Colecciones y APIs

Sesión Conceptual 01

{desafío} latam_

- Objetivo de la sesión
- Activación de conceptos clave

- Conceptualización
- Ejercicios
- Quiz

- Cierre

Inicio

{desafío}
latam_

/*Comprender que son las colecciones*/

Objetivo

{desafío} latam_

Desarrollo

130 minutos

{desafío} latam_

/* Introducción a Colecciones */

Java Collections

{desafío}
latam_

Java Collections

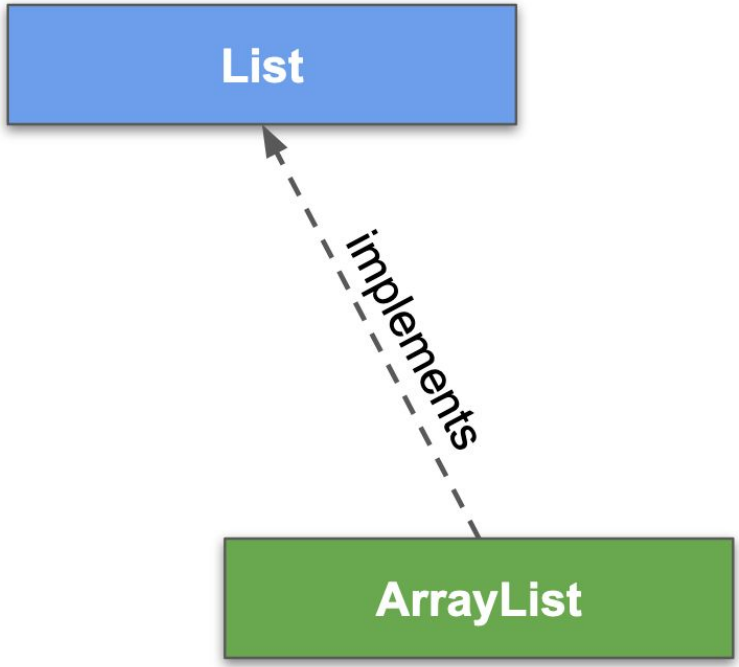
Quiz

{desafío}
latam_

/* Introducción a List */

List

ArrayList



ArrayList

- List es una interfaz secundaria de Collection.
- Es una colección ordenada de objetos en los que se pueden almacenar valores duplicados.
- Cómo List conserva el orden de inserción, permite el acceso posicional y la inserción de elementos.
- La interfaz List se implementa mediante las clases **ArrayList**, **LinkedList**, **Vector** y **Stack**.

```
List<String> list = new ArrayList<>();
list.add("Java");
list.add("Scala");
list.add("Kotlin");
```

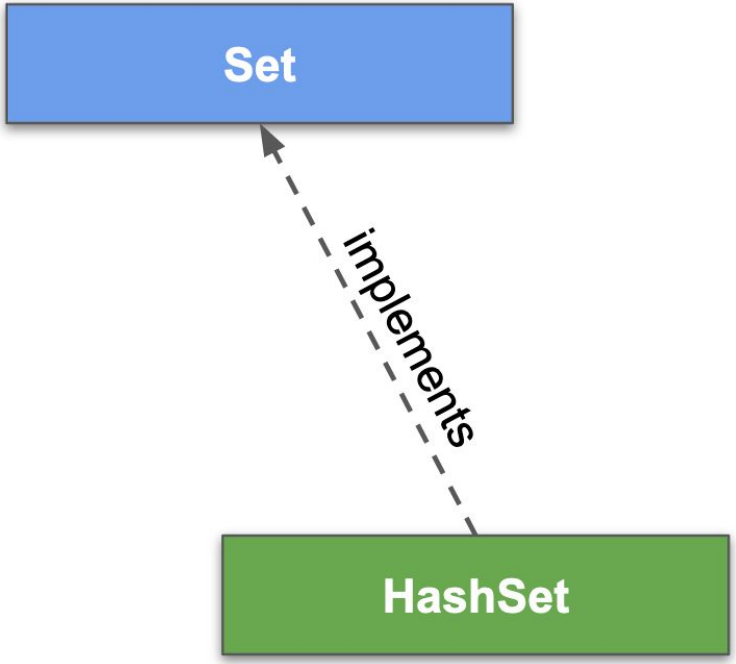
Quiz

{desafío}
latam_

/* Introducción a Set */

Set

HashSet



HashSet

- Set es una interfaz que extiende de Collection.
- Es una colección desordenada de objeto, en donde a diferencia de List no se pueden almacenar valores duplicados.
- Básicamente, Set está implementado por HashSet, LinkedHashSet o TreeSet (representación ordenada).
- Set tiene varios métodos como add, remove, clear, size, etc. para mejorar el uso de esta interfaz.

```
Set<String> langs = new HashSet<>();
langs.add("Java");
langs.add("Go");
langs.add("Erlang");
langs.add("Java");
```

Quiz

{desafío}
latam_

/* Introducción a Queue */

Queue

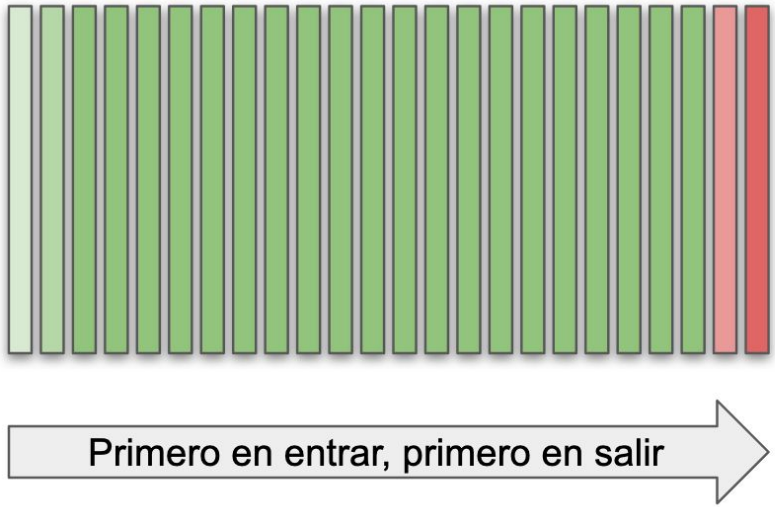
Queue

- Una cola es una estructura lineal que sigue un orden particular en el que se realizan las operaciones.
- El orden es Primero en entrar, primero en salir (FIFO). Un buen ejemplo de una cola es cualquier cola de consumidores para un recurso donde el consumidor que vino primero se sirve primero.

```
Queue tickets = new LinkedList<>();
for (int i = 2; i < 20; i+=3) tickets.add(i); //[2, 5, 8, 11, 14, 17]
```

Queue

Inserción y eliminación pasa en diferentes finales



Quiz

{desafío}
latam_

/* Introducción a Map */

Map

HashMap

- La interfaz Map representa una asignación entre una clave y un valor. La interfaz de mapa no es un subtipo de Collection. Por lo tanto, se comporta un poco diferente del resto de los tipos de colección
- Un mapa no puede contener claves duplicadas y cada clave puede mapearse a más de un valor. Algunas implementaciones permiten clave nula y valor nulo como HashMap y LinkedHashMap, pero otras no como TreeMap.

```
Map<String, List<String>> continentes = new HashMap<>();

continentes.put("America", Arrays.asList("Chile", "Peru", "Argentina", "Colombia"));
```

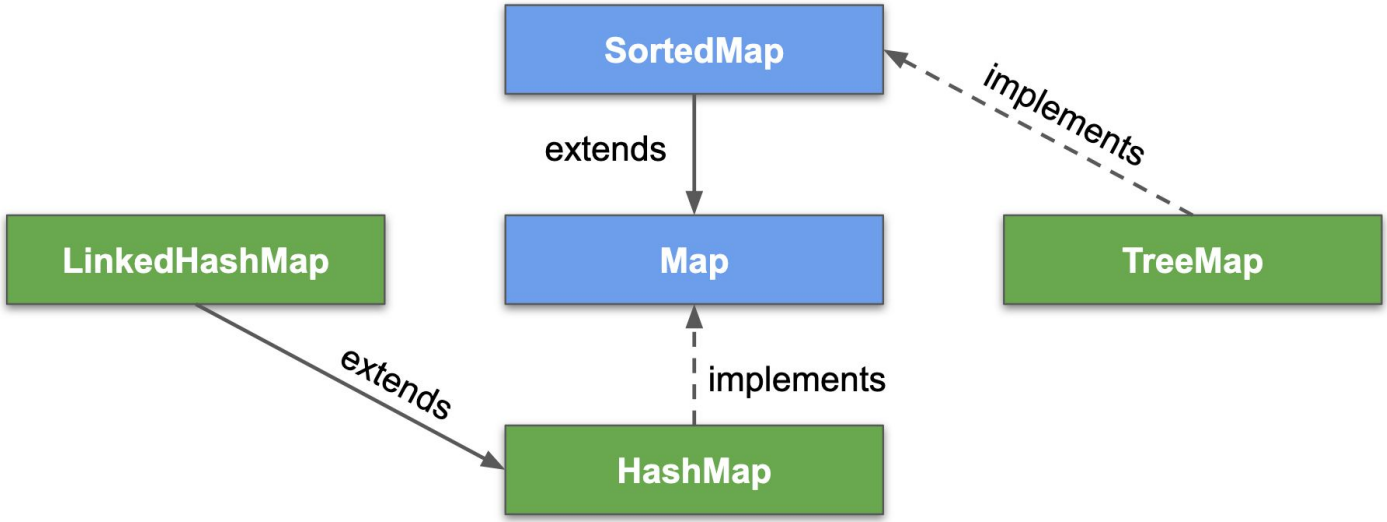
¿Por qué usar Map?

- Los mapas son perfectos para usar con el mapeo de asociación de valores clave, como los diccionarios. Los mapas se utilizan para realizar la búsqueda por claves o cuando desean recuperar y actualizar elementos por claves.

  Algunos ejemplos son:

- Un mapa de códigos de error y sus descripciones.
- Un mapa de códigos postales y ciudades.
- Un mapa de clases y estudiantes. Cada clase está asociada con una lista de estudiantes.

Jerarquía en Map



Quiz

{desafío}
latam_

Cierre

30 minutos

{desafío} latam_

¿Existe algún concepto que no hayas comprendido?

Volvamos a revisar los conceptos que más te hayan costado antes de seguir adelante

Reflexionemos

{desafío} latam_

{desafío} latam_

Academia de talentos digitales