***CODE FOR PART1***

package gradedassignment;

import static gradedassignment.GradedAssignment.listCom;

import static gradedassignment.GradedAssignment.listOrder;

import java.util.\*;

import java.lang.\*;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.FileReader;

import java.io.\*;

import java.util.Arrays;

/\*\*

\*

\* @author sharm

\*/

public class GradedAssignment {

static int numOfcompany=0;

static int numOftraders=0;

static int numOforders=0;

static ArrayList<Companyfn> listCom=new ArrayList<>();

static ArrayList<Traderfn> listTrader=new ArrayList<>();

static ArrayList<PlaceOrderfn> listOrder=new ArrayList<>();

static int tip=0;

public static void main(String[] args) {

// TODO code application logic here

BufferedReader br= null;

String line;

try {

br= new BufferedReader(new FileReader("C:\\Users\\sharm\\OneDrive\\Desktop\\sample\_input2.txt"));

while((line=br.readLine())!=null)

{

line=line.replaceAll("[,:{}]"," ");

line=line.replaceAll("\\s{2,}"," ").trim();

if(line.contentEquals(" "))

{

continue;

}

String[] tokens=line.split(" ");

List<String>wordlist=Arrays.asList(tokens);

//System.out.println(wordlist.size());

if(wordlist.size()>=2)

{

if(wordlist.get(0).equals("Add") && wordlist.get(1).equals("scrip"))

{

//call company constructor

int i=numOfcompany;

listCom.add(new Companyfn(wordlist.get(2),wordlist.get(4),wordlist.get(6),wordlist.get(8),wordlist.get(10),wordlist.get(12)));

Companyfn.printAddition(listCom.get(i).tickerCom);

numOfcompany++;

}

else if(wordlist.get(0).equals("Add") && wordlist.get(1).equals("user"))

{

//call trader constructor

int t=numOftraders;

listTrader.add(new Traderfn(wordlist,wordlist.get(2),wordlist.get(4)));

Traderfn.printTrader(listTrader.get(t).name);

numOftraders++;

}

else if(wordlist.get(0).equals("Place") && wordlist.get(1).equals("order"))

{

//call place order

if(tip==0){

System.out.println("Market Opens:");

tip++;

}

int t=numOforders;

listOrder.add(new PlaceOrderfn(wordlist.get(3),wordlist.get(5),wordlist.get(7),wordlist.get(9),wordlist.get(11)));

int cd=PlaceOrderfn.checkConditions(listOrder.get(t));

if(cd==0)

{

System.out.println("Order placed for user: "+listOrder.get(t).userPlacing+", type: "+listOrder.get(t).typeOforder+", scrip: "+listOrder.get(t).scripOfcom+", qty:"+listOrder.get(t).quantityDemanded+", rate: "+listOrder.get(t).rateOfshare);

numOforders++;

}

else if(cd==1)

{

System.out.println("Order rejected for user: "+listOrder.get(t).userPlacing+", type: "+listOrder.get(t).typeOforder+", scrip: "+listOrder.get(t).scripOfcom+", qty:"+listOrder.get(t).quantityDemanded+", rate: "+listOrder.get(t).rateOfshare+", reason: Insufficient funds.");

listOrder.remove(t);

}

else if(cd==2)

{

System.out.println("Order rejected for user: "+listOrder.get(t).userPlacing+", type: "+listOrder.get(t).typeOforder+", scrip: "+listOrder.get(t).scripOfcom+", qty:"+listOrder.get(t).quantityDemanded+", rate: "+listOrder.get(t).rateOfshare+", reason: lower circuit violation.");

listOrder.remove(t);

}

else if(cd==3)

{

System.out.println("Order rejected for user: "+listOrder.get(t).userPlacing+", type: "+listOrder.get(t).typeOforder+", scrip: "+listOrder.get(t).scripOfcom+", qty:"+listOrder.get(t).quantityDemanded+", rate: "+listOrder.get(t).rateOfshare+", reason: upper circuit violation.");

listOrder.remove(t);

}

}

else if(wordlist.get(0).equals("Show"))

{

if(wordlist.get(1).equals("Orderbook"))

{

//call Orderbook function

System.out.println("Orderbook:");

for(int fr=0;fr<listOrder.size();fr++)

{

System.out.println(listOrder.get(fr).typeOforder+" order "+listOrder.get(fr).scripOfcom+":"+listOrder.get(fr).quantityDemanded+" at "+listOrder.get(fr).rateOfshare);

}

}

else if(wordlist.get(1).equals("Scrips"))

{

//call show scrips

System.out.println("Scrips:");

for(int j=0;j<listCom.size();j++)

{

System.out.println("scrip: "+listCom.get(j).tickerCom+", sector: "+listCom.get(j).sectorCom+", O:"+listCom.get(j).openPrice+", H:"+listCom.get(j).highPrice+", L:"+listCom.get(j).lowPrice+", C:"+listCom.get(j).closePrice);

}

}

else if(wordlist.get(1).equals("Users"))

{

//call show user

System.out.println("Users:");

for(int j=0;j<listTrader.size();j++)

{

int n=listTrader.get(j).holdingCom.size();

System.out.print("user: "+listTrader.get(j).name+", funds:"+listTrader.get(j).fundsAtpresent+", holding: {");

for(int p=0;p<n;p++)

{

System.out.print(listTrader.get(j).holdingCom.get(p)+":"+listTrader.get(j).holdingCount.get(p)+", ");

}

System.out.print("}\n");

}

}

else if(wordlist.get(1).equals("sector"))

{

//call show sector

System.out.println("Scrips listed in sector: "+wordlist.get(2));

for(int j=0;j<listCom.size();j++)

if(wordlist.get(2).equals(listCom.get(j).sectorCom))

{

System.out.println((listCom.get(j).tickerCom)+", OHLC = <"+(listCom.get(j).openPrice)+", "+(listCom.get(j).highPrice)+", "+(listCom.get(j).lowPrice)+", "+(listCom.get(j).closePrice)+">");

}

}

}

if(wordlist.get(0).equals("Delete"))

{

if(wordlist.get(1).equals("scrip"))

{

//call delete scrip

for(int j=0;j<listCom.size();j++)

if(wordlist.get(2).equals(listCom.get(j).tickerCom))

{

System.out.println("Deleted scrip: "+listCom.get(j).tickerCom);

listCom.remove(j);

}

}

else if(wordlist.get(1).equals("User"))

{

//call delete user

for(int j=0;j<listTrader.size();j++)

if(wordlist.get(2).equals(listTrader.get(j).name))

{

System.out.println("Deleted User: "+listTrader.get(j).name);

listTrader.remove(j);

}

}

}

}

else if(wordlist.size()==1)

{

if(wordlist.get(0).equals("Execute"))

{

//transaction function

System.out.println("Executed transactions:");

PlaceOrderfn.checkExecution();

}

else if(wordlist.get(0).equals("Exit"))

{

System.out.println("Market closed.");

break;

}

}

}

}catch (IOException e){

e.printStackTrace();

}

}

}

class Companyfn extends GradedAssignment{

String tickerCom;

String sectorCom;

int openPrice;

int highPrice;

int lowPrice;

int closePrice;

String stockExchange;

public Companyfn(String tick,String sec,String O,String H,String L,String C)

{

tickerCom=tick;

sectorCom=sec;

openPrice=Integer.parseInt(O);

highPrice=Integer.parseInt(H);

lowPrice=Integer.parseInt(L);

closePrice=Integer.parseInt(C);

stockExchange=((int)(Math.random()\*2)==0)?"NSE":"BSE";

}

public static void printAddition(String str )

{

System.out.println("Added scrip: "+str+" with a new instantiation of Class Companyfn ");

}

}

class Traderfn extends GradedAssignment{

String name;

int fundsAtpresent;

int customerID;

ArrayList<String> holdingCom=new ArrayList<>();

ArrayList<Integer> holdingCount=new ArrayList<>();

public Traderfn(List<String>wordlist,String s1,String s2)

{

name=wordlist.get(2);

fundsAtpresent=Integer.parseInt(s2);

if(wordlist.size()%2==0)

{

for(int k=6;k<wordlist.size();k=k+2)

{

String str1=wordlist.get(k);

String str2=wordlist.get(k+1);

int pt=Integer.parseInt(str2);

holdingCom.add(str1);

holdingCount.add(pt);

}

}

else if(wordlist.size()%2!=0)

{

holdingCom.add(wordlist.get(6));

holdingCount.add(0);

}

customerID=((int)(Math.random()\*1000000));

}

public static void printTrader(String str )

{

System.out.println("Added user: "+str+" with a new instantiation of Class Traderfn ");

}

}

class PlaceOrderfn extends GradedAssignment{

String userPlacing;

String typeOforder;

String scripOfcom;

int quantityDemanded;

int rateOfshare;

public PlaceOrderfn(String user,String type,String scrip,String quantity,String rate)

{

userPlacing=user;

typeOforder=type;

scripOfcom=scrip;

quantityDemanded=Integer.parseInt(quantity);

rateOfshare=Integer.parseInt(rate);

}

public static int checkConditions(PlaceOrderfn order)

{

int flag=0;

if(order.typeOforder.equals("buy")){

for(int st=0;st<listTrader.size();st++)

{

if(listTrader.get(st).name.equals(order.userPlacing))

{

if(checkFunds(listTrader.get(st).fundsAtpresent,(order.quantityDemanded\*order.rateOfshare))==1)

flag=1;

}

}

for(int st=0;st<listCom.size();st++)

{

if(listCom.get(st).tickerCom.equals(order.scripOfcom))

{

if(checkLowercircuit(listCom.get(st).closePrice,order.rateOfshare)==1)

{

flag=2;

}

}

}

for(int st=0;st<listCom.size();st++)

{

if(listCom.get(st).tickerCom.equals(order.scripOfcom))

{

if(checkUppercircuit(listCom.get(st).closePrice,order.rateOfshare)==1)

{

flag=3;

}

}

}

}

return flag;

}

public static int checkFunds(int having,int wanted)

{

if(having<wanted)

{

//System.out.println(having+" " +wanted);

return 1;

}

else

return 0;

}

public static int checkLowercircuit(int comrate,int traderrate)

{

if(traderrate<(((double)comrate)-((double)comrate\*0.1)))

return 1;

else

return 0;

}

public static int checkUppercircuit(int comrate,int traderrate)

{

if(traderrate>(((double)comrate)+((double)comrate\*0.1)))

return 1;

else

return 0;

}

public static void checkExecution()

{

for(int ch=0;ch<listOrder.size();ch++)

{

if(listOrder.get(ch).typeOforder.equals("buy"))

{

for(int tt=0;tt<listOrder.size();tt++)

{

if(listOrder.get(tt).typeOforder.equals("sell"))

{

if(listOrder.get(ch).scripOfcom.equals(listOrder.get(tt).scripOfcom) && listOrder.get(ch).rateOfshare>=listOrder.get(tt).rateOfshare)

{

if(listOrder.get(ch).quantityDemanded>listOrder.get(tt).quantityDemanded)

{

System.out.println(listOrder.get(tt).quantityDemanded+" qty of scrip:"+listOrder.get(ch).scripOfcom+" sold for INR "+listOrder.get(tt).rateOfshare+"; Buyer: "+listOrder.get(ch).userPlacing+", Seller: "+listOrder.get(tt).userPlacing);

int fundexchange=(listOrder.get(tt).quantityDemanded)\*(listOrder.get(tt).rateOfshare);

String buyer=(listOrder.get(ch).userPlacing);

String seller=(listOrder.get(tt).userPlacing);

String scripused=listOrder.get(ch).scripOfcom;

int numofstocks=listOrder.get(tt).quantityDemanded;

for(int q=0;q<listTrader.size();q++)

{

if(listTrader.get(q).name.equals(buyer))

{

listTrader.get(q).holdingCom.add(scripused);

listTrader.get(q).holdingCount.add(numofstocks);

listTrader.get(q).fundsAtpresent=listTrader.get(q).fundsAtpresent-fundexchange;

}

else if(listTrader.get(q).name.equals(seller))

{

listTrader.get(q).holdingCom.remove(scripused);

listTrader.get(q).holdingCount.remove(Integer.valueOf(numofstocks));

listTrader.get(q).fundsAtpresent=listTrader.get(q).fundsAtpresent+fundexchange;

}

}

}

else if(listOrder.get(ch).quantityDemanded<listOrder.get(tt).quantityDemanded)

{

System.out.println(listOrder.get(ch).quantityDemanded+" qty of scrip:"+listOrder.get(ch).scripOfcom+" sold for INR "+listOrder.get(tt).rateOfshare+"; Buyer: "+listOrder.get(ch).userPlacing+", Seller: "+listOrder.get(tt).userPlacing);

int fundexchange=(listOrder.get(tt).quantityDemanded)\*(listOrder.get(tt).rateOfshare);

String buyer=(listOrder.get(ch).userPlacing);

String seller=(listOrder.get(tt).userPlacing);

String scripused=listOrder.get(ch).scripOfcom;

int numofstocks=listOrder.get(ch).quantityDemanded;

for(int q=0;q<listTrader.size();q++)

{

if(listTrader.get(q).name.equals(buyer))

{

listTrader.get(q).holdingCom.add(scripused);

//listTrader.get(q).holdingCount.add(listOrder.get(tt).quantityDemanded);

listTrader.get(q).fundsAtpresent=listTrader.get(q).fundsAtpresent-fundexchange;

}

else if(listTrader.get(q).name.equals(seller))

{

// listTrader.get(q).holdingCount.remove(numofstocks);

listTrader.get(q).fundsAtpresent=listTrader.get(q).fundsAtpresent+fundexchange;

}

}

}

else if(listOrder.get(ch).quantityDemanded==listOrder.get(tt).quantityDemanded)

{

System.out.println(listOrder.get(tt).quantityDemanded+" qty of scrip:"+listOrder.get(ch).scripOfcom+" sold for INR "+listOrder.get(tt).rateOfshare+"; Buyer: "+listOrder.get(ch).userPlacing+", Seller: "+listOrder.get(tt).userPlacing);

int fundexchange=(listOrder.get(tt).quantityDemanded)\*(listOrder.get(tt).rateOfshare);

String buyer=(listOrder.get(ch).userPlacing);

String seller=(listOrder.get(tt).userPlacing);

String scripused=listOrder.get(ch).scripOfcom;

int numofstocks=listOrder.get(tt).quantityDemanded;

for(int q=0;q<listTrader.size();q++)

{

if(listTrader.get(q).name.equals(buyer))

{

listTrader.get(q).holdingCom.add(scripused);

listTrader.get(q).holdingCount.add(numofstocks);

listTrader.get(q).fundsAtpresent=listTrader.get(q).fundsAtpresent-fundexchange;

}

else if(listTrader.get(q).name.equals(seller))

{

listTrader.get(q).holdingCom.remove(scripused);

listTrader.get(q).holdingCount.remove(Integer.valueOf(numofstocks));

listTrader.get(q).fundsAtpresent=listTrader.get(q).fundsAtpresent+fundexchange;

}

}

}

}

}

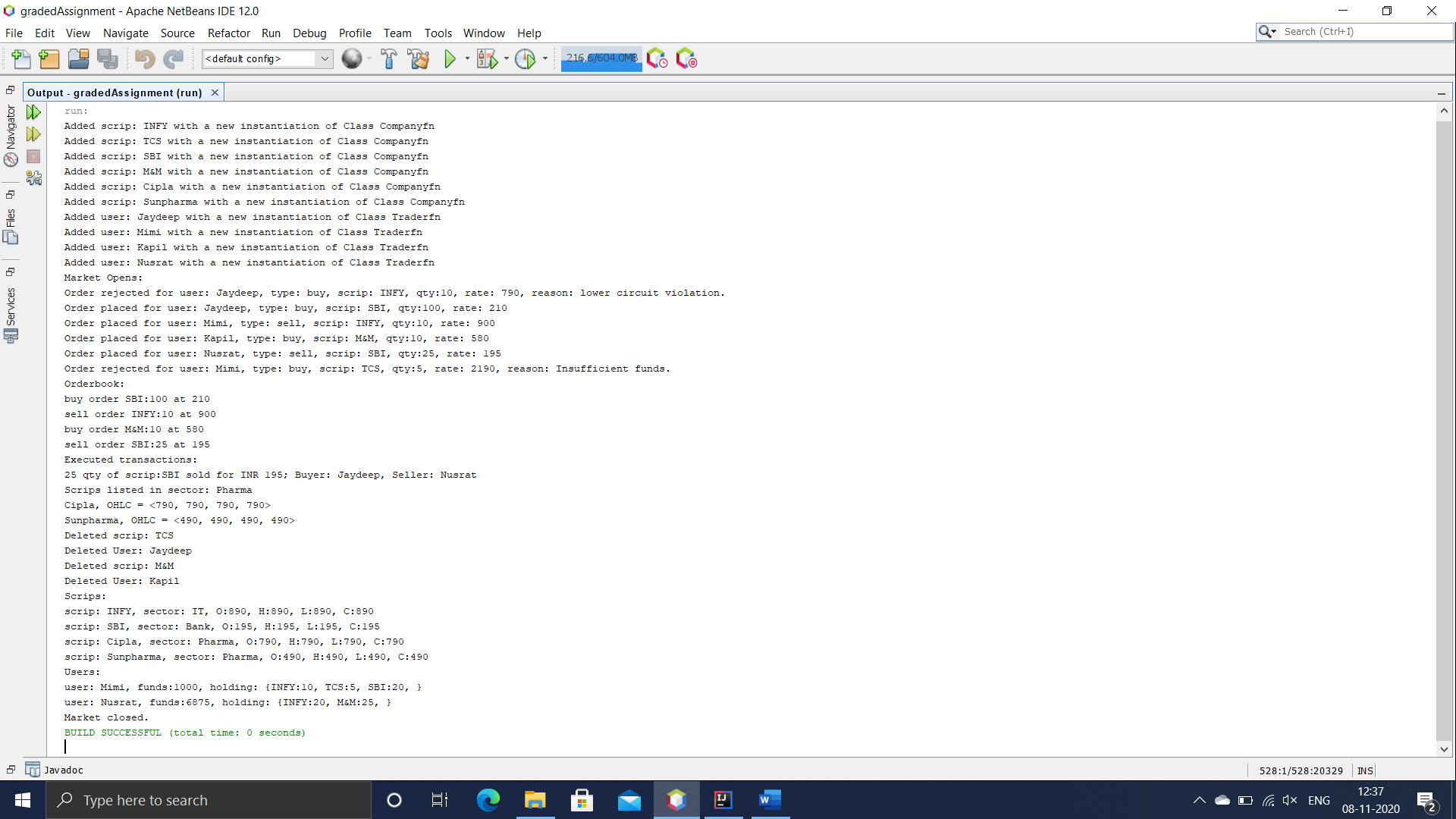
}

}

}

}

}



***CODE FOR PART 2:***

package excelimport;

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

/\*\*

\*

\* @author sharm

\*/

public class ExcelImport {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

BufferedReader bt= null;

String line;

ArrayList<Double> PrevclosePrice=new ArrayList<>();

ArrayList<Double> OpenPrice=new ArrayList<>();

ArrayList<Double> HighPrice=new ArrayList<>();

ArrayList<Double> LowPrice=new ArrayList<>();

ArrayList<Double> LastPrice=new ArrayList<>();

ArrayList<Double> ClosePrice=new ArrayList<>();

try {

bt= new BufferedReader(new FileReader("C:\\Users\\sharm\\OneDrive\\Desktop\\sample\_input22.txt"));

while((line=bt.readLine())!=null)

{

line=line.replaceAll("[,:{}-]"," ");

line=line.replaceAll("\\s{2,}"," ").trim();

if(line.contentEquals(" "))

{

continue;

}

//System.out.println(line);

String[] tokens=line.split(" ");

List<String>wordlist=Arrays.asList(tokens);

PrevclosePrice.add(Double.parseDouble(wordlist.get(4)));

OpenPrice.add(Double.parseDouble(wordlist.get(5)));

HighPrice.add(Double.parseDouble(wordlist.get(6)));

LowPrice.add(Double.parseDouble(wordlist.get(7)));

LastPrice.add(Double.parseDouble(wordlist.get(8)));

ClosePrice.add(Double.parseDouble(wordlist.get(9)));

//System.out.println(OpenPrice);

if(ClosePrice.size()==15)

{

averageStockprice(LastPrice);

maxDrawdown(ClosePrice,PrevclosePrice);

maxReturnPotential(OpenPrice,ClosePrice);

}

}

}catch (IOException e){

e.printStackTrace();

}

}

public static void averageStockprice(ArrayList LastPrice)

{

double sum=0.0;

for(int i=0;i<LastPrice.size();i++)

{

sum=sum+ (double)LastPrice.get(i);

}

System.out.println("The average price of a given stock: "+sum/15);

}

public static void maxDrawdown(ArrayList ClosePrice,ArrayList prevClosePrice)

{

double max=(double)prevClosePrice.get(0)-(double)ClosePrice.get(0);

for(int i=0;i<ClosePrice.size();i++)

{

if(max<((double)prevClosePrice.get(i))-((double)ClosePrice.get(i)))

{

max=((double)prevClosePrice.get(i))-((double)ClosePrice.get(i));

}

}

System.out.println("MaxDrawdown of the stock is: "+max);

}

public static void maxReturnPotential(ArrayList OpenPrice,ArrayList ClosePrice)

{

double maxpotential=0;

double maxpotentialper=0;

for(int i=0;i<OpenPrice.size();i++)

{

if(((double)OpenPrice.get(i)-(double)ClosePrice.get(i))>0)

{

maxpotential= maxpotential+((double)OpenPrice.get(i)-(double)ClosePrice.get(i));

}

else if(((double)ClosePrice.get(i)-(double)OpenPrice.get(i)>0))

{

maxpotential= maxpotential+(double)ClosePrice.get(i)-(double)OpenPrice.get(i);

}

}

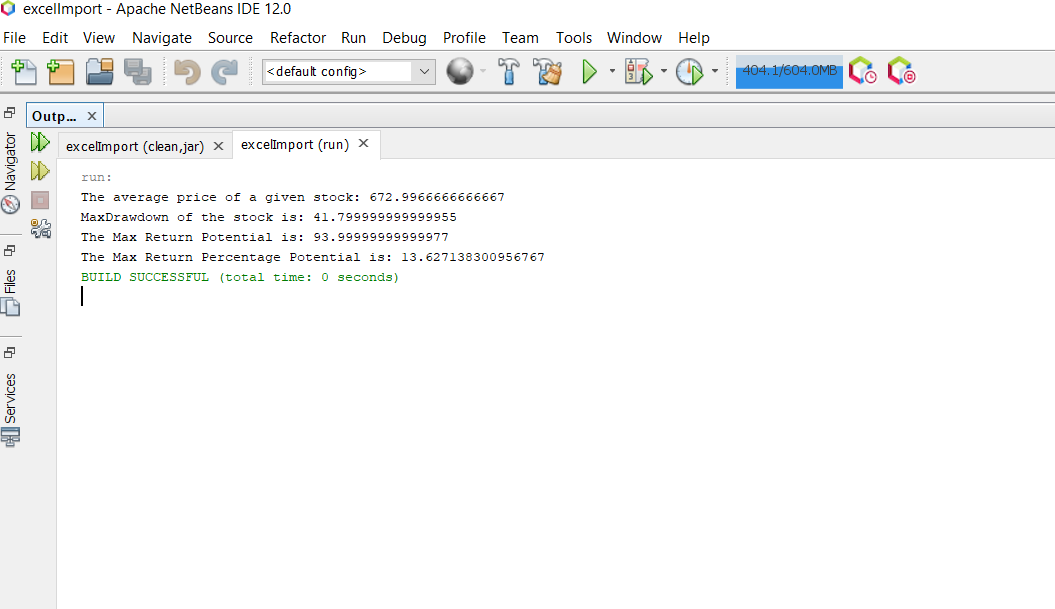
System.out.println("The Max Return Potential is: " + maxpotential);

maxpotentialper=((100\*maxpotential)/(double)OpenPrice.get(0));

System.out.println("The Max Return Percentage Potential is: " + maxpotentialper);

}

}



***UML DIAGRAM:***

