**APUNTES PROGRAMACIÓN UD5: Colecciones e iteradores**

**BUCLE FOR MEJORADO (FOR EACH)**

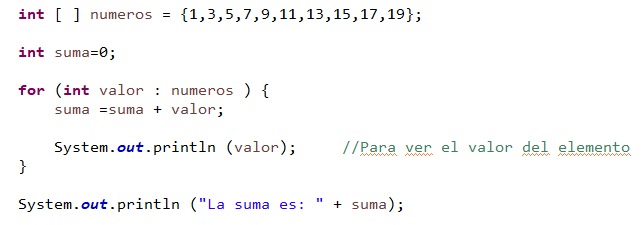
**Este bucle “for” puede usarse para hacer los bucles más compactos y fáciles de leer y suele usarse mucho en colecciones.**

**for (TipoARecorrer nombreVariableTemporal : nombredelArray) {...}**

**ESTE BUCLE SIEMPRE EMPIEZA POR EL PRIMER ELEMENTO, NO PUEDES ELEGIR POR QUÉ ELEMENTO EMPEZAR A ACCIONAR EL BUCLE. TAMPOCO SE PUEDE HACER QUE RECORRA EL ARRAY AL REVÉS NI MODIFICAR EL CONTENIDO DEL ARRAY MIENTRAS EL BUCLE ESTÁ ACTIVO.**

Este **“for”** es mejor para recorrer listas y encontrar datos concretos en listas. Además es mucho más rápido de escribir. Eso sí, NO SE DEBE USAR SIEMPRE, a veces es preferible usar bucles normales.

Ejemplo del nuevo bucle for:



**OPERADOR TERNARIO**

**Es una forma abreviada del bucle “if-else” que se escribe en una sola línea de código.**

**Condición ? ValorCumple : ValorNoCumple;**

Esta nomenclatura es lo mismo que:

**Condición if ValorCumple else ValorNoCumple;**

**nota>5 ? "aprobado" : "suspenso";**

El operador ternario se utiliza para ocasiones en los que el bucle que queremos accionar es simple y no realiza demasiado trabajo, como en el ejemplo anterior (por eso se escribe en una línea).

**SI EL BUCLE IF-ELSE QUE NECESITAMOS PROGRAMAR NO SE PUEDE ESCRIBIR EN UNA SOLA LÍNEA, NO SE PUEDE UTILIZAR EL OPERADOR TERNARIO, SE UTILIZARÍA UN BUCLE NORMAL.**

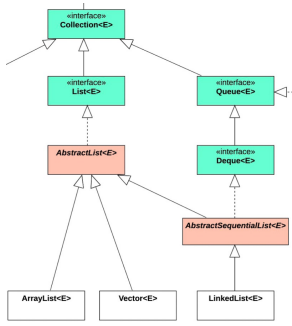
**LA INTERFAZ List <E>**

NOTACIÓN DE COLORES DE LOS DIAGRAMAS:

AZUL: Un tipo de colección

Se trata de una Collection <E> que permite duplicados y añade a dicha interfaz, funcionalidades como:

* Acceso posicional.
* Búsqueda.
* Iteración extendida.
* Operaciones sobre un rango de elementos.



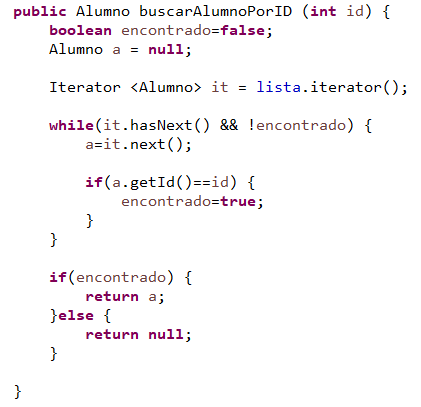
* **ArrayList <E>**
  + Más adecuada en la mayoría de las situaciones.
  + Acceso por índice en O(1).
  + Inserción, en media, en O(1).
  + Ocupa menos espacio que LinkedList.
* **LinkedList <E>**
  + Suele tener peor rendimiento.
  + Acceso por índice en O(n).
  + Inserción/borrado: O(1) extremos, O(n) por índice, O(1) en iteración.
  + Necesita más espacio (debe incluir dos referencias).

**LA INTERFAZ Iterator**

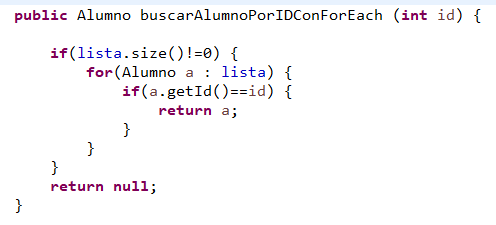
**Un iterador es un objeto “paseador” que nos permite ir obteniendo todos los objetos al ir invocando progresivamente su método next().** También, si la colección es modificable, podemos remover un objeto durante el recorrido mediante el método **remove( )** del iterador.

**Iterator it = nombreDeLaColección.iterator ();**

Buscar por ID con Iterator:

****

Buscar por ID con forEach:



//¡EN LAS CLASES QUE NO SEAN CRUD PONEMOS COMO ATRIBUTO UN OBJETO CRUD!

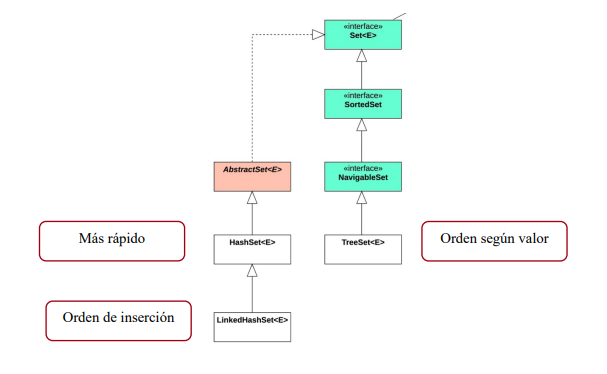
//EN ESTE CASO TENGO LA CLASE Secretaria QUE VA A TENER COMO ATRIBUTO UN OBJETO CRUDAlumno.

**LA INTERFAZ Set <E>**

**Se trata de una Colletion que no permite duplicados y es la abstracción del concepto matemático de "conjunto".**

**ESTA COLECCIÓN NO TIENE ÍNDICE, GUARDA ELEMENTOS COMO UN ARRAY, ¡PERO NO TIENE ÍNDICE! NO SE PUEDEN BUSCAR DE NINGUNA MANERA LOS DATOS CONTENIDOS EN SUS CASILLAS.**

**SI VAS A REALIZAR UN PROGRAMA EN EL QUE EL ORDEN DE LOS OBJETOS DE LA COLECCIÓN IMPORTA, NO USES SET<E>.**



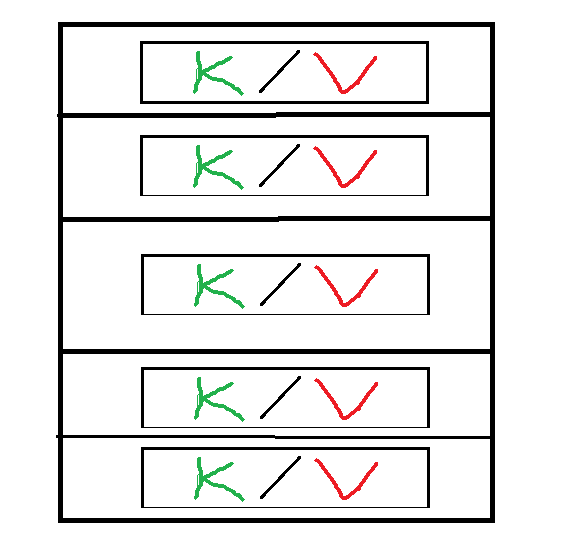
* **HashSet <E>**
  + No podemos predecir nada sobre el orden.
  + Mejor rendimiento de todas.
  + Proporciona tiempo constante (O(1)) en las operaciones básicas.
  + Permite insertar valores nulos.
  + No sincronizada.
  + Se mejora el rendimiento si se establece una capacidad inicial no muy elevada.

**LA INTERFAZ Map<E>**

**Un Map es una estructura de datos agrupados en parejas clave/valor. Pueden ser considerados como una tabla de dos columnas.**

CLAVE **(K)** / VALOR **(V)**

Se vería de esta forma:



Lo que le diferencia a esta colección de un array es que guarda valores en pareja y tanto la clave como el valor pueden ser de cualquier tipo. Por ejemplo, se puede tener un mapa en el que la clave sea una Persona y el valor sea un String: **Map <Persona/String>**.

**POR CIERTO, LAS CLAVES SON ÚNICAS E IRREPETIBLES ¡NO PUEDE HABER 2 IGUALES! PERO AL CONTRARIO QUE LAS CLAVES, ¡LOS VALORES SÍ SE PUEDEN REPETIR!**

Los opciones para recorrer un mapa son 2:

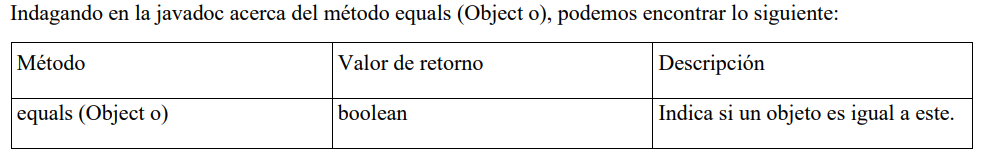
* **Opción 1:**
  + forEach o Obtener un Set con las claves.
  + Para cada clave, obtener los valores.
* **Opción 2:** 
  + Usando Lambdas (se explicará más adelante).
  + Método forEach. o Expresión lambda (Biconsumer).

**//MÉTODOS A TENER EN CUENTA:**

**KeySet() -** Devuelve un set (una especie de lista) de las claves. IDEAL PARA CREAR EL MÉTODO BUSCAR.

Mirar clase “Entry”.

**SOBREESCRIBIR EL MÉTODO equals()**



Este método lo utilizamos para saber si un objeto es igual que otro. Este método utiliza el operador == para comparar a dos objetos y decidir si son iguales.

**Reglas que sigue el método equals():**

* **Reflexivo:** Para cualquier referencia al valor x, x.equals(x) debe devolver true.
* **Simétrico:** Para cualquier referencia a los valores x y z, x.equals(z) debe devolver true si y solo si z.equals(x) es true.
* **Transitivo:** Para cualquier referencia a los valores w, x y z, si w.equals(x) regresa true y x.equals(z) regresa true, entonces w.equals(z) debe devolver true.
* **Consistente:** Para cualquier referencia a los valores x y z, múltiples invocaciones a x.equals(z) consistentemente devolverán true o false, si es que los valores utilizados para la comparación de los objetos no ha sido modificada.
* Para cualquier referencia no nula al valor x, x.equals(null), debe devolver false.

**SOBREESCRIBIR EL MÉTODO hashcode()**

**ORDENAR COLECCIONES**

**Interface Comparable<T>**

**T** → Tipo del objeto que guarda la colección (String, Integer, Persona).

**BELISARIO:** [**https://www.youtube.com/watch?v=mBqx4ze3-x4**](https://www.youtube.com/watch?v=mBqx4ze3-x4)

**NombreClase Identificador = new NombreConstructor ();**

**Producto p = new Producto ();**

Una clase siempre se puede siempre se puede instanciar (aunque esta esté vacía), PERO NUNCA SE DEBE HACER, SE DEBE INSTANCIAR UNA CLASE LLENA.

Lo que ocurre al instanciar una clase es que se reserva un espacio en la memoria del ordenador al que se le llama como el identificador. Y en ese espacio se guarda la dirección de memoria de esa clase.