import java.util.Scanner;

public class assignQ1 {

    public static void main(String[] args) {

        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();

            int result=dividend/divisor;

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

            scanner.close();

        }

        catch(Exception e)

        {

            e.printStackTrace();

        }

        finally {

            if(scanner!=null)

            scanner.close();

        }

    }

}

2.

import java.util.Scanner;

class assignQ2  {

    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 assignmentQ2 {

    public static void main(String[] args)  {

        assignQ2.solve();

    }

}

3.

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 assignmentQ3 {

    public static void main(String[] args)  {

        savingAccount Srija = new savingAccount(10,9000);

        Srija.deposit(-1);

        Srija.withdraw(9100);

        System.out.println(Srija.getBalance());

    }

}