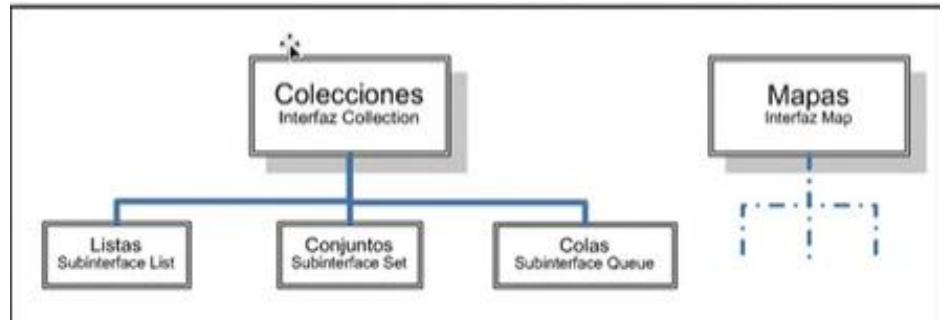


TEMA 12. COLECCIONES

¿Qué son las colecciones?

Almacenes de objetos dinámicos. La colección puede ampliarse o reducirse durante la ejecución del programa.

Inconvenientes: No podemos almacenar datos primitivos



Los arrays también son almacenes, pero no son dinámicos; ventajas respecto a los arrays:

- Pueden cambiar de tamaño dinámicamente.
- Podemos ordenar los objetos que están en una colección.
- Se puede insertar y eliminar elementos de forma dinámica

Hay que tener en cuenta el **framework** de colecciones, todas las clases e interfaces que hay al respecto.



1. Una lista (list).
2. Un conjunto (set).
3. Una cola (queue).
4. Un mapa (map).

extienden otra serie de interfaces genéricas. Estas subinterfaces aportan distintas funcionalidades sobre la interfaz anterior.

a) Una lista (list) es una colección de objetos donde cada uno de ellos lleva un índice asociado. Podríamos tener una lista con los nombres de las personas que han hecho una compra de un servicio: usuarios --> (Juan, Sara, Rodolfo, Pedro).

La **funcionalidad más importante** exclusiva de las listas es el **acceso posicional** a sus elementos por medio de índices.

Cada elemento va asociado a un índice, usuario(0) sería Juan, usuario(1) sería Sara...

En una lista podemos insertar y eliminar objetos de posiciones intermedias.

Puede haber **duplicados** y el **orden** en el que se encuentran los elementos es relevante.

b) Un conjunto (set) es una colección de objetos que **no admite duplicados**.

Siguiendo el ejemplo anterior, un conjunto nos serviría para saber los usuarios distintos que han comprado el servicio.

No tendríamos la información ordenada y una misma persona no aparecería más de una vez aunque hubiera utilizado el servicio varias veces.

b) Un conjunto (set) es una colección de objetos que **no admite duplicados**.

Siguiendo el ejemplo anterior, un conjunto nos serviría para saber los usuarios distintos que han comprado el servicio.

No tendríamos la información ordenada y una misma persona no aparecería más de una vez aunque hubiera utilizado el servicio varias veces.

c) Una cola (queue) es una colección de objetos que se comportan como lo haría un grupo de personas en la cola de una **caja de un supermercado**. Los objetos se van poniendo en cola y el primero en salir es el primero que llegó.

No tendríamos la información ordenada y una misma persona no aparecería más de una vez aunque hubiera utilizado el servicio varias veces.

c) Una cola (queue) es una colección de objetos que se comportan como lo haría un grupo de personas en la cola de una **caja de un supermercado**. Los objetos se van poniendo en cola y el primero en salir es el primero que llegó.

d) Un mapa (map) es una colección de objetos formados por **pares, clave-valor**. No hay duplicados. Un ejemplo de colección tipo mapa es un diccionario, las palabras son las claves, y el significado el valor.

Tipos de colecciones. Principales tipos de colecciones en Java.


List

La interfaz List define una **sucesión de elementos**. A diferencia de la interfaz Set, la interfaz List **sí admite elementos duplicados**. A parte de los métodos heredados de Collection, añade métodos que permiten mejorar los siguientes puntos:

- Acceso posicional a elementos: manipula elementos en función de su posición en la lista.
- Búsqueda de elementos: busca un elemento concreto y devuelve su posición.
- Iteración sobre elementos: mejora el Iterator por defecto.
- Rango de operación: permite realizar ciertas operaciones sobre rangos de elementos dentro de la propia lista.

Dentro de la interfaz List existen varios tipos de implementaciones. Las más típicas:

- ArrayList: esta es la implementación típica. Se basa en un array redimensionable que aumenta su tamaño según crece la colección de elementos. Es la que mejor rendimiento tiene sobre la mayoría de situaciones.

- 
- Iteración sobre elementos: mejora el Iterator por defecto.
 - Rango de operación: permite realizar ciertas operaciones sobre rangos de elementos dentro de la propia lista.

Dentro de la interfaz List existen varios tipos de implementaciones. Las más típicas:

- ArrayList: esta es la implementación típica. Se basa en un array redimensionable que aumenta su tamaño según crece la colección de elementos. Es la que mejor rendimiento tiene sobre la mayoría de situaciones.

Muy eficiente para recorrer pero ineficiente para insertar o eliminar elementos que se no encuentren en el final.

- usa internamente un array dinámico para almacenar los elementos.
 - Inserción y eliminación de elementos lenta
 - mejor opción **para almacenar y acceder a datos o elementos consecutivos**
 - **muy rápida en las operaciones que supongan recorrer la lista para la lectura o modificación de elementos.**
- LinkedList: esta implementación se basa en una lista doblemente enlazada de los elementos, teniendo cada uno de los elementos un puntero al anterior y al

Salvador Sánchez Fernández

- **ArrayList**: esta es la implementación típica. Se basa en un array redimensionable que aumenta su tamaño según crece la colección de elementos. Es la que mejor rendimiento tiene sobre la mayoría de situaciones.

Muy eficiente para recorrer pero ineficiente para insertar o eliminar elementos que se no encuentren en el final.

- usa internamente un array dinámico para almacenar los elementos.
 - Inserción y eliminación de elementos lenta
 - mejor opción **para almacenar y acceder a datos o elementos consecutivos**
 - **muy rápida en las operaciones que supongan recorrer la lista para la lectura o modificación de elementos.**
- **LinkedList**: esta implementación se basa en una lista doblemente enlazada de los elementos, teniendo cada uno de los elementos un puntero al anterior y al siguiente elemento.

Lista doblemente enlazada, lenta para recorrer pero sumamente eficiente para insertar o eliminar elementos.

Salvador Sánchez-Fernández