**COLLECTION**

For each loop:

It is like a for loop which repeatedly executes a group of statement for each element of the collection.

For(variable:collection-object)

{

Statement;

}

The loop is executed as many times as there are number of elements In the collection object.if collection object has n elements, the loop is executed exactly n times and the variable stores each element in each step.

Iterator interface:

It is an interface that contain methods to retrieve the elements one by one from the collection object.

It has three methods:

1. boolean hasNext(): it return true if the iterator has more elements.

2. element next(): it returns the next element in the iterator.

3. void remove(): it removes from the collection thet last element returned by an iterator.

ListIterator Interface:

It is an interface that contains methods to retrieve the elements from a collection object.

It has both forward and reverse directions.

It has five methods:

1. Boolean has next(): it returns true,when it has more element it traversing forward direction.

2. boolen hasPreviour(): it returns true,when is has more elements while traversing the list is reverse direction.

3. element next(): it returns next element.

4. element previous(): it returns the previous element.

5. void remove(): it removes from the list the last element that was returned by next().

HashSet Class:

It represents a set of element it does not guarantee the order of elements and does not allow the duplicate elements to be stored.

Class HashSet<T>

Where <T> represents the generic type parameter.

Suppose we want to create a hashset to store a group of string then we can create the object as,

HashSet<string>hs=new HasgSet <>();

Following constructors are available in HashSet:

1. HashSet();

2. HashSet(int capacity);

It represent how many elements can be stored into the HashSet.

3.HashSet(int capacity,float loadfactor);

It determines the point where the capacity of HashSet would be increased internally.

HashSet Class Method:

1. Boolean add(obj): adds an element obj to the HashSet and it returns true if the element is added to the HashSet, else it returns false.

2. Boolean remove(obj): it remove the element it is true if the element is removed.

3. void clear(): it removes all the elements from the HashSet.

4. Boolean contains(obj): the returns true is HashSet contains the specific element obj.

5. Boolean is Empty(): it returns true if the HashSet contains no element.

6. int size(): It returns the number of element present in the HashSet.

Stack Class:

* It represents a group of element stored in LIFO(last in first out)
* It means that the element which is stored as a last element into the stack will be the first element to be removed from the stack.
* Inserting element into the stack is class ‘PUSH’.
* Removing element from the stack is called ‘POP’.
* Searching for an element in the stack is called a ‘PEEP’.
* Inserting and deletion of element take place only from one side of the stack called ‘TOP’ of the stack.

Stack can be written as,

Class Stack<E> where E is element type

Suppose we want to create a stack object that contains integer objects.

Stack<integer> obj = new stack<>();

Stack Class Method:

1. Boolean empty(): it tests whether the stack is empty or not, if it is empty it is true.

2. element peek(): it return the top most object from the stack without removing it.

3. element pop(): it pops the top most elements from the stack and returns it.

4. element push(element obj): pushes an element obj onto the top of the stack an returns the position of an element obj from the top of the stack.

LinkedList Array:

It contains a group of element in the form of nodes each node will have three fields,

1. the data field contain data.

2. the link field contain references to previous>

3. next node.

Linked list is very convenient to store data inserting the element into the linked list and removing the element from the linked list is done quickly.

It can be written as:

Class LinkedList<E>

LinkedList Methods:

1. Boolean add (element obj) : it add the element to the linkedlist.

2. void add (int position, element obj) : it insert the element obj.

3. void addFirst(elements obj) : it add the element to the first position.

4. void addLast(element obj) : add the element to the last position.

5. element removeFirst() : it removes the first element.

6. element removeLast() : it removes the last element.

7. int Size() : it returns the number of element in the linkedList.

8. element getFirst() : it returns the first element .

9. element setLast() : it returns the last element.

10. element set (int positon, element obj) : it replaces the element at the specific position.

ArrayList Class:

It is like an array ,which can grow in memory dynamically which is not synchronized and threads acts simultaneously.

It can be written as

Class ArrayList<E> where E- Represent element to be stored in ArrayList

ArrayList Class Method:

1. boolean add (element obj) : the element is added to the end of the list.

2. void add (int position,element obj) : the element is added to the specific position.

3. element remove (int position) : it remove the element at the specific position.

4. Boolean remove(object obj) : it removes the first occurrence of the specific element present in the ArrayList.

5. element Set() : it replace an element at the specific position.

Vector Class:

A vector also stores element similar to ArrayList,but vector Synchronized.

Vector Class Method:

It is same as ArrayList.

Queue Interface:

A queue represent the group of element where the first element that enter the queue will leave first from the queue (FIFO).

In some cases,FIFO order is not compulsory.

Methods of queue interface:

* To add an element in queue it has two methods

1. add() : add the element.

2. offer() : add the element successfully and false if it fails.

* To delete an element from the beginning of the queue ,we can use two methods

1. remove() : delete the beginning element.

2. poll() : delete and returns the first elements of the queue.

* To retrieve an element without deleting,we can use element() and peek() .

1. element() : returns the first element .

2. peek() : returns the first elements and returns null if the queue is empty.

* To return the number of elements In the queue ,we can use the size() method.
* To remove all the elements from the queue,we can clear() method.
* To check whether or not the queue we can use empty() method.

HashMap Class:

It is collection that stores element in the form of key value pairs.the key should be unique.it means we cannot use duplicate data for keys in the HashMap and it is not synchronized.

Class HashMap<K,V> where K-Key V-type of value element

HashMap Class Method:

1. value put(key,value) : it stores key value pair.

2. value get(object key) : it returns the corresponding value when key is given.

3. set<k> key set() : when applied on a HashMap converts it into a set.

4. collection<V> values() : when applied on a HashMap object returns all the value of the HashMap.

5. value remove(object key) : remove the key and corresponding value.

6. value clear() : removes all the key pairs.

7. Boolean isEmpty() : it returns true when there is no key value pair.

8. int size() : returns number of keys value pair.

HashTable Class:

It is similar to HashMap which can be store element in the form of key vaule pairs but it is synchronized,even multiple threads act on it simultaneously.

HashTable Class Method:

It is same as HashMap class method.

**STREAMS**

If we want to represent a group of objects as a single entity then we should go for a collections

But if we want to process objects from the collections then we should go for streams.

Streams are included I the package java.util.streams

A stream is not a data structure instead it takes input from the collections, arrays or I/O channels

Streams don’t change the original data structure, they only provide the result as per the pipelined method.

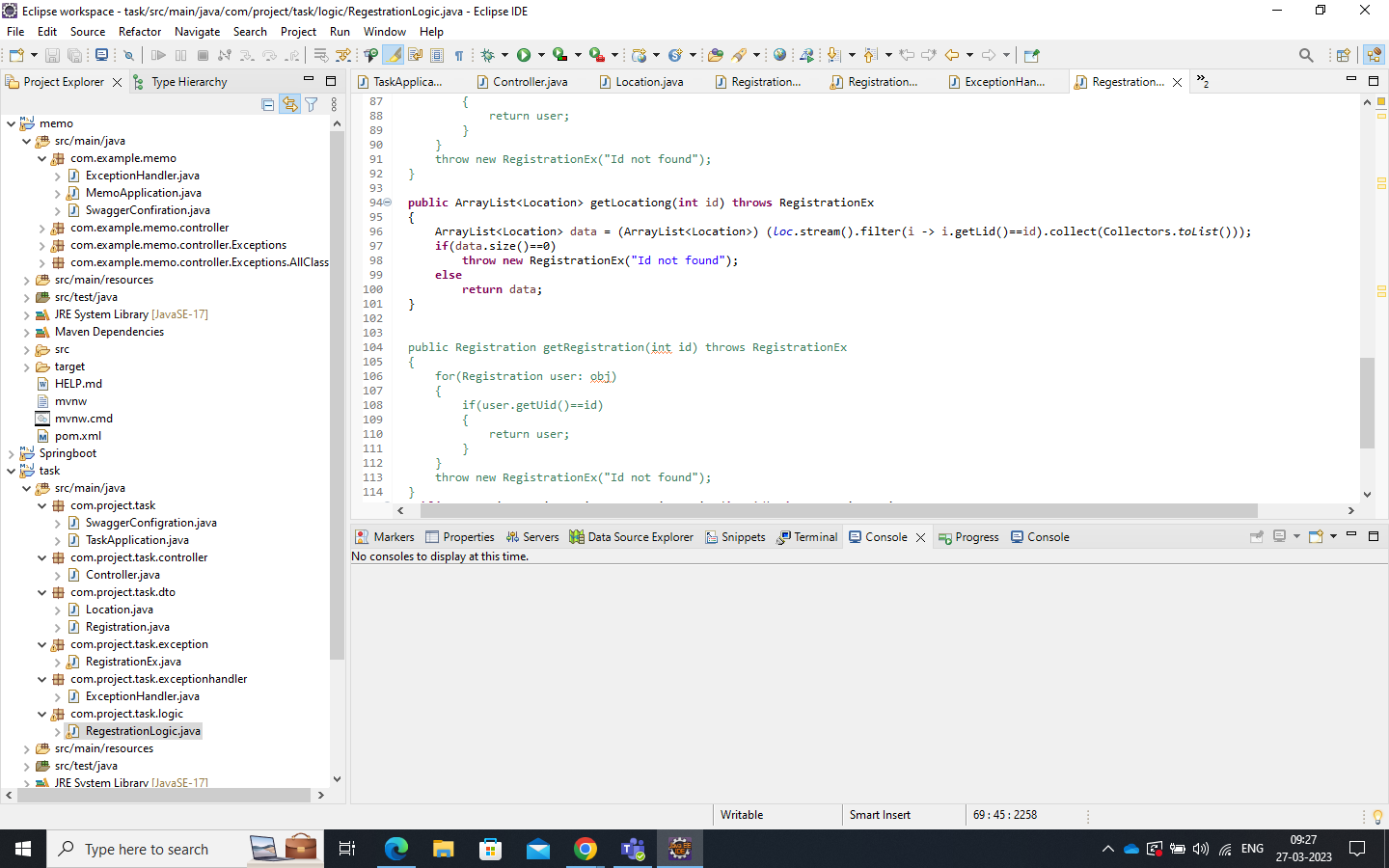
**Filter() method :-**

Filter method used to filter the required objects in the given colllections.

To filter out from the objects we have the fuction filter()

To collect elements we will be using Collectors.toList().

Lastly we will store these elements in a list and display the output on the console.



**Map() Method**

Map method is used change the entire objects i.e we can add something to the objects it will reflect in all the objects in the collection.

Example :- adding 6 to list<Integer>

List<Integer> updatedMarks = marks.stream().map( i->I+6).collect(Collectors.toList())