//Xiaqin Qiu

public abstract class Coordinate{

public double x=0, y=0;

public Coordinate(){

x=0;

y=0;

}

public Coordinate(double xt, double yt){

x=xt;

y=yt;

}

public void setX(double xt){

x=xt;

}

public void setY(double yt){

y=yt;

}

public double getX(){

return x;

}

public double getY(){

return y;

}

public String toString(){

String str="X is: "+x+" Y is: "+y;

return str;

}

}

//Xiaqin Qiu

public class Charge extends Coordinate{

double Q=0;

int ID;

static int staticID;

public Charge(){

super();

Q=0;

staticID++;

ID=staticID;

}

public Charge(double xt, double yt, double qt){

super(xt, yt);

Q=qt;

staticID++;

ID=staticID;

}

public void setQ(double qt){

Q=qt;

}

public void setID(int idt){

ID=idt;

}

public void setStaticID(int sid){

staticID=sid;

}

public double getQ(){

return Q;

}

public int getID(){

return ID;

}

public int getStaticID(){

return staticID;

}

public String toString(){

String str="The charge is: "+Q+"\nThe charge coordinates are: "+x+" "+y;

return str;

}

}

//Xiaqin Qiu

import java.io.\*;

public class Point extends Coordinate implements Formula, Serializable{

String name;

int ID;

static int staticID;

double ResultantField=0;

public Point(){

super();

name="";

staticID++;

ID=staticID;

}

public Point(double xt, double yt, String nm){

super(xt, yt);

name=nm;

staticID++;

ID=staticID;

}

public void setName(String nm){

name=nm;

}

public String getName(){

return name;

}

public void setID(int idt){

ID=idt;

}

public void setStaticID(int sid){

staticID=sid;

}

public int getID(){

return ID;

}

public int getStaticID(){

return staticID;

}

public void setResultantField(double rf){

ResultantField=rf;

}

public double getResultantField(){

return ResultantField;

}

public double distance(Charge c){

double d=Math.sqrt(Math.pow((c.getX()-x),2)+Math.pow((c.getY()-y),2));

return d;

}

public double cos(Charge c){

double cosans=0;

try{

if(distance(c)==0){

throw new CoincideException();

}

cosans=(x-c.getX())/distance(c);

}

catch(CoincideException ce){

System.out.println(ce.getMessage());

System.out.println(ce.toString());

}

return cosans;

}

public double sin(Charge c){

double sinans=0;

try{

if(distance(c)==0){

throw new CoincideException();

}

sinans=(y-c.getY())/distance(c);

}

catch(CoincideException ce){

System.out.println(ce.getMessage());

System.out.println(ce.toString());

}

return sinans;

}

public String toString(){

String str="The x coordinate is x="+this.getX()+" The y coordinate is y="+this.getY()+" The point is: "+name+"\nThe ResultantField is: "+ResultantField;

return str;

}

}

//Xiaqin Qiu

import java.util.\*;

import java.text.\*;

public class E{

public final long k=8990000000L;

public double ESingle(Charge c, Point p){

DecimalFormat df=new DecimalFormat("0.00");

double esTemp=0;

esTemp=k\*c.getQ()/(Math.pow(p.distance(c),2));

String st=df.format(esTemp);

double es=Double.parseDouble(st);

return es;

}

public double Ex(Charge c, Point p){

DecimalFormat df=new DecimalFormat("0.00");

double Exc=0;

Exc=p.cos(c)\*ESingle(c,p);

String st=df.format(Exc);

double Excreturn=Double.parseDouble(st);

return Excreturn;

}

public double Ey(Charge c, Point p){

DecimalFormat df=new DecimalFormat("0.00");

double Eyc=0;

Eyc=p.sin(c)\*ESingle(c,p);

String st=df.format(Eyc);

double Eycreturn=Double.parseDouble(st);

return Eycreturn;

}

public String ExSign(Charge c, Point p){

if(Ex(c,p)>0)

return "Positive";

else if(Ex(c,p)==0)

return "No direction";

else

return "Negative";

}

public String EySign(Charge c, Point p){

if(Ey(c,p)>=0)

return "Positive";

else if(Ey(c,p)==0)

return "No direction";

else

return "Negative";

}

public double EResultant(ArrayList<Charge> alc, Point p){

double er;

double ExR=0, EyR=0;

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

ExR=ExR+Ex(alc.get(i),p);

EyR=EyR+Ey(alc.get(i),p);

}

er=Math.sqrt(Math.pow(ExR,2)+Math.pow(EyR,2));

return er;

}

}

//Xiaqin Qiu

public class CoincideException extends Exception{

public CoincideException(){

super();

}

public String getMessage(){

return "The point is on the charge position";

}

public String toString(){

return "CoincideException occurred";

}

}

//Xiaqin Qiu

public interface Formula{

public double distance(Charge c);

public double cos(Charge c);

public double sin(Charge c);

}

//Xiaqin Qiu

import java.util.\*;

import java.io.\*;

public class Client{

public static void main(String[] args){

double ix=0, iy=0, iq=0;

String Name="";

ArrayList<Charge> alCharge=new ArrayList<Charge>();

ArrayList<Point> alPoint=new ArrayList<Point>();

E eField=new E();

boolean flag=true;

try{

FileOutputStream fos=new FileOutputStream("PointElectricFields.ser");

ObjectOutputStream oos=new ObjectOutputStream(fos);

Scanner inScan=new Scanner(System.in);

System.out.println("Please input the filename:");

String input=inScan.next();

FileInputStream myFile=new FileInputStream(input);

BufferedReader fileReader=new BufferedReader(new InputStreamReader(myFile));

String line="";

while((line=fileReader.readLine())!=null){

StringTokenizer strtkColon=new StringTokenizer(line,":");

String initial=strtkColon.nextToken();

String rest=strtkColon.nextToken();

if(initial.equals("Q")){

StringTokenizer strtkComma=new StringTokenizer(rest, ",");

String strtk0=strtkComma.nextToken();

String strtk1=strtkComma.nextToken();

String strtk2=strtkComma.nextToken();

ix=Double.parseDouble(strtk0);

iy=Double.parseDouble(strtk1);

iq=Double.parseDouble(strtk2);

Charge cTemp=new Charge(ix,iy,iq);

alCharge.add(cTemp);

}

else if(initial.equals("P")){

StringTokenizer strtkComma=new StringTokenizer(rest, ",");

String strtk0=strtkComma.nextToken();

String strtk1=strtkComma.nextToken();

String strtk2=strtkComma.nextToken();

ix=Double.parseDouble(strtk1);

iy=Double.parseDouble(strtk2);

Point pTemp=new Point(ix,iy,strtk0);

alPoint.add(pTemp);

}

}

myFile.close();

fileReader.close();

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

if(alPoint.size()>1 && i>0){

System.out.println("--------------NEXT POINT--------------");

}

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

System.out.println("The magnitude of electric field at point: "+alPoint.get(i).getName()+" due to charge point #"+alCharge.get(j).getID()+" is "+eField.ESingle(alCharge.get(j), alPoint.get(i)));

System.out.println("the point coordinates are: "+alPoint.get(i).getX()+" "+alPoint.get(i).getY());

System.out.println("The charge is: "+alCharge.get(j).getQ());

System.out.println("The charge coordinates are: "+alCharge.get(j).getX()+" "+alCharge.get(j).getY());

System.out.println("cosine of the angle= "+alPoint.get(i).cos(alCharge.get(j)));

System.out.println("The value of the x component of the Electric field at point: "+alPoint.get(i).getName()+" due to charge point #"+alCharge.get(j).getID()+" is "+eField.Ex(alCharge.get(j), alPoint.get(i)));

System.out.println("sine of the angle= "+alPoint.get(i).sin(alCharge.get(j)));

System.out.println("The value of the y component of the Electric field at point: "+alPoint.get(i).getName()+" due to charge point #"+alCharge.get(j).getID()+" is "+eField.Ey(alCharge.get(j), alPoint.get(i)));

System.out.println("The direction of the x-component is "+eField.ExSign(alCharge.get(j), alPoint.get(i))+" and the direction of the y component is "+eField.EySign(alCharge.get(j), alPoint.get(i)));

}

System.out.println("The magnitude of the resultant field is: "+eField.EResultant(alCharge, alPoint.get(i))+" for point: "+alPoint.get(i).getName());

alPoint.get(i).setResultantField(eField.EResultant(alCharge, alPoint.get(i)));

}

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

oos.writeObject(alPoint.get(i));

}

oos.close();

System.out.println("\nDo you want to have the PointElectricFields.ser binary coded file read? yes to read");

Scanner scanSer=new Scanner(System.in);

String temp=scanSer.nextLine();

FileInputStream fis=null;

ObjectInputStream ois=null;

try{

if(temp.equals("yes")){

fis=new FileInputStream("PointElectricFields.ser");

ois=new ObjectInputStream(fis);

while(true){

Point pTemp=(Point)ois.readObject();

System.out.println(pTemp);

}

}

}

catch(EOFException eof){

System.out.println("Closing the file");

}

catch(ClassNotFoundException cnf){

System.out.println(cnf.getMessage());

}

}

catch(IOException ioe){

System.out.println("input file error");

System.out.println(ioe.getMessage());

}

catch(NullPointerException npe){

System.out.println(npe.getMessage());

}

catch(IllegalArgumentException iae){

System.out.println(iae.getMessage());

}

}

}