Assignment 13

Title -

Java collection Libraries.

Problem Statement –

Write a Java program for implementation of different data structures using Java collection libraries at least 5 data structures are used to design a suitable application.

Objectives -

- To understand the use of Java collection libraries.
- To be able to use a Java Library.
- To use Java collections for implementing different types of data structures.

Outcome - We will be able to use JAVA collection Libraries in an application.

Theory - The JAVA collection framework is a set of classes and interface that implement commonly reusable collection of data structures. It works in the manner of a library, Implementation for fundamental collections. The framework had to allow different types of collections to work in a similar manner with high degree operability.

Types of interfaces:

1. Collection interface.
2. List Interface.
3. Set.
4. Sorted set.
5. Map.
6. Map entry.
7. Sorted Map.
8. Enumeration.

Algorithm –

```
public void Linked()
    {
        LinkedList<Integer> l= new
LinkedList<Integer>();
        int op,flag=0,data;
        while(flag!=1)
        {
            System.out.println("\n1.Add First \n2.Add Last \n3.Remove First \n4.Remove Last \n5.Display whole list \n6.Exit");
            op=obj.nextInt();
        switch(op)
        {
        case 1:
            System.out.println("Enter data to be added in list-: ");
            data=obj.nextInt();
            l.addFirst(data);
            break;

        case 2:
            System.out.println("Enter data to be added in list-: ");
            data=obj.nextInt();
            l.addLast(data);
            break;
        case 3:
            l.removeFirst();
            break;

        case 4:
            l.removeLast();
            break;

        case 5:
            System.out.println("Contents of Linked List are-: "+l);
            break;

        case 6:
            flag=1;
            break;
            default:
                System.out.println("Enter valid
choice!!");
                break;
        }
        }

        //main_func();
    }

    public void Stack()
    {
        Stack<Integer> st=new Stack<Integer>();
        int op,flag=0,data;
        while(flag!=1)
        {
            System.out.println("\n1.Push \n2.Pop
\n3.Display Top \n4.IsEmpty \n5.Exit");
            op=obj.nextInt();
        switch(op)
        {
        case 1:
                System.out.println("Enter data to be added in list-: ");
                data=obj.nextInt();
                st.push(data);
                break;

        case 2:
                System.out.println("Data popped from stack is-: " +st.pop());
                break;

        case 3:
                System.out.println("Data at top of stack is-: " +st.peek());
                break;
        case 4:
                if(st.isEmpty())
                {
                    System.out.println("Stack is
Empty!!");
                }
                else
                {
                    System.out.println("Stack is not
empty!!");
                }
                break;
        case 5:
                flag=1;
                break;

        default:
                System.out.println("Enter valid choice!!");
                break;
        }

        }

        //main_func();

    }


    public void Queue()

    {

        PriorityQueue<Integer> pq= new 
PriorityQueue<Integer>();

        int op,flag=0,data;

        while(flag!=1)

        {

            System.out.println("\n1.Add Data \n2.Pop 
\n3.Display head \n4.size \n5.Exit");

            op=obj.nextInt();

        switch(op)

        {

        case 1:
            System.out.println("Enter data to be added in Priority Queue-: ");
            data=obj.nextInt();
            pq.add(data);
            break;

        case 2:
            System.out.println("Data popped from Priority Queue is-: " +pq.poll());
            break;

        case 3:
            System.out.println("Data at top of Priority Queue is-: " +pq.peek());
            break;

        case 4:
            System.out.println("Size of Priority Queue is-: " +pq.size());
            break;
        case 5:
            flag=1;
            break;

        default:
            System.out.println("Enter valid choice!!");
            break;
        }
        }
        //main_func();
    }

    public void Dequeue()
    {
        ArrayDeque<Integer> dq= new
ArrayDeque<Integer>();
        int op,flag=0,data;
        while(flag!=1)

        {

                System.out.println("\n1.Add First \n2.Add last \n3.Display head \n4.Display Tail \n5.Remove First \n6.Remove Last \n7.Exit");

                op=obj.nextInt();

        switch(op)

        {

        case 1:

                System.out.println("Enter data to be added in Deque-: ");

                data=obj.nextInt();

                dq.addFirst(data);

                break;


        case 2:

                System.out.println("Enter data to be added in Deque-: ");

                data=obj.nextInt();
                dq.addLast(data);

                break;


        case 3:

                System.out.println("Element at first
position is-:  "+dq.peekFirst());

                break;


        case 4:

                System.out.println("Element at last
position is-:  "+dq.peekLast());

                break;


        case 5:

                System.out.println("Data removed from
front is-: " +dq.pollFirst());

                break;


        case 6:
            System.out.println("Data removed from Last is-: " +dq.pollLast());
            break;

        case 7:
            flag=1;
            break;

        default:
            System.out.println("Enter valid choice!!");
            break;
        }
        }
        //main_func();
    }

    public void HashSet()
    {
        HashSet<Integer> hs= new
HashSet<Integer>();

        int op,flag=0,data;

        while(flag!=1)

        {

            System.out.println("\n1.Add element
\n2.Remove element \n3.Display whole
\n4.size\n5.Exit");

            op=obj.nextInt();

        switch(op)

        {

        case 1:

            System.out.println("Enter data to be
added in HashSet-: ");

            data=obj.nextInt();

            hs.add(data);

            break;



        case 2:
            System.out.println("Enter data to be
removed from Deque-: ");
            data=obj.nextInt();
            hs.remove(data);
            break;

        case 3:
            System.out.println("Elements in whole
Hashset are-:  "+hs);
            break;

        case 4:
            System.out.println("Size of HashSet is-:
"+hs.size());
            break;

        case 5:
            flag=1;
            break;
        default:
            System.out.println("Enter valid choice!!");
            break;
        }
        }
        //main_func();
    }
```

Test case –

Description	Input	Output	Expected O/P	Result
Stack	Insert 2, 4, 6, 8 Pop 2 times	4 2	4 2	Pass
Queue	Enqueue 1, 2, 3, 4, 5 Dequeue 2 times	3 4 5	3 4 5	Pass
Linked List	Insert 2, 4, 6, 8 Remove 6	2 4 8	2 4 8	Pass
Array List	Add 10, 20, 30 Remove 20	Array 10 30 Size = 2	Array 10 30 Size = 2	Pass

Hash Map	Insert(A, 1)
	Insert(B, 2)
	Insert(C, 3)
		A 1
		C 3
		Size 2
			A 1
			C 3
			Size 2
				Pass

	Remove 2			

Conclusion –

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