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.

**public** **class** Exceptionhandling {

**public** **static** **void** main(String[] args)

{

**int** a = 8;

**int** b = 3;

**try** {

System.***out***.println(a / b);

}

**catch** (ArithmeticException e) {

System.***out***.println(

"Divided by zero operation cannot possible");

}

}

}

1. Carrying forward with the above problem, handle ArithmeticException by raising UnsupportedOperationException as a solution

**import** java.util.Scanner;

**class** UnSupportedOperationException {

**public** **static** **void** solve() **throws** UnsupportedOperationException {

Scanner scanner = **null**;

**try** {

scanner = **new** Scanner(System.***in***);

System.***out***.println("Enter Dividend number:");

**int** dividend = scanner.nextInt();

System.***out***.println("Enter Divisor number:");

**int** divisor = scanner.nextInt();

**if** (divisor == 0) {

**throw** **new** UnsupportedOperationException("Invalid operation for Division by Zero");

}

**int** result = dividend / divisor;

System.***out***.println("Result is:" + result);

} **catch** (UnsupportedOperationException e) {

System.***out***.println(e.getMessage());

e.printStackTrace();

} **finally** {

**if** (scanner != **null**)

scanner.close();

}

}

}

**public** **class** Q2 {

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

UnSupportedOperationException.*solve*();

}

}

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);

**class** InsufficientBalanceException **extends** Exception {

**public** InsufficientBalanceException(String str)

{

**super**(str);

}

}

**class** IllegalBankTransactionException **extends** InsufficientBalanceException {

**public** IllegalBankTransactionException(String str) {

**super**(str);

}

}

**class** savingAccount{

**private** **final** **long** id ;

**private** **double** balance ;

**public** savingAccount(**long** id,**double** balance){

**this**.id=id;

**this**.balance=balance;

};

**public** **double** withdraw(**double** amount) **throws** RuntimeException {

**try** {

**if**(balance<amount){

**throw** **new** InsufficientBalanceException ("insufficient balance in the account");

}

balance -= amount ;

}

**catch** (InsufficientBalanceException e) {

System.***out***.println(e.getMessage());

e.printStackTrace();

}

**return** balance ;

}

**public** **double** deposit(**double** amount) **throws** RuntimeException {

**try** {

**if**(amount<=0){

**throw** **new** IllegalBankTransactionException("The amount should always be greater than 0");

}

balance -= amount ;

}

**catch** (IllegalBankTransactionException ez) {

System.***out***.println(ez.getMessage());

ez.printStackTrace();

}

**finally** {

balance = balance ;

}

**return** balance ;

}

**public** **double** getBalance() {

**return** balance ;

}

}

**public** **class** Q3 {

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

savingAccount Ayushman = **new** savingAccount(10,1000000);

Ayushman.deposit(-1);

Ayushman.withdraw(900);

System.***out***.println(Ayushman.getBalance());

}

}