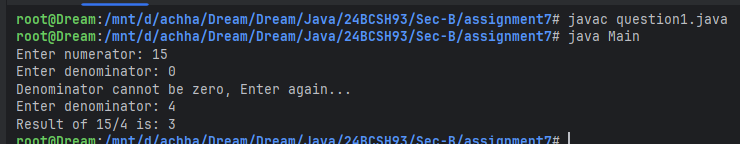
**Assignment – 7**

**Q1. Write an exception handling java program to read two numbers num1, num2 and calculate and print the result of num1/num2. If num2 is Zero (0) then it will be handled by exception handler and again ask the value of num2. In the exception handler the program should display appropriate message to the user.**

import java.util.Scanner;  
class Main {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter numerator: ");  
 int numerator = sc.nextInt();  
  
 while (true) {  
 try {  
 System.*out*.print("Enter denominator: ");  
 int denominator = sc.nextInt();  
 System.*out*.println("Result of " + numerator + "/" + denominator + " is: " + numerator / denominator);  
 return;  
 } catch (ArithmeticException e) {  
 System.*out*.println("Denominator cannot be zero, Enter again...");  
 }  
 }  
 }  
}

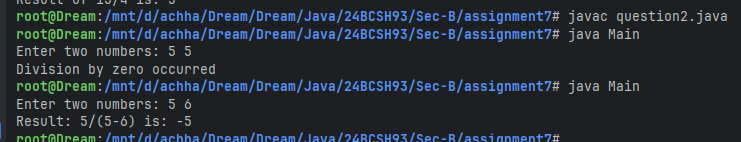
**Output:**

****

**Q2. Write a java program to read two numbers x and y and calculate x/(x-y). The program should check the value of x-y. Before dividing with x, it should throw an exception if x-y is zero. In the exception handler the program should display appropriate message to the user.**

import java.util.Scanner;  
class Main {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter two numbers: ");  
 int x = sc.nextInt();  
 int y = sc.nextInt();  
  
 try {  
 if (x - y == 0) {  
 throw new ArithmeticException("Division by zero occurred");  
 } else {  
 System.*out*.printf("Result: %d/(%d-%d) is: %d\n", x, x, y, x / (x - y));  
 }  
 } catch (ArithmeticException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
}

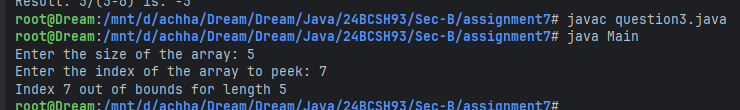
**Output:**



**Q3. Write an exception handling java program to print the index position of an existing integer array. The index value will be entered by user. It will be handled by exception handler if index position is greater than the size of array. In the exception handler the program should display appropriate message to the user.**

import java.util.Scanner;  
class Main {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter the size of the array: ");  
 int[] array = new int[sc.nextInt()];  
 System.*out*.print("Enter the index of the array to peek: ");  
 int idx = sc.nextInt();  
  
 try {  
 System.*out*.println("The value at " + idx + " in the array is: " + array[idx]);  
 } catch (ArrayIndexOutOfBoundsException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
}

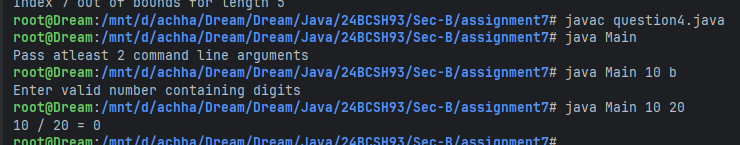
**Output:**

****

**Q4. Write a program to illustrate the use of multiple catch blocks associated with a single try block.**

class Main {  
 public static void main(String[] args) {  
 try {  
 int numerator = Integer.*parseInt*(args[0]);  
 int denominator = Integer.*parseInt*(args[1]);  
 int quotient = numerator / denominator;  
 System.*out*.println(numerator + " / " + denominator + " = " + quotient);  
 } catch (ArrayIndexOutOfBoundsException e) {  
 System.*out*.println("Pass atleast 2 command line arguments");  
 } catch (NumberFormatException e) {  
 System.*out*.println("Enter valid number containing digits");  
 } catch (ArithmeticException e) {  
 System.*out*.println("Division by zero occurred");  
 }  
 }  
}

**Output:**

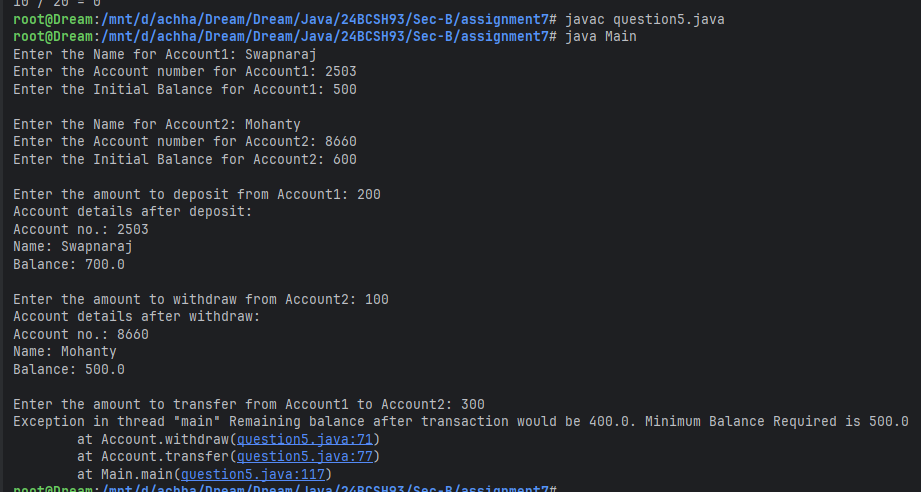
****

**Q5. Write a class called Account with the following properties and methods:  
 Properties: String name, int accountNo, double balance  
 Methods: void deposit(double amount)  
 void withdraw(double amount)  
 void transfer(Account account1, Account account1, double amount)  
Assume that an account needs to have a minimum balance of 500. If an attempt is made to withdraw or transfer, which results in balance going below 500, throw a user defined exception called MinimumBalanceException. Use throw and throws wherever necessary.**

import java.util.Scanner;  
class NonPositiveValueException extends Exception {  
 String name;  
 double value;  
  
 public NonPositiveValueException(String name, double value) {  
 this.name = name;  
 this.value = value;  
 }  
  
 public String toString() {  
 return name + " must be a positive value. " + value + " was provided";  
 }  
}  
  
class MinimumBalanceException extends Exception {  
 double balance;  
  
 public MinimumBalanceException(double balance) {  
 this.balance = balance;  
 }  
  
 public String toString() {  
 return "Remaining balance after transaction would be " + balance + ". Minimum Balance Required is " + Account.*MIN\_BALANCE*;  
 }  
}  
  
class Account {  
 public static final double *MIN\_BALANCE* = 500.0;  
 private String name;  
 private int accountNo;  
 private double balance;

public Account(String name, int accountNo, double balance) throws NonPositiveValueException, MinimumBalanceException {  
 if (accountNo <= 0) {  
 throw new NonPositiveValueException("Account Number", accountNo);  
 }  
 if (balance <= 0) {  
 throw new NonPositiveValueException("Initial Balance", balance);  
 }  
 if (balance < *MIN\_BALANCE*) {  
 throw new MinimumBalanceException(balance);  
 }  
 this.name = name;  
 this.accountNo = accountNo;  
 this.balance = balance;  
 }  
  
 public void deposit(double amount) throws NonPositiveValueException {  
 if (amount <= 0) {  
 throw new NonPositiveValueException("Deposit Amount", amount);  
 }  
 balance += amount;  
 }  
  
 public void withdraw(double amount) throws NonPositiveValueException, MinimumBalanceException {  
 if (amount <= 0) {  
 throw new NonPositiveValueException("Withdrawal Amount", amount);  
 }  
 if (balance - amount < *MIN\_BALANCE*) {  
 throw new MinimumBalanceException(balance - amount);  
 }  
 balance -= amount;  
 }  
  
 public static void transfer(Account account1, Account account2, double amount) throws NonPositiveValueException, MinimumBalanceException {  
 account1.withdraw(amount);  
 account2.deposit(amount);  
 }  
  
 public String toString() {  
 return "Account no.: " + accountNo + "\nName: " + name + "\nBalance: " + balance;  
 }  
}  
  
class Main {  
 public static void main(String[] args) throws NonPositiveValueException, MinimumBalanceException {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter the Name for Account1: ");  
 String name1 = sc.next();  
 System.*out*.print("Enter the Account number for Account1: ");  
 int accountNo1 = sc.nextInt();  
 System.*out*.print("Enter the Initial Balance for Account1: ");  
 double balance1 = sc.nextDouble();  
 Account account1 = new Account(name1, accountNo1, balance1);  
 System.*out*.print("\nEnter the Name for Account2: ");  
 String name2 = sc.next();  
 System.*out*.print("Enter the Account number for Account2: ");  
 int accountNo2 = sc.nextInt();  
 System.*out*.print("Enter the Initial Balance for Account2: ");  
 double balance2 = sc.nextDouble();  
 Account account2 = new Account(name2, accountNo2, balance2);  
  
 System.*out*.print("\nEnter the amount to deposit from Account1: ");  
 double amount1 = sc.nextDouble();  
 account1.deposit(amount1);  
 System.*out*.println("Account details after deposit:\n" + account1);  
  
 System.*out*.print("\nEnter the amount to withdraw from Account2: ");  
 double amount2 = sc.nextDouble();  
 account2.withdraw(amount2);  
 System.*out*.println("Account details after withdraw:\n" + account2);  
  
 System.*out*.print("\nEnter the amount to transfer from Account1 to Account2: ");  
 double amount3 = sc.nextDouble();  
 Account.*transfer*(account1, account2, amount3);  
 System.*out*.println("Account details after transfer:\n" + account1 + "\n" + account2);  
 }  
}

**Output:**

****

**Q6. Write a program that prompts the user to enter a length in feet and inches and outputs the equivalent length in centimetres. If the user enters a negative number or a non-digit number, throw and handle an appropriate exception and prompt the user to enter another set of numbers.**

import java.util.InputMismatchException;  
import java.util.Scanner;  
class Main {  
 public static void main(String[] args) {  
 while (true) {  
 try {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter the length in feet and inches: ");  
 int feet = sc.nextInt();  
 int inches = sc.nextInt();  
  
 if (feet < 0.0 || inches < 0.0) {  
 throw new IllegalArgumentException();  
 }  
 double centimeter = (feet \* 12 + inches) \* 2.54;  
 System.*out*.println("Equivalent length in centimeters: " + centimeter);  
 return;  
 } catch (InputMismatchException e) {  
 System.*out*.println("Enter a valid number containing digits");  
 } catch (IllegalArgumentException e) {  
 System.*out*.println("Negative value(s) was/were entered. Try again...");  
 }  
 }  
 }  
}

**Output:**

