**Exception Handling Assignment**

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 where there is an attempt to divide a number by zero.

public class Arithmetic {

     public static void main(String[] args)

     {

try

{

         int a,b;

Scanner sc=new Scanner(“System.in”);

System.out.println(“Enter a:”);

a=sc.nextInt();

         System.out.println(“Enter b:”);

         b=sc.nextInt();

          }

catch(ArithmeticException e)

{

System.out.println(“Exception Handled”+e);

}

}

}

Output:

Enter a:5

Enter b:0

Exception Handled java.lang.ArithmeticException: / by zero

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

public class Unsupport

{

public static void main(String[] args)

{

Scanner s=new Scanner(System.in);

try

{

System.out.println(“Enter a: ”);

a=sc.nextInt();

         System.out.println(“Enter b:”);

         b=sc.nextInt();

int c = a/b;

System.out.println(“C = ”+c);

}

catch(ArithmeticException e)

{

System.lout.println(“Exception handled”+e);

System.out.println(“UnsupportedOperationException”);

}

}

}

Output:

Enter a:5

Enter b:0

Exception handled Java.lang.ArithmeticException: / by zero

UnsupportedOperationException

1. 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. Eg: 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 a positive value.

class InsufficientBalanceException extends RuntimeException{

}

public class SavingAccount{

Scanner s = new Scanner(System.in);

public void withdraw(double a)

{

System.out.println(“Enter your ID: ”);

long id = s.nextLong();

System.out.println(“Enter balance amount:”);

double b=s.nextDouble();

try{

if(a<=b)

{

b=-a;

System.out.println(“Your balance amount is ”+b);

}

else

{

throw new InsufficientBalanceException();

}

}

catch(InsufficientBalanceException e)

{

e.printStackTrace();

}

}

public static void main(String[] args)

{

SavingAccount sa=new SavingAccount ();

sa.withdraw(2000);

}

}

Output:

Enter your ID: 234

Enter balance amount: 1000

Account.InsufficientBalanceException

at Account.SavingAccount.withdrawal(SavingAccount.java:21)

at Account.SavingAccount.withdrawal(SavingAccount.java:32)

b. Raise a **IllegalBankTransaction:** If you are trying to withdraw negative value from your balance Eg: 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)

class IllegalBankTransactionException extends RuntimeException{

}

public class SavingAccount{

Scanner s = new Scanner(System.in);

public void withdraw(double a)

{

System.out.println(“Enter your ID: ”);

long id = s.nextLong();

System.out.println(“Enter balance amount:”);

double b=s.nextDouble();

try{

if(a>0)

{

System.out.println(“Your Balance amount: ”+b);

}

else{

throw new IllegalBankTransactionException();

}

}

catch (IllegalBankTransactionException e){

e.printStackTrace();

}

}

Public static void main(String[] args){

SavingAccount sa =new SavingAccount();

sa.withdraw(-10201);

}

}

Output:

Enter your ID: 234

Enter balance amount: 1000

Account.IllegalBankTransactionException

at Account.SavingAccount.withdraw(SavingAccount.java:21)

at Account.SavingAccount.withdraw(SavingAccount.java:32)