COLLECTIONS

Collections

- Las colecciones son un tipo de **estructura dinámica** que permite el almacenamiento de un conjunto de objetos o elementos.
- En Java, las colecciones se emplean mediante la interfaz “**Collection**”, que permite implementar una serie de métodos comunes como ser: Añadir, Eliminar, Obtener el tamaño de la colección, etc.
- A partir de esta interfaz “**Collection**” se extiende otra serie de **interfaces genéricas**. Cada una de ellas aportan distintas funcionalidades sobre la interfaz principal.

Tipos de Collections

LIST

- La interfaz **LIST** nos permite agrupar una colección de elementos en forma de lista, es decir, uno detrás de otro.
- En **Java** existen diferentes tipos de listas, y entre las más conocidas se encuentran:

ArrayLists (o lista de arreglos)

LinkedLists (o listas enlazadas)

Stack(o pilas)

ArrayList

- Los **ArrayList**, tal como lo dice su nombre son arrays representados en forma de lista.
- Permiten **elementos duplicados**.
- Cumplen el orden **FIFO (First In First Out)** para la inserción y lectura de sus elementos.
- Permiten el acceso a un elemento en particular mediante un **índice** (tal como un array).

// Declaración e inicialización

```
ArrayList<String> listaPersonas = new ArrayList<String>();
```

// Carga

```
//forma 1
listaPersonas.add("messie");

//forma 2
String messi = "messie";
listaPersonas.add(messi);
```

// Recorrido

```
//forma 1
for (Persona pers : listaPersonas) {
     System.out.println ("Hola soy: " + pers.getNombre());
}

//forma 2
for (int i=0; i<listaPersonas.size(); i++) {
     System.out.println ("Hola soy: " + listaPersonas.get(i).getNombre());
}
```

LinkedList

- Las listas enlazadas o LinkedList se caracterizan por ser estructuras dinámicas que tienen punteros que permiten enlazar “doblemente” cada uno de sus elementos.
- Al igual a las ArrayLists, permite duplicados.
- Permite poder insertar o eliminar elementos tanto al principio como al final de la colección.
- Puede ser tratada no solo como una lista, sino también como una pila o una cola.

// Declaración e inicializalo ción

```
List<Persona> listaPersonas = new LinkedList<Persona>();
```

// Carga

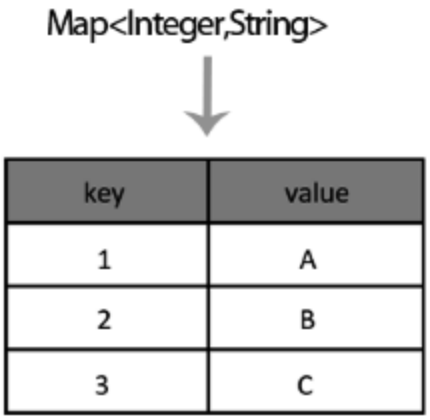
```
//Agregar al final
listaPersonas.add(new Persona(12333458,”Cristiano”, “Ronaldo”)); //igual que ArrayList

//Agregar al principio
listaPersonas.add(0, new Persona(12333458,”Cristiano”, “Ronaldo”));
```

// Recorrido

```
//A diferencia a los ArrayList, solo se puede recorrer de esta manera
for (Persona pers : listaPersonas) {
     System.out.println (“Hola soy: ” + pers.getNombre());
}
```

Mapas o Diccionarios

- En las collections de tipo **map** cada elemento se representa como un par {**key**, **value**} (clave/llave, valor).
- **Al agregar un valor, va a quedar vinculado a una key. Dicha key, posteriormente, nos permitirá recuperarlo, modificarlo o eliminarlo.**



// Declaración e inicialización

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

// Carga y eliminación

```
mapa.put(1,"Uno");
mapa.put(2,"Dos");
mapa.put(3,"Tres");
mapa.put(4,"Cuatro");

String valor = mapa.get(2); //traemos el valor asociado a la key 2
System.out.println(valor); //mostramos por pantalla el valor asociado a la key
mapa.remove(2); //eliminamos el elemento con la key 2
```

// Recorrido

```
for (Map.Entry<Integer,String> entrada : mapa.entrySet()) {
     Integer clave = entrada.getKey();
     String valor = entrada.getValue();
     System.out.println ("La clave es: " + clave + " El valor es: " + valor);
}
```