```java
import java.util.*;

// implementing a doubly linked list


class Node{

    private int data;
    private Node nextNodePtr;
    private Node prevNodePtr;

    public Node(){}

    public void setData(int d){
        data = d;
    }

    public int getData(){
        return data;
    }

    public void setNextNodePtr(Node nodePtr){
        nextNodePtr = nodePtr;
    }

    public Node getNextNodePtr(){
        return nextNodePtr;
    }

    public void setPrevNodePtr(Node nodePtr){
        prevNodePtr = nodePtr;
    }

    public Node getPrevNodePtr(){
        return prevNodePtr;
    }

}

class Stack{

    private Node headPtr;
    private Node tailPtr;

    public  Stack(){
        headPtr = new Node();
        tailPtr = new Node();
        headPtr.setNextNodePtr(null);
        tailPtr.setPrevNodePtr(null);
    }

    public Node getHeadPtr(){
        return headPtr;
    }

    public  Node getTailPtr(){
        return tailPtr;
    }

    public  boolean isEmpty(){

        if (headPtr.getNextNodePtr() == null)
            return true;

```

```
65          return false;
66      }
67
68
69      public  void push(int data){
70
71          Node newNodePtr = new Node();
72          newNodePtr.setData(data);
73          newNodePtr.setNextNodePtr(null);
74
75          Node lastNodePtr = tailPtr.getPrevNodePtr();
76
77          if (lastNodePtr == null){
78
79              headPtr.setNextNodePtr(newNodePtr);
80              newNodePtr.setPrevNodePtr(null);
81
82          }
83          else{
84
85              lastNodePtr.setNextNodePtr(newNodePtr);
86              newNodePtr.setPrevNodePtr(lastNodePtr);
87
88          }
89
90          tailPtr.setPrevNodePtr(newNodePtr);
91
92      }
93
94
95      public int pop(){
96
97          Node lastNodePtr = tailPtr.getPrevNodePtr();
98          Node prevNodePtr = null;
99
100         int poppedData = -100000; //empty stack
101
102         if (lastNodePtr != null){
103             prevNodePtr = lastNodePtr.getPrevNodePtr();
104             poppedData = lastNodePtr.getData();
105         }
106         else
107             return poppedData;
108
109         if (prevNodePtr != null){
110             prevNodePtr.setNextNodePtr(null);
111             tailPtr.setPrevNodePtr(prevNodePtr);
112         }
113         else{
114             headPtr.setNextNodePtr(null);
115             tailPtr.setPrevNodePtr(null);
116         }
117
118         return poppedData;
119
120     }
121
122
123     public  int peek(){
124
125         Node lastNodePtr = tailPtr.getPrevNodePtr();
126
127         if (lastNodePtr != null)
128             return lastNodePtr.getData();
```

```
129             else
130                 return -100000; //  empty stack
131 
132         }
133 
134 
135         public  void IterativePrint(){
136 
137             Node currentNodePtr = headPtr.getNextNodePtr();
138 
139             while (currentNodePtr != null){
140                 System.out.print(currentNodePtr.getData() + " ");
141                 currentNodePtr = currentNodePtr.getNextNodePtr();
142             }
143 
144             System.out.println();
145 
146         }
147 
148 
149 
150         public  void ReversePrint(){
151 
152             Node currentNodePtr = tailPtr.getPrevNodePtr();
153 
154             while (currentNodePtr != null){
155 
156                 System.out.print(currentNodePtr.getData() + " ");
157                 currentNodePtr = currentNodePtr.getPrevNodePtr();
158             }
159 
160             System.out.println();
161         }
162 
163 
164     }
165 
166 
167     class DoublyLinkedList{
168 
169         public static void main(String[] args){
170 
171         Scanner input = new Scanner(System.in);
172 
173         int stackSize;
174         System.out.print("Enter the number of elements you want to insert: ");
175         stackSize = input.nextInt();
176 
177         int maxValue;
178         System.out.print("Enter the maximum value for an element: ");
179         maxValue = input.nextInt();
180 
181         Random randGen = new Random(System.currentTimeMillis());
182 
183         Stack stack = new Stack(); // Create an empty stack
184 
185         for (int i = 0; i < stackSize; i++){
186             int value = randGen.nextInt(maxValue);
187             stack.push(value);
188             System.out.print(value + " ");
189         }
190 
191         System.out.println();
192 
```

```
193       //cout << "Contents of the Stack: ";
194       //stack.IterativePrint();
195
196
197       while (!stack.isEmpty()){
198
199           System.out.print(stack.pop() + " ");
200       }
201
202       System.out.println();
203
204      }
205
206   }
```

```
Enter the number of elements you want to insert: 10
Enter the maximum value for an element: 50
16 2 23 36 24 1 3 27 1 26
26 1 27 3 1 24 36 23 2 16
```