

NoInteraction exception:

Solution code

Please choose a language and write your code.

✓ SUBM

● ACCEPTED Score: 100 points (details)

CODE INPUT OUTPUT Java 8 ▾ ● RUN CODE

```
1 import java.io.*;
2 import java.util.*;
3 import java.text.*;
4 import java.math.*;
5 import java.util.regex.*;
6
7+ class NoInteractionException extends Exception{
8     NoInteractionException()
9     {
10         System.out.println("This Exception is Occured due to No-Interaction");
11     }
12 }
13
14+ class Trainer {
15     void askQuestion(String ques, String ans)
16     {
17         if(ans.equals("null"))
18         {
19             NoInteractionException n = new NoInteractionException();
20             System.out.println("NoInteractionException");
21         }
22     }
23 }
24
25+ public class Source extends Trainer{
26     public static void main(String args[] ) throws Exception {
27         Scanner sc=new Scanner(System.in);
28         String ques = sc.nextLine();
29         String ans=sc.nextLine();
30         System.out.println("Question:"+ques);
31         System.out.println("Answer:"+ans);
32 }
```

```
8     NoInteractionException()
9     {
10    |     System.out.println("This Exception is Occured due to No-Interaction");
11   }
12 }
13
14+ class Trainer {
15     void askQuestion(String ques, String ans)
16     {
17     |     if(ans.equals("null"))
18     {
19     |         NoInteractionException n = new NoInteractionException();
20     |         System.out.println("NoInteractionException");
21     }
22     }
23 }
24
25+ public class Source extends Trainer{
26+     public static void main(String args[] ) throws Exception {
27         Scanner sc=new Scanner(System.in);
28         String ques = sc.nextLine();
29         String ans=sc.nextLine();
30         System.out.println("Question:"+ques);
31         System.out.println("Answer:"+ans);
32
33         Trainer t=new Trainer();
34         t.askQuestion(ques,ans);
35
36     }
37 }
38 }
```

ⓘ 5 revisions found for this solution

⌚ SHOW REVISIONS

Stack:

The screenshot shows a Java code editor interface with the following layout:

- CODE**: The active tab where the Java code is written.
- INPUT**: A text input field for providing test data.
- OUTPUT**: A text output field showing the results of the program execution.
- Java 8**: The selected Java version.
- Run**: A small circular icon for running the code.

The Java code implements a stack using a Scanner for input and a UserStack class for storage. It takes user input to choose between pushing integers or floating-point numbers onto the stack, and then displays the stack's contents.

```
1 import java.io.*;
2 import java.util.*;
3 import java.text.*;
4 import java.math.*;
5 import java.util.regex.*;
6
7 // Class name should be "Source",
8 // otherwise solution won't be accepted
9 public class Source {
10    public static void main(String args[] ) throws Exception {
11        Scanner sc=new Scanner(System.in);
12        UserStack us=new UserStack();
13        int choice=sc.nextInt();
14        try{
15            if(choice==1)
16            {
17                while (sc.hasNext()){
18                    int data=sc.nextInt();
19                    us.push(data);
20                }
21            else if(choice==2){
22                while(sc.hasNext()){
23                    int data1=sc.nextInt();
24                    us.push(data1);
25                }
26                us.display();
27                for(int i=0;i<10;i++)
28                    us.pop();
29            }
30        }
31    }
32}
```

```
        user.pop();
29     }
30     }catch(FullStackException e){
31         System.out.print(e.getMessage());
32     }
33     }catch(EmptyStackException e){
34         System.out.print(e.getMessage());
35     }
36 }
37 }
38 class UserStack{
39     public UserStack(){}
40     int array1[]={};
41     public static int index=0;
42     public void push(int data) throws FullStackException{
43         if(index==9)
44             throw new FullStackException("Stack overflow");
45         array1[index]=data;
46         index++;
47     }
48     public int pop() throws EmptyStackException{
49         if(index==0)
50             throw new EmptyStackException("Stack empty");
51         return array1[--index];
52     }
53 }
54     public void display(){
55         for(int i=0;i<index;i++)
56             System.out.println(array1[i]);
57     }
58 }
```

```
53
54     public void display(){
55         for(int i=0;i<index;i++)
56             System.out.println(array1[i]);
57     }
58 }
59 }
60 class EmptyStackException extends Exception{
61     public EmptyStackException(String message){
62         super(message);
63     }
64 }
65 class FullStackException extends Exception{
66     public FullStackException(String msg){
67         super(msg);
68     }
69 }
```