Collection

Q)Why do we need collection framework in Java?

Ans)To store multiple objects or group of objects together we can use Array, But, Array has some limitations.

Limitation of Arrays:-

1) The size of the array is fixed, we cannot reduce or increase dynamically during the execution of the program.
2) Array is a collection of homogeneous elements.
3) Array manipulation such as:-

    a)Removing an element from an array,

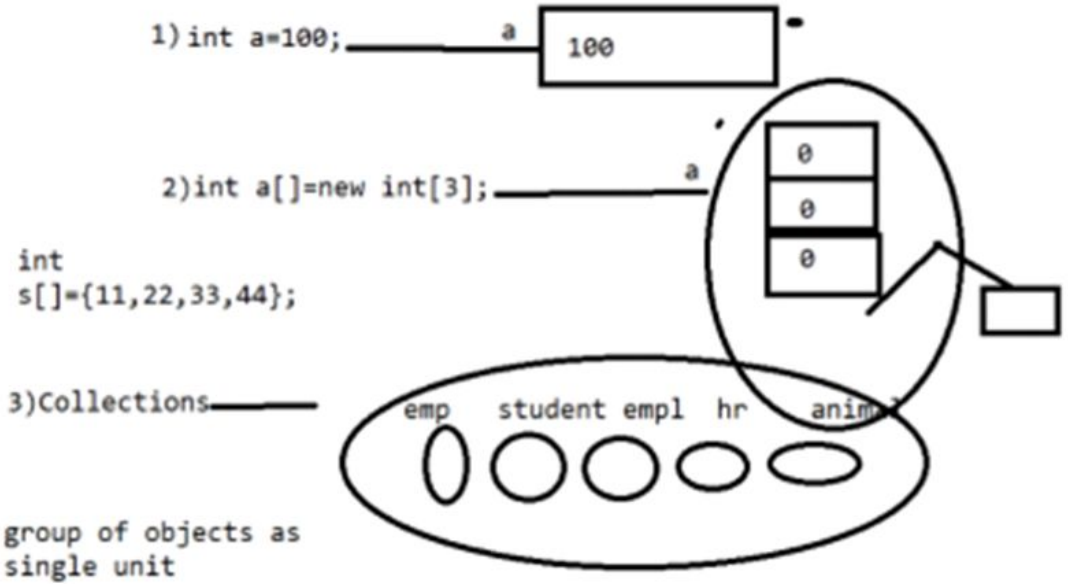
    b)Adding the element in between the array etc

require complex logic to solve. Therefore to avoid the limitations of the array we can store the group of objects or elements using different data structure such as:-

1. List 2. Set 3. Queue 4. Maps/Dictionary

Collection

- Collection is a group of objects represents as single unit.
- Its main purpose is to store huge amount of data.
- It provides multiple API(methods) to store and manipulate data.



Collections Framework

- It is an Architecture, which provides the group of classes and interfaces, which is used to store multiple objects as single unit.
- All the classes and interfaces of Collections Framework are present in java.util package.

CRUD operations

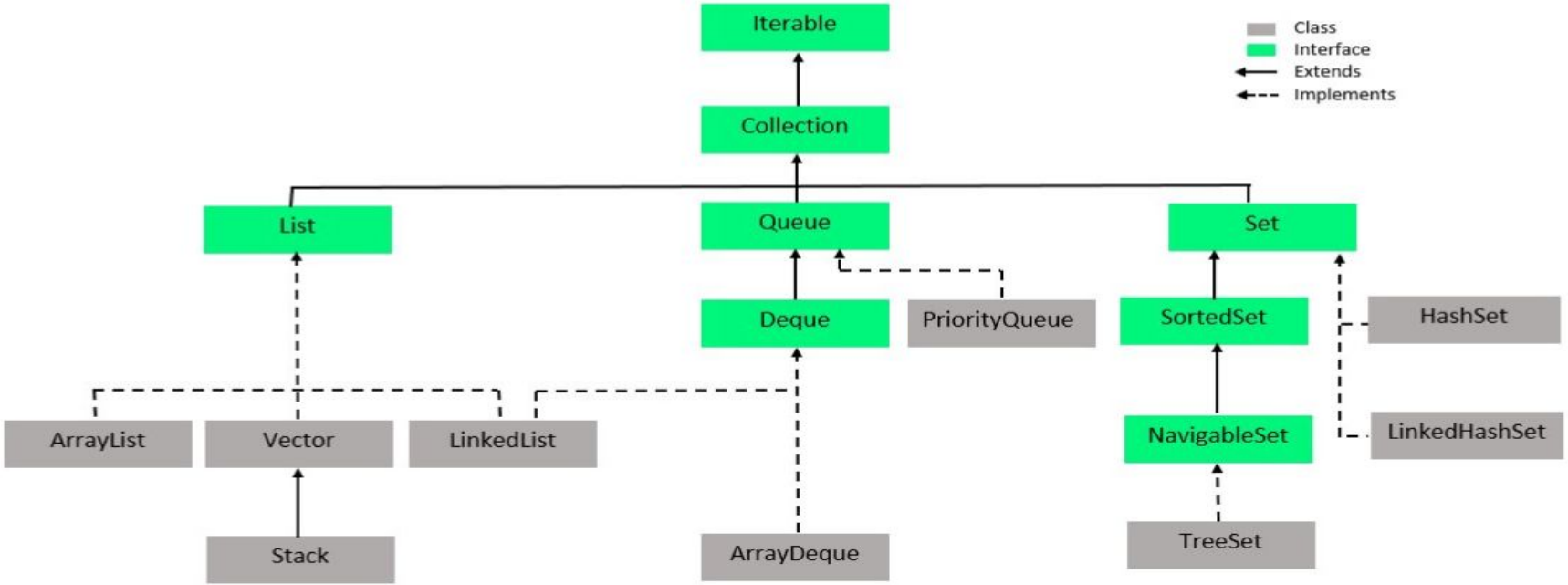
It also provides mechanism to perform action such as:-

1. Create and add an element
2. Access the element
3. remove / Delete the elements
4. Search elements
5. Update elements
6. Sort the elements

Important Hierarchies of collection framework:-

1. Collection Hierarchy
2. Map Hierarchy

Collection Hierarchy



Collection Interface

1)Collection is an interface defined in java.util.

2)Collection interface provides the mechanism to store group of objects(elements) together.

3) All the elements in the collections are stored in the form of objects(ie, only non-primitive is allowed)

4)It helps programmer to perform following operations

1. Add an element into the collection
2. Search an element in the collection
3. Remove an element from the collection
4. Access the element present in the collection

Most common methods for all collection objects.

1)isEmpty()

2)add(Object ref)

3)addAll(Collection ref)

4)size()

5)remove(Object ref)

6)contains(Object ref)

7)removeAll(Collection ref)

8)retainAll(Collection ref)

9)containsAll(Collection ref)

10)clear()

11)toArray()

12)iterator(

We can classify collection into 2 categories:-

1. Non-generic collection
2. Generic collection

Non-generic collection

- It is a heterogeneous (different types) collections of elements.
- Every elements is converted and stored as java.lang.Object class type.

  Syntax to create reference variable for Generic type

  Collection_Type<NonPrimitiveDataType> variable;

Generic collection

- It is a homogeneous(same type) collection of elements.

  Syntax to create reference variable for Non-generic type

  Collection_Type variable;

List Interface

1)It is a child of Collection Interface

2)It is used to store group of objects together as a single entity where duplicates are allowed and insertion order is preserved.

3)Index plays an important role here.

List interface contains all methods of Collection interface. It also contains following methods,

1)add(int index,Object ref)

2)addAll(int index,Collection ref)

3)remove(int index)

4)lastindexOf(Object ref)

5)indexOf(Object ref)

6)get(int index)

7)set(int index,Object ref)

8)listIterator()

ArrayList

It is a concrete subclass of List interface.

Characteristics of ArrayList

- accepts duplicate objects
- it maintains insertion order
- iteration order predictable
- dynamically growable or shrinkable
- accepts both homogeneous and heterogeneous

Steps to create ArrayList

1) import arraylist class from java.util package
2) create an object

Constructors of ArrayList

ArrayList()

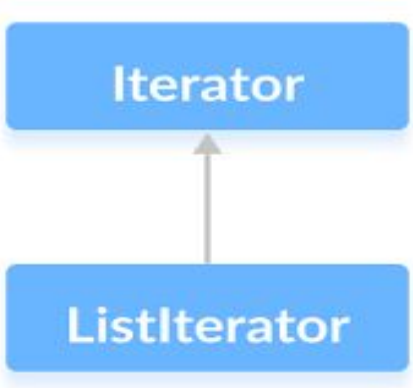
ArrayList(Collection ref)

ArrayList(int initialcapacity)

To access the elements in the arraylist we can access element in arraylist with the help of iterator, listiterator,get() and foreach loop.

1) Iterator

Iterator is used to iterate through each and every element in the collection.



- Iterator hierarchy is defined in java.util package.
- Iterator is used to access the elements present in the collection object.

Iterator interface provides following abstract methods:-

1)hasNext()-> It is used to check weather there are more elements for iteration. If yes it returns true, if not it returns false.

2)next()->It returns an element pointed by the cursor and moves the cursor forward.

3)remove()->It removes an element from the collection which was most recently returned by the iterator.

Note:-

- We can access element from the collection which was most recently returned by the iterator.
- We can access element of any collection with the help of iterator by creating an object of iterator type.
- We can create an object of iterator type by calling iterator() of the collection object.

Steps to create Iterator

step1: Import Iterator from java.util package.
step2: create an Iterator with the help of iterator() method.
step3:Using hasNext() and next() iterate through collection as shown below.

ListIterator

- The ListIterator interface of the Java collections framework provides the functionality to access elements of a list
- It is bidirectional. This means it allows us to iterate elements of a list in both the direction.
- It extends the Iterator interface.

ListIterator interface provides following abstract methods:-

1)hasNext()-> It is used to check weather next elements to be allowed is present or not.

2)next()->It returns an element pointed by the cursor and moves the cursor forward.

3)remove()->It removes an element from the collection which was most recently returned by the iterator.

4) previous()->It gives the previous element pointed by the cursor and removes the cursor backward.

5)hasPrevious()-> to check whether previous element to be allowed or not.

ArrayList specific methods

1)contains(Object ref)->It is used to check whether given element is present in the collection or not.

2)containsAll(Collection ref)-> It is used to check whether the ArrayList contains all the elements of given arraylist.

3)clear() -> It is used to remove all the elements from the arraylist.

4)indexOf(Object ref) ->It gives the index of given object(first occurence).

5)lastIndexOf(Object ref) -> It gives the index of given object(last occurence).

6)set(int index,Object ref) -> It is used to replace the element present at specified index with the given element.

7)remove(Object obj)->Removes given object

8)remove(int index) ->Removes object based on given index

Advantages of ArrayList:

1. ArrayList is variable length.
2. Add any type of data into ArrayList.
3. Traverse in both directions.
4. Insert and remove elements also at particular position of ArrayList.
5. ArrayList allows multiple null values.
6. ArrayList allows to add duplicate elements.
7. ArrayList has many methods to manipulate stored objects.
8. Retrieval is faster in ArrayList

Disadvantages of ArrayList:

1. If a data entry is added to or removed from an array-based list, data needs to be shifted to update the list.
2. Time taken to insert and remove element will be more .

LinkedList

- LinkedList is concrete subclass of List interface.
- It belongs to java.util package.

Characteristics of LinkedList

1)Maintains the insertion order
2)Iteration order is predictable
3)Dynamically growable or shrinkable
4)Accepts duplicate objects
5)Can store both homogeneous as well as heterogeneous objects.

Steps to create a Linked List

step1:- import LinkedList class from java.util package.
step2:- Create an object using the constructors.

Constructors of LinkedList

1)LinkedList()

2)LinkedList(Collection ref)

LinkedList specific methods

1)addFirst(Object ref)

2)addLast(Object ref)

3)getFirst()

4)getLast()

5)removeFirst()

6)removeLast()

Advantages of Linked List

1. The linked list is a dynamic data structure.
2. We can also decrease and increase the linked list at run-time ie, we can allocate and deallocate memory at run-time.
3. We can easily perform insertion and deletion operations.
4. Memory is well utilized in the linked list, because we do not have to allocate memory in advance.
5. Its access time is very fast, and it can be accessed at a certain time without memory overhead.
6. You can easily implement linear data structures using the linked list like a stack, queue.

Disadvantages of Linked List

1. The linked list requires more memory to store the elements than an array, because each node of the linked list points a pointer, due to which it requires more memory.
2. It is very difficult to traverse the nodes in a linked list. In this, we cannot access randomly to any one node. (As we do in the array by index.) For example: - If we want to traverse a node in an n position, then we have to traverse all the nodes that come before n, which will spoil a lot of our time.
3. Reverse traversing in a linked list is very difficult, because it requires more memory for the pointer.

Difference between ArrayList and LinkedList

ArrayList

1) ArrayList internally uses a dynamic array to store the elements.

2)ArrayList is better for storing and accessing data.

3)The memory location for the elements of an ArrayList is contiguous.

4)Generally, when an ArrayList is initialized, a default capacity of 10 is assigned to the ArrayList.

5)ArrayList is a resizable array.

LinkedList

1)LinkedList internally uses a **doubly linked list** to store the elements.

2)LinkedList is **better for manipulating** data.

3)The location for the elements of a linked list is not contagious.

4)There is no case of default capacity in a LinkedList. In LinkedList, an empty list is created when a LinkedList is initialized.

5)LinkedList implements the doubly linked list of the list interface.

Vector

- It is a concrete subclass of List interface.
- It belongs to java.util package.
-

Characteristics of Vector

1)Maintains the insertion order
2)Iteration order is predictable
3)Dynamically growable or shrinkable
4)Accepts duplicate objects
5)Can store both homogeneous as well as heterogeneous objects.

Note:- Vector came before ArrayList,List and Collection(I). i.e, in java 1.0 itself. After java 1.2 vector is the child of Collection(I) and List(I). Therefore it has 3 methods for adding an element :-

1)add(Object ref) --->from Collection(I)
2)add(int index,Object ref)--->from List(I)
3)addElement(Object ref) --->Vector f

For removing,

1)remove(object ref)-->from Collection(I)

2)remove(int index)-->from List(I)

3)removeElement(Object ref)-->from Vector

4)clear()-->from Collection(I)

5)removeAllElements()-->from Vector

For Accessing elements,

1)get(int index) -->from List(I)

2)elementAt(int index) -->from Vector

3)firstElement()-->from Vector

4)lastElement()-->from Vector

Stack

- Stack is subclass of Vector.
- It belongs to java.util package.
- It is specially designed for First in last out.

Constructor

Stack()

Methods

1)push(Object obj) -> It is used to insert an object to the stack.

2)pop() -> It is used to remove and return top of the stack.

3)peek() -> Returns the top of the stack without removal of object.

SET

1.Do not store duplicate objects.

2.If insertion order is not required.

3. It belongs to java.util package.

HashSet

HashSet is the concrete subclass of Set(I).

Characteristics of HashSet

1.Doesn't accept duplicate objects.

2.Doesn't maintain insertion order.

3.We cannot predict iteration order.

4.We can add and remove elements.

5.Accepts both homogeneous as well as heterogeneous objects.

Constructors

1.HashSet()

2.HashSet(Collection ref)

TreeSet

- TreeSet is a concrete subclass of Set(I)
- It belongs to java.util package.

Characteristics of TreeSet

1.Doesn't accept duplicate objects.

2.TreeSet gives the objects in ascending order.

3.We can add and remove elements.

4.Accepts homogeneous objects.

Constructors

1.TreeSet()

2.TreeSet(Collection ref)

3.TreeSet(Comparator ref)

LinkedHashSet

- LinkedHashSet is an implementing class of Set interface.
- It belongs to java.util package.

Characteristics of LinkedHashSet

1.Doesn't accept duplicate objects.

2.maintains Insertion order.

3.We can predict iteration order.

4.We can add and remove elements.

5.Accepts homogeneous and heterogeneous objects.

Queue

Queue(I) extends Collection(I).

It belongs to java.util package.

Queue is specially designed for FIRST IN FIRST OUT.

Methods

1.offer(Object) ->It is used to add the element to the queue.

2.poll() -> It is used to removes and returns the element from the queue. 3

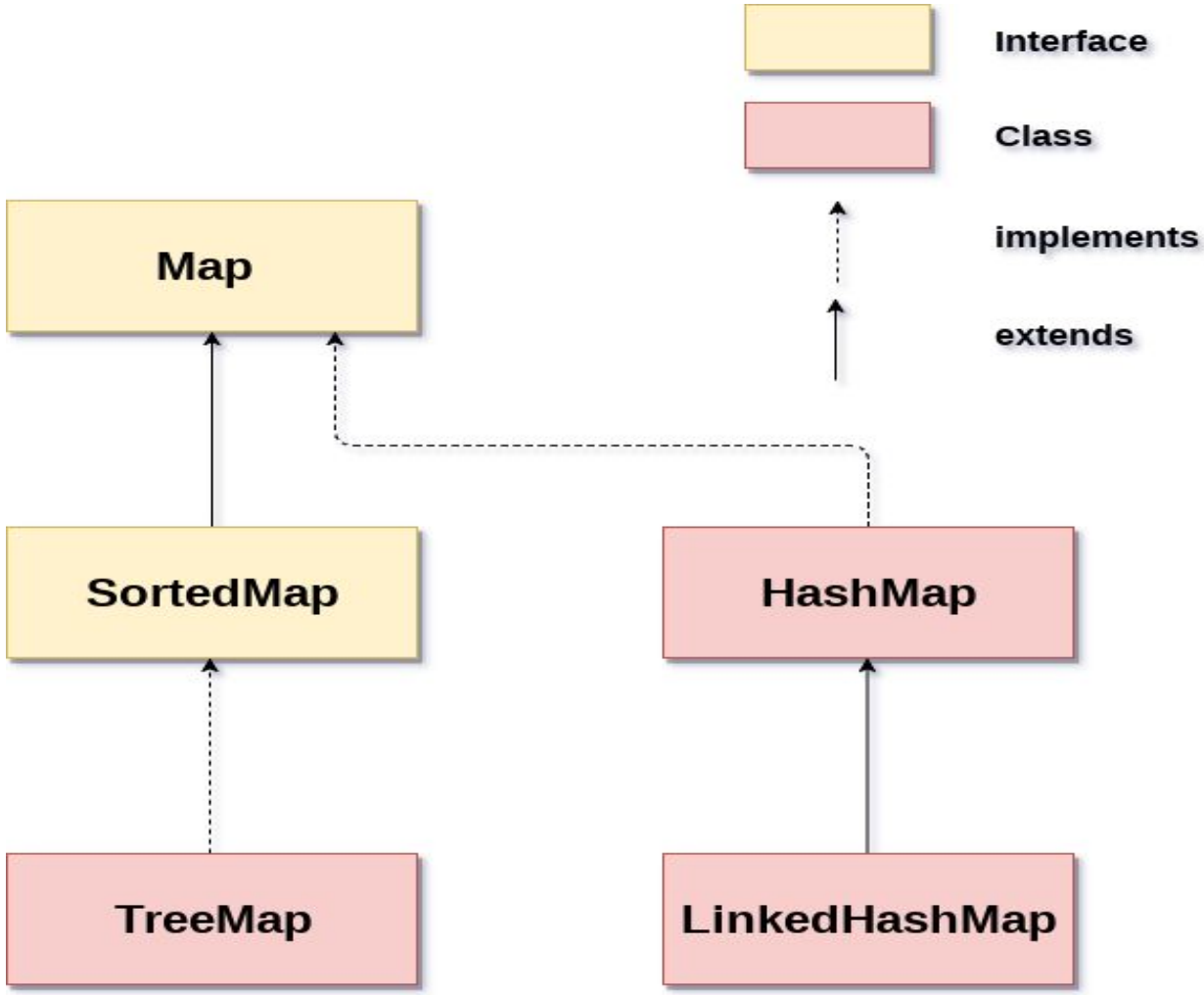
3.peek() ->It is used to fetch the object which is ready to be removed.

LinkedList and PriorityQueue are implementing classes of Queue interface.

PriorityQueue

- PriorityQueue is an implementing class of Queue interface.
- It is used to process the objects based on priority.

MAP


Interface
Class
implements
extends
Map
SortedMap
HashMap
TreeMap
LinkedHashMap

Map

Map is an interface, which is defined in java.util package.

Map interface helps to store the data in the form of key-value pairs.

Each key value pair in a Map is called Entry.

Eg:-RollNumber-Student, Eid-Employee etc.

Characteristics of Map

1)A map can cannot contain duplicate keys.

2)Map can contain duplicate values.

3)Each key can contain at most one value, and not more than one.

Map interface allows 3 types of views

1)A set of keys (As keys cannot be duplicate)

2)A List of values(As values can be duplicate)

3)A set of key-value Mapping

Methods

1)put(k,v)
2)putAll(Map)
3)size()
4)clear()
5)isEmpty()
6)containsKey(key)
7)containsValue(value)
8)get(key)
9)equals(Object)
10)remove(key)
11)keySet()
12)values()
13)entrySet()

HashMap

HashMap is an implementing class of Map interface.

HashMap is present in java.util package.

Characteristics

1)Does not maintain insertion order of the entries

2)We cannot predict the iteration order.

TreeMap

1)TreeMap doesn't maintain insertion order.

2)It stores the elements as per the natural sorting order of keys.

3)We can predict the iteration order(As per the natural sorting order of keys).

4)Doesn't accept heterogeneous keys.

5)Introduced in 1.2v

LinkedHashMap

1)It was introduced in 1.4v

2)Only heterogenous data allowed

3)Data structure is hashtable

4)Duplicate keys are not allowed but values can be duplicate,if we add duplicate key it

5)replaces with original one.

6)As per Insertion order

7)only one null key is allowed and multiple null values are allowed.