**Colecciones**

**Definición:**

La interfaz raíz en la jerarquía de la colección. Una colección representa un grupo de objetos, conocidos como sus elementos. Algunas colecciones permiten elementos duplicados y otras no. Unos están ordenados y otros desordenados.

**Ventajas con respecto a Arrays:**

* Pueden cambiar de tamaño de forma dinámica.
* Pueden ordenarse los objetos que existen dentro de la colección.
* Se pueden insertar o eliminar elementos.

**Tipos de colecciones:**

Hay tres tipos de colecciones, cada uno con un interfaz común y diferentes implementaciones. Las diferentes implementaciones de un mismo interfaz realizan la misma tarea, aunque la diferencia está en que unas implementaciones son más rápidas en algunas operaciones y más lentas en otras:

**Set**

La interfaz Set define una colección que no puede contener elementos duplicados.

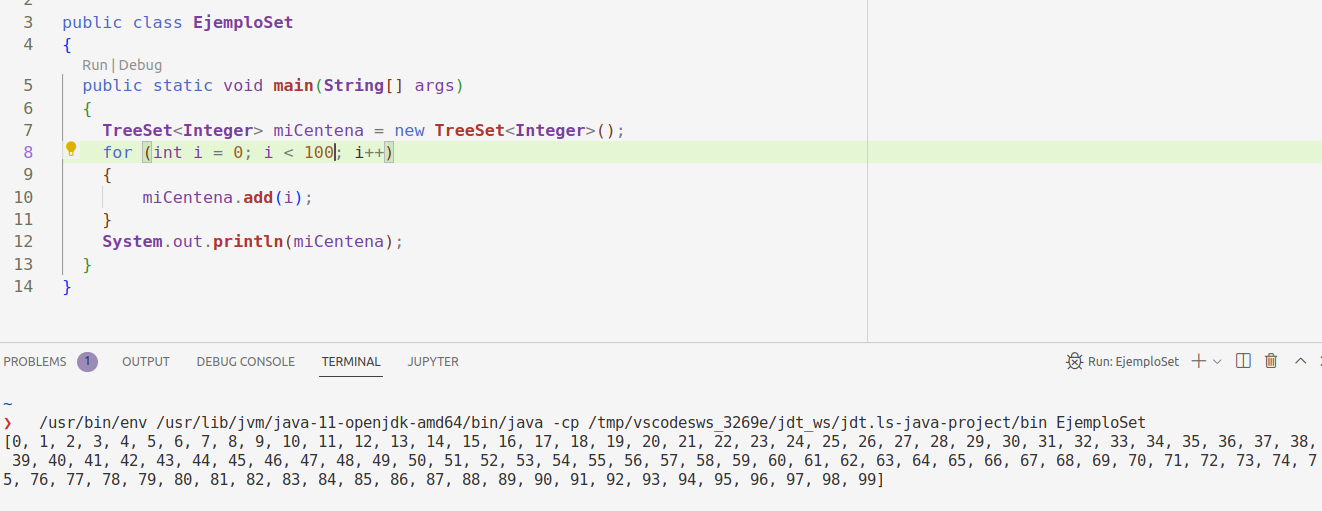
**Tipos de implementaciones de Set**

HashSet: esta implementación almacena los elementos en una tabla *hash*.

* Los elementos son almacenados por la clase HashSet usando el mecanismo hash.
* La clase HashSet solo permite elementos únicos.
* La clase HashSet permite valores nulos.
* No está sincronizado.
* Los elementos de la clase HashSet se insertan en función de su código hash, por lo que esta clase no mantiene el orden de inserción.
* Se desempeña mejor en las operaciones de búsqueda.

Treeset: esta implementación almacena los elementos ordenándolos en función de sus valores. Es bastante más lento que HashSet.

* Al igual que HashSet, la clase Java TreeSet solo contiene elementos únicos.
* Los tiempos de acceso y recuperación de la clase TreeSet son muy rápidos.
* No da acceso al elemento nulo.
* Mantiene el orden ascendente.
* No está sincronizado.



LinkedHashSet: esta implementación almacena los elementos en función del orden de inserción. Es, simplemente, un poco más costosa que HashSet.

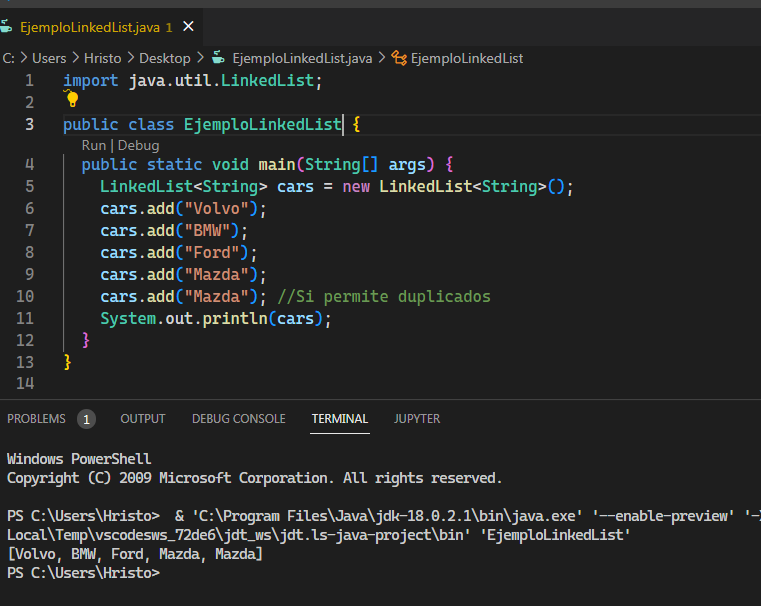
**List**

Una lista representa un grupo ordenado o secuenciado de elementos. Puede contener elementos duplicados.

* Los elementos se pueden insertar o recuperar por su posición en la lista. El valor de posición o índice comienza desde 0.
* La interfaz de lista define sus propios métodos además de los métodos de la interfaz de recopilación.

**Tipos de implementaciones de List**

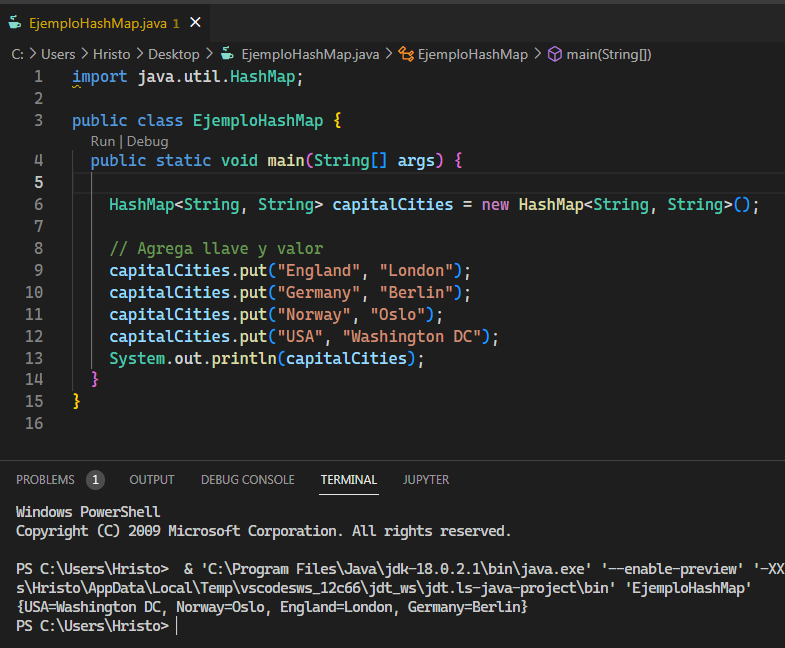
ArrayList: esta es la implementación típica. Se basa en un *array* redimensionable que aumenta su tamaño según crece la colección de elementos.

LinkedList: Tiene las mismas clases y métodos que ArrayList pero la diferencia con ArrayList es que LinkedLIst guarda los elementos en “contenedores” donde cada contenedor tiene un enlace al contenedor siguiente de la lista.

**Map**

Son estructuras de datos donde cada elemento tiene asociado una clave que usaremos para recuperarlo (en lugar del índice de una lista).

**Tipos de implementaciones de Map**

HashMap**:** Esta implementación proporciona todas las operaciones de mapa opcionales y permite valores nulos y la clave nula. (La clase HashMap es aproximadamente equivalente a Hashtable, excepto que no está sincronizada y permite valores nulos). Esta clase no garantiza el orden del mapa; en particular, no garantiza que el orden se mantenga constante en el tiempo.

TreeMap: El mapa es ordenado de acuerdo al ordenamiento natural de sus llaves.

LinkedHashMap: Implementación de tabla hash y lista enlazada de la interfaz Map, con orden de iteración predecible. Esta implementación se diferencia de HashMap en que mantiene una lista de enlaces dobles que se ejecuta en todas sus entradas. Esta lista enlazada define el orden de las iteraciones.