DATE:09.07.2019 **STACK IMPLEMENTATION USING ARRAY**

EX.NO:01

PROGRAM:

```
import java.io.*;
import java.util.*;
public class Stack
{
   static final int MAX=5;
   int top=-1;
   int[] stack=new int[MAX];
   public static void main(String args[])
   {
      Stack s1=new Stack();
      intopt,val;
      System.out.println("1.PUSH");
      System.out.println("2.POP");
      System.out.println("3.PEEP");
      System.out.println("4.DISPLAY STACK");
      do
      {
      System.out.println("\n Enter your option:");
      Scanner s=new Scanner(System.in);
      opt=s.nextInt();
      switch(opt)
      {
      case 1:System.out.println("Enter the value to be added to the stack");
      val=s.nextInt();
      s1.push(val);
      break;
      case 2:s1.pop();
      break;
      case 3:s1.peep();
      break;
      case 4:s1.display();
```

```
      break;
      }
      }while(opt!=5);
  }
  public void push(int val)
  {
      if(top==(MAX-1))
      {
      System.out.println("Stack is full!");
      }
      else
      {
      top++;
      stack[top]=val;
      System.out.println("element added to the stack is :"+val);
      display();
      }
  }
  public void pop()
  {
    int x;
    if(top==-1)
  {
    System.out.println("stack is EMPTY!");
  }
  else
  {
       x=stack[top];
      System.out.println("The element deleted from the stack is:"+x);
      top--;
      display();
  }
}
public void peep()
```

```
    {
    int n;
     n=stack[top];
    System.out.println("the value at the top of the stack is:"+n);
    }
  public void display()
  {
    int i;
    if(top==-1)
    System.out.println("stack is EMPTY!");
    else
    {
    for(i=0;i<=top;i++)
    System.out.println("the element in the stack are:"+stack[i]);
    }
  }
}
```

OUTPUT:

1.PUSH
2.POP
3.PEEP
4.DISPLAY STACK
Enter your Option: 1
Enter the value to be added to the Stack: 11
The elements in the Stack is: 10
Enter your Option: 1
Enter the value to be added to the Stack: 20
The elements in the Stack is: 10 20
Enter your Option: 2
The element deleted from the Stack is: 20
The element in the Stack is: 10
Enter your Option: 3
The Value at the top of the Stack is: 10
Enter your Option: 4
The elements in the Stack is: 10

DATE: 12.07.2019 **QUEUE IMPLEMENTATION USING ARRAY**

EX.NO: 02

PROGRAM:

```
import java.io.*;
class QueueArr
{
    static int i,front,rear,item,max=5,ch;
    static int a[]=new int[5];
    QueueArr()
    {
        front=-1;
        rear=-1;
    }
    public static void main(String args[])throws IOException
    {
        while((boolean)true)
        {
        try
        {
            System.out.println("Select option 1.insert 2.delete
        3.display 4.Exit");
            BufferedReader br=new BufferedReader(new
        InputStreamReader(System.in));
            ch=Integer.parseInt(br.readLine());
        }
        catch(Exception e)
        {}
        if(ch==4)
            break;
        else
        {
            switch(ch)
            {
            case 1:
```

```
                    insert();
                    break;
                case 2:
                    delete();
                    break;
                case 3:
                    display();
                    break;
                }
            }
        }
}
static void insert()
{
        if(rear>=max)
        {
            System.out.println("Queue is full");
        }
        else
        {
            try
            {
                BufferedReader br=new BufferedReader(new
            InputStreamReader(System.in));
                System.out.println("Enter the element:");
                item=Integer.parseInt(br.readLine());
            }
            catch(Exception e)
            {}
            rear=rear+1;
            a[rear]=item;
        }
}
        static void delete()
```

```
        {
            if(front==-1)
        {
            System.out.println("Queue is empty");
        }
        else
        {
            front=front+1;
            item=a[front];
            System.out.println("Deleted Item:"+item);
        }
    }
    static void display()
    {
        System.out.println("Elements in the queue are:");
        for(int i=front+1;i<=rear;i++)
        {
            System.out.println(a[i]);
        }
    }
}
```

OUTPUT:

```
Select Option 1.Insert 2.Delete 3.Display 4.Exit
1
Enter the element: 100
Select Option 1.Insert 2.Delete 3.Display 4.Exit
1
Enter the element: 200
Select Option 1.Insert 2.Delete 3.Display 4.Exit
2
Deleted item: 100
Select Option 1.Insert 2.Delete 3.Display 4.Exit
3
Elements in the queue are: 200
Select Option 1.Insert 2.Delete 3.Display 4.Exit
4
```

DATE:15.07.2019 **STACK IMPLEMENTATION USING LINKED LIST**

EX.NO:03

PROGRAM:

```
import java.util.*;
class Node
{
    protected int data;
    protected Node link;
    public Node()
    {
        link=null;
        data=0;
    }
    public Node(int d,Node n)
    {
        data=d;
        link=n;
    }
    public void setData(int d)
    {
        data=d;
    }
    public Node getLink()
    {
        return link;
    }
    public void setLink(Node n)
    {
        link=n;
    }
    public int getData()
    {
        return data;
    }
```

```
}
class linkedStack
{
        protected Node top;
        protected int size;
        public linkedStack()
        {
                top=null;
                size=0;
        }
        public booleanisEmpty()
        {
                return top==null;
        }
        public intgetSize()
        {
          return size;
        }
        public void push(int data)
        {
            Node nptr=new Node(data,null);
            if(top==null)
            top=nptr;
            else
            {
               nptr.setLink(top);
               top=nptr;
            }
            size++;
        }
        public int pop()
        {
                if(isEmpty())
```

```
                throw new NoSuchElementException("Underflow
        Exception");
                Node ptr=top;
                top=ptr.getLink();
                size--;
                return ptr.getData();
        }
        public int peep()
        {
                if(isEmpty())
                        throw new NoSuchElementException("Underflow
        Exception");
                return top.getData();
        }
        public void display()
        {
                System.out.println("\nStack=");
                if(size==0)
                {
                   System.out.println("Empty\n");
                   return;
                }
                Node ptr=top;
                while(ptr!=null)
                {
                        System.out.println(ptr.getData()+"");
        ptr=ptr.getLink();
                }
                System.out.println();
        }
}
public class LinkedStackImplement
{
        public static void main(String args[])
```

```
{
    Scanner scan=new Scanner(System.in);
    linkedStack ls=new linkedStack();
    System.out.println("Linked Stack Test\n");
                        char ch;
    do
    {
     System.out.println("\nLinked Stack Operations");
     System.out.println("1.push");
     System.out.println("2.pop");
     System.out.println("3.peep");
     System.out.println("4.check empty");
     System.out.println("5.size");
     int choice=scan.nextInt();
     switch (choice)
     {
      case 1:
      System.out.println("Enter integer element to push");
      ls.push(scan.nextInt());
      break;
      case 2:
      try
      {
      System.out.println("Popped Element ="+ls.pop());
      }
      catch(Exception e)
      {
      System.out.println("Error:"+e.getMessage());
      }
      break;
      case 3:
      try
      {
      System.out.println("Peep Element="+ ls.peep());
```

```
                    }
                    catch(Exception e)
                    {
                      System.out.println("Error :"+e.getMessage());
                    }
                    case 4:
                    System.out.println("Empty status="+ls.isEmpty());
                    break;
                    case 5:
                    System.out.println("Size="+ls.getSize());
                    break;
                    case 6:
                    System.out.println("Stack=");
                    ls.display();
                    break;
                    default:
                    System.out.println("Wrong Entry \n");
                      break;
                  }
                  ls.display();
                  System.out.println("\n Do you want to continue(Type y or
n)\n");
                  ch=scan.next().charAt(0);
               }while(ch=='Y'||ch=='y');
        }
}
```

OUTPUT:

```
 Linked Stack Test
Linked Stack Operations
1.PUSH
2.POP
3.PEEP
4.CHECK EMPTY
5.SIZE
1
Enter integer element to push: 10
Stack=10
Do you want to continue(Type y or n)
Y
 Linked Stack Operations
1.PUSH
2.POP
3.PEEP
4.CHECK EMPTY
5.SIZE
3
Peep element=10
Do you want to continue (Type y or n)
Y
Linked Stack Operations
1.PUSH
2.POP
3.PEEP
4.CHECK EMPTY
5.SIZE
4
Empty status=false
Do you want to continue (Type y or n)
Y
Linked Stack Operations
```

1.PUSH
2.POP
3.PEEP
4.CHECK EMPTY
5.SIZE
5
Size=1
Do you want to continue(Type y or n)
y
Linked Stack Operations
1.PUSH
2.POP
3.PEEP
4.CHECK EMPTY
5.SIZE
2
Popped Element=20
Stack=10
Do you want to continue (Type y or n)
n

DATE: 20.07.2019 **QUEUE IMPLEMENTATION USING LINKED LIST**

EX.NO:04

PROGRAM:

```
import java.io.*;
class Node
{
 publicint data;
 public Node next;
 public Node (int x)
 {
  data=x;
 }
 public void displayNode()
 {
  System.out.println(data+"");
 }
}
classLinkList
{
 private Node first;
 private Node last;
 publicLinkList()
 {
 first=null;
 last=null;
 }
 public void insertLast(int x)
 {
 Node newNode=new Node(x);
 newNode.next=null;
 if(isEmpty())
 first=newNode;
 else
 last.next=newNode;
```

```
  last=newNode;
  }
  public int deleteFirst()
  {
   int t=first.data;
   if(first.next==null)
   last=null;
   first=first.next;
   return t;
  }
  public int peekFirst()
  {
   return (first.data);
  }
  public booleanisEmpty()
  {
   return (first==null);
  }
  public void displayList()
  {
     Node current=first;
     while(current!=null)
    {
      current.displayNode();
      current=current.next;
    }
  }
}
class Queue
{
  private LinkListl;
  public Queue()
  {
   l=new LinkList();
```

```
  }
  public void insert(int x)
  {
   l.insertLast(x);
   System.out.println("Inserted");
  }
  public int delete()
  {
   returnl.deleteFirst();
  }
  public booleanisQueueEmpty()
  {
   returnl.isEmpty();
  }
  public void display()
  {
   l.displayList();
  }
  public int peek()
  {
   returnl.peekFirst();
  }
 }
 class QueueList
 {
      public static void main(String args[])throws IOException
      {
       Queue q=new Queue();
       int ch,d;
       while((boolean)true)
       {
        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
        System.out.println("MENU");
```

```
System.out.println("_____");
System.out.println("1.INSERT");
System.out.println("2.DELETE");
System.out.println("3.PEEK");
System.out.println("4.DISPLAY");
System.out.println("5.EXIT");
System.out.println("Enter your choice:");
ch=Integer.parseInt(br.readLine());
if(ch==5)
break;
else
{
 switch(ch)
 {
 case 1:
   System.out.println("Enter Number of elements");
   int n1=Integer.parseInt(br.readLine());
  System.out.println("\n Enter elements:");
  for(int i=0;i<n1;i++)
  {
    d=Integer.parseInt(br.readLine());
    q.insert(d);
   }
   break;
   case 2:
   if(q.isQueueEmpty())
         System.out.println("Queue is Empty");
  else
  {
         d=q.delete();
         System.out.println("Deleted data:-"+d);
}
break;
case 3:
```

```
            if(q.isQueueEmpty())
            System.out.println("Queue is empty");
            else
            {
            d=q.peek();
            System.out.println("First item:-"+d);
        }
        break;
        case 4:
        if(q.isQueueEmpty())
        System.out.println("Queue is empty");
        else
        {
        System.out.println("datas in queue");
        q.display();
        }
        break;
        default:
        System.out.println("Invalid choice");
        }
        }
        System.out.println("");
        }
```

OUTPUT:

Menu

1.INSERT

2.DELETE

3.PEEK

4.DISPLAY

5.EXIT

Enter your Choice: 1

Enter number of elements:1

1

Inserted
Menu
1.INSERT
2.DELETE
3.PEEK
4.DISPLAY
5.EXIT
Enter your choice: 3
First Item: -1
Menu
1.INSERT
2.DELETE
3.PEEK
4.DISPLAY
5.EXIT
Enter your choice: 4
Datas in queue: 1
Menu
1.INSERT
2.DELETE
3.PEEK
4.DISPLAY
5.EXIT
Enter your choice: 2
Deleted data: -1
Menu
1.INSERT
2.DELETE
3.PEEK
4.DISPLAY
5.EXIT
Enter your choice: 5

DATE: 24.07.2019 **BINARY TREE TRAVERSAL**

EX.NO:05

PROGRAM:

```
class Node
{
int key;
 Node left,right;
public Node(int item)
{
key=item;
left=right=null;
}
}
class BinaryTree
{
Node root;
BinaryTree()
{
root=null;
}
void printPostorder(Node node)
{
if(node==null)
return;
printPostorder(node.left);
printPostorder(node.right);
System.out.println(node.key+"");
}
void printInorder(Node node)
{
if(node==null)
return;
printInorder(node.left);
System.out.println(node.key+"");
```

```
        printInorder(node.right);
        }
        void printPreorder(Node node)
        {
        if(node==null)
        return;
        System.out.println(node.key+"");
        printPreorder(node.left);
        printPreorder(node.right);
        }
        void printPostorder() { printPostorder(root);}
        void printInorder(){ printInorder(root);}
        void printPreorder(){ printPreorder(root);}
        public static void main(String args[])
        {
        BinaryTree tree = new BinaryTree();
        tree.root=new Node(1);
        tree.root.left=new Node(2);
        tree.root.right=new Node(3);
        tree.root.left.left=new Node(4);
        tree.root.left.right=new Node(5);
System.out.println("Preorder traversal of binary tree is");
        tree.printPreorder();
System.out.println("Inorder traversal of binary tree is");
        tree.printInorder();
System.out.println("Postorder traversal of binary tree is");
        tree.printPostorder();
        }
        }
```

OUTPUT:

Preorder traversal of binary tree is
1
2
4
5
3
Inorder traversal of binary tree is
4
2
5
1
3
Postorder traversal of binary tree is
4
5
2
3

DATE: 29.07.2019 **BREADTH FIRST GRAPH TRAVERSAL**

EX.NO: 08

PROGRAM:

```
import java.io.*;
import java.util.*;
class Graph
{
private int V;
private LinkedList<Integer>adj[];
Graph(int v)
{
V=v;
adj=new LinkedList[V];
for(int i=0;i<v;i++)
adj[i]=new LinkedList();
}
void addEdge(intv,int w)
{
adj[v].add(w);
}
void BFS(int s)
{
boolean visited[]=new boolean[V];
LinkedList<Integer>queue=new LinkedList<Integer>();
visited[s]=true;
queue.add(s);
while(queue.size()!=0)
{
s=queue.poll();
System.out.println(s+"");
Iterator<Integer>i=adj[s].listIterator();
while(i.hasNext())
{
int n=i.next();
```

```
if(!visited[n])
{
visited[n]=true;
queue.add(n);
}
}
}
}
public static void main(String args[])
{
Graph g=new Graph(4);
g.addEdge(0,1);
g.addEdge(0,2);
g.addEdge(1,2);
g.addEdge(2,0);
g.addEdge(2,3);
g.addEdge(3,3);
System.out.println("Following is breadth first traversal"+"(Starting from vertex 2)");
g.BFS(2);
}
}
```

OUTPUT:

```
Following is breadth first traversal(Starting from vertex 2)
2
0
3
1
```

DATE: 01.08.2019 **DEPTH FIRST GRAPH TRAVERSAL**

EX.NO:09

PROGRAM:

```
import java.io.*;
import java.util.*;
class Graph7
{
private int V;
private LinkedList<Integer>adj[];
Graph7(int v)
{
V=v;
adj=new LinkedList[v];
for(int i=0;i<v;i++)
adj[i]=new LinkedList();
}
void addEdge(intv,int w)
{
adj[v].add(w);
}
void DFSUtil(intv,boolean visited[])
{
visited[v]=true;
System.out.println(v+" ");
Iterator<Integer>i=adj[v].listIterator();
while(i.hasNext())
{
int n=i.next();
if(! visited[n])
DFSUtil(n,visited);
}
}
void DFS(int v)
{
```

```
boolean visited[]=new boolean[V];
DFSUtil(v,visited);
}
public static void main(String args[])
{
Graph7 g=new Graph7(4);
g.addEdge(0,1);
g.addEdge(0,2);
g.addEdge(1,2);
g.addEdge(2,0);
g.addEdge(2,3);
g.addEdge(3,3);
System.out.println("following is depth first traversal "+"(starting from vertex2)");
g.DFS(2);
}}
```

OUTPUT:

Following is depth first traversal(Starting from vertex 2)

2

0

1

3

DATE: 06.08.2019 **LINEAR SEARCH**

EX.NO:10

PROGRAM:

```
class LinearSearch
{
public static int search(intarr[],int x)
  {
int n=arr.length;
for(int i=0;i<n;i++)
   {
if(arr[i]==x)
return i;
   }
return-1;
  }
public static void main(String args[])
  {
int arr[]={2,3,4,10,40};
int x=10;
int result=search(arr,x);
if(result==-1)
                    System.out.println("element  not present in the array");
else
System.out.println("element at index"+result);
  }
 }
```

OUTPUT:

Element at index: 3

DATE: 09.08.2019 **BINARY SEARCH**

EX.NO:11

PROGRAM:

```
class BinarySearch
 {
int binarySearch(int arr[],int i,int r,int x)
  {
if(r>=1)
   {
      int mid=i+(r-1)/2;
      if(arr[mid]==x)
      return mid;
      if(arr[mid]>x)
      return binarySearch(arr,i,mid-1,x);
      return binarySearch(arr,mid+1,r,x);
   }
return-1;
  }
public static void main(String args[])
  {
BinarySearch ob=new BinarySearch();
intarr[]={2,3,4,10,40};
int n=arr.length;
int x=10;
int result=ob.binarySearch(arr,0,n-1,x);
if(result==-1)
System.out.println("Element not present");
else
System.out.println("Element found at index"+result);
  }
 }
```

OUTPUT:

Element found at index: 3

DATE: 14.08.2019 **BUBBLE SORT**

EX.NO:12

PROGRAM:

```
class BubbleSort
{
void BubbleSort(intarr[])
{
int n=arr.length;
for(int i=0;i<n-1;i++)
for(int j=0;j<n-i-1;j++)
if(arr[j]>arr[j+1])
{
int temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
}
}
void printArray(int arr[])
{
int n=arr.length;
for(int i=0;i<n;i++)
System.out.println(arr[i]+"");
System.out.println();
}
public static void main(String args[])
{
BubbleSort ob=new BubbleSort();
int arr[]={10,90,40,50};
ob.BubbleSort(arr);
System.out.println("Sorted array");
ob.printArray(arr);
}
}
```

OUTPUT:

Sorted array:

10

40

50

90

DATE: 19.08.2019 **SELECTION SORT**

EX.NO:13

PROGRAM:

```
class SelectionSort
{
void Sort (int arr[])
{
int n=arr.length;
for(int i=0;i<n-i;i++)
{
int min_idx=i;
for(int j=i+1;j<n;j++)
if(arr[j] <arr[min_idx])
min_idx=j;
int temp=arr[min_idx];
arr[min_idx]=arr[i];
arr[i]=temp;
}
}
void printArray(int arr[])
{
int n=arr.length;
for(int i=0;i<n;++i)
System.out.print(arr[i]+"");
System.out.println();
}
public static void main(String args[])
{
SelectionSort ob=new SelectionSort();
int arr[]={64,25,12,22,11};
ob.Sort(arr);
System.out.println("Sorted array");
ob.printArray(arr);
}
```

```
}
```

OUTPUT:

```
Sorted array:
11
12
22
25
64
```

DATE:04.09.2019 **INSERTION SORT**

EX.NO:14

PROGRAM:

```
class InsertionSort
{
void sort(int arr[])
 {
int n=arr.length;
for(int i=1;i<n;++i)
   {
int key=arr[i];
int j=i-1;
while(j>=0&&arr[j]>key)
     {
arr[j+1]=arr[j];
       j=j-1;
     }
arr[j+1]=key;
   }
  }
static void printArray(int arr[])
  {
int n=arr.length;
for(int i=0;i<n;++i)
System.out.println(arr[i]+" ");
System.out.println();
  }
public static void main(String args[])
  {
intarr[]={12,11,13,5,6};
InsertionSort ob =new InsertionSort();
System.out.println(“Sorted array:”);
ob.sort(arr);
printArray(arr);
```

```
        }
    }
```

OUTPUT:

Sorted array:

5

6

11

12

13

DATE: 10.09.2019 **HASHING TECHNIQUE**

EX.NO:15

PROGRAM:

```
import java.util.*;
import java.util.Scanner;
public class Hashtable1
{
    public static void main(String args[])
   {
        Hashtable<Integer, String>hm=new Hashtable<Integer, String>();
        hm.put(100,"Vignesh");
        hm.put(101,"Santhosh");
        hm.put(102,"Sapeek");
        hm.put(103,"Hemanath");
        Scanner s=new Scanner(System.in);
        int key;
        for(Map.Entry m:hm.entrySet())
        {
           System.out.println(m.getKey()+" "+m.getValue());
        }
        System.out.println("Before remove:"+hm);
        System.out.println("Enter the key to be removed:");
        key=s.nextInt();
        hm.remove(key);
        System.out.println("After remove:"+hm);
        System.out.println("Finding the respective key:");
        System.out.println(hm.getOrDefault(101,"Not found"));
        System.out.println(hm.getOrDefault(105,"Not found"));
        System.out.println("Updating the table:");
        hm.putIfAbsent(104,"Chandru");
        System.out.println("Updated Table:"+hm);
        hm.putIfAbsent(100,"Vignesh");
        System.out.println("Updated Table:"+hm);
    }
```

```
}
```

OUTPUT:

```
103 Hemanath
102 Sapeek
101 Santhosh
100 Vignesh
Before remove:{103=Hemanath, 102=Sapeek, 101=Santhosh, 100=Vignesh}
Enter the key to be removed: 102
After remove:{103=Hemanath, 101=Santhosh, 100=Vignesh}
Finding the respective key:
Santhosh
Not found
Updating the table:
Updated Table:{104=Chandru, 103=Hemanath, 101=Santhosh, 100=Vignesh}
Updated Table:{104=Chandru, 103=Hemanath, 101=Santhosh, 100=Vignesh}
```