23-10-20

1

import java.io.\*;

public class Main

{

public static void main(String[] args) throws Exception {

int n1,n2,d;

String num1,num2;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

try{

num1=br.readLine();

num2=br.readLine();

n1=Integer.parseInt(num1);

n2=Integer.parseInt(num2);

d=n1/n2;

System.out.println("output"+d);

}

catch(NumberFormatException e){

System.out.println("Inputs are not valid");

}

catch(ArithmeticException e){

System.out.println("Divide by Zero error");

}

}

}

2. .Java programs to create an bank account with minimum balance, deposit amount,

withdraw amount and throws LessBalanceException, create a

LessBalanceException class which returns a statement says withdraw amount is

not valid, creates 2 accounts and try to withdraw more money than account and

see which type of exception occurs.

import java.io.\*;

class BalanceCheck extends Exception{

BalanceCheck(){

super("Transaction Denied: No min balance found");

}

}

class Bank{

int accountno;

String name;

double balance;

static int min\_amount=500;

Bank(int ano,String nm,double bal,int min){

accountno=ano;

name=nm;

balance=bal;

}

void Withdraw(double cash) throws BalanceCheck{

if((balance-cash)>=min\_amount){

balance=balance-cash;

System.out.println("Transaction Succesful");

System.out.println("the balance after withdrawl is "+ balance);

}

else{

throw new BalanceCheck();

}

}

void Deposit(double cash){

balance = balance+cash;

System.out.println("Transaction Succesful");

System.out.println("the balance after deposit is "+balance);

}

void CheckBal(){

System.out.println("the balance is"+balance);

}

}

public class Main

{

public static void main(String[] args) throws Exception {

Bank b1 = new Bank(49960,"haru",10000,500);

Bank b2 = new Bank(40935,"harish",10000,500);

try{

b1.Withdraw(6800);

b2.Withdraw(10000);

}

catch(BalanceCheck b){

System.out.println(b);

}

b1.CheckBal();

b2.CheckBal();

b1.Deposit(7000);

b2.Deposit(4000);

b1.CheckBal();

b2.CheckBal();

}

}

3. Write a Java program to check whether the age entered is a valid number by creating user defined exception.

import java.util.\*;

class AgeCheck extends Exception{

AgeCheck(){

super("Age is invalid");

}

}

class Age{

Scanner s = new Scanner(System.in);

int age=s.nextInt();

void checkage() throws AgeCheck{

if(age>0&&age<120){

System.out.println("Age is valid");

}

else{

throw new AgeCheck();

}

}

}

public class Main

{

public static void main(String[] args) throws Exception {

Age a1 = new Age();

try{

a1.checkage();

}

catch(AgeCheck a){

System.out.println(a);

}

}

}

4. Design a Java interface for ADT Stack. Implement this interface using array.

Provide necessary exception handling in both the implementations.

class FullStack extends Exception

{

FullStack()

{

super("Stack is Full");

}

}

class EmptyStack extends Exception

{

EmptyStack()

{

super("Stack is Empty ");

}

}

class Stack

{

int top;

int arr[];

static int max=10;

Stack()

{

top=-1;

arr=new int[max];

}

void push(int x)throws FullStack

{

if(top==max-1)

{

throw new FullStack();

}

else

{

arr[++top]=x;

}

}

int pop()throws EmptyStack

{

if(top==-1)

{

throw new EmptyStack();

}

else

{

return(arr[top--]);

}

}

void print()

{

for(int i=0;i<arr.length;i++)

System.out.print(arr[i]+" ");

System.out.println();

}

}

public class MyClass {

public static void main(String args[]) {

Stack s1=new Stack();int x;

for(int i=1;i<=12;i++)

{

try{

s1.push(i);

s1.print();

}

catch(FullStack fs)

{

System.out.println(fs);

}

}

for (int i=1;i<=12;i++)

{

try

{

x=s1.pop();

System.out.print(x +" ");

}

catch(EmptyStack es)

{

System.out.println(es);

}

}

}

}