Laboratory Work

Subject: Java Technologies
Branch: B.Tech. (CE)
Semester: IV
Batch: A1

Student Roll No: CE001
Student Name: NISARG KALPESHBHAI AMLANI

Department of Computer Engineering,
Faculty of Technology,
Dharmsinh Desai University, Nadiad – 387001.
Gujarat, INDIA.

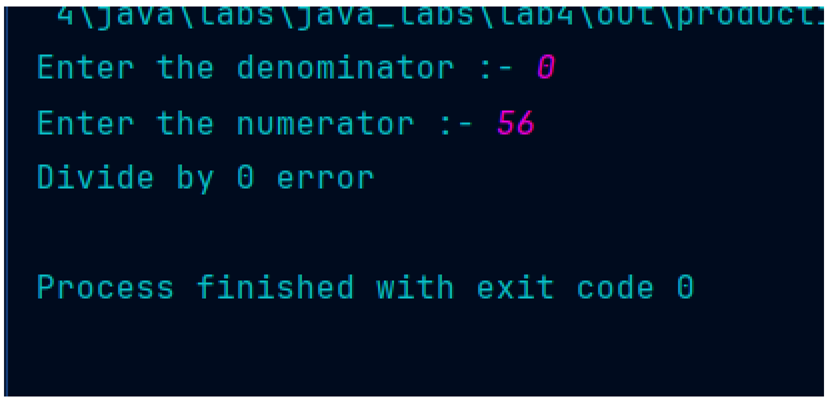
Question 1)

Write a program that catches the divide-by-zero exception using the try-catch mechanism. Take a numeric value and perform division by zero. Catch the ArithmeticException.

Solution

```
import java.util.Scanner;

public class lab4_pg1 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the denominator :- ");
        int denominator = sc.nextInt();
        System.out.print("Enter the numerator :- ");
        int numerator = sc.nextInt();
        try{
            int ans = numerator/denominator;
        }
        catch (ArithmeticException e) {
            System.out.println("Divide by 0 error");
        }
    }
}
```

Screenshot



```
 4\java\labs\java_labs\lab4\out\product
Enter the denominator :- 0
Enter the numerator :- 56
Divide by 0 error

Process finished with exit code 0
```

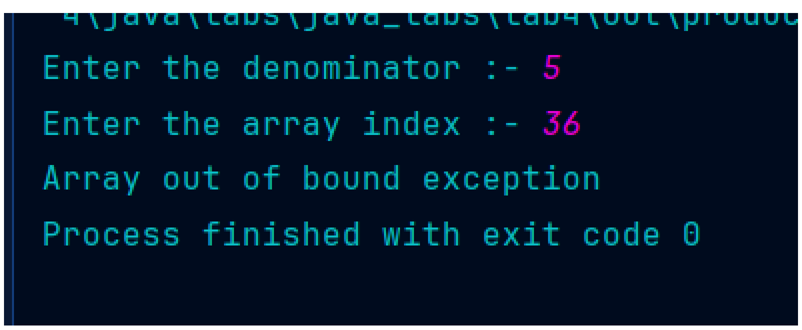
Question 2)

Write a java program using multiple catch blocks. Create a class CatchExercise inside the try block declare an array a[] and initialize with value a[5] =30/5; . In each catch block show Arithmetic exception and ArrayIndexOutOfBoundsException.

Solution

```
import java.util.Scanner;


public class CatchExercise {
    public static void main(String[] args) {
        Scanner obj = new Scanner(System.in);
        System.out.print("Enter the denominator :- ");
        int deno = obj.nextInt();
        System.out.print("Enter the array index :- ");
        int index = obj.nextInt();
        int[] arr  = new int[5];
        try{
            arr[index] = 30/5;
            int ans = 5/deno;
        }
        catch(ArithmeticException a)
        {
            System.out.print("Divide by 0 exception");
        }
        catch(ArrayIndexOutOfBoundsException aob)
        {
            System.out.print("Array out of bound exception ");
        }

    }
}
```

Screenshots



```
Enter the denominator :- 5
Enter the array index :- 36
Array out of bound exception
Process finished with exit code 0
```

Question 3)

Write a program that demonstrates use of finally block. Observe the output of your program for different cases as mentioned below.

● Case A: **exception does not occur. Perform 25/5 mathematical operation. Catch the NullPointerException.**

● Case B: exception occurs but not handled. Perform 25/0 mathematical operation. Catch NullPointerException.

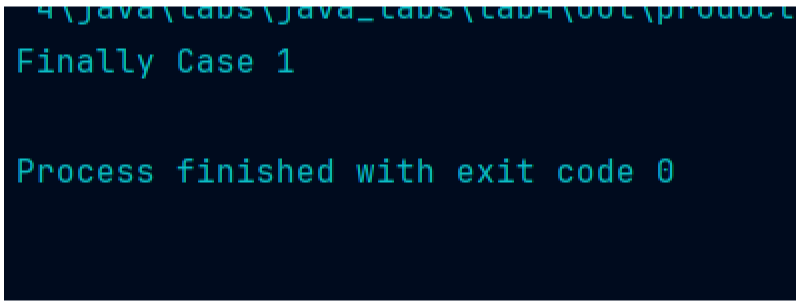
● Case C: exception occurs and handled. Perform 25/0 mathematical operation. Catch ArithmeticException

Solution

Case 1 :-

```
public class FinallyExercise_Case_1 {
    public static void main(String[] args) {
        try{
            int ans = 25/5;
        }
        catch(NullPointerException npe){
            System.out.println("Nullpointer Exception");

        }
        finally {
            System.out.println("Finally Case 1");
        }
    }

}
```

Screenshot case 1 :-

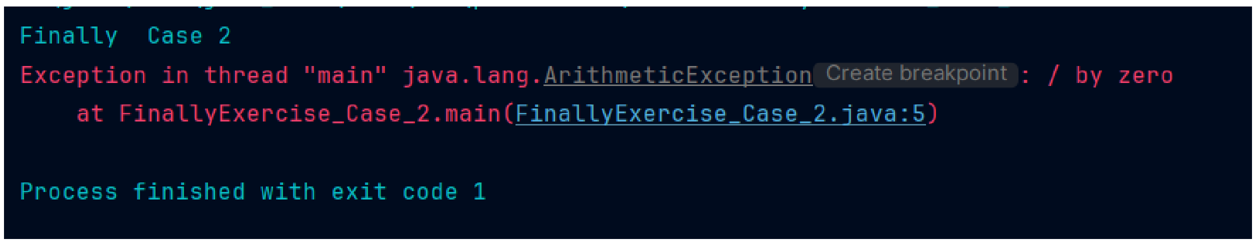


```
Finally Case 1

Process finished with exit code 0
```

Case 2 :-

```
public class FinallyExercise_Case_2 {
    public static void main(String[] args) {

        try{
            int ans = 25/0;
        }
        catch (NullPointerException nep) {
            System.out.println("NullPointer Exception");
        }
        finally {
            System.out.println("Finally  Case 2");
        }
    }
}
```

ScreenShot Case 2 :-

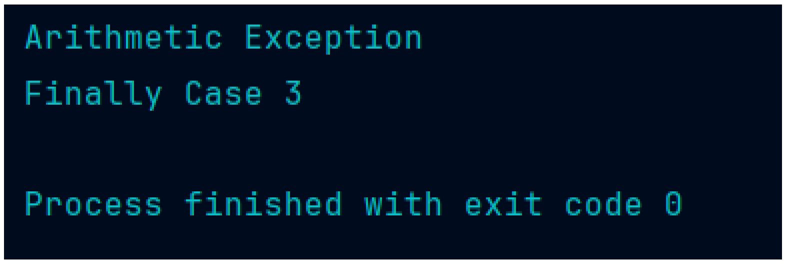


```
Finally  Case 2
Exception in thread "main" java.lang.ArithmeticException Create breakpoint : / by zero
    at FinallyExercise_Case_2.main(FinallyExercise_Case_2.java:5)

Process finished with exit code 1
```

Case 3 :-

```
public class FinallyExercise_Case_3 {
    public static void main(String[] args) {
        try{
            int ans = 25/0;

        }
        catch (ArithmeticException ae)
        {
            System.out.println("Arithmetic Exception");
        }
        finally {
            System.out.println("Finally Case 3");
        }
    }
}
```

ScreenShot Case 3:-



```
Arithmetic Exception
Finally Case 3

Process finished with exit code 0
```

Question 4)

Create an interface Account with two methods: deposit and withdraw. Create class SavingsAccount which implements the interface. Write a custom Exception handler for SavingsAccount to handle the scenarios when the withdrawn amount is larger than the balance in the account.

Solution

Interface Account ➖

```
public interface Account {
    void deposite (int amount) ;

    void withdraw (int amount) throws Custom_Exception;
}
```

Class SavingsAccount :-

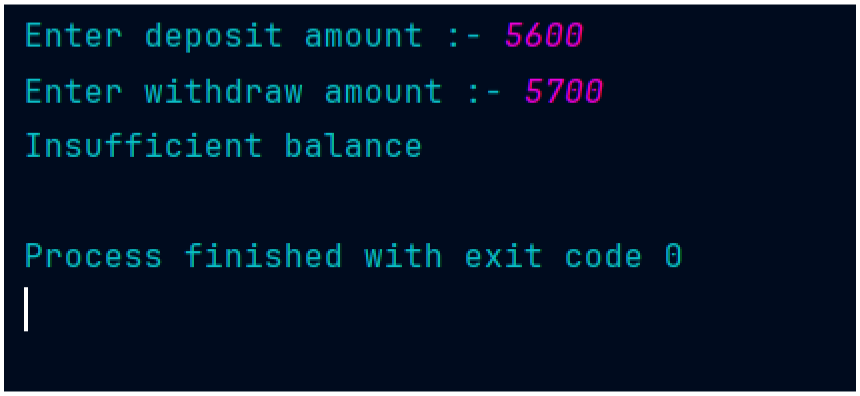
```
public class SavingsAccount implements Account{
    private int balance = 50000;

    @Override
    public void deposite(int amount) {
        this.balance = amount;
    }

    @Override
    public void withdraw(int amount) throws Custom_Exception{

        if(amount> this.balance)
        {
            throw new Custom_Exception("Insufficient balance");
        }
    }
}
```

Solution :-

```
import java.util.Scanner;
```

```
public class Lab4_pg4 {
    public static void main(String[] args) {
        Scanner obj = new Scanner(System.in);
        System.out.print("Enter deposit amount :- ");
        int amount = obj.nextInt();
        System.out.print("Enter withdraw amount :- ");
        int wamount = obj.nextInt();

        SavingsAccount a1 = new SavingsAccount();

        try{
            a1.deposite(amount);
            a1.withdraw(wamount);
        }
        catch (Custom_Exception c)
        {
            System.out.println(c.getMessage());
        }
    }
}
```

ScreenShots :-



```
Enter deposit amount :- 5600
Enter withdraw amount :- 5700
Insufficient balance


Process finished with exit code 0
```