Core Java Assignment 4

**Assignments on Exception Handling**

1. Write an application that accepts two numbers, divide 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 a zero.

**package** EceptionHandling;

**public** **class** ArithEcep {

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

**try** {

**int** a=10;

**int** b=0;

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

}

**catch**(ArithmeticException e)

{

System.***out***.println("Exception Handled "+ e);

}

}

}

**Output:**

Exception Handled java.lang.ArithmeticException: / by zero

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

**package** EceptionHandling;

**public** **class** UnsupportedException {

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

{

**try**

{

**int** a=10;

**int** b=0;

**int** c=a/b;

System.***out***.println("answer is: "+c);

}

**catch**(ArithmeticException e)

{

e.printStackTrace();

}

**throw** **new** UnsupportedOperationException("Invalid");

}

}

**Output:**

java.lang.ArithmeticException: / by zero

at EceptionHandling.UnsupportedException.main(UnsupportedException.java:13)

Exception in thread "main" java.lang.UnsupportedOperationException: Invalid

at EceptionHandling.UnsupportedException.main(UnsupportedException.java:21)

1. Perform withdraw functionality with saving account object.

i)Raise InsufficientBalanceException if you are trying to withdraw more than balance.

**package** EceptionHandling;

**import** java.util.Scanner;

**class** IBException **extends** RuntimeException{

}

public class SavingAccount {

Scanner sc = new Scanner(System.in);

public void withdrawal(double a)

{

System.out.println("Enter your Id ");

long id = sc.nextLong();

System.out.println("Enter your balance ");

double b = sc.nextDouble();

try {

if(a<=b) {

b = b - a;

System.out.println("Balance= " + b);

}

else {

throw new InsufficientBalanceException();

}

}

catch (InsufficientBalanceException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

SavingAccount s = new SavingAccount();

s.withdrawal(2000);

}

}

**Output:**

Enter your Id

2222

Enter your balance

500

Account.InsufficientBalanceException

at Account.SavingAccount.withdrawal(SavingAccount.java:23)

at Account.SavingAccount.main(SavingAccount.java:34)

ii) Raise a illigalBankTransaction if you are trying to withdraw negative amount from account.

**package** EceptionHandling;

**import** java.util.Scanner;

**class** ITException **extends** RuntimeException{

}

public class SavingsAccount {

Scanner sc = new Scanner(System.in);

public void withdrawal(double a)

{

System.out.println("Enter your Id ");

long id = sc.nextLong();

System.out.println("Enter your balance ");

double b = sc.nextDouble();

try {

if(a>0) {

System.out.println("Balance= " + b);

}

else {

throw new IllegalBankTransactionException();

}

}

catch (IllegalBankTransactionException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

SavingsAccount s = new SavingsAccount();

s.withdrawal(-10201);

}}

**Output:**

Enter your Id

22222

Enter your balance

2000

ExceptionHandling.illigalTransactionException

at ExceptionHandling.Exception1.withdrawal(Exception1.java:22)

at ExceptionHandling.Exception1.main(Exception1.java:32)