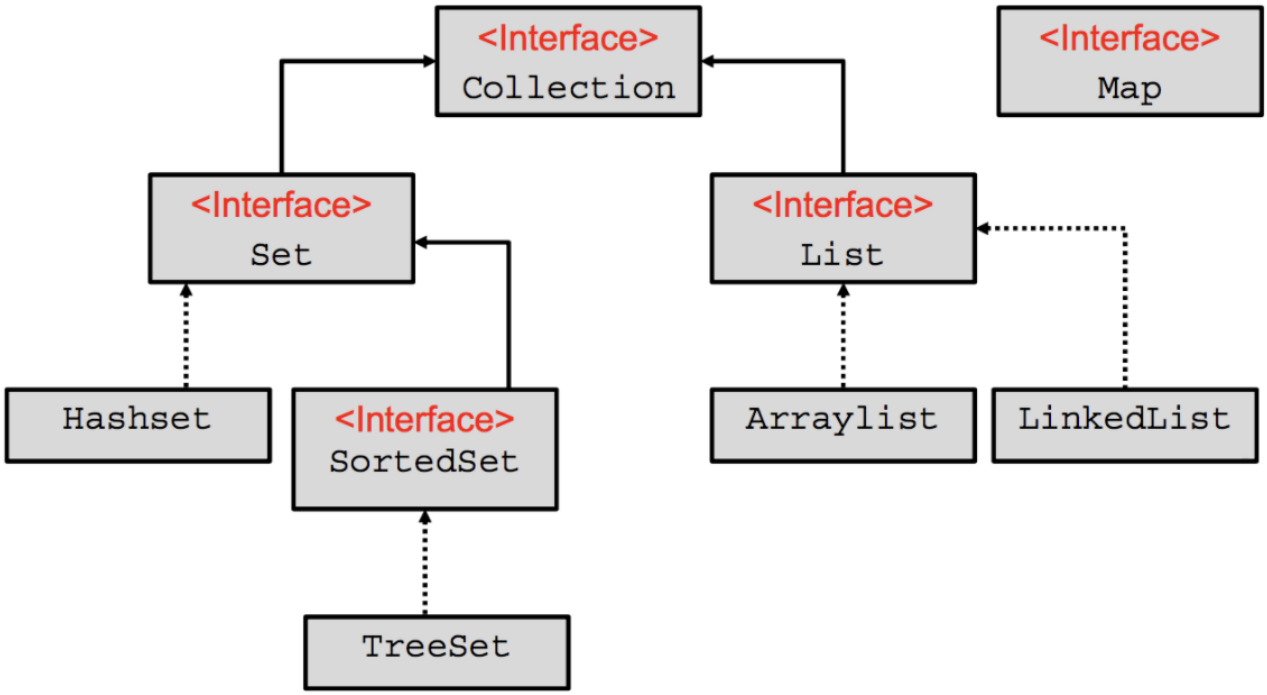
colecciones

anncode 14 de Febrero de 2018

En la clase anterior conociste las colecciones, aprendiste que estas nos ayudan a generar listas de objetos, únicamente objetos, para definir el tipo de objetos que almacenará la colección usamos la sintaxis diamante < >.

En Java existen colecciones que podemos usar para muchos tipos de casos, comenzaremos aprendiendo su árbol familiar:



Como podemos observar el elemento más alto es la interfaz **Collection**, para lo cual, partiendo de su naturalidad de interface, entendemos que tiene una serie de métodos “básicos” dónde su comportamiento será definido a medida

que se vaya implementando en más elementos. De ella se desprenden principalmente las interfaces **Set** y **List**.

La interface **Set** tendrá las siguientes características:

- Almacena objetos únicos, no repetidos.
- La mayoría de las veces los objetos se almacenarán en desorden.
- No tenemos índice.

La interface **List** tiene éstas características:

- Puede almacenar objetos repetidos.
- Los objetos se almacenan en orden secuencial.
- Tenemos acceso al índice.

Si seguimos analizando las familias tenemos que de **Set** se desprenden:

- Clase HashSet
- Interfaz SortedSet y de ella la clase TreeSet.

HashSet los elementos se guardan en **desorden** y gracias al mecanismo llamado hashing (obtiene un identificador del objeto) permite **almacenar objetos únicos.**

TreeSet almacena **objetos únicos**, y gracias a su estructura de árbol el **acceso** es sumamente **rápido.**

Ahora si analizamos la familia **List**, de ella se desprenden:

- Clase **ArrayList** puede tener duplicados, no está sincronizada por lo tanto es más rápida
- Clase **Vector** es sincronizada, los datos están más seguros pero es más lento.

- Clase **LinkedList**, puede contener elementos duplicados, no está sincronizada (es más rápida) al ser una estructura de datos doblemente ligada podemos añadir datos por encima de la pila o por debajo.

fig- doubly linked list

Como pudiste notar hay un elemento que pareciera no estar incluída en la familia, me refiero a la **interfaz Map**, está tendrá como naturaleza tener los componentes `Key, Value`, donde el Value será el objeto insertado y el Key será el valor clave para obtenerlo.

De Map se desprenden algunas otras interfaces y clases, cuéntale a tus compañeros en la sección de comentarios cuáles son y cómo funcionan.