a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.

Specifications:

```
class definitions:
   class Header:
        Variables:
            String from
            String to
        Implement a parameterized constructor to initialize

   class Email:
        Variables:
            Header header
            String body
            String greetings
        Implement a parameterized constructor to initialize
  class EmailOperations:
        Methods:
            emailVerify(Email e): Use regular expression to
                Return type:int
                Visibility: public

            bodyEncryption(Email e): Use Ceasar cipher(Shift
                Return type:String
                Visibility: public

            greetingMessage(Email e): In this method you hav
                Return type:String
                Visibility: public
```

Autocomplete ready

```
74          return cipher;
75
76      }
77      public String greetingMessage(Email e){
78          String name = e.header.from;
79          int index = name.indexOf('@');
80          String only_name = name.substring(0,index);
81          String final_greet = only_name.concat(e.greetings);
82          return final_greet;
83      }
84      }
85
86
87      public class Source {
88          public static void main(String args[] ) throws Exception {
89              /* Enter your code here. Read input from STDIN. Print output to STDOUT
    */
90          // You can Implement your main() to check your Program.
91          Header h = new Header("anshuman@cap.com", "aditya@cap.com");
92          Email e = new Email(h, "Hi", "Thank You");
93          EmailOperations eop = new EmailOperations();
94          eop.emailVerify(e);
95          eop.bodyEncryption(e);
96          eop.greetingMessage(e);
97
98      }
```

Ln 32, Col 67 Java 8

1 revision found for this solution. SHOW REVISIONS

Class Variables:

- **class Header:** It contains two email id 'from' and 'to'. 'from' signifies the sender's email address and 'to' signifies receiver's email address.
- **class Email:** This class contains three parts: first Header header which has two email address from and to,the second body which contains the message to send and third greetings which contains greetings such as "Regards", "Thank you", etc.

To access a variable in Header class through Email object we use:

```
‹Email(obj)›.‹Email(variable)›.‹Header(variable)›
```

Example to access "from" address from the Email object e we use : e.header.from;

Tasks:

- Implement the two classes Email and Header class according to the specifications.
- Implement the three methods in the EmailOperations class:

1. emailVerify (Email e)
2. bodyEncryption (Email e)
3. greetingMessage (Email e)

Method Description:

1. emailVerify(Email e):

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**.

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

a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.

Specifications:

```
class definitions:
   class Header:
        Variables:
            String from
            String to
        Implement a parameterized constructor to initialize

   class Email:
        Variables:
            Header header
            String body
            String greetings
        Implement a parameterized constructor to initialize
  class EmailOperations:
        Methods:
            emailVerify(Email e): Use regular expression to
                Return type:int
                Visibility: public

            bodyEncryption(Email e): Use Ceasar cipher(Shift
                Return type:String
                Visibility: public

            greetingMessage(Email e): In this method you hav
                Return type:String
                Visibility: public
```

Autocomplete ready

```
49  public String bodyEncryption(Email e){
50      String plaintext = e.body;
51      int shift =3;
52      String cipher="";
53      char alb;
54      for(int i=0;i<plaintext.length();i++){
55          alb = plaintext.charAt(i);
56          if(alb>='a'&& alb<='z'){
57              alb=(char)(alb+shift);
58              if(alb>'z'){
59                  alb = (char)(alb + 'a'-'z'-1);
60              }
61              cipher = cipher +alb;
62          }
63          else if(alb>='A' && alb <= 'Z'){
64              alb=(char)(alb+shift);
65              if(alb>'Z'){
66                  alb=(char)(alb + 'A'-'Z'-1);
67              }
68              cipher=cipher+alb;
69          }
70          else{
71              cipher=cipher+alb;
72          }
73      }
74          return cipher;
```

Ln 32, Col 67 Java 8

1 revision found for this solution.

SHOW REVISIONS

a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.

Specifications:

```
class definitions:
   class Header:
        Variables:
            String from
            String to
        Implement a parameterized constructor to initialize

   class Email:
        Variables:
            Header header
            String body
            String greetings
        Implement a parameterized constructor to initialize
  class EmailOperations:
        Methods:
            emailVerify(Email e): Use regular expression to
                Return type:int
                Visibility: public

            bodyEncryption(Email e): Use Ceasar cipher(Shift
                Return type:String
                Visibility: public

            greetingMessage(Email e): In this method you hav
                Return type:String
                Visibility: public
```

Autocomplete ready

```
1  import java.io.*;
6  class Email{
7    // Implement Email Class according to the specifiaction.
8    Header header;
9    String body;
10   String greetings;
11 Email(Header h,String b, String greet){
12       this.header=h;
13       this.body=b;
14       this.greetings=greet;
15   }
16   }
17 class Header{
18   // Implemet the Header Class according to the specifiaction.
19   String from;
20   String to;
21 Header(String fr,String to){
22       this.from=fr;
23       this.to=to;
24 }
26
27
28 class EmailOperations{
29   // Implemet the Three methods specified in the specified.
30
31 public int emailVerify(Email e){
```

Ln 6, Col 13 Java 8

1 revision found for this solution. SHOW REVISIONS

a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.

Specifications:

```
class definitions:
   class Header:
        Variables:
            String from
            String to
        Implement a parameterized constructor to initialize

   class Email:
        Variables:
            Header header
            String body
            String greetings
        Implement a parameterized constructor to initialize
  class EmailOperations:
        Methods:
            emailVerify(Email e): Use regular expression to
                Return type:int
                Visibility: public

            bodyEncryption(Email e): Use Ceasar cipher(Shift
                Return type:String
                Visibility: public

            greetingMessage(Email e): In this method you hav
                Return type:String
                Visibility: public
```

Autocomplete ready

```
27
28  class EmailOperations{
29    // Implemet the three methods specified in the specified.
30
31  public int emailVerify(Email e){
32    String regex = "^[\\w!#$%&'*+/=?`{|}~^-]+(?:\\.[\\w!#$%&'*+/=?`{|}~^-]+)*@(?
      :[a-zA-Z0-9
                Pattern pattern = EmailOperations.emailVerify(Email)
33
                Pattern pattern = EmailOperations.emailVerify(Email)
34
35      Pattern pattern = Pattern.compile(regex);
36      Matcher match1 = pattern.matcher(e.header.from);
37      Matcher match2 = pattern.matcher(e.header.to);
38      boolean first = match1.matches();
39      boolean second = match2.matches();
40      if(match1.matches()==true && match2.matches()==true)
41      return 2;
42      if(first==true && second==false || first==false && second==true)
43      return 1;
44      if(first==false && second==false)
45      return 0;
46
47      return 0;
48  }
49  public String bodyEncryption(Email e){
50      String plaintext = e.body;
51      int shift =3;
52      String ciphere="";
```

Ln 32, Col 67 Java 8

1 revision found for this solution. SHOW REVISIONS

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 [redacted]:
    data fields:
      String string1
      String string2
      String operator
    Constructor to initialize the class [redacted].

class [redacted]:
       method definitions:
           handleException(Activity a): implement try-catch
               return type: String

           doOperation(Activity a): implement switch statem
               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

Autocomplete ready

```
26    public String handleException(Activity a){
27
28      try{
29            if(a.string1 == null && a.string2 == null){
30            throw new NullPointerException();
31          }
32          if((a.operator != "+") && (a.operator != "-" )){
33            throw new Exception();
34          }
35          return "No Exception Found";
36          }
37      catch(NullPointerException e){
38        return "Null values found";
39      }
40       catch(Exception e){
41              return a.operator;
42          }
43  }
44    public String doOperation(Activity a){
45      String result = "";
46      if(a.operator == "+"){
47          result = a.string1.concat(a.string2);
48      }
49      else if(a.operator == "-"){
50        result = a.string1.replace(a.string2,"");
51      }
```

Ln 55, Col 3 Java 8

4 revisions found for this solution. SHOW REVISIONS

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 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

10 seconds

REPORT AN ISSUE

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 [redacted]:
    data fields:
      String string1
      String string2
      String operator
    Constructor to initialize the class [redacted].

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

            doOperation(Activity a): implement switch statem
                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

Autocomplete ready

```
1  import java.io.*;
6  class Activity{
7    //Implement Activity class here..
8    String string1;
9    String string2;
10   String operator;
11
12   Activity(){}
13
14   Activity(String st1,String st2,String op){
15     this.string1=st1;
16     this.string2=st2;
17     this.operator=op;
18   }
19 }
20
21
22
23 public class Source {
24   //implement the two function given in description in here...
25
26     public String handleException(Activity a){
27
28       try{
29           if(a.string1 == null && a.string2 == null){
30           throw new NullPointerException();
```

Ln 55, Col 3 Java 8

4 revisions found for this solution. SHOW REVISIONS

a precise manner. Consider default visibility of classes, data fields and methods unless mentioned otherwise.

Specifications:

```
class definitions:
   class Header:
        Variables:
            String from
            String to
        Implement a parameterized constructor to initialize

   class Email:
        Variables:
            Header header
            String body
            String greetings
        Implement a parameterized constructor to initialize
  class EmailOperations:
        Methods:
            emailVerify(Email e): Use regular expression to
                Return type:int
                Visibility: public

            bodyEncryption(Email e): Use Ceasar cipher(Shift
                Return type:String
                Visibility: public

            greetingMessage(Email e): In this method you hav
                Return type:String
                Visibility: public
```

Autocomplete ready

```
70          else{
71              cipher=cipher+alb;
72          }
73      }
74          return cipher;
75
76  }
77  public String greetingMessage(Email e){
78      String name = e.header.from;
79      int index = name.indexOf('@');
80      String only_name = name.substring(0,index);
81      String final_greet = only_name.concat(e.greetings);
82      return final_greet;
83  }
84  }
85
86
87  public class Source {
88      public static void main(String args[] ) throws Exception {
89          /* Enter your code here. Read input from STDIN. Print output to STDOUT
    */
90      // You can implement your main() to check your Program.
91      Header h = new Header("anshuman@cap.com", "aditya@cap.com");
92      Email e = new Email(h, "Hi", "Thank You");
93      EmailOperations eop = new EmailOperations();
94      eop.emailVerify(e);
```

Ln 32, Col 67 Java 8

1 revision found for this solution. SHOW REVISIONS

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 [redacted]:
    data fields:
      String string1
      String string2
      String operator
    Constructor to initialize the class [redacted].

class [redacted]:
        method definitions:
            handleException(Activity a): implement try-catch
                return type: String

            doOperation(Activity a): implement switch statem
                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

Autocomplete ready

```
    catch(NullPointerException e){
      return "Null values found";
    }
     catch(Exception e){
           return a.operator;
       }
  }
  public String doOperation(Activity a){
    String result = "";
    if(a.operator == "+"){
       result = a.string1.concat(a.string2);
    }
    else if(a.operator == " "){
      result = a.string1.replace(a.string2,"");
    }
    return result;

  }
public static void main(String args[] ) throws Exception {
  //Write your own main to check the program...
  Activity ac = new Activity("Hello","World","+");
  Source s = new Source();
  s.handleException(ac);
  s.doOperation(ac);
 }
}
```

Ln 55, Col 3 Java 8

4 revisions found for this solution. SHOW REVISIONS