

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package assignment2;

import static assignment2.Brokerage\_System.Listofentity;

import static assignment2.Brokerage\_System.writer;

import java.io.\*;

import java.text.\*;

import java.util.\*;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\*

\* @author sam

\*/

abstract class company

{

String Ticker;

String Sector;

}

class PubListCo extends company

{

String Ticker;

String Sector;

Date day;

double OpenPrice;

double HighPrice;

double LowPrice;

double ClosePrice;

int TransactionNo;

double LastPrice;

double PrevClosePrice;

int[] Trading = new int[10];

PubListCo(String t, String s, double o, double h, double l, double c)

{

Ticker = t;

Sector = s;

OpenPrice = o;

HighPrice = h;

LowPrice = l;

ClosePrice = c;

TransactionNo=0;

LastPrice=c;

}

public int getTransactionNo()

{

return TransactionNo;

}

public void incrementTransactionNo()

{

TransactionNo++;

}

public double getLastPrice()

{

return LastPrice;

}

public void setLastPrice(double lp)

{

LastPrice=lp;

}

public void setPrevClosePrice(double pcp)

{

PrevClosePrice=pcp;

}

public String getTicker()

{

return Ticker;

}

public String getSector()

{

return Sector;

}

public double getOPrice()

{

return OpenPrice;

}

public double getHPrice()

{

return HighPrice;

}

public double getLPrice()

{

return LowPrice;

}

public double getCPrice()

{

return ClosePrice;

}

public void setOPrice(double newO)

{

OpenPrice= newO;

}

public void setHPrice(double newH)

{

HighPrice= newH;

}

public void setLPrice(double newL)

{

LowPrice= newL;

}

public void setCPrice(double newC)

{

ClosePrice= newC;

}

public void setDate(Date dt)

{

day=dt;

}

}

class StockEx//make 2 object Nse and Bse

{

String name;

ArrayList<PubListCo> regco ;

StockEx(String n)

{

name = n;

regco = new ArrayList<PubListCo>();

}

public void register(PubListCo comp)//will add trader or add public listed company to its list

{ //is same as add/list a new company with stock exchange

regco.add(comp);

}

public void deregister(ArrayList<PubListCo>plc)//will add trader or add public listed company to its list

{ //is same as add/list a new company with stock exchange

regco = plc;

}

}

class entity\_stock// class for making arraylist of the stocks owned by the entity

{

String Ticker;

int qty;

entity\_stock(String t, int q)

{

Ticker = t;

qty = q;

}

public String getTicker()

{

return Ticker;

}

public int getqty()

{

return qty;

}

public void setqty(int q)

{

qty=q;

}

}

class entity

{

String name;

double funds;

ArrayList<entity\_stock> holding = new ArrayList<entity\_stock>(); //list of all the stocks of various PubListCo owned by the entity

//int custId;

//String Type;

//ArrayList<Integer> order = new ArrayList<Integer>();

entity(String n, double f, ArrayList<entity\_stock> es)

{

name = n;

funds = f;

holding = es;

}

public String getname()

{

return name;

}

public double getfunds()

{

return funds;

}

public void setfundsb(double f)

{

funds-=f;

}

public void setfundss(double f)

{

funds+=f;

}

public ArrayList<entity\_stock> getHolding()

{

return holding;

}

public void setHolding(ArrayList<entity\_stock> es)

{

holding = es;

}

public String DisplayStockList()

{

//user: Mimi, funds:1000, holding: {INFY:10, TCS:5, SBI:20}

String str = "{"+holding.get(0).getTicker()+":"+holding.get(0).getqty();

for(int i = 1;i < holding.size()-1;i++)

str += ", "+holding.get(i).getTicker()+":"+holding.get(i).getqty();

str += ", "+holding.get(holding.size()-1).getTicker()+":"+holding.get(holding.size()-1).getqty()+"}";

return str;

}

}

class order

{

entity e;

String type;//Buy, Sell

PubListCo company;

int qty;

double rate;

String Status;//placed or rejected

String Reason;//will contain the reason for rejection of the order

public order(entity en, String t, PubListCo plc, int q, double r)

{

e = en;

type = t;

company = plc;

qty = q;

rate = r;

Status = "placed";

}

public void validate()

{

if(type.equals("buy"))

{

if(qty\*rate >= e.getfunds())

{

Status = "rejected";

Reason = "Insufficient funds.";

}

else if(rate<(company.getLPrice()-0.1\*company.getLPrice()))

{

Status = "rejected";

Reason = "lower circuit violation.";

}

else if(rate>(company.getHPrice()+0.1\*company.getHPrice()))

{

Status = "rejected";

Reason = "upper circuit violation.";

}

}

}

public String status()

{

return Status;

}

public String reason()

{

return Reason;

}

public String Type()

{

return type;

}

public PubListCo comp()

{

return company;

}

public entity user()

{

return e;

}

public int quantity()

{

return qty;

}

public void updateqty(int q)

{

qty+= q;

}

public double Rate()

{

return rate;

}

}

public class Brokerage\_System

{

/\*\*

\* @param args the command line arguments

\*/

static StockEx Bse;

static StockEx Nse;

static FileWriter writer;

static String dashes = "-------------------------------------\n";

static ArrayList<PubListCo> ListofCo,DayWiseListofCo;

static ArrayList<entity> Listofentity;

static ArrayList<order> orderList;

static File file;

static BufferedReader br;

static String inputFile="D:\\SNU\\Sem3\\CSD203\\lab\\assignment2\\src\\assignment2\\sample\_input.txt";

static String inputFile1="D:\\SNU\\Sem3\\CSD203\\lab\\assignment2\\src\\assignment2\\INFY\_15days\_data.csv";

static String outputFile="D:\\SNU\\Sem3\\CSD203\\lab\\assignment2\\src\\assignment2\\output.txt";

static String outputFile1="D:\\SNU\\Sem3\\CSD203\\lab\\assignment2\\src\\assignment2\\output1.txt";

public static entity getEntity(String s)

{

entity e = null;

for (entity a : Listofentity)

{

if(s.equals(a.getname()))

{

e=a;

break;

}

}

return e;

}

public static PubListCo getCompany(String s)

{

PubListCo c = null;

for (PubListCo a : ListofCo)

{

if(s.equals(a.getTicker()))

{

c = a;

break;

}

}

return c;

}

public static void readInputFile(String filename) throws IOException

{

file = new File(filename);

if(!file.exists())

{

throw new IOException("Demo");

}

br = new BufferedReader(new FileReader(file));

}

public static void add() throws IOException

{

double o,h,l,c,funds;

String str, addplc,user;

String[] arr = new String[100];

ListofCo = new ArrayList<>();

Listofentity = new ArrayList<>();

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

{

if(!(str.isEmpty())&&(str.length()>=8))//extracting add commands from file

{

addplc = str.substring(0,8);

switch (addplc)

{

case "Add scri":

//System.out.println("Add Scri");

arr = str.split(":|,");

o = Double.parseDouble(arr[5]);

h = Double.parseDouble(arr[7]);

l = Double.parseDouble(arr[9]);

c = Double.parseDouble(arr[11]);

arr[1] = arr[1].split(" ")[1];

arr[3] = arr[3].split(" ")[1];

PubListCo company = new PubListCo(arr[1],arr[3],o,h,l,c);

ListofCo.add(company);

writer.write("Added scrip: "+arr[1]+" with a new instantiation of PubListCo\n");

break;

case "Add user":

arr = str.split(":|,");

user = arr[1].split(" ")[1];

arr = arr[3].split(" ");

funds = Double.parseDouble(arr[0]);

Pattern pattern = Pattern.compile("[/{]");

Matcher matcher = pattern.matcher(str);

boolean matchFound = matcher.find();

ArrayList<entity\_stock> es = new ArrayList<>();

if(matchFound)

{

arr = str.split("[/{]",2);

arr = arr[1].split("[/}]",2);

arr = arr[0].split(":|,");

int counter=0,q;

String strTicker = "";

for(String a : arr)

{

counter++;

if (counter%2==0)

{

q = Integer.parseInt(a);

entity\_stock e = new entity\_stock(strTicker,q);

counter=0;

es.add(e);

}

else strTicker = a;

}

}

entity ent = new entity(user, funds, es);

Listofentity.add(ent);//list of all the users

writer.write("Added user: "+user+" with a new instantiation of Entity\n");

break;

default:

//System.out.println("In switch- default case, str= "+str);

break;

}

}

}

br.close();

}

public static void execute() throws IOException

{

int i,j;

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

writer.write("Execute transactions:\n");

for(i = 0;i < orderList.size();i++)

{

order oi= orderList.get(i);

for(j = 1;j < orderList.size();j++)

{

order oj= orderList.get(j);

if(oi.Type().equals("buy")&&oj.Type().equals("sell")&&oi.comp().getTicker().equals(oj.comp().getTicker())&&(oi.Rate()>=oj.Rate())

&&(oi.quantity()>=oj.quantity()))

{

//setting values of o,h,l for the company

if (oi.comp().getTransactionNo()==0)

{

oi.comp().setOPrice(oj.Rate());

oi.comp().setHPrice(oj.Rate());

oi.comp().setLPrice(oj.Rate());

}

else

{

//set H and L price

if (oi.comp().getHPrice()<oj.Rate())

{

oi.comp().setHPrice(oj.Rate());

}

if (oi.comp().getLPrice()>oj.Rate())

{

oi.comp().setLPrice(oj.Rate());

}

}

oi.comp().incrementTransactionNo();

oi.comp().setLastPrice(oj.Rate());

oi.user().setfundsb(oj.Rate()\*oj.quantity());

oj.user().setfundss(oj.Rate()\*oj.quantity());

//have to change the stocks

ArrayList<entity\_stock> esi=oi.user().getHolding();

ArrayList<entity\_stock> esj=oj.user().getHolding();

int found=0;

for (entity\_stock es : esi)

{

if (es.getTicker().equals(oi.comp().getTicker()))

{

es.setqty(es.getqty()+oj.quantity());

found=1;

break;

}

}

if (found==0)

{

entity\_stock es\_new= new entity\_stock(oi.comp().getTicker(),oj.quantity());

esi.add(es\_new);

}

for (entity\_stock es1 : esj)

{

if (es1.getTicker().equals(oj.comp().getTicker()))

es1.setqty(es1.getqty()-oj.quantity());

}

writer.write(oj.quantity()+" qty of scrip:"+oj.comp().getTicker()+" sold at the rate of INR "+oj.Rate()+" ; Buyer: "

+oi.user().getname()+" , Seller: "+oj.user().getname()+"\n");

if (oi.quantity()>oj.quantity())//decrement oi.quantity() in order list

{

oi.updateqty(-oj.quantity());

}

/\*

if (oi.quantity()==oj.quantity())// remove oi from orderlist

{

// orderList.remove(i);

ai.add(i);

}\*/

ai.add(i);

ai.add(j);

//orderList.remove(j);//remove oj from order list

}

if(oi.Type().equals("buy")&&oj.Type().equals("sell")&&oi.comp().getTicker().equals(oj.comp().getTicker())&&(oi.Rate()>=oj.Rate())

&&(oi.quantity()<oj.quantity()))

{

//setting values of o,h,l for the company

if (oi.comp().getTransactionNo()==0)

{

oi.comp().setOPrice(oj.Rate());

oi.comp().setHPrice(oj.Rate());

oi.comp().setLPrice(oj.Rate());

}

else

{

//set H and L price

if (oi.comp().getHPrice()<oj.Rate())

{

oi.comp().setHPrice(oj.Rate());

}

if (oi.comp().getLPrice()>oj.Rate())

{

oi.comp().setLPrice(oj.Rate());

}

}

oi.comp().incrementTransactionNo();

oi.comp().setLastPrice(oj.Rate());

oi.user().setfundsb(oj.Rate()\*oi.quantity());

oj.user().setfundss(oj.Rate()\*oi.quantity());

//have to change the stocks

ArrayList<entity\_stock> esi=oi.user().getHolding();

ArrayList<entity\_stock> esj=oj.user().getHolding();

int found=0;

for (entity\_stock es : esi)

{

if (es.getTicker().equals(oi.comp().getTicker()))

{

es.setqty(es.getqty()+oi.quantity());

found=1;

break;

}

}

if (found==0)

{

entity\_stock es\_new= new entity\_stock(oi.comp().getTicker(),oi.quantity());

esi.add(es\_new);

}

for (entity\_stock es1 : esj)

{

if (es1.getTicker().equals(oj.comp().getTicker()))

es1.setqty(es1.getqty()-oi.quantity());

}

writer.write(oj.quantity()+" qty of scrip:"+oj.comp().getTicker()+" sold at the rate of INR "+oj.Rate()+" ; Buyer: "

+oi.user().getname()+" , Seller: "+oj.user().getname()+"\n");

if (oj.quantity()>oi.quantity())//decrement oj.quantity() in order list

{

oj.updateqty(-oi.quantity());

}

/\*

if (oi.quantity()==oj.quantity())// remove oi from orderlist

{

//orderList.remove(j);

ai.add(j);

}\*/

//orderList.remove(i);//remove oi from order list

ai.add(i);

ai.add(j);

}

else if(oi.Type().equals("sell")&&oj.Type().equals("buy")&&oi.comp().getTicker().equals(oj.comp().getTicker())&&(oi.Rate()<=oj.Rate())

&&(oj.quantity()<oi.quantity()))

{

//setting values of o,h,l for the company

if (oi.comp().getTransactionNo()==0)

{

oi.comp().setOPrice(oi.Rate());

oi.comp().setHPrice(oi.Rate());

oi.comp().setLPrice(oi.Rate());

}

else

{

//set H and L price

if (oi.comp().getHPrice()<oi.Rate())

{

oi.comp().setHPrice(oi.Rate());

}

if (oi.comp().getLPrice()>oi.Rate())

{

oi.comp().setLPrice(oi.Rate());

}

}

oi.comp().incrementTransactionNo();

oi.comp().setLastPrice(oi.Rate());

oi.user().setfundss(oi.Rate()\*oj.quantity());

oj.user().setfundsb(oi.Rate()\*oj.quantity());

//have to change the stocks

ArrayList<entity\_stock> esi=oi.user().getHolding();

ArrayList<entity\_stock> esj=oj.user().getHolding();

int found=0;

for (entity\_stock es : esj)

{

if (es.getTicker().equals(oj.comp().getTicker()))

{

es.setqty(es.getqty()+oj.quantity());

found=1;

break;

}

}

if (found==0)

{

entity\_stock es\_new= new entity\_stock(oj.comp().getTicker(),oj.quantity());

esj.add(es\_new);

}

for (entity\_stock es1 : esi)

{

if (es1.getTicker().equals(oi.comp().getTicker()))

es1.setqty(es1.getqty()-oj.quantity());

}

writer.write(oi.quantity()+" qty of scrip:"+oi.comp().getTicker()+" sold at the rate of INR "+oi.Rate()+" ; Buyer: "

+oj.user().getname()+" , Seller: "+oi.user().getname()+"\n");

if (oi.quantity()>oj.quantity())//decrement oi.quantity() in order list

{

oi.updateqty(-oj.quantity());

}

/\*

if (oi.quantity()==oj.quantity())// remove oi from orderlist

{

orderList.remove(i);

ai.add(i);

}\*/

//orderList.remove(j);//remove oj from order list

ai.add(j);

ai.add(i);

}

else if(oi.Type().equals("sell")&&oj.Type().equals("buy")&&oi.comp().getTicker().equals(oj.comp().getTicker())&&(oi.Rate()<=oj.Rate())&&(oj.quantity()>=oi.quantity()))

{

//setting values of o,h,l for the company

if (oi.comp().getTransactionNo()==0)

{

oi.comp().setOPrice(oi.Rate());

oi.comp().setHPrice(oi.Rate());

oi.comp().setLPrice(oi.Rate());

}

else

{

//set H and L price

if (oi.comp().getHPrice()<oi.Rate())

{

oi.comp().setHPrice(oi.Rate());

}

if (oi.comp().getLPrice()>oi.Rate())

{

oi.comp().setLPrice(oi.Rate());

}

}

oi.comp().incrementTransactionNo();

oi.comp().setLastPrice(oi.Rate());

oi.user().setfundss(oi.Rate()\*oi.quantity());

oj.user().setfundsb(oi.Rate()\*oi.quantity());

//have to change the stocks

ArrayList<entity\_stock> esi=oi.user().getHolding();

ArrayList<entity\_stock> esj=oj.user().getHolding();

int found=0;

for (entity\_stock es : esj)

{

if (es.getTicker().equals(oj.comp().getTicker()))

{

es.setqty(es.getqty()+oi.quantity());

found=1;

break;

}

}

if (found==0)

{

entity\_stock es\_new= new entity\_stock(oj.comp().getTicker(),oi.quantity());

esj.add(es\_new);

}

for (entity\_stock es1 : esi)

{

if (es1.getTicker().equals(oi.comp().getTicker()))

es1.setqty(es1.getqty()-oi.quantity());

}

writer.write(oi.quantity()+" qty of scrip:"+oi.comp().getTicker()+" sold at the rate of INR "+oi.Rate()+" ; Buyer: "

+oj.user().getname()+" , Seller: "+oi.user().getname()+"\n");

if (oj.quantity()>oi.quantity())//decrement oj.quantity() in order list

{

oj.updateqty(-oi.quantity());

}

/\*

if (oi.quantity()==oj.quantity())// remove oi from orderlist

{

//orderList.remove(j);

ai.add(j);

}\*/

//orderList.remove(i);//remove oi from order list

ai.add(i);

ai.add(j);

}

}

}

for(Integer a: ai)

{

orderList.remove(a);

}

}

public static void Show\_scrip(String t) throws IOException

{

writer.write("Scrips listed in Sector: "+t+"\n");

for(PubListCo a: ListofCo)

{

if(a.getSector().equals(t))

{

writer.write(a.getTicker()+", OHLC = <"+a.getOPrice()+", "+a.getHPrice()+", "+a.getLPrice()+", "+a.getCPrice()+">\n");

}

}

}

public static void marketOpen() throws IOException

{

double o,h;

String str, addplc;

String[] arr = new String[100];

orderList = new ArrayList<>();

readInputFile(inputFile);

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

{

if(!(str.isEmpty())&&(str.length()>=4))//extracting add commands from file

{

if (str.length()>=7)

addplc = str.substring(0,7);

else addplc = str.substring(0,4); //for exit

switch (addplc)

{

case "Place o":

arr = str.split(":|,| ");

o = Integer.parseInt(arr[16]);

h = Double.parseDouble(arr[20]);

entity ent1 = getEntity(arr[5]);//getting null here as the Listofentity has not yet been fully formed restart the loop

PubListCo plc = getCompany(arr[13]);

order Order = new order(ent1,arr[9],plc,(int)o,h);

//validate the order, also allowing short selling

Order.validate();

if(Order.status().equals("rejected"))

writer.write("Order "+Order.status()+ " for user: "+arr[5]+", type: "+arr[9]+", scrip: "+arr[13]+", qty:"+(int)o+", rate: "+h+" reason: "+Order.reason()+"\n");

else

{

writer.write("Order "+Order.status()+ " for user: "+arr[5]+", type: "+arr[9]+", scrip: "+arr[13]+", qty:"+(int)o+", rate: "+h+"\n");

orderList.add(Order);

}

writer.write(dashes);

break;

case "Show Or":

for(order a: orderList)

{

writer.write(a.type+" order "+a.comp().getTicker()+" : "+a.quantity()+" at "+a.Rate()+"\n");

}

writer.write(dashes);

break;

case "Execute":

execute();

writer.write(dashes);

break;

case "Show se":

arr = str.split(":| ");

//System.out.println("arr[0] "+arr[0]+" arr[1] "+arr[1]+" arr[2] "+arr[2]+" arr[3] "+arr[3]);

Show\_scrip(arr[3]);

writer.write(dashes);

break;

case "Delete ":

// String ch = str.substring(8,9);

//PubListCo p;

arr = str.split(":| ");

//System.out.println("arr[0] "+arr[0]+" arr[1] "+arr[1]+" arr[2] "+arr[2]+" arr[3] "+arr[3]);

if(arr[1].equals("scrip"))

{

//System.out.println("Inside scrip del");

Iterator<PubListCo> itr = ListofCo.iterator();

while (itr.hasNext())

{

//System.out.println("ticker:"+a.getTicker()+" arr[3]"+arr[3]);

PubListCo a = itr.next();

if(a.getTicker().equals(arr[3]))

{

writer.write("Deleted scrip: "+arr[3]+"\n");

//System.out.println("Inside scrip for each");

//p = a;

itr.remove();

Nse.deregister(ListofCo);

Bse.deregister(ListofCo);

}

}

//ListofCo.remove(p);

}

else if(arr[1].equals("User"))

{

Iterator<entity> itr1 = Listofentity.iterator();

while (itr1.hasNext())

{

entity a = itr1.next();

if(a.getname().equals(arr[3]))

{

writer.write("Deleted User: "+arr[3]+"\n");

itr1.remove();

}

}

}

writer.write(dashes);

break;

case "Show Sc":

//update closing price for companies

for (PubListCo co:ListofCo)

{

co.setCPrice(co.getLastPrice());

}

writer.write("Scrips:\n");

for (PubListCo a: ListofCo)

{

writer.write("scrip: "+a.getTicker()+", sector: "+a.getSector()+", O:"+a.getOPrice()+", H:"+a.getHPrice()+

", L:"+a.getLPrice()+", C:"+a.getCPrice()+"\n");

}

writer.write(dashes);

break;

case "Show Us":

writer.write("Users:\n");

for (entity e: Listofentity)

{

writer.write("user: "+e.getname()+", funds: "+e.getfunds()+", holding: ");

writer.write(e.DisplayStockList()+"\n");

}

writer.write(dashes);

break;

default:

if(str.equals("Exit"))

{

writer.write("Market Closed.\n");

}

break;

}

}

}

br.close();

}

public static void regCoStockEx() throws IOException

{

String str;

for(PubListCo a : ListofCo)

{

str = a.getTicker();

Bse.register(a);

writer.write(a.getTicker()+" registered in BSE\n");

Nse.register(a);

writer.write(a.getTicker()+" registered in NSE\n");

}

}

public String ListOpenOrders(String userName)

{

String str ="";

for(order a: orderList)

{

if (a.user().getname().equals(userName))

{

str += a.type+" order "+a.comp().getTicker()+" : "+a.quantity()+" at "+a.Rate();

}

}

return str;

}

public static void processInpFile() throws IOException

{

//Open the input file for reading and creare a bufferedreader object

readInputFile(inputFile);

//Creating output file

writer = new FileWriter(outputFile);

writer.write("Opened the writing file\n");

writer.write(dashes);

writer.write("Reading the input file ...\n");

writer.write(dashes);

//Creating stock exchange objects

Bse = new StockEx("BSE");

writer.write("BSE created\n");

Nse = new StockEx("NSE");

writer.write("NSE created\n");

writer.write(dashes);

//Executing add scrip and add user commands from the input file

writer.write("Executing add scrip and add user commands from the input file\n");

writer.write(dashes);

add();

writer.write(dashes);

//Registering the public listed companies in stock exchanges

writer.write("Registering the public listed companies in stock exchanges\n");

writer.write(dashes);

regCoStockEx();

writer.write(dashes);

//Noting the event of market opening

writer.write("Market opens\n");

writer.write(dashes);

marketOpen();

writer.write(dashes);

//Closing the output file

writer.close();

br.close();

}

public static void processInpFile1() throws IOException, ParseException

{

//Open the input file for reading and creare a bufferedreader object

readInputFile(inputFile1);

//Creating output file

writer = new FileWriter(outputFile1);

writer.write("Opened the writing file for historical input data\n");

writer.write(dashes);

String str;

String[] arr = new String[100];

DayWiseListofCo= new ArrayList<>();

str = br.readLine(); //read heading of the csv file

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

{

//System.out.println(str);

if (!str.isEmpty())

{

arr = str.split(",");

//System.out.println("Ticker= "+arr[0]+" Date= "+arr[1]+" Prev Close= "+arr[2]+" Open Price= "+arr[3]+" High Price"+arr[4]+" Low Price= "+arr[5]+" Last Price= "+arr[6]+" Close Price= "+arr[7]);

PubListCo co = new PubListCo(arr[0],"-",Double.parseDouble(arr[3]),Double.parseDouble(arr[4]),Double.parseDouble(arr[5]),Double.parseDouble(arr[7]));

co.setLastPrice(Double.parseDouble(arr[6]));

co.setDate(new SimpleDateFormat("dd-MMM-yy").parse(arr[1]));

co.setPrevClosePrice(Double.parseDouble(arr[2]));

DayWiseListofCo.add(co);

}

//process the file data

}

writer.write(arr[0]+":\n");

double avgClosePrice=0, maxClosePrice=0, minClosePrice=0, maxDrawdown, maxRet=0, maxRetPer,openPriceDay1=0;

int noOfDays=0;

for (PubListCo a: DayWiseListofCo)

{

noOfDays++;

if (noOfDays==1)

openPriceDay1=a.getOPrice();

avgClosePrice += a.getCPrice();

if (a.getCPrice()>maxClosePrice)

maxClosePrice= a.getCPrice();

if (a.getCPrice()<minClosePrice)

minClosePrice= a.getCPrice();

maxRet += Math.abs(a.getCPrice()-a.getOPrice());

}

avgClosePrice /= noOfDays;

maxDrawdown=maxClosePrice-minClosePrice;

maxRetPer=100\*maxRet/openPriceDay1;

writer.write("Number of Days= "+noOfDays+"\n");

writer.write("Average Price of the stock over "+ noOfDays+" days= "+avgClosePrice+"\n");

writer.write("Max. Draw down of the stock over "+ noOfDays+" days= "+maxDrawdown+"\n");

writer.write("Max. return potential of the stock over "+ noOfDays+" days= "+maxRet+"\n");

writer.write("Max. return potential percentage of the stock over "+ noOfDays+" days= "+maxRetPer+"\n");

writer.write(dashes);

writer.close();

br.close();

}

public static void main(String[] args) throws IOException, ParseException

{

processInpFile(); // for first input file

//System.out.println("Starting 2nd file");

System.out.println("The validate function allows short selling");

System.out.println("DisplayStockList is also implemented and displays the list of stocks owned by the person");

System.out.println("We have also implemented ListOpenOrders wich shows the list of open orders for a perticular user");

processInpFile1();// for second input file

}

}









