Autocomplete not supported

```java
import java.util.*;

class ArrayListOps{
    public ArrayList<Integer> makeArrayListInt(int n){
        ArrayList<Integer> list = new ArrayList<Integer>();
        for(int i=0;i<n;i++)
        {
            list.add(0);
        }
        return list;
    }
    public ArrayList<Integer> reverseList(ArrayList<Integer> list){
        Collections.reverse(list);
        return list;
    }
public ArrayList<Integer> changeList(ArrayList<Integer> list, int m, int n){
    Collections.replaceAll(list,m,n);
    return list;
    }
}
public class Source extends ArrayListOps{
    public static void main(String[] args) {
        Scanner scr = new Scanner(System.in);
        String str = scr.nextLine();
        String[] s = str.split(", ");
        int len = s.length;
```

Ln 2, Col 1 Java

01 H 07 M 32 S

```
21  public class Source extends ArrayListOps{
22      public static void main(String[] args) {
23          Scanner scr = new Scanner(System.in);
24          String str = scr.nextLine();
25          String[] s = str.split(", ");
26          int len = s.length;
27          ArrayList<Integer> list = new ArrayList<Integer>();
28          for(int i=0;i<len;i++)
29          {
30              list.add(Integer.parseInt(s[i]));
31          }
32          ArrayList<Integer> revList = new ArrayList<Integer>();
33          for(int i=0;i<len;i++)
34          {
35              revList.add(list.get(len-1-i));
36          }
37          int n=scr.nextInt();
38          int m=scr.nextInt();
39          int p=scr.nextInt();
40
41          ArrayListOps es = new ArrayListOps();
42
43          System.out.println(es.makeArrayListInt(n));
44          System.out.println(es.reverseList(list));
45          System.out.println(es.changeList(revList,m,p));
46      }
```

Ln 31, Col 5

Handling Stuff

Coding

DESCRIPTION

In **Java**, we can use more than one catch block with the try block. Generally, multiple catch block is used to handle different types of exceptions, which means each catch block is used to handle different types of exceptions.

If you use multiple catch blocks for the same type of exception, then it will give you a compile-time error because **Java does not allow you to use multiple catch block for the same type of exception**. A catch block is always preceded by the try block.

Write a program to demonstrate Multiple Exceptions.

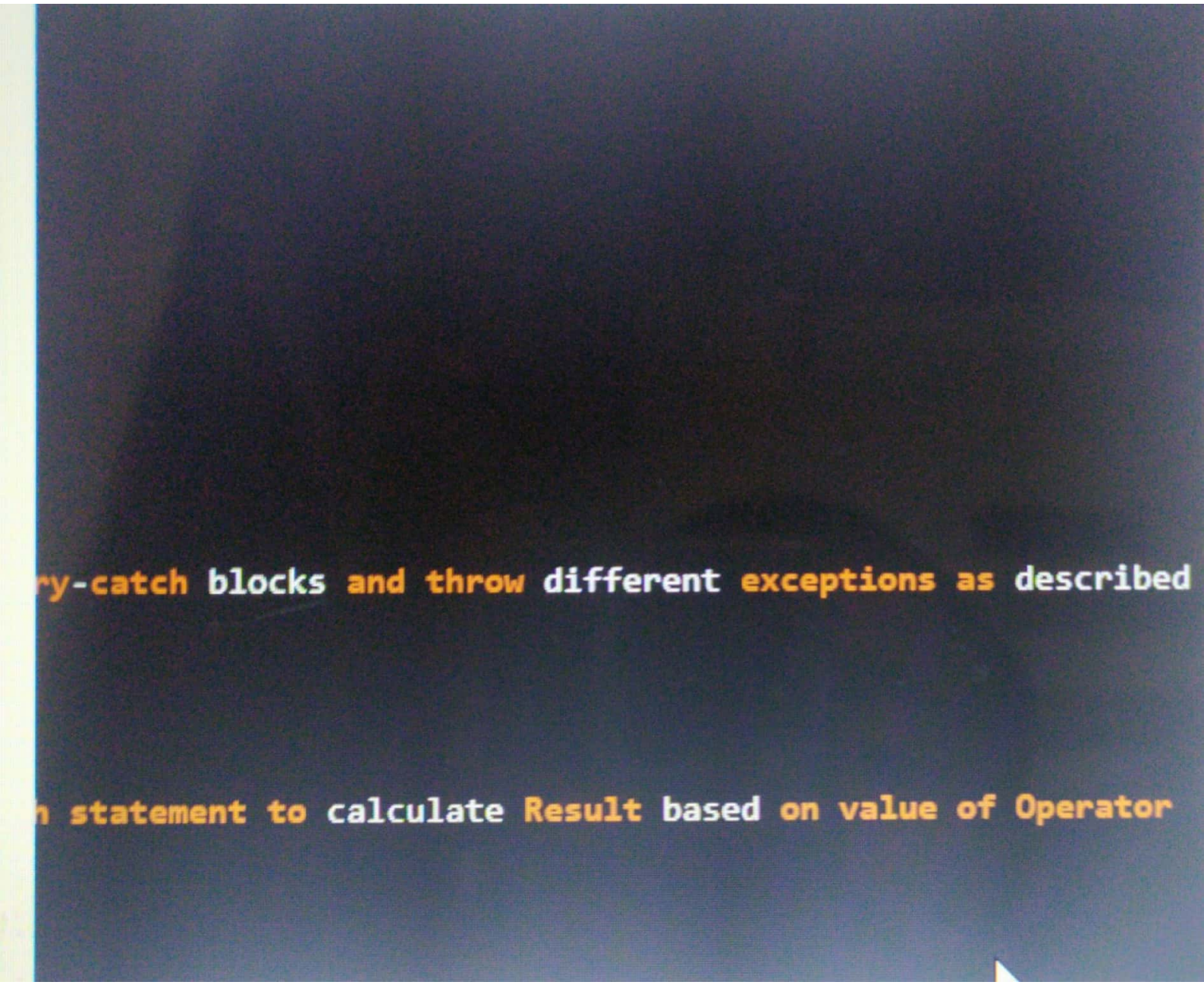
Specifications:

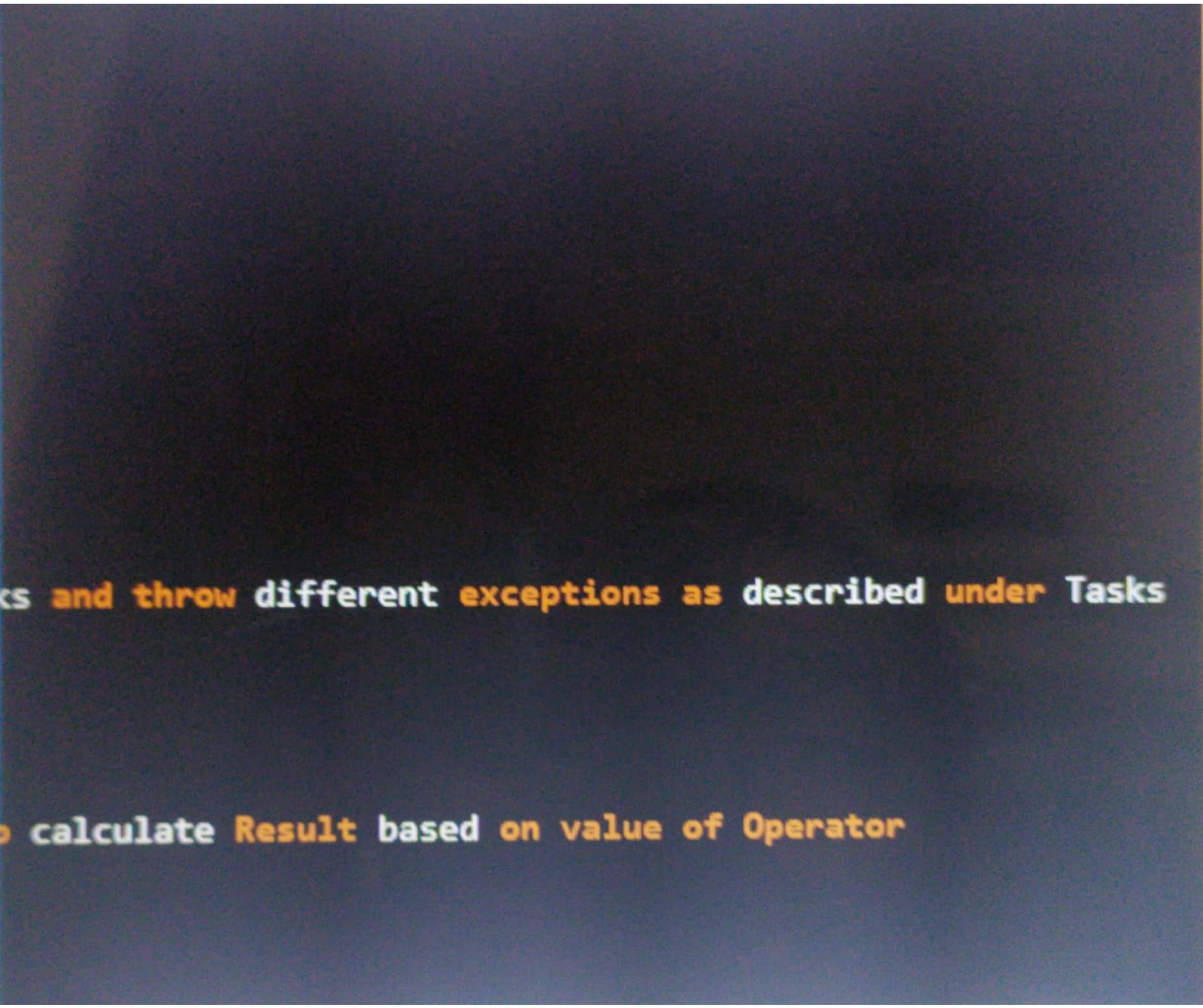
```
class Activity:
```

```
class          :
    data fields:
      String string1
      String string2
      String operator
    Constructor to initialize the class variables.

class Source:
        method definitions:
            handleException(Activity a): implement try-cat
                return type: String
                visibility: public

            doOperation(Activity a): implement switch stat
                return type: String
                visibility: public
```





You have to implement the following methods under Source class:

- **handleException (Activity a)** - In this function you have to check for exceptions.
- **doOperation (Activity a)** - this function should implement the string operation between **string1** and **string2** for the operator **operator**.
- If **operator = '+'**, concat the strings **string1** and **string2**.
- **e.g.** for **string1 = "hello"** and **string2 = "world"**, then **result = "helloworld"**
- If **operator = '-'**, replace the contents of **string2** in **string1** with empty string.
- **e.g.** If **string1 = "helloworld"** and **string2 = "world"**, then **result = "hello"**

Input Format

- The main() method has already been implemented, which will pass values for the variables: **string1**, **string2** and **operator**.

Input Format

- The **main()** method has already been implemented, which will pass values for the variables: **string1**, **string2** and **operator**.

Tasks:

In the function **handleException (Activity a)**:

- Check that the value of either **string1** or **string2** variable is **null**, then throw appropriate exception for **NullPointerException** and return "**Null values found**".
- Check if the value of **operator** variable is not equal to these string operators ((+ or -) using logical AND operator. If the condition is true then throw and return the default exception with the Operator as the return message.
- If no exception is found return "**No Exception Found**".

In the function **doOperation (Activity a)**:

- perform the string operations, using switch statement and

condition is true then throw and return the default exception with the Operator as the return message.

- If no exception is found return **"No Exception Found".**

In the function **doOperation (Activity a):**

- perform the string operations, using switch statement and return the correct value.

IMPORTANT:

- If you want to test your program, you can implement a **main()** function given in the stub and you can use **RUN CODE** to test your main() provided you have made valid function calls with valid data required.

EXECUTION TIME LIMIT

Default.

REPORT AN ISSUE

```
1  import java.io.*;…
6  class Activity{
7    //Implement Activity class here..
8    String string1;
9    String string2;
10   String operator;
11
12   public Activity(){
13     super();
14   }
15   public Activity(String string1,String string2,String operator){
16     this.string1 = string1;
17     this.string2 =  string2;
18     this.operator = operator;
19   }
20
21 }
22 public class Source extends Activity{
23   //Implement the two function given in description in here...
24   public String handleException(Activity a){
25     try{
26       if(string1==null || string2==null){
27
28       }
29       if(!operator.equals("+") && !operator.equals("-")){
30
```

```
try{
      if(string1==null || string2==null){

      }
      if(!operator.equals("+") && !operator.equals("-")){

      }
    }
    catch( NullPointerException e)
    {
      return "No values found";
    }
    catch(Exception e){
      return "No Exception Found";
    }
  }
  public String doOperation(Activity a){
    String result="";
    int num=0;
    if(operator.equals("+"))
    {
      num=1;
    }
    else if(operator.equals("-")){
      num=2;
    }
    switch(num){
```

```
49          num=2;
50      }
51      switch(num){
52          case 1:
53              result = string1+string2;
54
55          case 2:
56              if(string1.contains(string2))
57              {
58                  result = string1.replaceAll(string2,"");
59              }
60      }
61      return result;
62  }
63
64  public static void main(String args[] ) throws Exception {
65      //Write your own main to check the program...
66      Scanner scr = new Scanner(System.in);
67     // Source obj = new Source();
68      String s1 = scr.nextLine();
69      String s2 = scr.nextLine();
70      String operator = scr.next();
71      Activity obj = new Source();
72      System.out.println(obj.doOperation());
73  }
74  }
```