

Assignments on Exception Handling

- 1) Write an application that accepts two numbers, divides the first number with the second number and display the result. Hint: You need to handle **ArithmeticException** which is thrown when there is an attempt to divide a number by zero.

```
1 package Assignment4.java;
2 import java.util.Scanner;
3 public class Arithmetic {
4     public static void main(String[] args) {
5         try {
6             Scanner sc=new Scanner(System.in);
7             System.out.println("Enter num1");
8             int num1=sc.nextInt();
9             System.out.println("Enter num2");
10            int num2=sc.nextInt();
11            int res=num1/num2;
12            System.out.println(res);
13        }
14        catch (ArithmeticException ae)
15        {
16            System.out.println("division can't be done.");
17        }
18    }
19 }
20
21
```

<terminated> Arithmetic [Java Application] C:\Users\valaanus\p2
Enter num1
20
Enter num2
0
division can't be done.

- 2) Carrying forward with the above problem, handle **ArithmeticException** by raising **UnsupportedOperationException** as a solution.

```
1 package Assignment4.java;
2 import java.util.Scanner;
3 public class Division {
4
5     public static void main(String[] args) {
6         System.out.println("enter a,b");
7         Scanner obj=new Scanner(System.in);
8         int a=obj.nextInt();
9         int b=obj.nextInt();
10        int c;
11        try {
12            c=a/b;
13            System.out.println(c);
14        }
15        catch (ArithmeticException ex) {
16            System.out.println("Division by zero");
17            throw new UnsupportedOperationException();
18        }
19    }
20 }
21
```

<terminated> Division [Java Application] C:\Users\valaanus\p2
enter a,b
7 0
Division by zero
Exception in thread "main" java.lang.UnsupportedOperationException
at Assignment4.java.Division.ma:

3) Write an application to perform withdraw functionality on a SavingAccount object. Point to note:

- a. Raise **InsufficientBalanceException** if you are trying to withdraw more than balance or when you balance is zero. E.g. if you balance is 2000 and if you are trying to withdraw 2100 or if you balance is 0 and you are trying to withdraw positive value.
- b. Raise **IllegalBankTransactionException** if you are trying to withdraw a negative value from your balance. E.g. if you try to withdraw a negative value `savingAcc.withdraw(-1000);`

Note: SavingAccount

```
-- long id
-- double balance
--double withdraw(double amount)
--double deposit(double amount)
```

The screenshot shows an IDE with two panes. The left pane displays the source code for `Assignment4.java`, which defines a `SavingAccount` class and an `InsufficientBalanceException` class. The right pane shows a stack trace for a terminated exception.

```
1 package Assignment4.java;
2
3
4
5 class SavingAccount {
6     int balance=0;
7     public SavingAccount(int balance) {
8         this.balance=balance;
9     }
10    public void setBal(int balance) {
11        this.balance=balance;
12    }
13    public int getBal() {
14        return balance;
15    }
16    public void withdraw(int amount) {
17        if(amount>balance || amount==0) {
18            throw new InsufficientBalanceException(amount);
19        }
20        else if(amount<0) {
21            throw new IllegalBankTransactionException();
22        }
23        else {
24            balance=balance-amount;
25            System.out.println("Remaining balance:"+balance);
26        }
27    }
28 }
29 class InsufficientBalanceException extends RuntimeException{
30     public InsufficientBalanceException() {
31         super();
32     }
33     public InsufficientBalanceException(int amount) {
34         System.out.println("Insufficient Balance:"+amount);
35     }
36 }
```

Stack Trace:

```
<terminated> Exception3 [Java Application] C:\Users\valaanus\
2000
Illegal Bank Transaction
Exception in thread "main" Assignment4.
at Assignment4.java.SavingAccou
at Assignment4.java.Exception3.i
```

```

37 class IllegalBankTransactionException extends RuntimeException{
38     public IllegalBankTransactionException() {
39         System.out.println("Illegal Bank Transaction");
40     }
41 }
42 public class Exception3 {
43     public static void main(String[] args) {
44         SavingAccount s= new SavingAccount(2000);
45         System.out.println(s.getBal());
46         s.withdraw(-7);
47         s.withdraw(2100);
48     }
49 }

```

3(b)

The screenshot shows an IDE with two panels. The left panel displays the source code for 'Assignment4.java', and the right panel shows the Java Exception Viewer.

Source Code (Assignment4.java):

```

1 package Assignment4.java;
2 class BankAccount {
3     private double balance;
4     public BankAccount() {
5         // TODO Auto-generated constructor stub
6         balance=0;
7     }
8     public BankAccount(double initialBalance)
9     {
10         balance=initialBalance;
11     }
12     public void deposit(double amount)
13     {
14         balance=balance+amount;
15     }
16     public void withdraw(double amount)
17     {
18         if ((amount>balance) || balance==0)
19         {
20             throw new IllegalArgumentException("InsufficientBalance");
21         }
22         balance=balance-amount;
23     }
24     public double getBalance()
25     {
26         return balance;
27     }
28 }
29 public class ThrowingException {
30     public static void main(String args[]) {
31         BankAccount b=new BankAccount();
32         b.deposit(2000);
33         b.withdraw(2100);
34         System.out.println(b.getBalance());
35     }
36 }

```

Exception Viewer:

```

<terminated> ThrowingException [Java Application] C:\Users\ve...
Exception in thread "main" java.lang.IllegalArgumentException: InsufficientBalance
    at Assignment4.java.BankAccount.withdraw(BankAccount.java:22)
    at Assignment4.java.ThrowingException.main(ThrowingException.java:33)

```