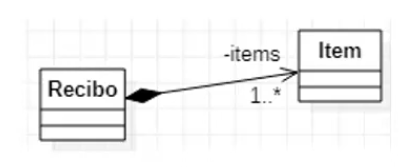
Programación Orientada a Objetos 1 – UNLP

Colecciones ~

Diagrama UML -> cardinalidad indica una colección cuando es 1..\* (1 a muchos).



List: elementos indexados desde el 0.

* Array List (no sincronizado) y Vector (sincronizado).

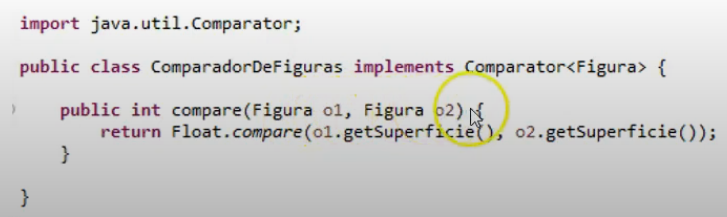
Set: no admite duplicados, sin indexación.

* TreeSet y HashSet.

Map: asocia objetos como claves para otros objetos que actúan como valores.

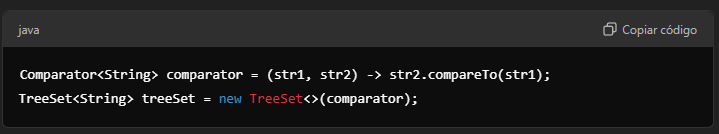
Queue: estructuras que indican el orden de entrada salida (LIFO, FIFO, priority…).

Al usar colecciones siempre hacemos las mismas operaciones casi siempre, por lo que queremos modelar las soluciones de la manera más genérica posible:



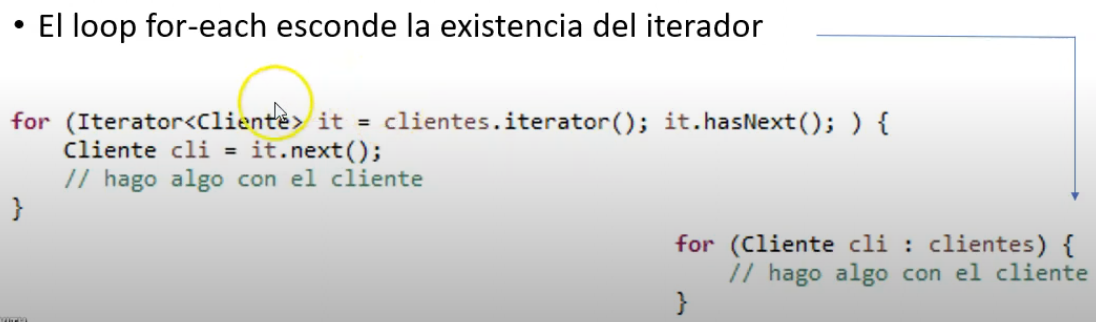
(Se llama con figuras.sort (new ComparadorDeFiguras());

Cuando una colección indexada recibe el comando “sort” espera en los parámetros un comparador (alguien que le explique cómo comparar).  
En un TreeSet se espera en el constructor el comparador, ya que lo contiene de entrada:



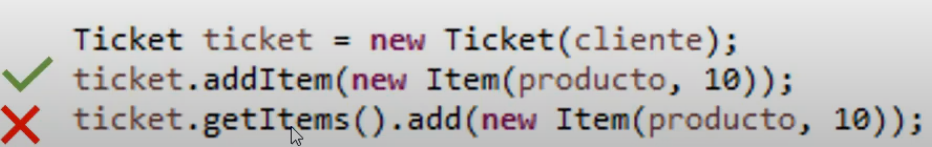
Recorridos:  
 Las colecciones entienden el mensaje iterator(), un iterador es polimórfico y encapsula el recorrido en 4 simples mensajes:

* hasNext();
* next() : E -> it.hasNext();
* remove();
* forEachRemaining(ConsumerAction);



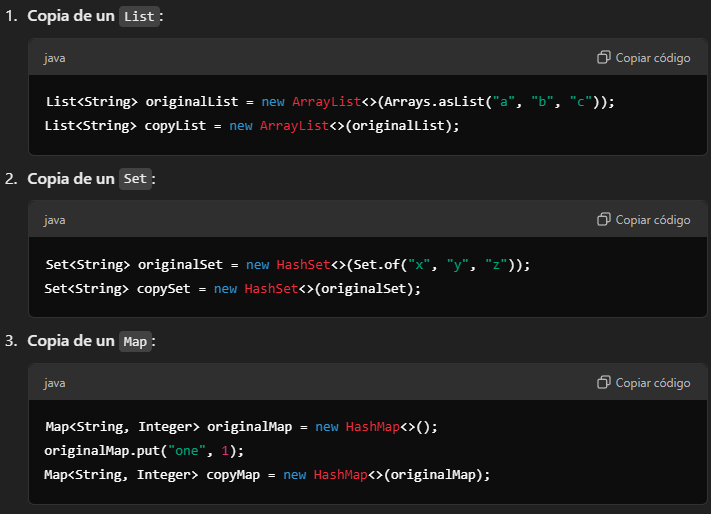
Son equivalentes.

Solo el dueño de una colección puede modificarla:



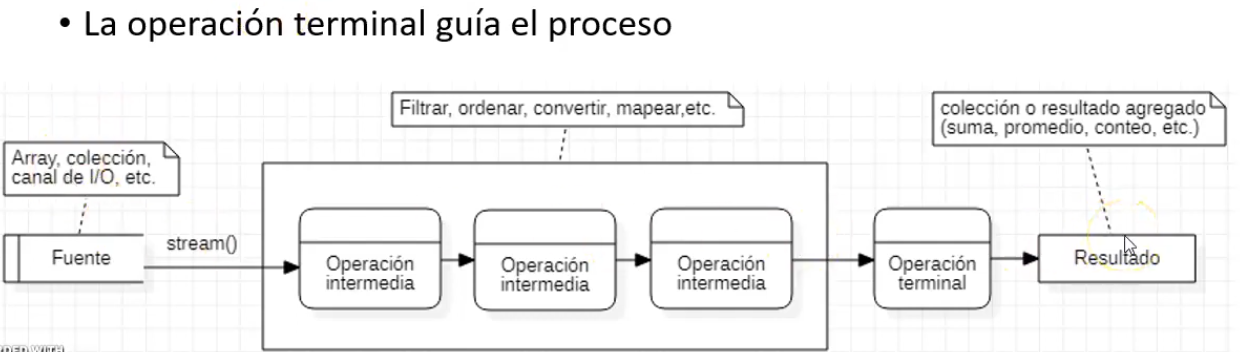
En la buena se usa el mensaje de ticket para avisarle que tiene que agregar un item, en el segundo se agrega “sin avisarle” o “por fuera” del objeto ticket, un item. **Debe respetarse encapsulamiento.**

Como corro el riesgo de que la colección sea modificada, debo crear una copia si necesitara la colección antes de ser modificada.

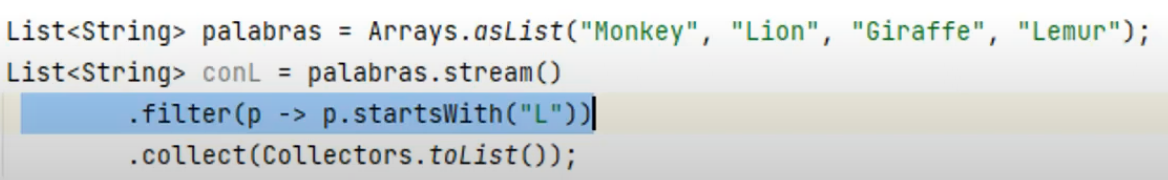


Streams:

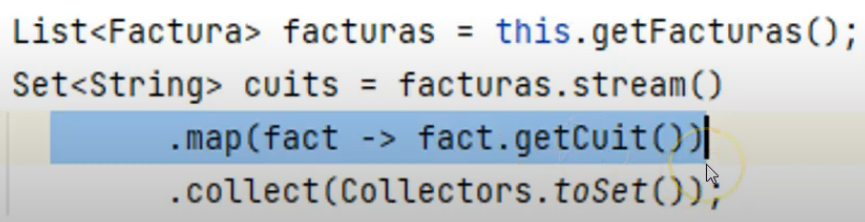
Mediante el mensaje stream() para obtener una colección, canal, I/O, etc) y tratarla como un flujo donde pasan uno a uno sin modificar el contenido original. “Río arriba” está la colección original y “río abajo” los resultados que pasaron por el stream de operaciones. Puede no tener final.



Intermedias opcionales, UNA terminal obligatoria que se dice “reductora”, me da algo en base a todo lo que pasó todo el stream: promedio, cantidad, nueva colección.



Dado un List<Strings> uso startsWith(“L”) y solo pasan las que arrancan con L, al final las recolecto con .collect(Collectors.tolist()) en la List<String> conL.



De la List<Facturas> creo un Set<String> (no se quiern duplicados -> set) que tomará de cada factura el cuit y lo recolectará en el set cuits.

