Laboratory

2

Data Structures-II

Implementation of Daa structures using Collection frame work

Obecjective

Data Structures. A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. Data structures provide a means to manage large amounts of data efficiently. efficient data structures are a key to designing efficient algorithms.

Java - The **Set** Interface. A **Set** is a Collection that cannot contain duplicate elements. It models the mathematical **set** abstraction. The **Set** interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited.

Java - The **Map** Interface. Advertisements. The **Map** interface **maps** unique keys to values. A key is an object that you use to retrieve a value at a later date. Given a key and a value, you can store the value in a **Map** object. The objective is to write java programs using Map and Set interface.

Overview

The java.util.Queue is a subtype of java.util.Collection interface. It is an ordered list of objects with its use limited to inserting elements at the end of list and deleting elements from the start of list i.e. it follows FIFO principle.

Since it is an interface, we need a concrete class during its declaration. There are many ways to initialize a Queue object, most common being-

1. As a Priority Queue
2. As a LinkedList

Operations on Queue :

- add()-Adds an element at the tail of queue. More specifically, at the last of linkedlist if it is used, or according to the priority in case of priority queue implementation.
- peek()-To view the head of queue without removing it. Returns null if queue is empty.
- element()-Similar to peek(). Throws NoSuchElementException if queue is empty.
- remove()-Removes and returns the head of the queue. Throws NoSuchElementException when queue is impty.
- poll()-Removes and returns the head of the queue. Returns null if queue is empt

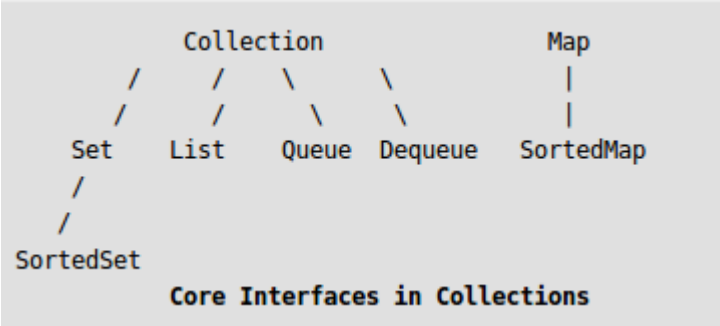
Map interface, which is also a part of **java collection** framework, doesn't inherit from**Collection** interface. **Collection** interface is a member of **java**.util package.**Collections** is an utility class in **java**.util package. It consists of only static methods which are used to operate on objects of type **Collection.**

Set in Java

- Set is an interface which extends Collection. It is an unordered collection of objects in which duplicate values cannot be stored.
- Basically, Set is implemented by HashSet, LinkedSet or TreeSet (sorted representation).
- Set has various methods to add, remove clear, size, etc to enhance the usage of this interface

Map in java

The java.util.Map interface represents a mapping between a key and a value. The Map interface is not a subtype of the Collection interface. Therefore it behaves a bit different from the rest of the collection types.



Core Interfaces in Collections

A Map cannot contain duplicate keys and each key can map to at most one value. Some implementations allow null key and null value (HashMap and LinkedHashMap) but some do not (TreeMap).

Procedure

Implement the following Data structures in Java

a)Queues
b) Set
c) Map

a) Implementation of Queues

```
import java.util.LinkedList;
import java.util.Queue;

public class QueueExample
{
  public static void main(String[] args)
  {
    Queue<Integer> q = new LinkedList<>();

    // Adds elements {0, 1, 2, 3, 4} to queue
    for (int i=0; i<5; i++)
     q.add(i);

    // Display contents of the queue.
```

```
    System.out.println("Elements of queue-"+q);

    // To remove the head of queue.
    int removedele = q.remove();
    System.out.println("removed element-" + removedele);

    System.out.println(q);

    // To view the head of queue
    int head = q.peek();
    System.out.println("head of queue-" + head);

    // Rest all methods of collection interface,
    // Like size and contains can be used with this
    // implementation.
    int size = q.size();
    System.out.println("Size of queue-" + size);
  }
}
```

Expected Output:

```
Elements of queue-[0, 1, 2, 3, 4]

removed element-0

[1, 2, 3, 4]

head of queue-1

Size of queue-4
```

b) set implementation

```
// Java code for adding elements in Set
import java.util.*;
public class Set_example
{
    public static void main(String[] args)
    {
        // Set deonstration using HashSet
        Set<String> hash_Set = new HashSet<String>();
        hash_Set.add("srinivas");
        hash_Set.add("shoba");
        hash_Set.add("srithan");
        hash_Set.add("krithik");
        hash_Set.add("shoba");
        System.out.print("Set output without the duplicates");

        System.out.println(hash_Set);

        // Set deonstration using TreeSet
        System.out.print("Sorted Set after passing into TreeSet");
        Set<String> tree_Set = new TreeSet<String>(hash_Set);
        System.out.println(tree_Set);
    }
}
```

Output:

```
Set output without the duplicates[srinivas, shoba, srithan sai, sai krithik]

Sorted Set after passing into TreeSet[sai krithik, shoba, srinivas,srithan sai]
```

working with HashSet:

```
import java.util.*;

class Test
{
    public static void main(String[]args)
    {
        HashSet<String> h = new HashSet<String>();

        // adding into HashSet
        h.add("India");
        h.add("Australia");
        h.add("South Africa");
        h.add("India");// adding duplicate elements

        // printing HashSet
        System.out.println(h);
        System.out.println("List contains India or not:" +
                           h.contains("India"));

        // Removing an item
        h.remove("Australia");
        System.out.println("List after removing Australia:"+h);

        // Iterating over hash set items
        System.out.println("Iterating over list:");
        Iterator<String> i = h.iterator();
        while (i.hasNext())
            System.out.println(i.next());
    }
}
```

Expected Output :

```
[Australia, South Africa, India]

List contains India or not:true

List after removing Australia:[South Africa, India]

Iterating over list:

South Africa

India
```

C)working with Map interface

```
import java.util.*;
class HashMapDemo
{
   public static void main(String args[])
   {
       HashMap< String,Integer> hm =
                           new HashMap< String,Integer>();
```

```
        hm.put("a", new Integer(100));
        hm.put("b", new Integer(200));
        hm.put("c", new Integer(300));
        hm.put("d", new Integer(400));

        // Returns Set view
        Set< Map.Entry< String,Integer> > st = hm.entrySet();

        for (Map.Entry< String,Integer> me:st)
        {
            System.out.print(me.getKey()+":");
            System.out.println(me.getValue());
        }
    }
}
```

Run on IDE

Output:

```
a:100

b:200

c:300

d:400
```

Questions

1. What is Queue?
2. Differentiate Set and List ?
3. Mention various methods used in Set?
4. With diagram explain class Hierarchy in Collection frame work ?
5. What is Hash Set?
6. What are the applications of Queues?
7. What is super class of SortedMap?

Conclusion

After successful completion of the program the Students are able to understand basics of Collection Frame Work .They are able to perform basic operations on Queues ,set and Map.

Prepared By

Name D. Srinivas

Branch Department of Computer Science and Engineering

College MVR College of Engineering and Technology