IRC_SKCT_Java2_COD_Exception

Test Summary

- No. of Sections: 1
- No. of Questions: 10
- Total Duration: 120 min

Section 1 - Coding

Section Summary

- No. of Questions: 10
- Duration: 120 min

Additional Instructions:

None

Q1. Input an integer as dividend, input another integer as divisor and print the result of division. If the divisor is 0, catch the resulting exception and print the message “divide by 0”

Input Format

The first line of input consists of an integer that represents the dividend
The second line of input consists of an integer that represents the divisor

Output Format

The output prints the result of division.
Refer to the sample input and output for formatting specifications.

Sample Input

```
6
3
```

Sample Output

```
2
```

Sample Input

```
1
0
```

Sample Output

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

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

Q2. **NumberFormatException**
Another common type of exception which you would have come across already. When you use BufferedReader to read input you need to parse String it into various datatype like Integer, Double. For example, If you try to parse a String ("abc") into Integer, it throws NumberFormatException. So let's try to handle this NumberFormat exception.
In our application, while acquiring attributes for classes like ItemType, this exception may occur. So try to handle it in this program.
Create a class ItemType with the following attribute,

Attributes	Data type
name	String
deposit	Double
costPerDay	Double

Add appropriate getter/setter, default and parameterized constructor. public ItemType(String name, Double deposit, Double costPerDay). Override toString() and print the details.
Handle the NumberFormatException in the Main Class.
Refer sample input/output for other further details and format of the output.

Input Format

The first line of the input consists of the name.
The second line of the input consists of the deposit.
The third line of the input consists of the costPerDay.

Output Format

The output prints the item details or the exception details.

Sample Input

```
Electronics
1000
100
```

Sample Output

```
Electronics 1000.0 100.0
```

Sample Input

```
Electronics
one thousand
```

Sample Output

```
java.lang.NumberFormatException: For input string: "one thousand"
```

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

Q3. Write a program to read the Register Number and Mobile Number of a student. Create user defined exception and handle the following:

1. If the Register Number does not contain exactly 9 characters in specified format(2 numbers followed by 3 characters followed by 4 numbers) or if the Mobile Number does not contain exactly 10 characters, throw an IllegalArgumentException.
2. If the Mobile Number contains any character other than a digit, raise a NumberFormatException.
3. If the Register Number contains any character other than digits and alphabets, throw a NoSuchElementException.
4. If they are valid, print the message ‘valid’ else ‘Invalid’.

Input Format

Register number as a string in the first line
Mobile number as a string in the second line

Output Format

Valid or Invalid with exception message

Refer sample outputs for format and exact text

Sample Input

```
19ABC1001
9949596920
```

Sample Output

```
Valid
```

Sample Input

```
19ABC1001
99495969209
```

Sample Output

```
Invalid
java.lang.IllegalArgumentException: Mobile Number does not contain exactly 1
```

Sample Input

```
19ABC10019
9949596920
```

Sample Output

```
Invalid
java.lang.IllegalArgumentException: Register Number does not contain exactly
```

Sample Input

```
195AC1001
9949596920
```

Sample Output

```
Invalid
java.util.NoSuchElementException: Registration Number cannot contain any cha
```

Sample Input

```
19ABC1001
994C596920
```

Sample Output

```
Invalid
java.lang.NumberFormatException: Mobile Number cannot contain any character
```

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

Q4. 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

Q5. 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

Q6. **ArrayIndexOutOfBoundsException:**
The prominent exception which you will see is ArrayIndexOutOfBoundsException. It occurs when the program try to access the array beyond its size. As we know arrays have fixed size. So when you try to use array beyond its size it throws this exception. Let's try to handle this exception.
Get an Array of size N and get an index, then print the Array[index]. If the index is greater or equal to array size(N), then print the Exception.

Divide by zero exception:
When you try to divide any number by Zero it will throw ArithmeticException: / by Zero
Get two numbers Then print the quotient if the divisor is 0 then print the Exception.

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

Input Format

The first line consists of array size(N).
The second line consists of N integers separated by space.
The third line consists of the Index value to retrieve the array element.
The fourth line consists of two integers(Dividend and Divisor) separated by space.
The fifth line consists of an index value to get the character from the string.

Output Format

The first line consists of Array[Index] or ArrayIndexOutOfBoundException.
The second line consists of the result of division or ArithmeticException.
The third line consists of String(Which is assigned to null value).
The fourth line consists of NullPointerException.
Refer to the sample input and output for formatting specifications.

Sample Input

```
5
1 2 3 4 5
6
4 0
```

Sample Output

```
Array index out of bound.
java.lang.ArithmeticException: / by zero
null
java.lang.NullPointerException
```

Sample Input

```
4
10 89 76 12
3
4 2
```

Sample Output

```
12
2
null
java.lang.NullPointerException
```

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

Q7. **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

```
44 2
```

Sample Output

```
22
```

Sample Input

```
2 0
```

Sample Output

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

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

Q8. **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

```
9
```

Sample Output

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

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

Q9. **User defined Exception**

Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, user can also create exceptions which are called 'user-defined Exceptions'.
Create a class **Bank** with the following private attributes and Create class BankBO with the following method.

Attributes	Datatype
accno	Integer
name	String
balance	Double

Method name	Description
static void validate(Bank b)	This method throws InvalidBalance exception if the balance is less than 1000.

Include appropriate getters/setters and add constructors.
Create a driver class called Main. In the Main method, obtain inputs from the user. Validate the balance and if there is an exception, handle the exception and prompt the user(Refer I/O)

Pass the exception message as "Balance is less than 1000".

Input Format

First line of the input consists of account number
Second line of the input consist of name of the account holder
Third line of the input consists of the account balance

Output Format

Output prints the account details if the balance is greater than 1000 otherwise throws an invalid balance exception.

Sample Input

```
10001
Ankit
5000
```

Sample Output

```
10001 Ankit 5000.0
```

Sample Input

```
10001
Ankit
500
```

Sample Output

```
Balance is less than 1000
InvalidBalanceException
```

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

Q10. 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

```
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 20 ch
```

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

Answer Key & Solution

Section 1 - Coding

Q1

Test Case

Input

```
6
3
```

Output

```
2
```

Weightage - 50

Input

```
9
5
```

Output

```
1
```

Weightage - 50

Sample Input

```
6
3
```

Sample Output

```
2
```

Sample Input

```
1
0
```

Sample Output

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

Solution

```
import java.util.*;
import java.lang.*;
import java.io.*;

class Q01Simple_List
{
    public static void main (String[] args) throws java.lang.Exception
    {
        Scanner input = new Scanner(System. in);
        // Enter dividend
        try{
                int dividend = input. nextInt();
                // Enter divisor
                int divisor = input. nextInt();

                try {
                    System.out.println(dividend/ divisor);
                } catch (ArithmeticException e) {
                    System.out.println(e);
                }
        }catch(Exception e){
            System.out.println(e);
        }

    }
}
```

Q2

Test Case

Input

```
Electronics
1000
100
```

Output

```
Electronics 1000.0 100.0
```

Weightage - 10

Input

```
Electronics
one thousand
```

Output

```
java.lang.NumberFormatException: For input string: "one thousand"
```

Weightage - 10

Input

```
Academics
5000
50
```

Output

```
Academics 5000.0 50.0
```

Weightage - 10

Input

```
Entertainment
8000
800
```

Output

```
Entertainment 8000.0 800.0
```

Weightage - 15

Input

```
Entertainment
eight thousand
```

Output

```
java.lang.NumberFormatException: For input string: "eight thousand"
```

Weightage - 15

Input

```
Commercial
4000
40
```

Output

```
Commercial 4000.0 40.0
```

Weightage - 20

Input

```
Commercial
four thousand
```

Output

```
java.lang.NumberFormatException: For input string: "four thousand"
```

Weightage - 20

Sample Input

```
Electronics
1000
100
```

Sample Output

```
Electronics 1000.0 100.0
```

Sample Input

```
Electronics
one thousand
```

Sample Output

```
java.lang.NumberFormatException: For input string: "one thousand"
```

Solution

```
import java.io.*;
import java.util.*;
class ItemType {
    private String name;
    private double deposit;
    private double costPerDay;
    public ItemType() {
        this.name = null;
        this.deposit = 0;
        this.costPerDay =0;
    }
    public ItemType(String name, double deposit,double costPerDay) {
        this.name = name;
        this.deposit = deposit;
        this.costPerDay = costPerDay;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getDeposit() {
        return deposit;
    }
    public void setDeposit(double deposit) {
        this.deposit = deposit;
    }
    public double getCostPerDay() {
```

```
        return costPerDay;
    }
    public void setCostPerDay(double costPerDay) {
        this.costPerDay = costPerDay;
    }
    public String toString() {
        return name+" "+deposit+" "+costPerDay;
    }
}
class Main {
    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        ItemType i = new ItemType();
        try {
            i.setName(sc.nextLine());
            i.setDeposit(Double.parseDouble(sc.nextLine()));
            i.setCostPerDay(Double.parseDouble(sc.nextLine()));
            System.out.println(i);
        }
        catch(NumberFormatException n) {
            System.out.println(n);
        }
    }
}
```

Q3

Test Case

Input

```
21XYZ0001
8899776655
```

Output

```
Valid
```

Weightage - 10

Input

```
21XYZ0001
7699776655
```

Output

```
Valid
```

Weightage - 10

Input

```
34BHY2001
956
```

Output

```
Invalid
java.lang.IllegalArgumentException: Mobile Number does not contain exac
```

Weightage - 10

Input

```
34BHY2001
9560251251251212
```

Output

```
Invalid
java.lang.IllegalArgumentException: Mobile Number does not contain exac
```

Weightage - 10

Input

```
12SDF2
9865321025
```

Output

```
Invalid
java.lang.IllegalArgumentException: Register Number does not contain ex
```

Weightage - 10

Input

```
12SDF21
9894359269
```

Output

```
Invalid
java.lang.IllegalArgumentException: Register Number does not contain ex
```

Weightage - 10

Input

```
12AB11234
9932587410
```

Output

```
Invalid
java.util.NoSuchElementException: Registration Number cannot contain any
```

Weightage - 10

Input

```
12ABCC234
9932587410
```

Output

```
Invalid
java.util.NoSuchElementException: Registration Number cannot contain any
```

Weightage - 10

Input

```
34CIT2345
9876543W10
```

Output

```
Invalid
java.lang.NumberFormatException: Mobile Number cannot contain any chara
```

Weightage - 10

Input

```
98VIT2345
965896321#
```

Output

```
Invalid
java.lang.NumberFormatException: Mobile Number cannot contain any chara
```

Weightage - 10

Sample Input

```
19ABC1001
9949596920
```

Sample Output

```
Valid
```

Sample Input

```
19ABC1001
99495969209
```

Sample Output

```
Invalid
java.lang.IllegalArgumentException: Mobile Number does not contain exac
```

Sample Input

```
19ABC10019
9949596920
```

Sample Output

```
Invalid
java.lang.IllegalArgumentException: Register Number does not contain ex
```

Sample Input

```
195AC1001
9949596920
```

Sample Output

```
Invalid
java.util.NoSuchElementException: Registration Number cannot contain any
```

Sample Input

```
19ABC1001
994C596920
```

Sample Output

```
Invalid
java.lang.NumberFormatException: Mobile Number cannot contain any chara
```

Solution

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

class Main{

    static void validate(String r, String n){
        if(r.length() != 9){
            System.out.println("Invalid");
            throw new IllegalArgumentException("Register Number does not contain exactly 9 characters");
        }
        if(n.length() != 10){
            System.out.println("Invalid");
            throw new IllegalArgumentException("Mobile Number does not contain exactly 10 characters");
        }

        // String pattern = "^[6|7|8|9]{1}\\d{9}";
        String pattern = "^[1-9]([0-9]){9,9}$";
        Pattern a = Pattern.compile(pattern);
        Matcher m1 = a.matcher(n);
        if(!m1.find()){
            System.out.println("Invalid");
            throw new NumberFormatException("Mobile Number cannot contain any character other than a digit");
        }

        String pattern2 = "^[1-9][0-9]([A-Z]){3,3}([0-9]){4,4}$";
```

```
        Pattern b = Pattern.compile(pattern2);
        Matcher m2 = b.matcher(r);
        if(!m2.find()){
            System.out.println("Invalid");
            throw new NoSuchElementException("Registration Number cannot contain any character other than digits and alphabets in format specified");
        }

    }

    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        String reg = sc.nextLine();
        String no = sc.nextLine();
        sc.close();
        try {
         validate(reg, no);
         System.out.println("Valid");
        }catch(Exception e)
        {
            System.out.println(e);
        }

    }
}
```

24

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();
}

}
```

Q5

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();
}

}
```

Q6

Test Case

Input

```
5
1 2 3 4 5
6
4 0
```

Output

```
Array index out of bound.
java.lang.ArithmeticException: / by zero
null
java.lang.NullPointerException
```

Weightage - 25

Input

```
4
10 89 76 12
3
4 2
```

Output

```
12
2
null
java.lang.NullPointerException
```

Weightage - 25

Input

```
6
9 8 7 6 5 6
10
10 0
```

Output

```
Array index out of bound.
java.lang.ArithmeticException: / by zero
null
java.lang.NullPointerException
```

Weightage - 25

Input

```
10
45 65 67 87 89 10 12 34 77 100
15
12 4
```

Output

```
Array index out of bound.
3
null
java.lang.NullPointerException
```

Weightage - 25

Sample Input

Sample Output

```
5
1 2 3 4 5
6
4 0
```

```
Array index out of bound.
java.lang.ArithmeticException: / by zero
null
java.lang.NullPointerException
```

Sample Input

Sample Output

```
4
10 89 76 12
3
4 2
```

```
12
2
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{
        int size = sc.nextInt();
        int[] intArray = new int[size];
        for (int i = 0; i < size; i++) {
            intArray[i] = sc.nextInt();
        }
        int index = sc.nextInt();
        System.out.println(intArray[index]);

    }
    catch(ArrayIndexOutOfBoundsException e){
        System.out.println("Array index out of bound.");
    }

    try
        {
        int a=sc.nextInt();
        int b=sc.nextInt();
            int  c= a/b;
            System.out.println(c);
        }
    catch(ArithmeticException e){
        System.out.println(e);
    }

    try {
            String str = null;
            int index = sc.nextInt();
            System.out.println(str);
            System.out.println(str.charAt(index));
        }
    catch(NullPointerException n){
        System.out.println(n);
    }
}
}
```

Q7

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

Output

65 5	13

Weightage - 20

Input **Output**

80 2	40

Weightage - 20

Sample Input **Sample Output**

44 2	22

Sample Input **Sample Output**

2 0	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);
        }
    }
}
```

Q8

Test Case

Input **Output**

8	null java.lang.NullPointerException

Weightage - 20

Input **Output**

10	null java.lang.NullPointerException

Weightage - 20

Input **Output**

12	null java.lang.NullPointerException

Weightage - 20

Input **Output**

7	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);
        }
    }
}
```

Q9

Test Case

Input

```
1002
Sharma
8000
```

Output

```
1002 Sharma 8000.0
```

Weightage - 20

Input

```
1003
Sanmar
700
```

Output

```
Balance is less than 1000
InvalidBalanceException
```

Weightage - 20

Input

```
1005
Messi
999
```

Output

```
Balance is less than 1000
InvalidBalanceException
```

Weightage - 20

Input

```
1007
Dav
6678
```

Output

```
1007 Dav 6678.0
```

Weightage - 20

Input

```
1009
Nirmal
100
```

Output

```
Balance is less than 1000
InvalidBalanceException
```

Weightage - 20

Sample Input

```
10001
Ankit
5000
```

Sample Output

```
10001 Ankit 5000.0
```

Sample Input

```
10001
Ankit
500
```

Sample Output

```
Balance is less than 1000
InvalidBalanceException
```

Solution

```
import java.io.*;
import java.util.*;
class Bank {
    private int accno;
    private String name;
    private double bal;
 public Bank() {
    this.accno = 0;
    this.name = null;
    this.bal = (double)0;
    }

 public Bank(int accno, String name,double bal) {
        this.accno = accno;
        this.name = name;
        this.bal = bal;
        }

public int getAccno() {
    return accno;
}

public void setAccno(int accno) {
    this.accno = accno;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public double getBal() {
    return bal;
}

public void setBal(double bal) {
    this.bal = bal;
}
public String toString() {
    return accno+" "+name+" "+bal;
}
}
class BankBO {
 public void validate(Bank b) throws InvalidBalanceException {
     if(b.getBal() < 1000) {
         throw new InvalidBalanceException("Balance is less than 1000");
     }
 }
}
class InvalidBalanceException extends Exception {
public InvalidBalanceException(String s) {
    System.out.println(s);
}
}
class Main {
public static void main(String [] args) {
    Scanner sc = new Scanner(System.in);
    Bank b = new Bank();
    b.setAccno(Integer.parseInt(sc.nextLine()));
    b.setName(sc.nextLine());
    b.setBal(Double.parseDouble(sc.nextLine()));
    BankBO bbo = new BankBO();
    try {
        bbo.validate(b);
        System.out.println(b);
    }
    catch(Exception e) {
        System.out.println(e);
    }
}
}
```

Q10

Test Case

Input

```
Kumar
1234567890
Kumar
Kuman123%
```

Output

```
Valid Password
```

Weightage - 10

Input

```
Kumar
1234567890
Kumar
Kuman123
```

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#nhoj
```

Sample Output

```
Valid Password
```

Sample Input

```
John
9874563210
john
john#nhoj
```

Sample Output

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

Sample Input

```
John
9874563210
john
john1nhoj
```

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);
        }



    }
}
```