StackAndQueueCode

//Stack

Ex.1

```
package A3Stack;
  //Stack Using LinkedList
public class A1StackClass {

           static class Node{  //this Node class for creating a Node of LinkedList
             int data;
             Node next;
             Node(int data){
                    this.data=data;
                    this.next=null;
             }
           }

           static class Stack{
             public static Node head;
             public static boolean isEmpty() {  //to check Stack is empty or not
                    return head==null;
             }
             public static void push(int data) {
                    Node newNode = new Node(data);
                    if(isEmpty()) {
                           head=newNode;
                           return;
                    }
                    newNode.next=head;
                    head=newNode;
             }

             public static int pop() {
                    if(isEmpty()) {
                           return -1;
                    }
                    int top=head.data;
                    head=head.next;  //we remove the head that is top of the Stack
                    return top;
             }

             public static int peek() {
                    if(isEmpty()) {
                           return -1;
                    }

                    return head.data;
             }
           }


       public static void main(String[] args) {
                  Stack s=new Stack();
                    s.push(1);
                    s.push(2);
```

```
                   s.push(3);
                   s.push(4);
                   s.push(5);

                   while(!s.isEmpty()) {
                     System.out.println(s.peek());  //Ans: 5 4 3 2 1
                     s.pop();
                   }

       }

}
```

Ex.2

```
package A3Stack;
  //Stack using ArrayList
import java.util.ArrayList;
public class A2StackArrayList {

          static class Stack{
             static ArrayList<Integer> list = new ArrayList<Integer>();

              public static boolean isEmpty() {  //to check stack is empty or not
                    return list.size()==0;
              }
              //push
              public static void push(int data) {
                    list.add(data); //it is automatically added to the last index
              }

              //pop
              public static int pop() {  //we remove last element of ArrayList

                    if(isEmpty()) {
                           return -1;
                    }
                    int top=list.get(list.size()-1);
                     list.remove(list.size()-1);
                     return top;
              }

              //peek
              public static int peek() {
                    if(isEmpty()) {
                           return -1;
                    }
                    int top = list.get(list.size()-1);
                    return top;
              }
```

```
            }

       public static void main(String[] args) {
             Stack s = new Stack();
             s.push(1);
             s.push(2);
             s.push(3);
             s.push(4);

             while(!s.isEmpty()) {
                    System.out.print(s.peek()+" ");  //Ans: 4 3 2 1
                    s.pop();
             }

       }

}
```

Ex.3

```
package A3Stack;
import java.util.Stack;  //Need to import Stack
public class A3StackusingJavaCollections {
     //Stack Using Java Collection framework
       public static void main(String[] args) {
             Stack<Integer> s = new Stack<Integer>();
        s.push(1);
        s.push(2);
        s.push(3);
        s.push(4);

        while(!s.isEmpty()) {
             System.out.print(s.peek()+" ");  //Ans: 4 3 2 1
             s.pop();
        }
       }

}
```

Ex.4

```
package A3Stack;
import java.util.*;
public class A4PushEleAtBottomOfStack {
  //Que. Push  element at the bottom of the Stack
  //here we use recursion approach to push element at bottom of the Stack
       // here stack is 3 2 1 and we need to push element 4 at bottom of stack so ans should be 3 2 1 4
       public static void pushAtBottom(Stack<Integer> s,int data) {
             if(s.isEmpty()) {
```

```
            s.push(data);
            return;
        }
        int top=s.pop();
         pushAtBottom(s,data);
         s.push(top);
    }
    public static void main(String[] args) {
        Stack<Integer> s = new Stack<Integer>();
        s.push(1);
        s.push(2);
        s.push(3);

        pushAtBottom(s,4);


        while(!s.isEmpty()) {
            System.out.println(s.peek());  //Ans: 3 2 1 4
            s.pop();
        }

    }

}
```

Ex.5

```
package A3Stack;
import java.util.*;
  //Que. Reverse a Stack
 //while solving this see a diagram or draw using pen and paper
public class A5ReverseAStack {

    public static void pushAtTheBottom(Stack<Integer> s,int data) {
        if(s.isEmpty()) {
            s.push(data);
            return;
        }
        int top=s.pop();
        pushAtTheBottom(s,data);
        s.push(top);
    }

     public static void reverseStack(Stack<Integer> s) {
         if(s.isEmpty()){
             return;
         }
         int top= s.pop();
         reverseStack(s);
         if(s.isEmpty()) {
             s.push(top);
         }else {
             pushAtTheBottom(s,top);
         }
        }

      public static void main(String[] args) {
        Stack<Integer> s = new Stack<Integer>();
      s.push(1);
      s.push(2);
      s.push(3);
     /* while(!s.isEmpty()) {
        System.out.println(s.peek()); //Ans: 3 2 1
        s.pop();
      } */

      reverseStack(s);
       while(!s.isEmpty()) {
           System.out.println(s.peek()); //Ans: 1 2 3
           s.pop();
       }
      }

}
```

//Queue

Ex.6

```
package A4Queue;
  //Queue using Array
public class A1QueueEx1 {

        static class Queue{
               static int arr[];
               static int size;
               static int rear=-1;

               Queue(int n){ //constructor
                      arr = new int[n];
                      this.size=n;
               }

               public static boolean isEmpty() {
                      return rear==-1;
               }

               //enqueue that is add
               public static void add(int data) {
                      if(rear==size-1) {
                             System.out.println("Queue is Full");
                             return;
                      }
                      rear++;
                      arr[rear]=data;
               }

               //Dequeue means remove here time complexity: O(n)
               public static int remove() {
                      if(isEmpty()) {
                             return -1;
                      }
                     int front = arr[0];
                     for(int i=0;i<rear;i++) {
                            arr[i]=arr[i+1];
                     }
                          rear--;
                      return front;
               }

               //peek
               public static int peek() {
                      if(isEmpty()) {
                             return -1;
                      }

                      return arr[0];
               }
         }

      public static void main(String[] args) {
         Queue q=new Queue(5);
         q.add(1);
         q.add(2);
         q.add(3);

         while(!q.isEmpty()) {
                System.out.println(q.peek());  //Ans: 1 2 3
                q.remove();
         }

      }

}
```

Ex.7

```
package A4Queue;
  //Circular Queues
 //Circular Queues time complexity is good that is O(1) than Queue using array
public class A2CircularQueue {
     static class Queue{
             static int size;
             static int rear=-1;
             static int front=-1;
             static int arr[];
             Queue(int n){
               arr = new int[n];
```

```
              this.size=n;
             }

             public static boolean isEmpty() {
               return rear==-1 && front==-1;
             }

             public static boolean isFull() {
               return (rear+1)%size==front;  //formula to check circular queue is
full or not
             }

             //enque means add
             public static void add(int data) {
               if(isFull()) {
                      System.out.println("Queue is full");
                      return;
               }
               rear= (rear+1)%size;  //for one step back
               //if first ele add
               if(front==-1) {
                      front=0;
               }
               arr[rear]=data;

             }

             //dequeue time complexity: O(1);
             public static int remove() {
               if(isEmpty()) {
                      return -1;
               }
               int result = arr[front];
               //single element condition
               if(rear==front) {
                      rear=-1;
                      front=-1;
               }
               else {
                      front=(front+1)%size;
               }

               return result;
             }
             //peek
             public static int peek() {
               if(isEmpty()) {
                      return -1;
               }
               return arr[front];
             }

     }
      public static void main(String[] args) {
               Queue q=new Queue(5);
```

```
        q.add(1);
        q.add(2);
     q.add(3);
     q.add(4);
     q.add(5);
     System.out.println(q.remove());//Ans:1
     q.add(6);
     System.out.println(q.remove());//Ans:2
     q.add(7);
     while(!q.isEmpty()) {
        System.out.println(q.peek());  //Ans : 3 4 5 6 7
        q.remove();
     }
   }

}
```

Ex.8

```
package A4Queue;
  //Queue Using LinkedList
public class A3QueueUsingLinkedList {



         static  class Node{
           int data;
           Node next;
           Node(int data){  //here we create new node
                this.data=data;
                this.next=null;
           }
          }

      static class Queue{
            static Node head=null;
            static Node tail=null;
        //isEmpty()
        public static boolean isEmpty() {
             return head==null && tail==null;
        }

        //add node at last
        public static void add(int data) {
           Node newNode= new Node(data);
           if(isEmpty()) {
                head=tail=newNode;
                return;
           }
           tail.next=newNode;
           tail=newNode;
        }
```

```
               //remove first node
               public int remove() {
                      if(isEmpty()) {
                            System.out.println("LinkList/Queue is empty");
                            return -1;
                      }
                      int first=head.data;
                      if(head==tail) {//when linkedlist have only one element
                           tail=null;
                      }
                      head=head.next;
                      return first;
               }

               //find the peek means head
               public int peek() {
                      if(isEmpty()) {
                            return -1;
                      }
                      return head.data;
               }

              }
      public static void main(String[] args) {
      //     A3QueueUsingLinkedList ll=new A3QueueUsingLinkedList();
             Queue q= new Queue();
               q.add(1);
               q.add(2);
               q.add(3);
               q.add(4);
               q.remove();

              while(!q.isEmpty()) {
                     System.out.println(q.peek());  //Ans 2 3 4
                     q.remove();
              }

      }

}
```

Ex.9

```
package A4Queue;
 import java.util.*;  //Need to import Queue
public class A4QueueJavaCollection {

      public static void main(String[] args) {
             // TODO Auto-generated method stub
       // Queue<Integer> q= new LinkedList<>();  //We can't create Object of Queue
because Queue is not a class it is a Interface
                                            //LinkedList is a Class
```

```
        Queue<Integer> q= new ArrayDeque<>();   //here ArrayDeque and LinkedList are 
class which can implement Queue interface
         q.add(1);
         q.add(2);
         q.add(3);
         q.add(4);
         q.add(5);
         q.remove();
         q.remove();
         
         while(!q.isEmpty()) {
              System.out.println(q.peek());  //Ans: 3 4 5
              q.remove();
         }
         
      }

}
```

Ex.10

```
package A4Queue;
import java.util.*;
public class A5QueueUsingTwoStack {
    //Que. Create a Queue(FIFO Structure) Using Two Stack(LIFO Structure)
    
      static class Queue{
             static Stack<Integer> s1=new Stack<Integer>();
             static Stack<Integer> s2=new Stack<Integer>();
             
             public static boolean isEmpty() {
                   return s1.isEmpty();
             }
             
              //time complexity:O(n) for adding the data
             public static void add(int data) {
                   while(!s1.isEmpty()) {
                          s2.push(s1.pop());
                   }
                   s1.push(data); //When Stack s1 becomes empty then we add data in 
s1 stack
                   
                   
                   while(!s2.isEmpty()) {
                          s1.push(s2.pop());
                   }
             }
             
             //here for removing time complexity: O(1)
             public static int remove() {
                   if(s1.isEmpty()) {
                          return -1;
```

```
            }

            return s1.pop();
        }

        //here time complexity: O(1)
        public static int peek() {
            if(s1.isEmpty()) {
                return -1;
            }
            return s1.peek();
        }
    }

    public static void main(String[] args) {
        Queue q = new Queue();
        q.add(1);
        q.add(2);
        q.add(3);
        q.add(4);

        while(!q.isEmpty()) {
            System.out.println(q.peek());  //Ans: 1 2 3 4 that is in
FIFO(First In First Out) structure
            q.remove();
        }

    }

}
```