IRC_SKCT_Java2_CY_COD_Exception

Test Summary

- No. of Sections: 1
- No. of Questions: 5
- Total Duration: 120 min

Section 1 - Coding

Section Summary

- No. of Questions: 5
- Duration: 120 min

Additional Instructions:

None

Q1. Write a program to valid the email address and display suitable exception if there is any mistake.

Create 3 custom exceptions class as below

1. **DotException**
2. **AtTheRateException**
3. **DomainException**

A typical email address should have a **" . "** character, **"@"** character and also the domain name should be valid. Valid domain names for practice be '**in**', '**com**', '**net**' or '**biz**'.

Display **Invalid Dot usage**, **Invalid @ usage** or **Invalid Domain** message based on email id.

Get the email address from the user, validate the email by checking the above-mentioned criteria and print the validity status of the input email address.

Input Format

First line of input contains the email to be validated

Output Format

Print **Valid email address** or **Invalid email address** along with suitable exception

Sample Input

```
sample@gmail.com
```

Sample Output

```
Valid email address
```

Sample Input

```
sample@gmail.com.
```

Sample Output

```
DotException: Invalid Dot usage
Invalid email address
```

Sample Input

```
sample@g@mail.com
```

Sample Output

```
AtTheRateException: Invalid @ usage
Invalid email address
```

Sample Input

```
sample@gmail.con
```

Sample Output

```
DomainException: Invalid Domain
Invalid email address
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q2. Write a program to valid the email address and display suitable exception if there is any mistake.

Create 3 custom exceptions class as below

1. **DotException**
2. **AtTheRateException**
3. **DomainException**

A typical email address should have a **" . "** character, **"@"** character and also the domain name should be valid. Valid domain names for practice be '**in**', '**com**', '**net**' or '**biz**'.

Display **Invalid Dot usage**, **Invalid @ usage** or **Invalid Domain** message based on email id.

Get the email address from the user, validate the email by checking the above-mentioned criteria and print the validity status of the input email address.

Input Format

First line of input contains the email to be validated

Output Format

Print **Valid email address** or **Invalid email address** along with suitable exception

Sample Input

```
sample@gmail.com
```

Sample Output

```
Valid email address
```

Sample Input

```
sample@gmail.com.
```

Sample Output

```
DotException: Invalid Dot usage
Invalid email address
```

Sample Input

Sample Output

sample@g@mail.com	AtTheRateException: Invalid @ usage Invalid email address

Sample Input **Sample Output**

sample@gmail.con	DomainException: Invalid Domain Invalid email address

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q3. **Divide by zero exception.**
Write a program to obtain two numbers and print their quotient. In case of exception print the same.

Input Format

Given a single line input separated by space.get the Integer N1 and N2

Output Format

Display the quotient if there is no Exception.else print the Exception,

Constraints

Integers only.

Sample Input **Sample Output**

44 2	22

Sample Input **Sample Output**

2 0	java.lang.ArithmeticException: / by zero

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q4. **NullPointerException**
Another prominent exception is NullPointerException. It occurs when you try to access a null value. Assign null value to a string and obtain an index position and try to access it. Print the exception.

Input Format

Input consists of an integer.

Output Format

Output prints the null pointer exception.

Sample Input **Sample Output**

9	null java.lang.NullPointerException

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q5. Write a program to validate the given password. A password is said to be strong if it satisfies the following criteria
i) It should be minimum of 10 characters and a maximum of 20 characters
ii) It should contain at least one digit
iii)It should contain at least one special character (!,@,#,$,%,^,&,*)

If the password fails any one of the criteria, it is considered as weak.

Create a class called **User** with the following private attributes.

1. name as String
2. mobile as String
3. username as String
4. password as String

Create a class called UserBO with following methods.

static void validate(User u) This method throws Exception with a suitable message if the Password is weak.

Create a Mainclass get inputs from the user. Validate the password and if there is an exception, handle the exception and prompt the user with a suitable message.

Refer Sample input and output for exact statement

Input Format

Name
Phone number
User Name
Password

Output Format

Print **Valid Password** or suitable exception

Constraints

Special characters are !,@,#,$,%,^,&,*

Sample Input **Sample Output**

John 9874563210 john john1#phoj	Valid Password

Sample Input **Sample Output**

```
John
9874563210
john
john#nhoi
```

Sample Input

```
java.lang.Exception: Should contain at least one digit
```

Sample Output

```
John
9874563210
john
john1nhoi
```

Sample Input

```
java.lang.Exception: It should contain at least one special character
```

Sample Output

```
John
9874563210
john
john9#
```

Sample Input

```
java.lang.Exception: Should be minimum of 10 characters and maximum of 20 ch
```

Sample Output

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Answer Key & Solution

Section 1 - Coding

Q1

Test Case

Input

```
a@b.v
```

Output

```
DomainException: Invalid Domain
Invalid email address
```

Weightage - 10

Input

```
abc@@gmail.com
```

Output

```
AtTheRateException: Invalid @ usage
Invalid email address
```

Weightage - 15

Input

```
abc@gmail
```

Output

```
DotException: Invalid Dot usage
Invalid email address
```

Weightage - 15

Input

```
abc@abc.co
```

Output

```
DomainException: Invalid Domain
Invalid email address
```

Weightage - 15

Input

```
abc@google.net
```

Output

```
Valid email address
```

Weightage - 15

Input

```
abc@ab.c.com
```

Output

```
DotException: Invalid Dot usage
Invalid email address
```

Weightage - 15

Input

```
examly@examly.in
```

Output

```
Valid email address
```

Weightage - 15

Sample Input

```
sample@gmail.com
```

Sample Output

```
Valid email address
```

Sample Input

```
sample@gmail.com.
```

Sample Output

```
DotException: Invalid Dot usage
Invalid email address
```

Sample Input

```
sample@g@mail.com
```

Sample Output

```
AtTheRateException: Invalid @ usage
Invalid email address
```

Sample Input

Sample Output

sample@gmail.con	DomainException: Invalid Domain Invalid email address

Solution

```
import java.util.Scanner;

class DomainException extends Exception {
       String expDescription;
        // public constructor with String argument
       DomainException(String expDescription) {
            super(expDescription);
        }
    }

class DotException extends Exception {
       String expDescription;
        // public constructor with String argument
       DotException(String expDescription) {
            super(expDescription);
        }
    }

class AtTheRateException extends Exception {
       String expDescription;
        // public constructor with String argument
       AtTheRateException(String expDescription) {
            super(expDescription);
        }
    }


class EmailValidationMain {
    public static void main(String[] args) {

        Scanner myObj = new Scanner(System.in);

        String email = myObj.next();

        boolean checkEndDot  = false;
        checkEndDot = email.endsWith(".");


        int indexOfAt = email.indexOf('@');
        int lastIndexOfAt = email.lastIndexOf('.');

        int countOfAt = 0;

        for (int i = 0; i < email.length(); i++) {
            if(email.charAt(i)=='@')
                countOfAt++;
            }

        String buffering = email.substring(email.indexOf('@')+1, email.length());
        int len = buffering.length();

        int countOfDotAfterAt = 0;
        for (int i=0; i < len; i++) {
            if(buffering.charAt(i)=='.')
                countOfDotAfterAt++; }

String userName = email.substring(0, email.indexOf('@'));
String domainName = email.substring(email.indexOf('.')+1, email.length());


int domainCheck=0;
if((domainName.equals("in")) || (domainName.equals("com")) || (domainName.equals("net")) || (domainName.equals("biz")))
    domainCheck=1;

try {
    if((checkEndDot) || (countOfDotAfterAt!=1)) {
        throw new DotException("Invalid Dot usage");
    }

    if(countOfAt!=1) {
        throw new AtTheRateException("Invalid @ usage");
    }

    if(domainCheck!=1) {
        throw new DomainException("Invalid Domain");
    }

}catch(DotException e) {
    System.out.println(e);
}catch(AtTheRateException e) {
    System.out.println(e);
}catch(DomainException e) {
    System.out.println(e);
```

```
        }

        if ((countOfAt==1) && (userName.endsWith(".")==false)  && (domainCheck==1) && (countOfDotAfterAt ==1) &&((indexOfAt+3) <= (lastIndexOfAt) && !checkEndDot)) {
            System.out.println("Valid email address");
        }

         else {
             System.out.println("Invalid email address");
         }
        myObj.close();
    }

}
```

Q2

Test Case

Input

```
a@b.v
```

Output

```
DomainException: Invalid Domain
Invalid email address
```

Weightage - 10

Input

```
abc@@gmail.com
```

Output

```
AtTheRateException: Invalid @ usage
Invalid email address
```

Weightage - 15

Input

```
abc@gmail
```

Output

```
DotException: Invalid Dot usage
Invalid email address
```

Weightage - 15

Input

```
abc@abc.co
```

Output

```
DomainException: Invalid Domain
Invalid email address
```

Weightage - 15

Input

```
abc@google.net
```

Output

```
Valid email address
```

Weightage - 15

Input

```
abc@ab.c.com
```

Output

```
DotException: Invalid Dot usage
Invalid email address
```

Weightage - 15

Input

```
examly@examly.in
```

Output

```
Valid email address
```

Weightage - 15

Sample Input

```
sample@gmail.com
```

Sample Output

```
Valid email address
```

Sample Input

```
sample@gmail.com.
```

Sample Output

```
DotException: Invalid Dot usage
Invalid email address
```

Sample Input

```
sample@g@mail.com
```

Sample Output

```
AtTheRateException: Invalid @ usage
Invalid email address
```

Sample Input

```
sample@gmail.con
```

Sample Output

```
DomainException: Invalid Domain
Invalid email address
```

Solution

```
import java.util.Scanner;

class DomainException extends Exception {
       String expDescription;
        // public constructor with String argument
       DomainException(String expDescription) {
            super(expDescription);
        }
    }

class DotException extends Exception {
       String expDescription;
        // public constructor with String argument
       DotException(String expDescription) {
            super(expDescription);
        }
    }

class AtTheRateException extends Exception {
       String expDescription;
        // public constructor with String argument
       AtTheRateException(String expDescription) {
            super(expDescription);
        }
    }


class EmailValidationMain {
    public static void main(String[] args) {

        Scanner myObj = new Scanner(System.in);

        String email = myObj.next();

        boolean checkEndDot  = false;
        checkEndDot = email.endsWith(".");


        int indexOfAt = email.indexOf('@');
        int lastIndexOfAt = email.lastIndexOf('.');
        int countOfAt = 0;

for (int i = 0; i < email.length(); i++) {
    if(email.charAt(i)=='@')
        countOfAt++;
    }

String buffering = email.substring(email.indexOf('@')+1, email.length());
int len = buffering.length();

int countOfDotAfterAt = 0;
for (int i=0; i < len; i++) {
    if(buffering.charAt(i)=='.')
        countOfDotAfterAt++; }

String userName = email.substring(0, email.indexOf('@'));
String domainName = email.substring(email.indexOf('.')+1, email.length());


int domainCheck=0;
if((domainName.equals("in")) || (domainName.equals("com")) || (domainName.equals("net")) || (domainName.equals("biz")))
    domainCheck=1;

try {
    if((checkEndDot) || (countOfDotAfterAt!=1)) {
        throw new DotException("Invalid Dot usage");
    }

    if(countOfAt!=1) {
```

```
                throw new AtTheRateException("Invalid @ usage");
            }

            if(domainCheck!=1) {
                throw new DomainException("Invalid Domain");
            }

        }catch(DotException e) {
            System.out.println(e);
        }catch(AtTheRateException e) {
            System.out.println(e);
        }catch(DomainException e) {
            System.out.println(e);
        }

        if ((countOfAt==1) && (userName.endsWith(".")==false)  && (domainCheck==1) && (countOfDotAfterAt ==1) &&((indexOfAt+3) <= (lastIndexOfAt) && !checkEndDot)) {
            System.out.println("Valid email address");
        }

         else {
             System.out.println("Invalid email address");
         }
        myObj.close();
    }

}
```

Q3

Test Case

Input

```
2 0
```

Output

```
java.lang.ArithmeticException: / by zero
```

Weightage - 20

Input

```
44 2
```

Output

```
22
```

Weightage - 20

Input

```
48 12
```

Output

```
4
```

Weightage - 20

Input

```
65 5
```

Output

```
13
```

Weightage - 20

Input

```
80 2
```

Output

```
40
```

Weightage - 20

Sample Input

```
44 2
```

Sample Output

```
22
```

Sample Input

```
2 0
```

Sample Output

```
java.lang.ArithmeticException: / by zero
```

Solution

```
import java.util.Scanner;
class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        try
        {
        int a=sc.nextInt();
        int b=sc.nextInt();
            int  c= a/b;
            System.out.println(c);
        } catch (Exception e)
        {
            System.out.println(e);
        }
    }
}
```

Q4

Test Case

Input

```
8
```

Output

```
null
java.lang.NullPointerException
```

Weightage - 20

Input

```
10
```

Output

```
null
java.lang.NullPointerException
```

Weightage - 20

Input

```
12
```

Output

```
null
java.lang.NullPointerException
```

Weightage - 20

Input

```
7
```

Output

```
null
java.lang.NullPointerException
```

Weightage - 20

Input

```
20
```

Output

```
null
java.lang.NullPointerException
```

Weightage - 20

Sample Input

```
9
```

Sample Output

```
null
java.lang.NullPointerException
```

Solution

```
import java.io.*;
import java.util.*;
class Main {
    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        try {
            String str = null;
            int index = Integer.parseInt(sc.nextLine());
            System.out.println(str);
            System.out.println(str.charAt(index));
        }
        catch(NullPointerException n) {
```

```
            System.out.println(n);
        }
    }
}
```

Q5

Test Case

Input

```
Kumar
1234567890
Kumar
Kumar123%
```

Output

```
Valid Password
```

Weightage - 10

Input

```
Kumar
1234567890
Kumar
Kumar123
```

Output

```
java.lang.Exception: Should be minimum of 10 characters and maximum of
```

Weightage - 10

Input

```
Kumar
1234567890
Kumar
Kumanntntn%
```

Output

```
java.lang.Exception: Should contain at least one digit
```

Weightage - 10

Input

```
Kumar
1234567890
Kumar
Kumanntntn%3dfddfdfdf
```

Output

```
java.lang.Exception: Should be minimum of 10 characters and maximum of
```

Weightage - 10

Input

```
Kumar
1234567890
Kumar
Kum1$
```

Output

```
java.lang.Exception: Should be minimum of 10 characters and maximum of
```

Weightage - 10

Input

```
Banu
89898567890
banubtech
BaNu
```

Output

```
java.lang.Exception: Should be minimum of 10 characters and maximum of
```

Weightage - 10

Input

```
Banu
89898567890
banubtech
BaNuldfldfdlfdlfldfdlfldfdlfd
```

Output

```
java.lang.Exception: Should be minimum of 10 characters and maximum of
```

Weightage - 10

Input

```
Banu
89898567890
banubtech
BaNu6*now
```

Output

```
Valid Password
```

Weightage - 10

Input

```
Banu
89898567890
banubtech
BaNu87654321
```

Output

```
java.lang.Exception: It should contain at least one special character
```

Weightage - 10

Input

```
Banu
89898567890
banubtech
BaNu**&^%$$%
```

Output

```
java.lang.Exception: Should contain at least one digit
```

Weightage - 10

Sample Input

```
John
9874563210
john
john1#nhoi
```

Sample Output

```
Valid Password
```

Sample Input

```
John
9874563210
john
john#nhoi
```

Sample Output

```
java.lang.Exception: Should contain at least one digit
```

Sample Input

```
John
9874563210
john
john1nhoi
```

Sample Output

```
java.lang.Exception: It should contain at least one special character
```

Sample Input

```
John
9874563210
john
john9#
```

Sample Output

```
java.lang.Exception: Should be minimum of 10 characters and maximum of
```

Solution

```
import java.util.Arrays;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

class User{
    String name;
    String mobile;
    String username;
    String password;

    public User(String name, String mobile, String username, String password) {
        super();
        this.name = name;
        this.mobile = mobile;
        this.username = username;
        this.password = password;
    }

}

class UserBO{
    static void validate(User u) throws Exception {

        String pattern = "[!|@|#|$|%|^|&|*]";
        Pattern a = Pattern.compile(pattern);
        Matcher m1 = a.matcher(u.password);

        String pattern2 = "[1|2|3|4|5|6|7|8|9|0]";
        Pattern b = Pattern.compile(pattern2);
        Matcher m2 = b.matcher(u.password);
       // System.out.println(u.password);

        if((u.password.length()<9) || (u.password.length()>20) ) {
            throw new Exception("Should be minimum of 10 characters and maximum of 20 characters");
        }

        else if(!m2.find()){
            throw new Exception("Should contain at least one digit");
        }

        else if(!m1.find()){
            throw new Exception("It should contain at least one special character");
        }
        else
            System.out.println("Valid Password");
    }
}

class PasswordMain{
    public static void main(String args[]) throws Exception {
        Scanner myObj = new Scanner(System.in);
```

```
        String name = myObj.nextLine();
        String mobile= myObj.nextLine();
        String username= myObj.nextLine();
        String password= myObj.nextLine();

        User userOne = new User(name, mobile, username, password);

        try{
            UserBO.validate(userOne);
        }
        catch(Exception e){
            System.out.println(e);
        }



    }
}
```