EXCEPTIONAL HANDLING

Array Manipulation - Use try with multi catch

Tom wants to store the price details of the products that he purchased from the departmental store. Help him do this by using the concept of Arrays.

To do this create a public class ArrayException with a method getPriceDetails as :

public String getPriceDetails() - This method should do the following

Get the size of an array as input and then get the elements of the array(all elements are int) as input.

Next, user should provide the index of the array. This method should return the element at that index as “The array element is “+<that value>

This program may generate ArrayIndexOutOfBoundsException / InputMismatchException

In case of ArrayIndexOutOfBoundsException, the function should return “Array index is out of range”.

When providing the input, if the input is not an integer, it will generate InputMismatchException. In this case the function should return “Input was not in the correct format”.

Use exception handling mechanism to handle the exception. Use separate catch block for handling each exception. In the catch block, return the appropriate message.

Write a main method and test the above function.

Sample Input 1:

Enter the number of elements in the array
5
Enter the price details
50

80

60

70

40
Enter the index of the array element you want to access
1

Sample Output 1:

The array element is 80

Sample Input 2:

Enter the number of elements in the array
2
Enter the price details
50
80
Enter the index of the array element you want to access
9

Sample Output 2:

Array index is out of range

Sample Input 3:

Enter the number of elements in the array
2
Enter the price details
30

j

Sample Output 3:

Input was not in the correct format

ArrayException.java

```
1 import java.util.*;
2 public class ArrayException
3 {
4    public String getPriceDetails()
5    { String res="";
6       try {
7          Scanner sc=new Scanner(System.in);
8       System.out.println("Enter the number of elements in the array ");
9       int n=sc.nextInt();
10       int arr[]=new int[n];
11       System.out.println("Enter the price details");
12       int inp;
13       for(int i=0;i<n;i++)
14       {
15          inp=sc.nextInt();
16          arr[i]=inp;
17       }
18       System.out.println("Enter the index of the array element you want to access");
19       int ele=sc.nextInt();
20       int e=arr[ele];
21       res="The array element is " +e;
22
23    }
24    catch(ArrayIndexOutOfBoundsException a)
25    {
```

```
26      res="Array index is out of range";
27    }
28    catch(InputMismatchException i)
29    {
30      res="Input was not in the correct format";
31    }
32    return res;
33    }
34          public static void main (String[] args) {
35      ArrayException ae=new ArrayException();
36      System.out.println(ae.getPriceDetails());
37    }
38 }
```

Divide two numbers - Use finally

Andrew wants to teach division of two numbers to his son. To make his kid understand it better, he wants to write a program that divides two numbers and displays the answer.

Help him do this by writing a java program.

Write a public **class Division**. Write a method divideTwoNumbers as

public String divideTwoNumbers(int number1,int number2) – This method should perform division as number1 / number2.

If number2 is zero, it will throw ArithmeticException.

Whether division is done successfully or not, it should concatenate the String as "Thanks for using the application.".

When division done successfully, it should return a message as

"The answer is <number1/number2>. Thanks for using the application."

If it results in ArithmeticException, it should return a message as

"Division by zero is not possible. Thanks for using the application."

Use try, catch and finally to perform the above task.

Write the main method and test the above function.

Sample Input 1:

Enter the numbers

15

3

Sample Output 1:

The answer is 5. Thanks for using the application.

Sample Input 2:

Enter the numbers

15

0

Sample Output 2:

Division by zero is not possible. Thanks for using the application.

Sample Input 3:

Enter the numbers

15

2

Sample Output 1:

The answer is 7. Thanks for using the application.

Division.java

```
1 import java.util.*;
2 public class Division
3 {
4    public String divideTwoNumbers(int number1,int number2)
5    {String res="";
6    try {
7        int result=number1/number2;
8        res="The answer is "+result+".";
9
10    } catch(ArithmeticException e) {
11        res="Division by zero is not possible.";
12    } finally {
13        res=res+"Thanks for using the application.";
14    }
15    return res;
16 }
17 public static void main (String[] args) {
18    Scanner sc=new Scanner(System.in);
19    System.out.println("Enter the numbers");
20    int n1=sc.nextInt();
21    int n2=sc.nextInt();
22    Division d=new Division();
23    System.out.println(d.divideTwoNumbers(n1,n2));
24 }
25 }
```

Register a Candidate - User defined Exception(with throw and throws)

Geneva Technologies is planning to conduct a Walk-in interview. The interview has 4 levels. To attend the interview, the candidates need to register the following information:
Name, Gender and Expected salary.

Help him do this by writing a java program.

Partial code is provided.

You are provided with a public class Candidate with private attributes :

String name

String gender

double expectedSalary

Appropriate getter and setters are provided.

You are provided with a public class Main.

Write a method getCandidateDetails as –

public static Candidate getCandidateDetails() – This method should get the candidate details, create the Candidate object using those details and return that object.

If the candidate's expected salary is less than 10000

- throw a user defined exception as InvalidSalaryException with the message "Registration Failed. Salary cannot be less than 10000." and return null.
- this method should throw / propagate InvalidSalaryException.

To do this, write a class InvalidSalaryException that inherits Exception class.

Write a constructor that takes a String as argument and set this string to the message attribute of the super class, Exception.

In the Main class, write the main method and test the method getCandidateDetails.

If it returns a valid Candidate object, then display "Registration Successful".

Use a catch block to handle the exception that is returned by the method getCandidateDetails. In catch block display the message by using the getMessage() method.

Sample Input 1:

Enter the candidate Details

Name

Margrett

Gender

Female

Expected Salary

50000

Sample Output 1:

Registration Successful

Sample Input 2:

Enter the candidate Details

Name

Robin

Gender

Male

Expected Salary

5000

Sample Output 2:

Registration Failed. Salary cannot be less than 10000.

Candidate.java

```
1
2 public class Candidate {
3
4         private String name;
5         private String gender;
6         private double expectedSalary;
7         public String getName() {
8                 return name;
9         }
10        public void setName(String name) {
11                this.name = name;
12        }
13        public String getGender() {
14                return gender;
15        }
16        public void setGender(String gender) {
17                this.gender = gender;
18        }
19        public double getExpectedSalary() {
20                return expectedSalary;
21        }
22        public void setExpectedSalary(double expectedSalary) {
23                this.expectedSalary = expectedSalary;
24        }
```

```
}
```

Main.java

```
import java.util.*;
public class Main
{
    public static Candidate getCandidateDetails() throws InvalidSalaryException
    {

        Scanner sc=new Scanner(System.in);
            System.out.println("Enter the candidate Details");
            System.out.println("Name");
             String name=sc.nextLine();
             System.out.println("Gender");
             String gender=sc.nextLine();
             System.out.println("Expected Salary");
             double sal=sc.nextDouble();
             Candidate c=new Candidate();
        c.setName(name);
        c.setGender(gender);
        c.setExpectedSalary(sal);
        if(sal<10000)
        {
            throw new InvalidSalaryException("Registration Failed.Salary cannot be less than 10000.");

        }
        else
        return c;

    }
    public static void main (String[] args)  {




            try
            {
                Candidate c1=getCandidateDetails();

            if(c1!=null)
            System.out.println("Registration Successful");
            }
            catch(InvalidSalaryException e)
            {
            System.out.println(e.getMessage());


            }
}
}
```

InvalidSalaryException.java

```
public class InvalidSalaryException extends Exception
{
    public InvalidSalaryException(String msg)
    {
        super(msg);
}
}
```