Assignment - 13

Problem Statement -

Write a JAVA program for the implementation of different data structures using JAVA collection libraries atleast 5 data structures, to be used to design a suitable application.

Objective -

- To understand the use of JAVA collection libraries.
- To be able to use a JAVA library.
- To use JAVA collection for implementing different types of data structures.

Outcome -

We will be able to use JAVA collection libraries in an application.

H/W & S/W Requirements -

Dell Optiplex 3020 MT, monitors, keyboard, mouse, Fedora 20 OS, Eclipse.

Theory -

The JAVA collections framework is a set of classes and interfaces that implement commonly reusable collection data structures.

It works in manner of a library.

Implementation for fundamental collection. The framework had to allow different types of collection to work in a similar manner with high degree of interoperability.

Types of interface -

1. Collection interface
2. List Interface
3. Set
4. ~~Linked~~ Sorted Set
5. Map
6. Map Entry
7. Sorted Map
8. Enumeration.

Paeudocode -

1. Stack -

```
public void Stack(){
    Stack<Integer> st = new Stack<Integer>();
    Print("1. Push \n 2. Pop \n");
    ~~scr~~ accept(ch);
    if (ch == 1)
        st.push(new Integer(sl.nextInt()));
    if (ch == 2)
        st.pop();
    if (ch == 3)
        st.peak();
}
```

② Priority Queue -

```
public void Queue() {
  PriorityQueue<Integer> pq = new
  PriorityQueue<Integer>();
Print(" 1. Add data \n 2. Poll \n
      3. Display head \n 4. Size ");
accept(ch);
if (ch == 1)
  pq.add(new Integer(sc.nextInt()));
if (ch == 2)
  pq.poll();
if (ch == 3)
  pq.peek();
if (ch == 4)
  pq.size();
else
  Print "Invalid choice";
}
```

③ Linked List -

```
public() void Linked() {
  LinkedList<Integer> l = new LinkedList
                          <Integer>();
Print(" 1. Add First \n 2. Add Last \n
3. Remove First \n 4. Remove Last \n 5. Display
List ");
Accept(ch);
if (ch == 1)
  l.addFirst(sc.nextLine());
```

```
if ( ch == 2)
    d. addLast ( sc. nextInt());
if (ch == 3)
    d. removeFirst ();
if (ch == 4)
    d. removeLast ();
if (ch == 5)
  Sopln (" contents of List - " + l);
else
    Print (" .Invalid choice ");
}
```

4. Array Deque -

```
public void Dqueue () {
ArrayDeque <Integer> dq = new ArrayDeque <Integer> ();
Print (" 1. Add First \n 2. Add Last \n 3. Display head
    \n 4. Display Tail \n 5. Remove First \n
    6. Remove Last );
Accept ( ch);
  if ( ch == 1)
     dq. addFirst ( sc. nextInt() );
  if (ch == 2)
    dq. addLast ( sc. nextInt());
  if ( ch == 3)
     dq. peekFirst ();
  if (ch == 4)
     dq. peekLast ();
  if ( ch == 5)
     dq. pollFirst ();
  if ( ch == 6)
     dq. pollLast ();
}
```

5. Hash Set -

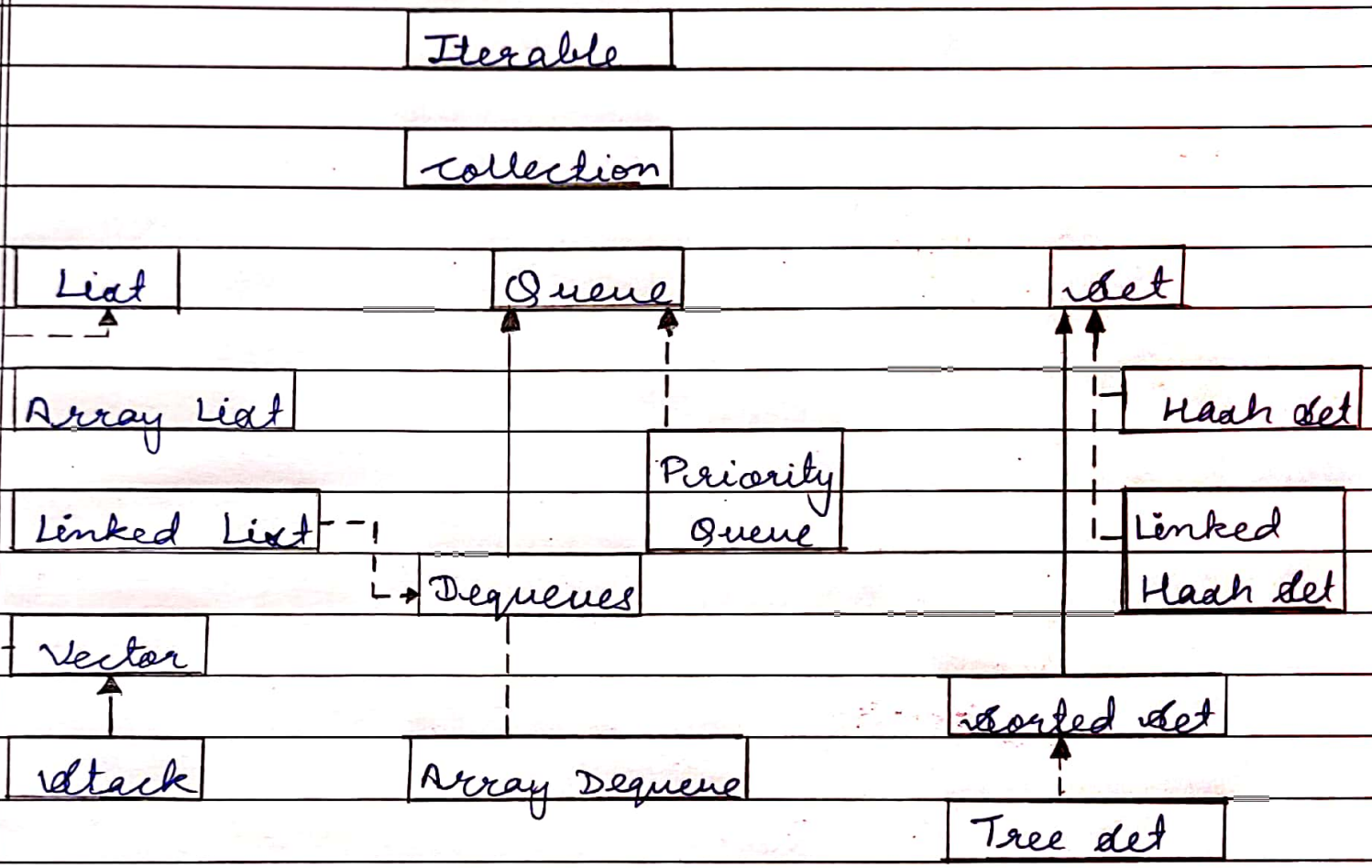
```
public void HashSet() {
HashSet <Integer> hs = new HashSet <Integer> ();
Print (" 1. Add Element \n 2. Remove element \n
      3. Display whole \n 4. Size ");
Accept ( ch);
 if ( ch == 1)
    hs. add ( sc. nextInt ());
 if ( ch == 2)
    hs. remove ( sc. next Int ());
 if ( ch == 3)
    Sofdn (" Elements in Hash - " + hs );
 if ( ch == 4)
    hs. size ();
 else
    print (" Invalid choice ! ");
}
```

Test cases -

	Description	Input	Expected O/P	Actual O/P
1.	Stack	Insert 1	7 5 3 1	Success
		(1, 3, 5, 7)		
		Pop	7 5 3	
		Pop	7 5	
		Insert - 9	9 7 5	

2. Linked List	Insert - (1, 3, 5, 7)	1 3 5 7	Success
	Delete - 3	1 5 7	
	Insert Last - 9	1 5 7 9	
	Delete First -	5 7 9	
3. Priority Queue	Insert - 2	2	Success
	Insert - 4	2 4	
	Insert - 0	0 2 4	
	Insert - 3	0 2 3 4	
	Pop	0 2 3	
	Top	3	
4. Array Dequeue	Insert First - 1	1	Success
	Insert First - 0	0 1	
	Insert Last - 2	0 1 2	
	Head	0	
	Tail	2	
	Delete First	0	
	Delete Last	2	
5. Hash Set	Insert (A, 1)	A1	Success
	Insert (B, 2)	B2	
	Insert (A, 1)	Cannot Insert as keyword already exists	
	Remove (A)	B 2	
	Insert (C, 3)	C 3	
	Size ( ) -	2	

Collection Framework Hierarchy



↑ - extends

↑ - implements

Conclusion -

We have successfully studied and implemented data structures using JAVA library collections.