Laboratory

1

Data Structures-I

Implementation of Daa structures using Collection frame work

Obecjective

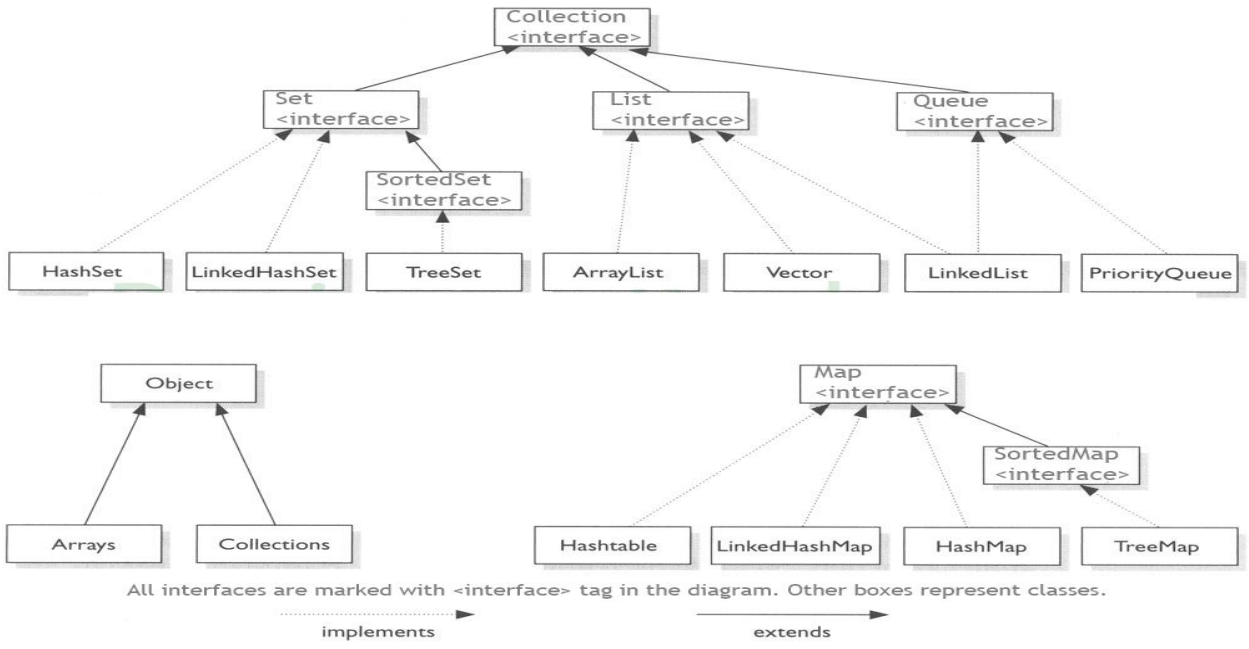
Data Structures. A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. Data structures provide a means to manage large amounts of data efficiently. efficient data structures are a key to designing efficient algorithms.

The **Java collections framework** (JCF) is a set of classes and interfaces that implement commonly reusable **collection** data structures. Although referred to as a **framework**, it works in a manner of a library. The JCF provides both interfaces that define various **collections** and classes that implement them .The objective of this program is to implement Linked list stack Queue data Structures.

Overview

Java Collection simply means a single unit of objects. **Java Collection** framework provides many interfaces (Set, List, Queue, Deque etc.) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet etc).

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**



Procedure

Implement the following Data structures in Java

a) Linked Lists
b) Stacks

a) Implementation of LinkedList

```
Include the package java.util.*;
 public class Test
{
   public static void main(String args[])
   {
      // Creating object of class linked list
      LinkedList<String> object = new LinkedList<String>();

      // Adding elements to the linked list
      object.add("A");
      object.add("B");
      object.addLast("C");
      object.addFirst("D");
      object.add(2, "E");
// similar to above add F and G
      System.out.println("Linked list : " + object);

      // Removing elements from the linked list
      object.remove("B");
      object.remove(3);
      object.removeFirst();
      object.removeLast();
      System.out.println("Linked list after deletion: " + object);

      // Finding elements in the linked list
      boolean status = object.contains("E");

      if(status)
         System.out.println("List contains the element 'E' ");
      else
         System.out.println("List doesn't contain the element 'E'");

      // Number of elements in the linked list
      int size = object.size();
      System.out.println("Size of linked list = " + size);

      // Get and set elements from linked list
      Object element = object.get(2);
      System.out.println("Element returned by get() : " + element);
      object.set(2, "Y");
      System.out.println("Linked list after change : " + object);
```

```
    }
}
```

Expected Output :

```
Linked list : [D, A, E, B, C, F, G]
Linked list after deletion: [A, E, F]
List contains the element 'E'
Size of linked list = 3
Element returned by get() : F
Linked list after change : [A, E, Y]
```

b) stack implementation

write the java statements for mporting packages java.io.*, java.util.*.

```
class MyStack
{
   // Pushing element on the top of the stack
   static void stack_push(Stack<Integer> stack)
   {
      for(int i = 0; i < 5; i++)
      {
         stack.push(i);
      }
   }

   // Popping element from the top of the stack
   static void stack_pop(Stack<Integer> stack)
   {
      System.out.println("Pop :");

      for(int i = 0; i < 5; i++)
      {
         Integer y = (Integer) stack.pop();
         System.out.println(y);
      }
   }

   // Displaying element on the top of the stack
   static void stack_peek(Stack<Integer> stack)
   {
      Integer element = (Integer) stack.peek();
      System.out.println("Element on stack top : " + element);
   }
```

```
    // Searching element in the stack
    static void stack_search(Stack<Integer> stack, int element)
    {
        Integer pos = (Integer) stack.search(element);

        if(pos == -1)
            System.out.println("Element not found");
        else
            System.out.println("Element is found at position " + pos);
    }


    public static void main (String[] args)
    {
        Stack<Integer> stack = new Stack<Integer>();

        stack_push(stack);
        stack_pop(stack);
        stack_push(stack);
        stack_peek(stack);
        stack_search(stack, 2);
        stack_search(stack, 6);
    }
}
```

Expected Output :

```
Pop :
4
3
2
1
0
Element on stack top : 4
Element is found at position 3
Element not found
```

Questions

1. What is Data structure?
2. Explain about Collection Frame work?
3. Mention various methods used in LinkedList?
4. What is interface?
5. What is abstract class?
6. What are the applications of Stack?

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 LinkedList and Stack.

Prepared By

Name D. Srinivas

Branch Department of Computer Science and Engineering

College MVR College of Engineering and Technology