**CORE JAVA ASSIGNMENT 4 – EXCEPTION HANDLING**

**Assignment on Exception Handling**

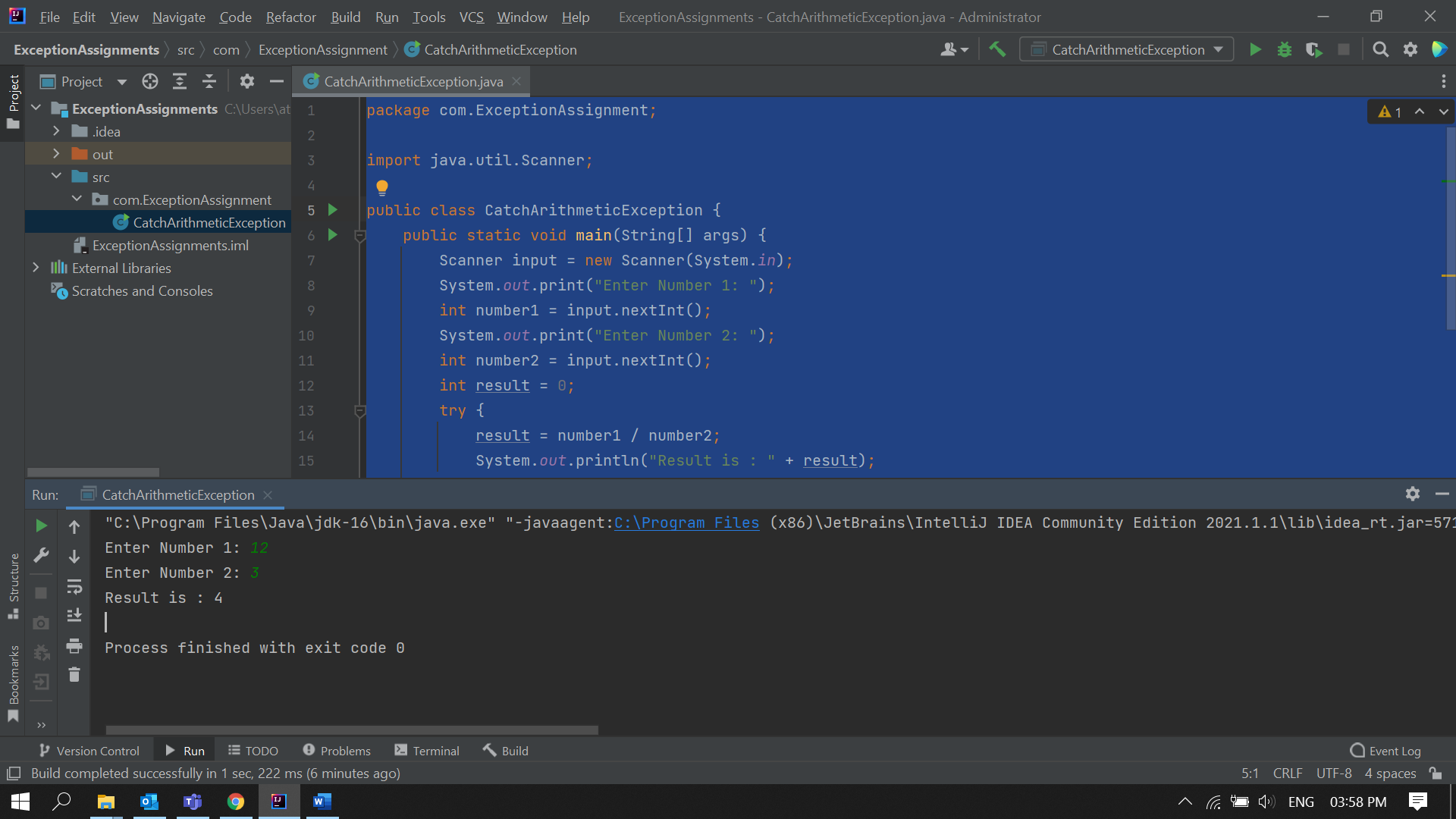
1. Write an application that accepts two numbers, divides the first number with the second number and display the result. Hint: You need to handle ArithmeticException which is thrown when there is an attempt to divide a number by zero.

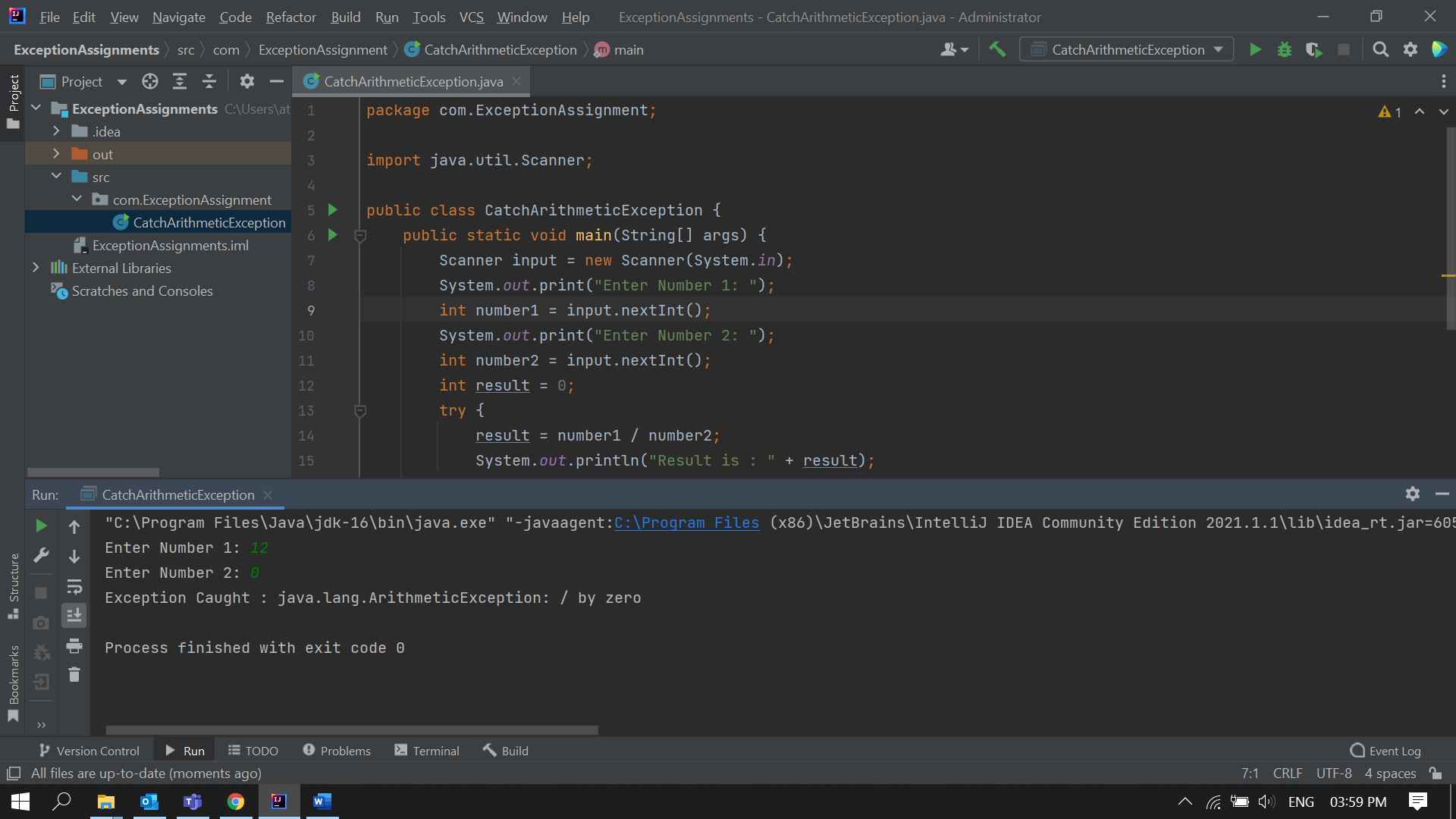
**Solution:**

**CatchArithmeticException.java**

package com.ExceptionAssignment;  
import java.util.Scanner;  
public class CatchArithmeticException {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("Enter Number 1: ");  
 int number1 = input.nextInt();  
 System.*out*.print("Enter Number 2: ");  
 int number2 = input.nextInt();  
 int result = 0;  
 try {  
 result = number1 / number2;  
 System.*out*.println("Result is : " + result);  
 } catch (ArithmeticException e) {  
 System.*out*.println("Exception Caught : " + e);  
 }  
 }  
}

**Output:**





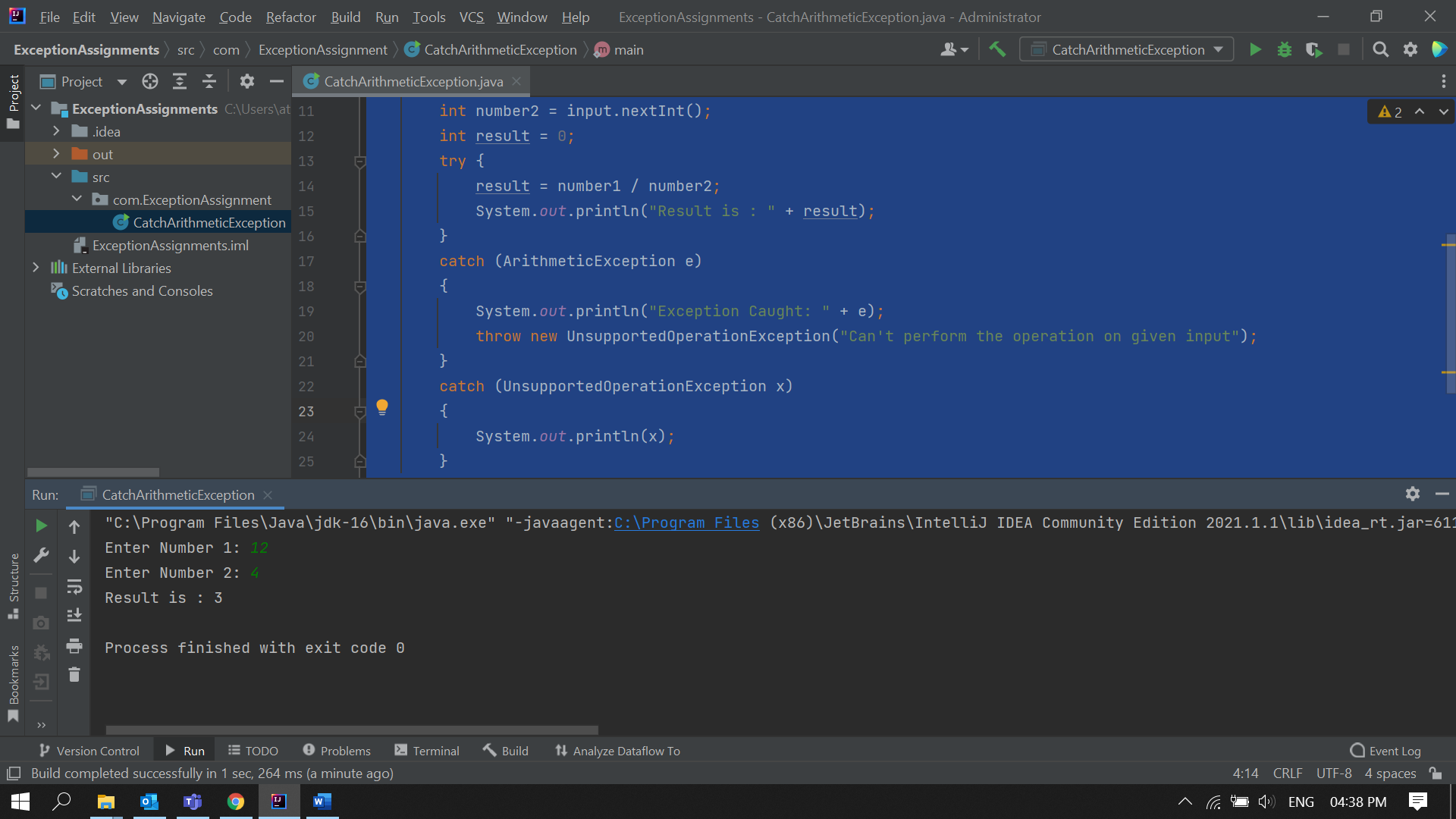
2. Carrying forward with the above problem, handle ArithmeticException by raising Unsupported OperationException as a solution.

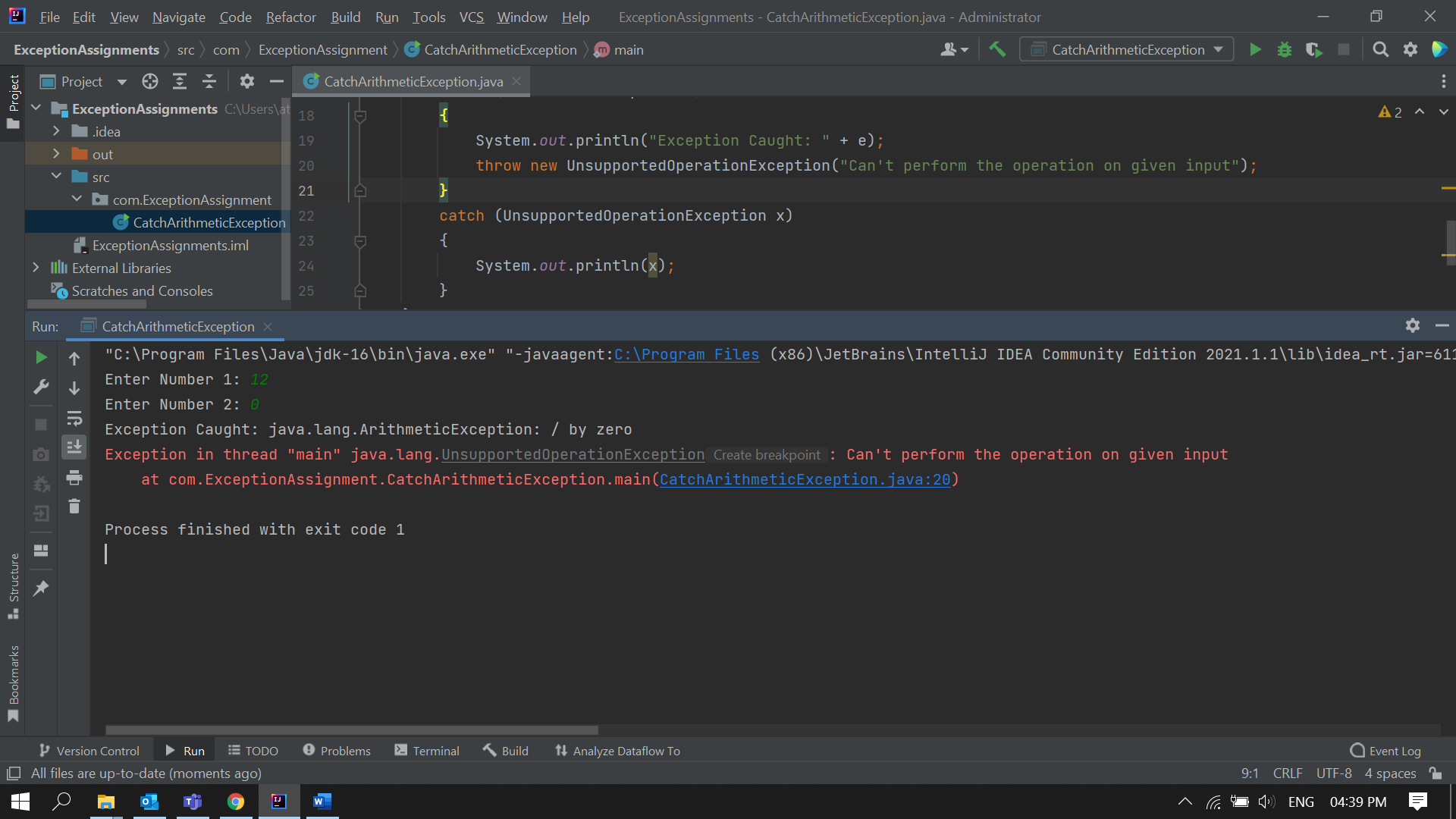
**Solution:**

**CatchArithmeticException.java**

package com.ExceptionAssignment;  
  
import java.util.Scanner;  
  
public class CatchArithmeticException {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("Enter Number 1: ");  
 int number1 = input.nextInt();  
 System.*out*.print("Enter Number 2: ");  
 int number2 = input.nextInt();  
 int result = 0;  
 try {  
 result = number1 / number2;  
 System.*out*.println("Result is : " + result);  
 }  
 catch (ArithmeticException e)  
 {  
 System.*out*.println("Exception Caught: " + e);  
 throw new UnsupportedOperationException("Can't perform the operation on given input");  
 }  
 catch (UnsupportedOperationException x)  
 {  
 System.*out*.println(x);  
 }  
 }  
}

**Output:**





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 O 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)

**Solution:**

**Main.java**

package com.ExceptionAssignment;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 SavingAccount newAccount = new SavingAccount(11223344,10000);  
 Scanner input = new Scanner(System.*in*);  
 int amount , choice;  
 do {  
 Menus.*Menu*();  
 choice = input.nextInt();  
 switch (choice) {  
 case 1:  
 Menus.*WithdrawMenu*();  
 amount = input.nextInt();  
 System.*out*.println("Balance : " + newAccount.withdraw(amount));  
 break;  
 case 2:  
 Menus.*DepositMenu*();  
 amount = input.nextInt();  
 System.*out*.println("Balance : " + newAccount.deposit(amount));  
 break;  
 case 3:  
 System.*out*.println("Current Balance: " + newAccount.balance);  
 break;  
 case 4:  
 System.*out*.println("Bye");  
 break;  
 }  
 }while(choice<4); } }

**SavingAccount.java**

package com.ExceptionAssignment;  
public class SavingAccount {  
  
 long id;  
 double balance;  
  
 SavingAccount(long id , double balance)  
 {  
 this.id = id;  
 this.balance = balance;  
 }  
  
 double withdraw(double amount){  
 try{  
 if(amount > balance)  
 throw new InsufficientBalanceException();  
 else if(amount < 0)  
 throw new illegalBankTransactionException();  
 else  
 {  
 System.*out*.println(amount + " Withdrawl Successful!");  
 this.balance -= amount;  
 }  
 }  
 catch(InsufficientBalanceException e){  
 System.*out*.println("Exception Caught: " + e);  
 }  
 catch (illegalBankTransactionException e)  
 {  
 System.*out*.println("Exception Caught: " + e);  
 }  
 return balance;  
 }  
 double deposit(double amount){  
 balance += amount;  
 System.*out*.println(amount + " has been deposited in your account");  
 return balance;  
 }  
}

**illegalBankTransactionException.java**  
package com.ExceptionAssignment;  
public class illegalBankTransactionException extends Exception{  
 illegalBankTransactionException()  
 {  
 System.*out*.print("Invalid Amount Entered! ");  
 }  
}

**InsufficientBalanceException.java**

package com.ExceptionAssignment;  
  
public class InsufficientBalanceException extends Exception{  
 InsufficientBalanceException(){  
 System.*out*.print("Insufficient Balance! ");  
 }  
}

**Menus.java**

package com.ExceptionAssignment;  
  
public class Menus {  
 static void Menu()  
 {  
 System.*out*.println();  
 System.*out*.println("Transactions Options");  
 System.*out*.println("1. Withdraw");  
 System.*out*.println("2. Deposit");  
 System.*out*.println("3. CheckBalance");  
 System.*out*.println("4. Exit");  
 System.*out*.print("Enter your Choice: ");  
 }  
 static void WithdrawMenu()  
 {  
 System.*out*.println();  
 System.*out*.print("Enter the amount to Withdraw: ");  
 }  
 static void DepositMenu(){  
 System.*out*.println();  
 System.*out*.print("Enter the amount to Deposit: ");  
 }  
}

**Output:**

