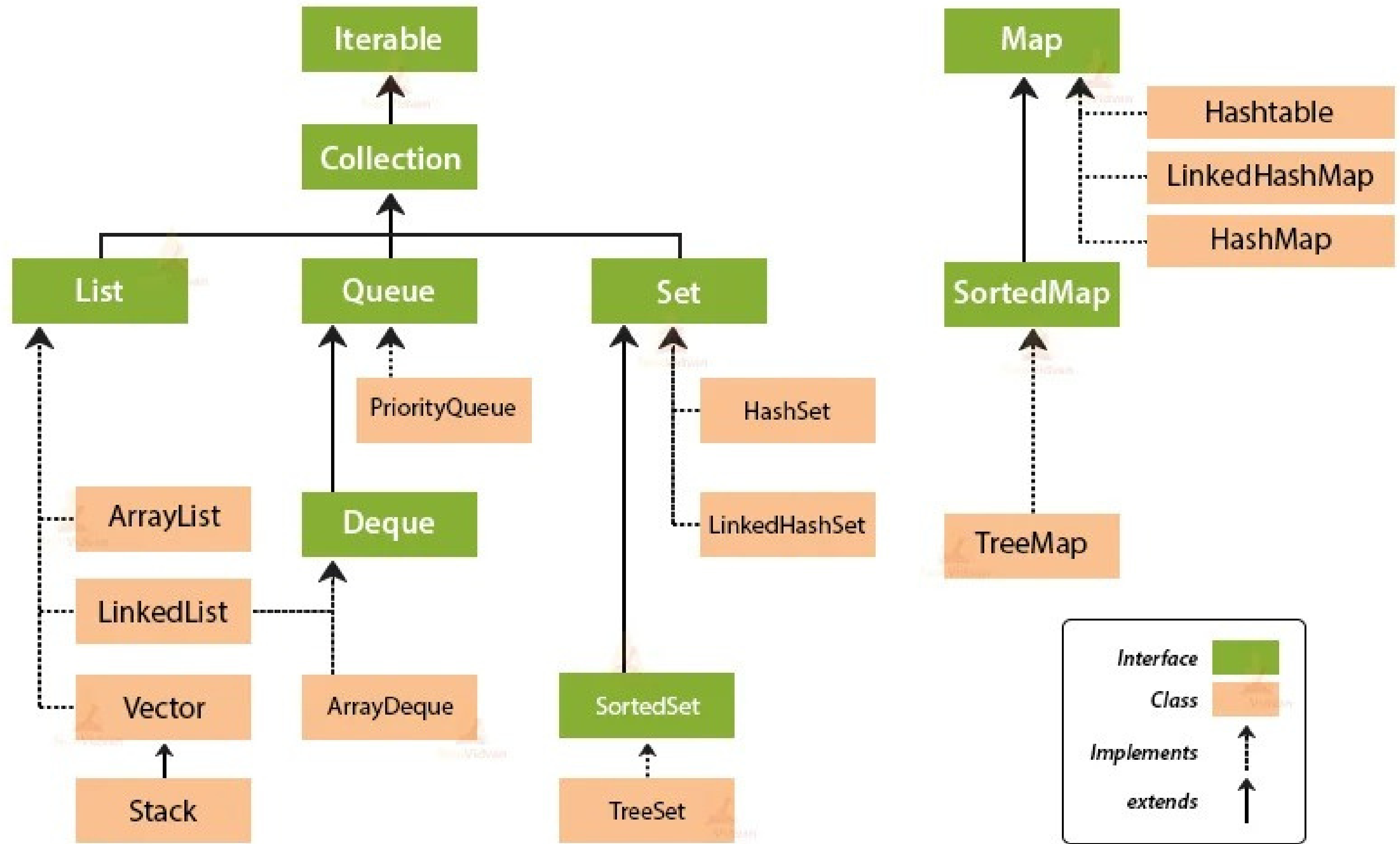


COLLECTIONS

Java



COLLECTION

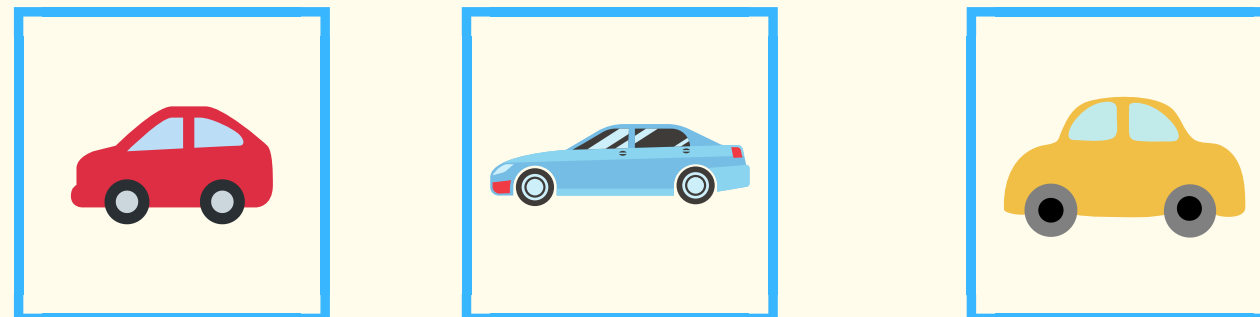
Es la interfaz raíz de la jerarquía de colecciones. Define operaciones básicas de una colección, como agregar, eliminar, consultar elementos y obtener su tamaño.

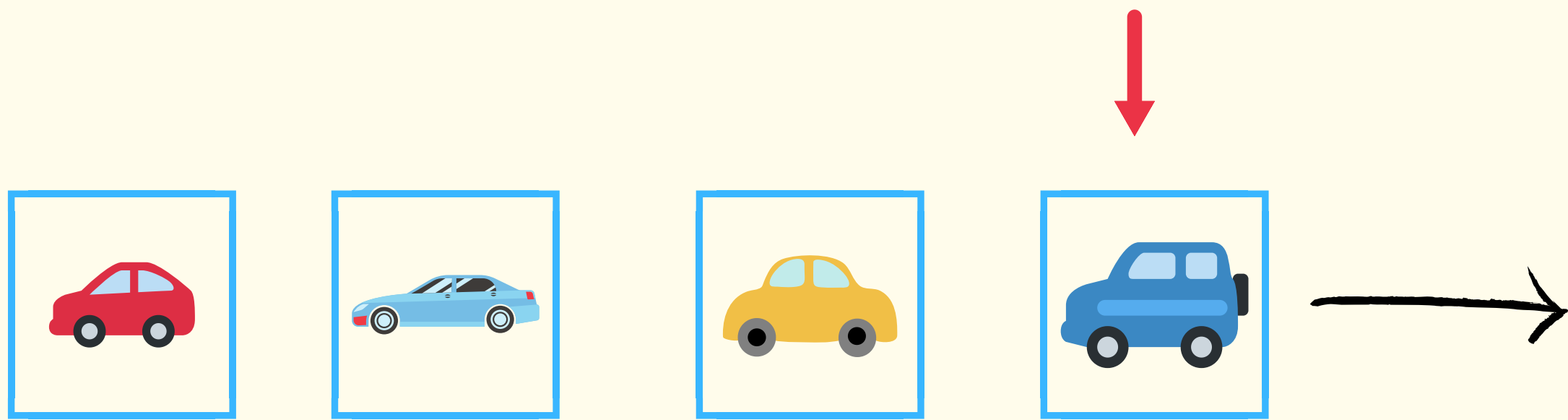
ArrayList

Es una implementación de List que utiliza un arreglo dinámico para almacenar elementos. Proporciona un acceso rápido a los elementos por su índice, pero puede ser menos eficiente en inserciones y eliminaciones.

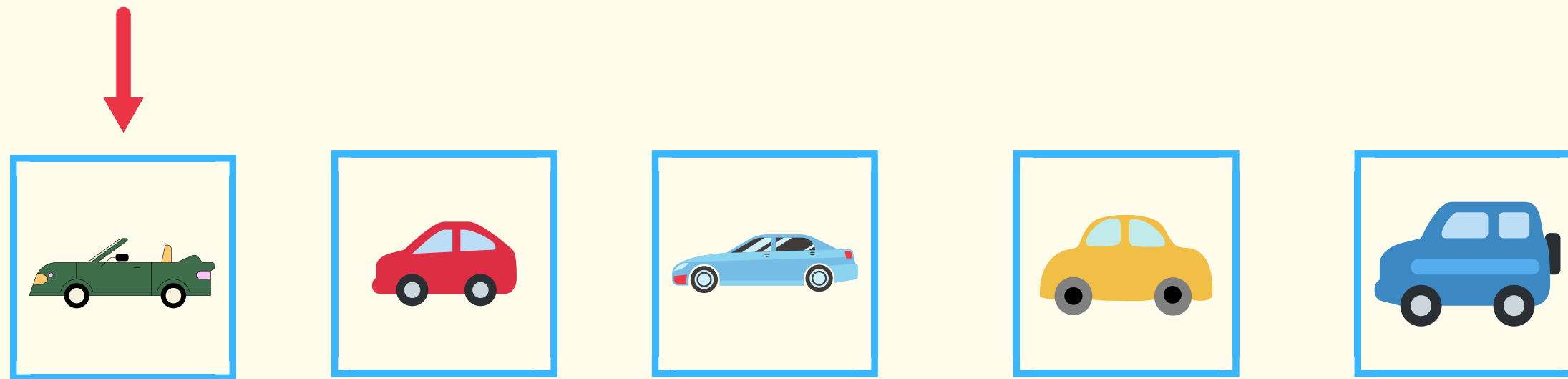
EJEMPLO TENEMOS UN ARRAYLIST DE AUTOS

Autos

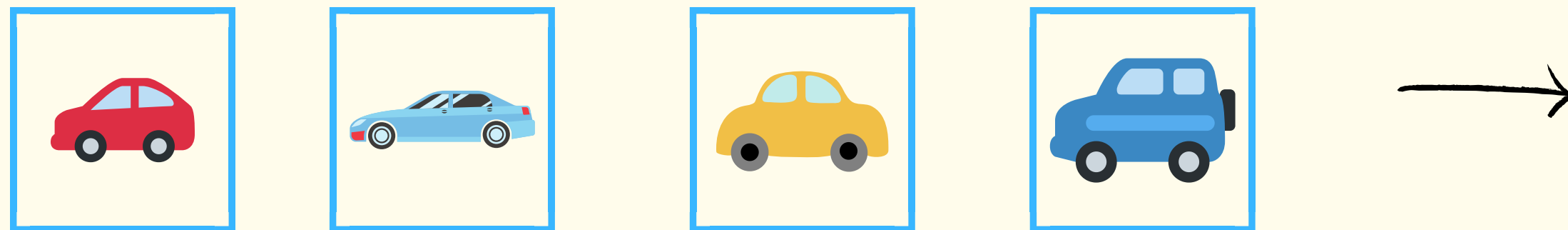




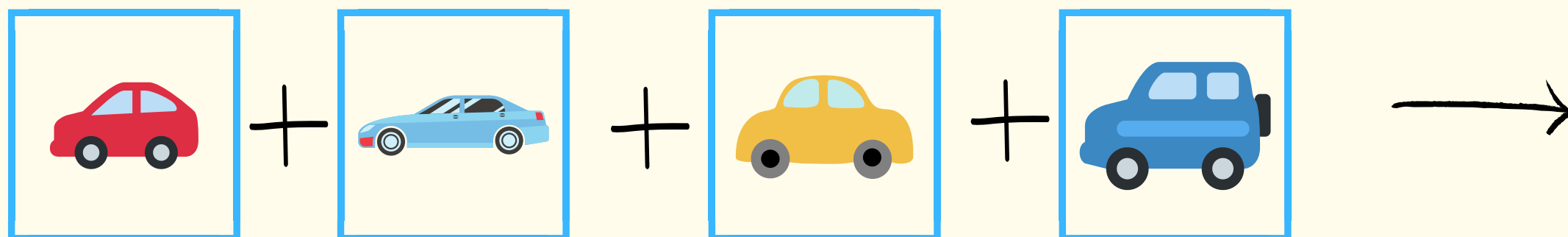
add(camioneta): Agrega el elemento especificado al final de la lista.



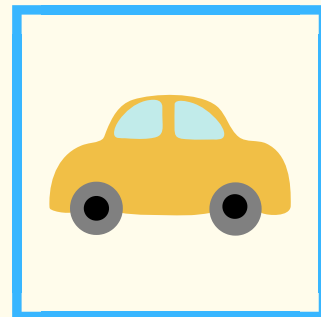
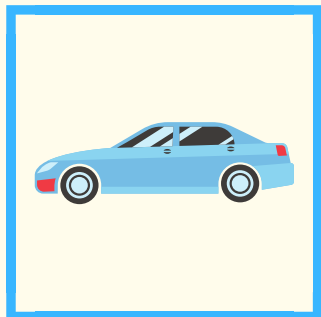
add(0, "convertible"):
Inserta el elemento especificado en la posición dada por el índice.



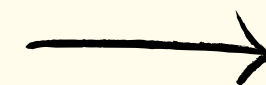
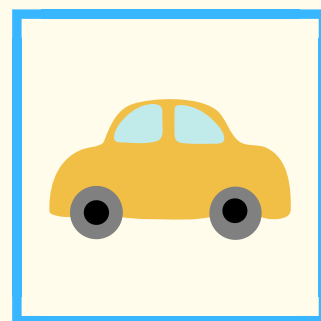
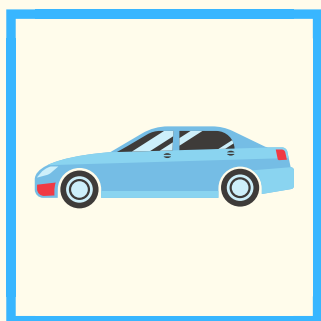
Eliminar un Coche por índice
listaAutos.remove(0);



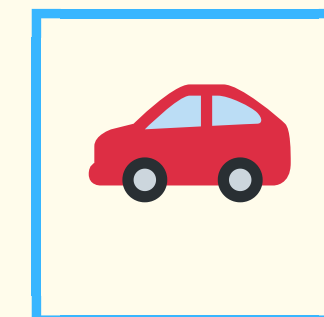
Obtener el tamaño de la lista
listaAutos.size();



Verificar si la lista está vacía
listaAutos.isEmpty();

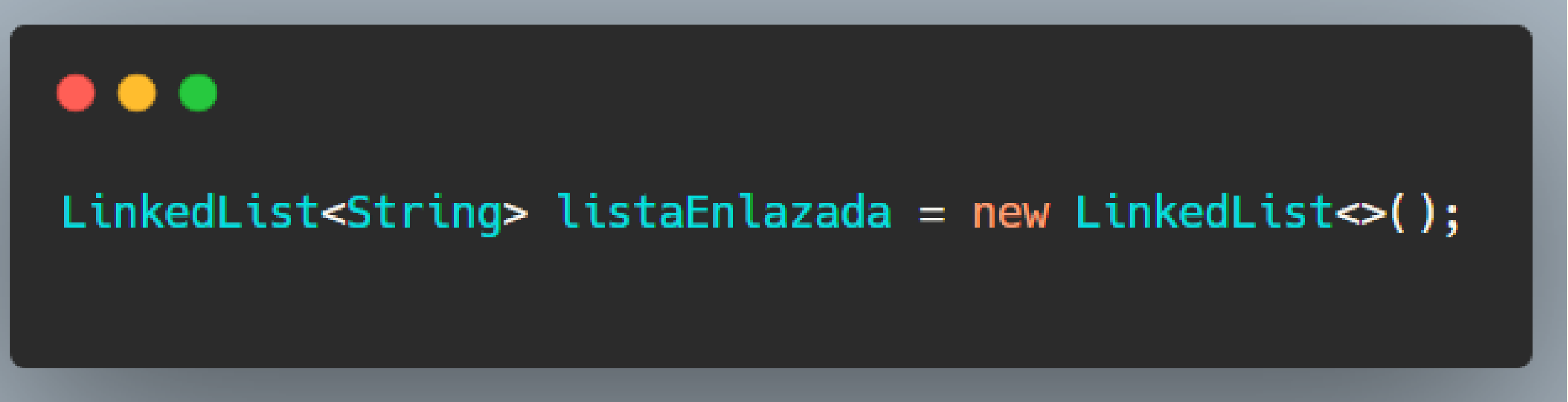


Obtener un Coche por
índice
listaAutos.get(1);

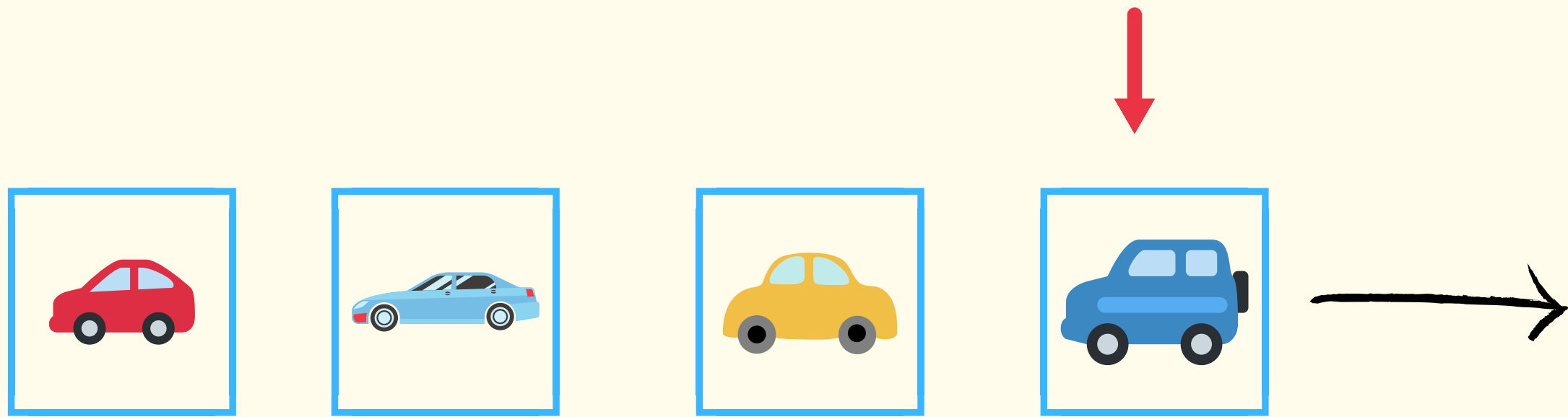


En Java, **LinkedList** es una clase que **implementa la interfaz List** y representa una lista enlazada doblemente en lugar de un arreglo dinámico como lo hace ArrayList.

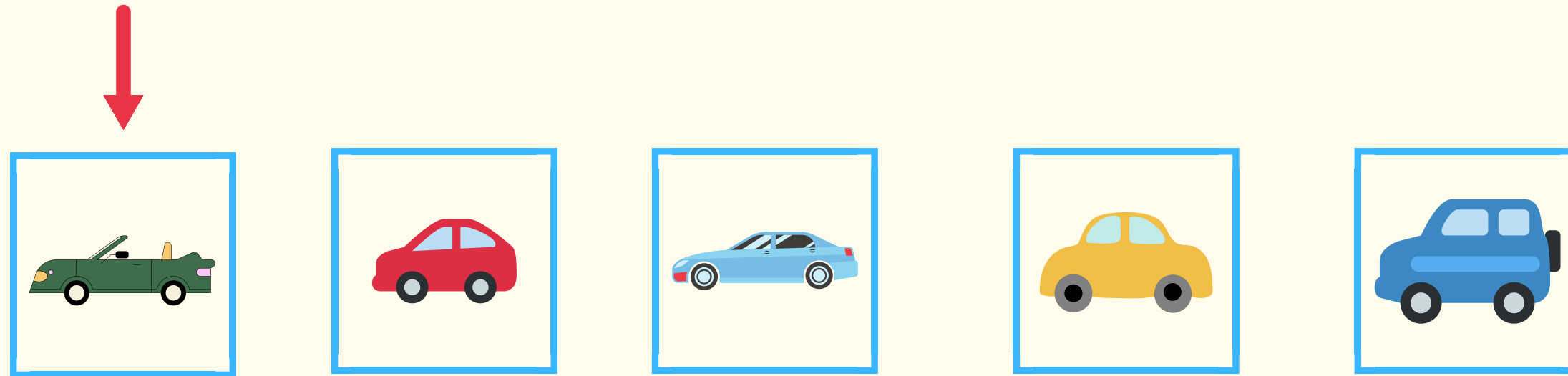
Una lista enlazada doblemente permite un acceso más rápido a los elementos en el medio de la lista, pero puede ser más lenta para acceder a elementos en posiciones aleatorias.



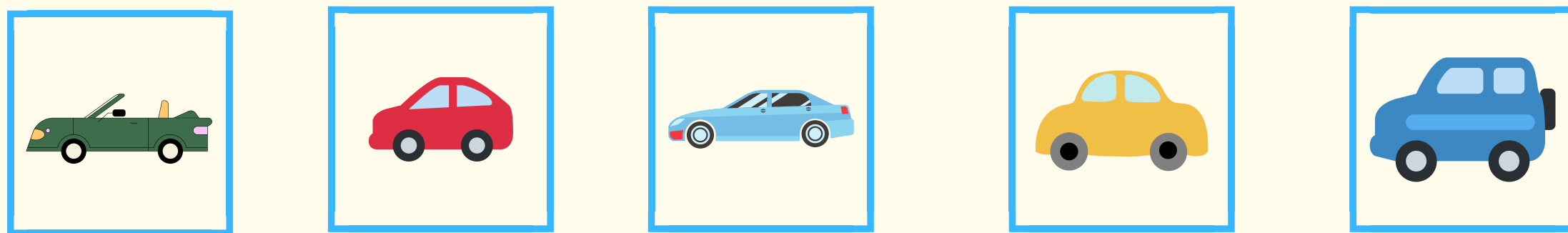
```
LinkedList<String> listaEnlazada = new LinkedList<>();
```



add(E e): Agrega el elemento especificado al final de la lista.

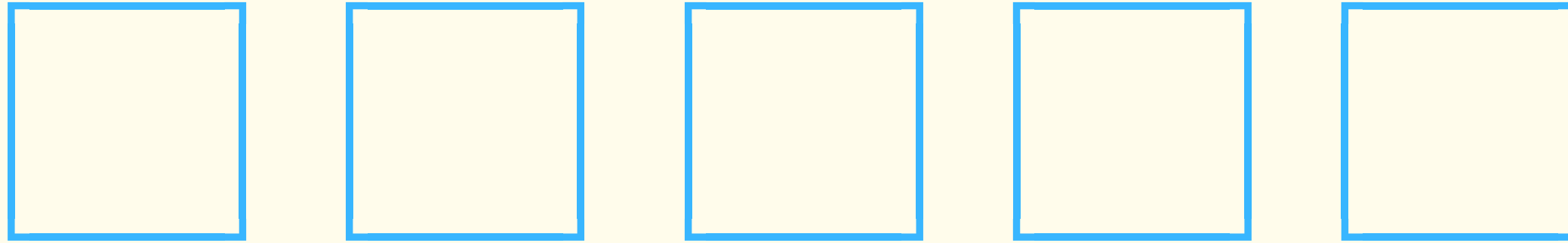


add(0, "convertible"):
Inserta el elemento especificado en la posición dada por el índice.

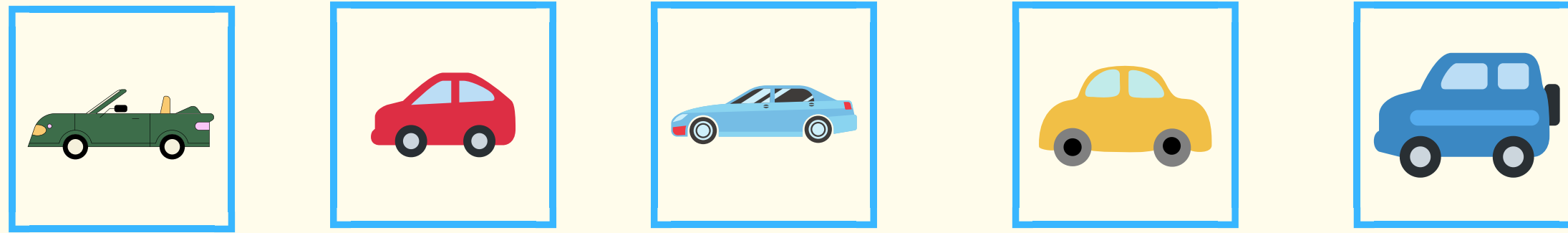


remove(int index):
Elimina el elemento en la posición especificada.

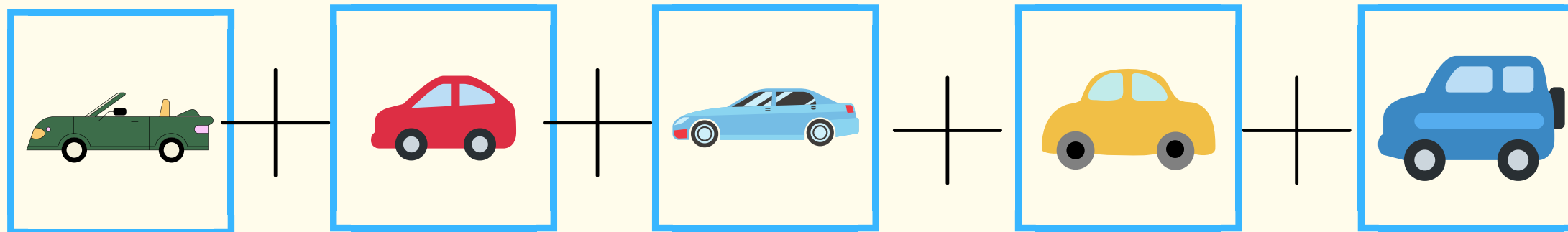
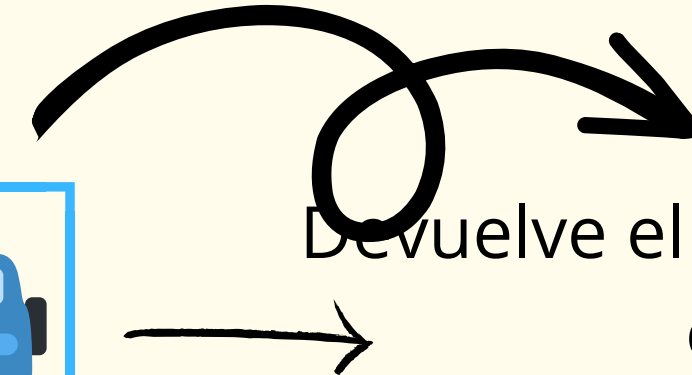




clear():
Elimina todos los elementos de la lista.



get(int 4):
Devuelve el elemento en la posición especificada.

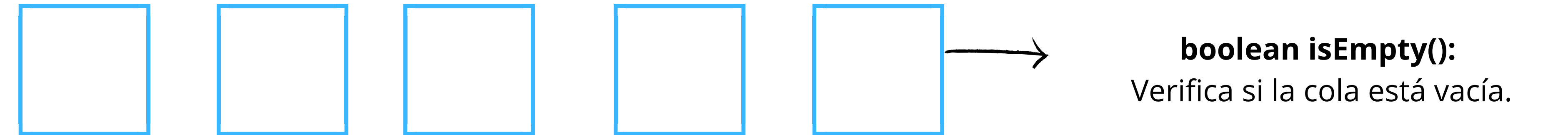
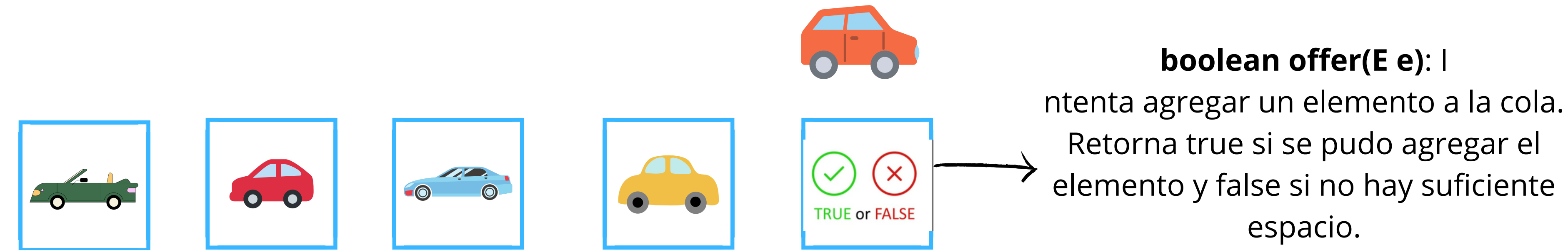


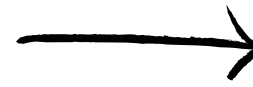
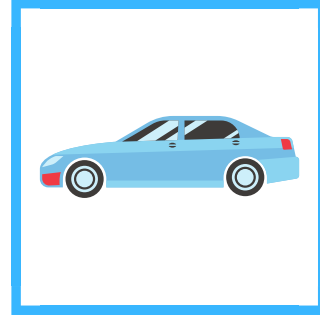
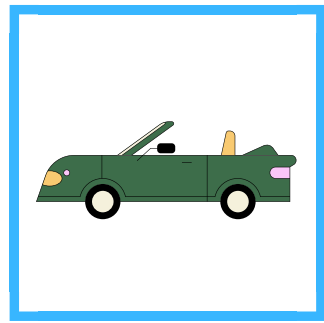
size():
Devuelve el número de elementos en la lista.

En Java, una cola (**Queue**) es una estructura de datos que sigue el principio de "**Primero en entrar, primero en salir**" (FIFO - First-In-First-Out). Esto significa que el elemento que se agrega primero a la cola será el primero en ser eliminado. J



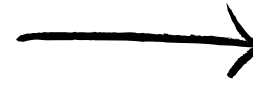
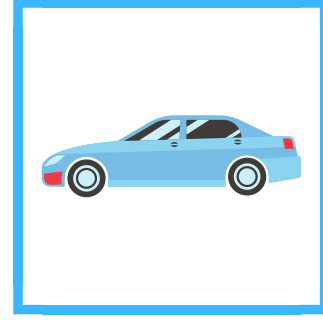
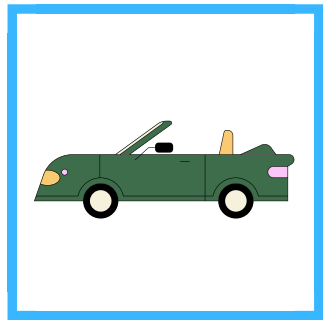
```
Queue<String> cola = new LinkedList<>();
```





E peek():

Devuelve el elemento al frente de la cola sin eliminarlo. Retorna null si la cola está vacía.



E poll():

Elimina y devuelve el elemento al frente de la cola. Retorna null si la cola está vacía.

```
System.out.println cola.poll();
```



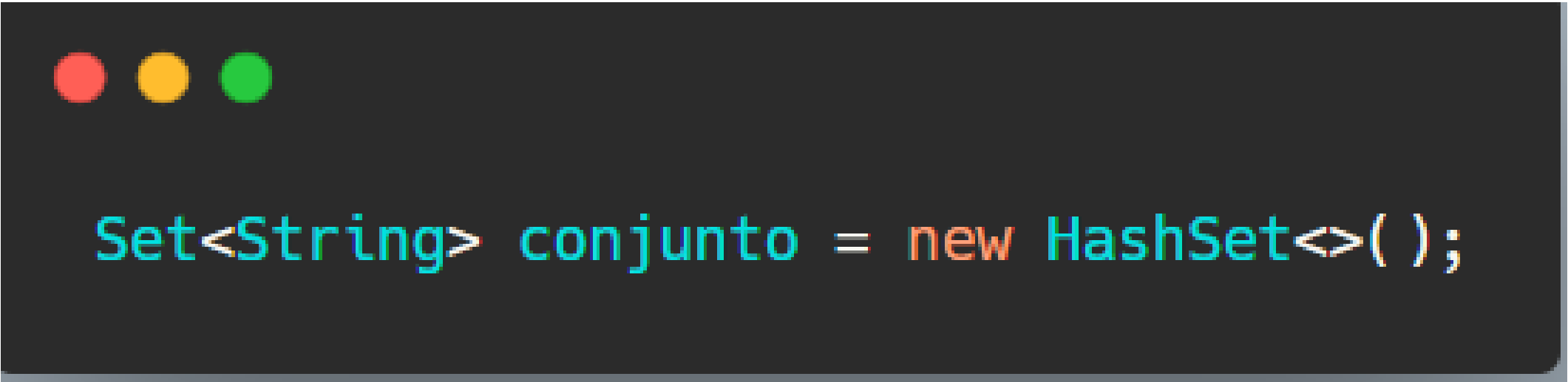


Cola doblemente terminada

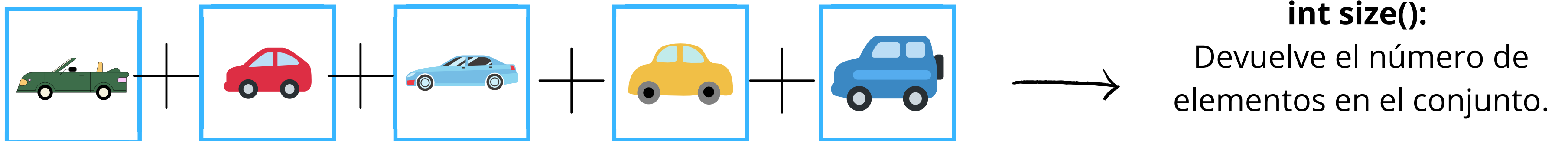
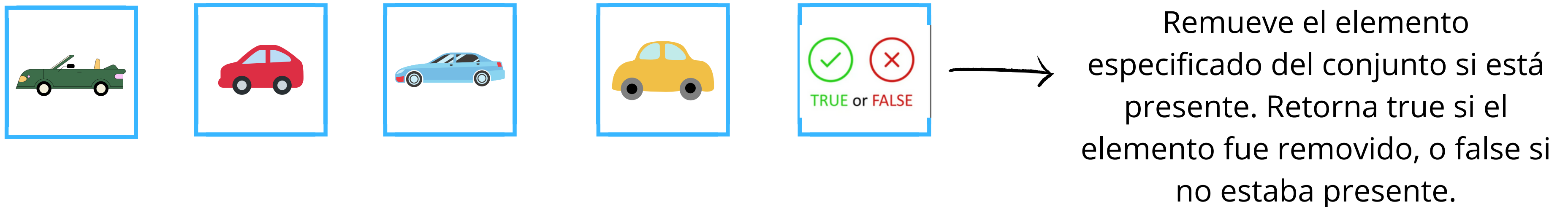


En Java, un conjunto (Set)

Es una colección que no permite elementos duplicados. La interfaz `Set<E>` en el paquete `java.util` define un conjunto, y las implementaciones concretas proporcionan distintas maneras de almacenar y acceder a los elementos sin permitir duplicados.



```
Set<String> conjunto = new HashSet<>();
```





boolean isEmpty():
Verifica si el conjunto está vacío.



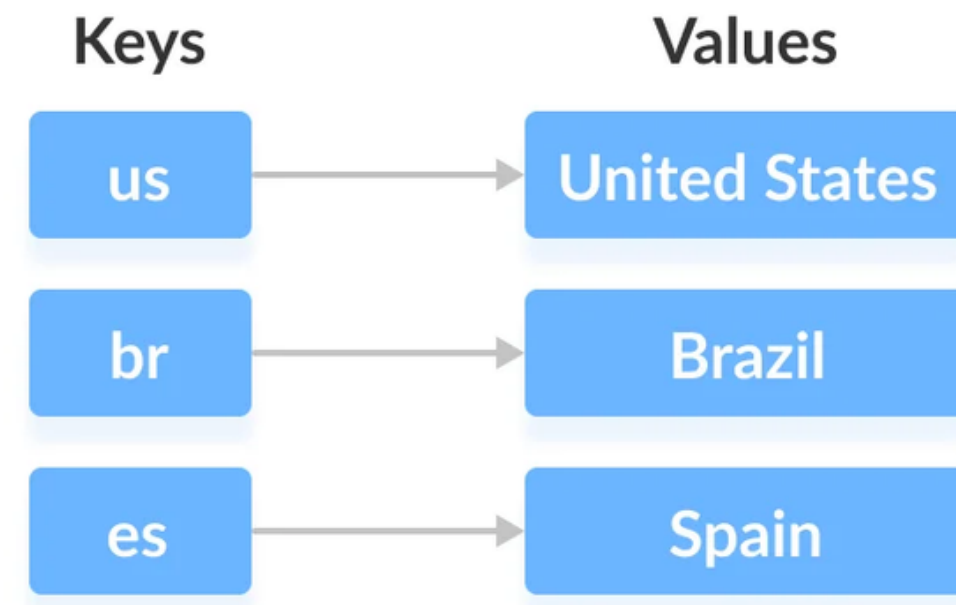
void clear():
Remueve todos los
elementos del conjunto.

En Java, un mapa (Map)

Es una estructura de datos que almacena pares clave-valor, donde cada clave es única y se utiliza para acceder a su correspondiente valor. La interfaz `Map<K, V>` en el paquete `java.util` define un mapa, y las implementaciones concretas proporcionan distintas maneras de almacenar y acceder a los pares clave-valor.

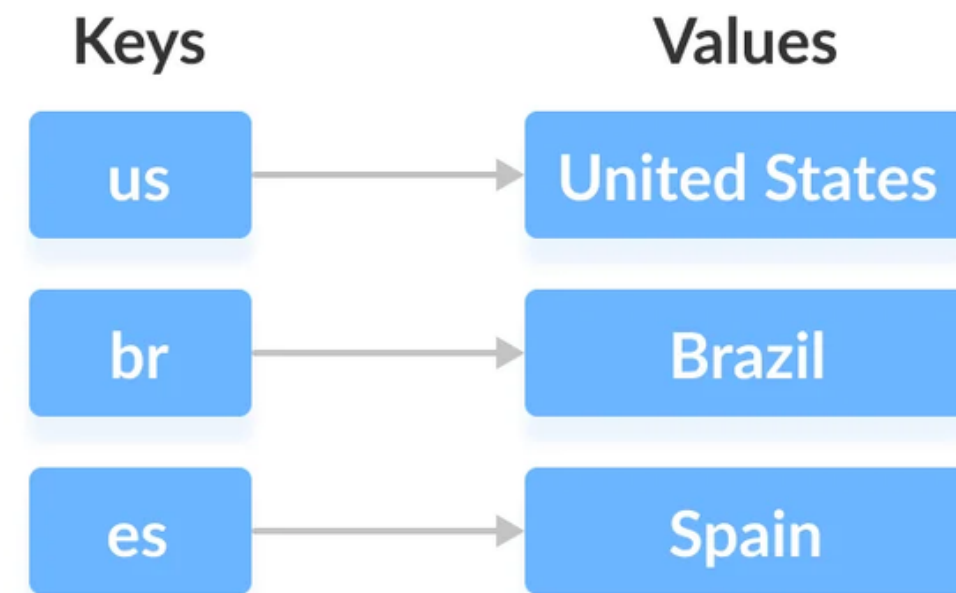


```
Map<String, Integer> mapa = new HashMap<>();
```



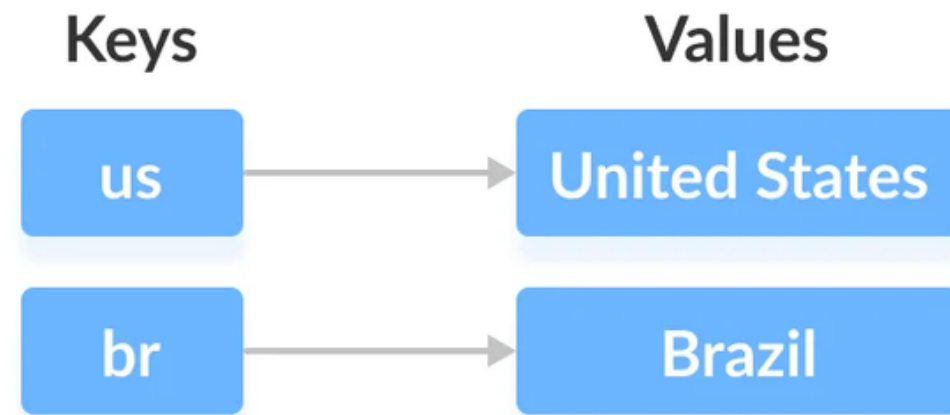
put(K us, V United States):

Asocia la clave dada con el valor dado en el mapa. Si la clave ya existe, el valor anterior asociado con la clave se reemplaza y se devuelve; si no, se agrega la nueva clave-valor y se devuelve null.

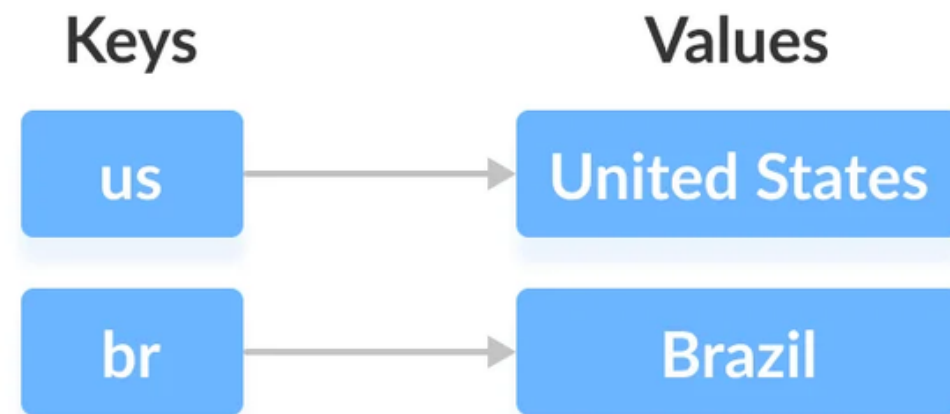


V get(Object br): Devuelve el valor asociado con la clave especificada. Si la clave no está presente, devuelve null.

Brazil

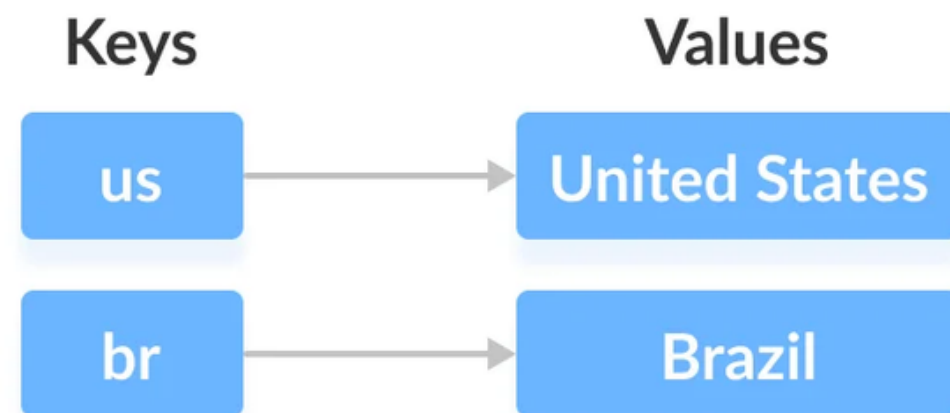


→ **V remove(Object es):**
Elimina la entrada correspondiente a la clave dada y devuelve el valor asociado. Si la clave no está presente, devuelve null.



→ **boolean isEmpty():**
Verifica si el mapa está vacío.

FALSE



→ **int size():**
Devuelve el número de pares clave-valor en el mapa.