**4th unit: collection framework**

What is Collection in java

Collection represents a single unit of objects i.e. a group.

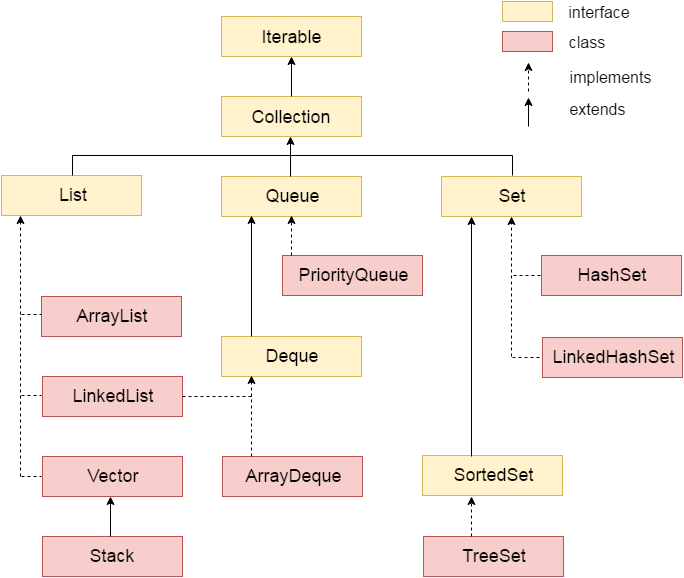
What is framework in java

* provides readymade architecture.
* represents set of classes and interface.

All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc. can be performed by Java Collections.

Hierarchy of Collection Framework

The **java.util** package contains all the classes and interfaces for Collection framework.



common methods in all the classes:

1.add():

2.remove():

3.removeAll():

4.clear();

5.size();

1.ArrayList:

ArrayList implements List

{

}

1.dynamic array

2.duplicate data

3.maintains the insertion order

4.non synchronized class

5.random access is possible

6.manipulate the data ,it is slow

---------------------------------------------------

there are two ways to traverse /iterate collection elements

1.for-each loop

for( datatype var:collection)

{

}

2.Iterator interface.

1.hasNext():it will return true,

2.next():

3.remove():

-------------------------------

1.ArrayList class:

import java.util.\*;

class demo

{

public static void main(String args[])

{

ArrayList <String> obj=new ArrayList <String>();

obj.add("padma");

obj.add("sreyas");

obj.add("college");

obj.add("hyd");

obj.remove("padma");

obj.removeAll();

obj.clear();

System.out.println(obj.size());

System.out.println("display data");

for( String i:obj)

{

System.out.println(i);

} } }

----------------------------------------------

by using Iterator interface:

import java.util.\*;

class test1

{

public static void main(String args[])

{

ArrayList <String> obj=new ArrayList <String>();

obj.add("padma");

obj.add("sreyas");

obj.add("college");

obj.add("hyd");

System.out.println("display data");

Iterator itr = obj.iterator();

while(itr.hasNext())

{

System.out.println(itr.next());

}

}

}

-----------------------------------------------------------------

LinkedList class: uses double linked list to store the elements.

2. add and remove the elements from both the sides

3. manuplation is fast as compared to arraylist

methods:

1.void addFirst(object):

2.void addLast(object)

3.object getFirst()

4.object getLast();

5.remove();

class test1

{

public static void main(String args[])

{

LinkedList<String> obj=new LinkedList <String>();

obj.add("padma");

obj.add("sreyas");

obj.add("college");

obj.add("hyd");

obj.addFirst("cseA");

OBJ.addLast("cseC");

System.out.println(obj.getFirst());

System.out.println(obj.getLast());

System.out.println("display data");

//for(String i:obj)//

// System.out.println(i);//

Iterator itr = obj.iterator();

while(itr.hasNext())

{

System.out.println(itr.next());

} }

}

-------------------------------------------------

3.HashSet class: class implements Set interface

1. unique values stored

2.stored in the hash table by using hashing techique.

import java.util.\*;

class test1

{

public static void main(String args[])

{

HashSet<String> obj=new HashSet<String>();

obj.add("gopal");

obj.add("sunitha");

obj.add("college");

obj.add("hyd");

System.out.println("display data");

Iterator itr = obj.iterator();

while(itr.hasNext())

{

System.out.println(itr.next());

} }

}

------------------------------------------------------

TreeSet class: it allows to store the unique data

2. stored as at different levels

3.data is maintained in ascending order

import java.util.\*;

class test1

{

public static void main(String args[])

{

TreeSet<String> obj=new TreeSet<String>();

obj.add("gopal");

obj.add("sunitha");

obj.add("college");

obj.add("hyd");

System.out.println("display data");

Iterator itr = obj.iterator();

while(itr.hasNext())

{

System.out.println(itr.next());

} }

}

op: college

gopal

hyd

sunitha

-----------------------------------------------

PriorityQueue class:

1.remove() :to retrive and remove the head element

2.poll():to retrive and remove the head element or it returns null if queue is empty

3.element():to retrive but doesnot remove the head element

4. peek():to retrive but doesnot remove the head element or it returns null if queue is empty

import java.util.\*;

class test1{

public static void main(String args[])

{

PriorityQueue<String> obj=new PriorityQueue<String>();

obj.add("gopal");

obj.add("sunitha");

obj.add("college");

obj.add("hyd");

obj.poll();

System.out.println(obj.peek());

System.out.println("display data");

//for(String i:obj)//

// System.out.println(i);//

Iterator itr = obj.iterator();

while(itr.hasNext())

{

System.out.println(itr.next());

} }

}

------------------------------------

ArrayDeque class: is double ended queue

2.add and remove the data at both the sides

3.no capacity restriction

4.faster than linked list.

methods:1.add()

2.remove();

3.size();

1.offerFirst():insert at front side

2.offerLast():insert at rear side

3.pollFirst():delete at front side

4.pollLast(): delete at rear side

import java.util.\*;

class test{

public static void main(String args[])

{

ArrayDeque<String> obj=new ArrayDeque<String>();

obj.add("arvind");

obj.add("vimal");

obj.add("mukul");

obj.offerFirst("jai");

obj.offerLast("padma");

System.out.println("After offerFirst and offerLast Traversal...");

for(String s:obj)

{

System.out.println(s);

}

obj.pollFirst();

obj.pollLast();

System.out.println("After pollLast() Traversal...");

for(String s:obj){

System.out.println(s);

} }

}

----------------------------------------------------------------