```java
package DataModel;

import java.io.*;
import java.util.*;


public class History {
    public String name;
    public int InitialNumberOfOperations;
    private int InitialNumberOfDataStructures;
    private int Operations[] [];
    private int DataStructures[] [];
    private int Dates[][];
    public int DiffDates[][];
    File file;
    private int counter=0;
    ArrayList names;
    public int idCounter=0;
    private String line,lastline;
    BufferedReader reader=null;
    public Report report;
    public History(File file,ArrayList names,int counter2){
        this.names=names;
        this.file=file;
        report=new Report(counter2);
        try{
            reader=new BufferedReader(new FileReader(file));
        }
        catch(FileNotFoundException e){
            System.err.print("error opening file");
        }
        try{
            line=reader.readLine();
            int counter=0;
            while (counter!=2){
                if(line==null){
                    counter++;
                    continue;
                }
                lastline=line;
                line=reader.readLine();
            }
            StringTokenizer st=new StringTokenizer(lastline,"line: ;");
            String token=st.nextToken();
            idCounter=Integer.parseInt(token);
            Operations=new int[idCounter] [3];
            DataStructures=new int[idCounter] [3];
            Dates=new int[idCounter][3];
            try{
                reader=new BufferedReader(new FileReader(this.file));
            }
            catch(FileNotFoundException e){
                System.err.print("error opening file");
            }
            try{
                name=reader.readLine();
                if(names.contains(name)){
                    System.out.println("Εχει καταχωρηθεί ήδη το αρχείο ιστορικού.");
                }
                else{
                    report.insertName(name);
                    line=reader.readLine();
                    StringTokenizer stO=new StringTokenizer(line,"Initial Number of Operations;");
                    InitialNumberOfOperations=Integer.parseInt(stO.nextToken());
                    line=reader.readLine();
                    StringTokenizer stDS=new StringTokenizer(line,"Initial Number of Data Structures;");
                    InitialNumberOfDataStructures=Integer.parseInt(stDS.nextToken());
                    line=reader.readLine();
                    line=reader.readLine();
                    line=reader.readLine();
                    StringTokenizer stT=new StringTokenizer(line,"line: ; ");
                    String date=null;
                    String token2=stT.nextToken();
                    for(int i=0;i<idCounter;i++){
                        if(i!=0){
                            token2=stT.nextToken();
                        }
                        date=stT.nextToken();
                        stT=new StringTokenizer(date,"/");
                        for(int j=0;j<3;j++){
                            Dates[i][j]=Integer.parseInt(stT.nextToken());
                        }
                        stT=new StringTokenizer(line,"line: ; ");
                        int idFirst=Integer.parseInt(stT.nextToken());
                        date=stT.nextToken();
                        for(int j=0;j<3;j++){
                            Operations[i][j]=Integer.parseInt(stT.nextToken());
                        }
                        for(int j=0;j<3;j++){
                            DataStructures[i][j]=Integer.parseInt(stT.nextToken());
                        }
                        line=reader.readLine();
                        if(i!=idCounter-1){
                            stT=new StringTokenizer(line,"line: ;");
                        }
                    }
                }
            }
            catch(IOException e){
                System.err.println("error reading line " + counter);
            }
        }
        catch(IOException e){
            System.err.println("error reading line " + counter);
        }
    }

    public int returnNumberOfOperations(int n){
        int numOfOper=InitialNumberOfOperations;
        for(int i=0;i<n+1;i++){
            numOfOper=numOfOper+Operations[i][0]-Operations[i][1];
        }
        return numOfOper;
    }

    public int returnGrowthOperations(int current){
        if(current==0){
            return 0;
        }
        return Operations[current][0]-Operations[current][1];
    }

    public int returnOperationChanges(int n){
        return Operations[n][0]+Operations[n][1]+Operations[n][2];
    }

    public int returnDataStructuresChanges(int n){
        return DataStructures[n][0]+DataStructures[n][1]+DataStructures[n][2];
    }

    public double returnComplexityO(int n){
        double numDAndU=0,numA=0;
        numA=numA+Operations[n][0];
        numDAndU=numDAndU+Operations[n][1]+Operations[n][2];
        if(numA==0 || numDAndU==0){
            return 0;
        }
        else{
            return numDAndU/numA;
        }
    }

    public double returnEmploymentRateO(int current){
        double numDAndUAndA=Operations[current][0]+Operations[current]
[1]+Operations[current][2];
        if(current==0){
            return 0;
        }
        else{
            int year=Dates[current][2]-Dates[current-1][2];
            int months=year*12;
            months=months+Dates[current][1]-Dates[current-1][1];
            int days=months*30;
            days=days+Dates[current][0]-Dates[current-1][0];
            return numDAndUAndA/days;
        }
    }

    public int[][] retrunNumberOfOpPerYear(){
        int c=0;
        for(int i=0;i<Dates.length;i++){
            if(i==Dates.length-1){
                if(Dates[i][2]!=Dates[i-1][2]){
                    c++;
                }
                break;
            }
            if(Dates[i][2]!=Dates[i+1][2]){
                c++;
            }
        }
        DiffDates=new int[c][2];
        int s=0;
        for(int i=0;i<Dates.length;i++){
            if(i==Dates.length-1){
                if(Dates[i][2]!=Dates[i-1][2]){
                    DiffDates[s][0]=Dates[i][2];
                }
                break;
            }
            if(Dates[i][2]!=Dates[i+1][2]){
                DiffDates[s++][0]=Dates[i][2];
            }
        }
        int times=0;
        for(int i=0;i<DiffDates.length;i++){
            for(int j=0;j<Dates.length;j++){
                if(DiffDates[i][0]==Dates[j][2]){
                    times++;
                }
            }
            DiffDates[i][1]=times;
            times=0;
        }
        return DiffDates;
    }

    public int returnNumberOfDataStructures(int n){
        int numOfDataSt=InitialNumberOfDataStructures;
        for(int i=0;i<n+1;i++){
            numOfDataSt=numOfDataSt+DataStructures[i][0]-DataStructures[i][1];
        }
        return numOfDataSt;
    }

    public int returnGrowthDataStructures(int current){
        if(current==0){
            return 0;
        }
        return DataStructures[current][0]-DataStructures[current][1];
    }

    public double returnComplexityDS(int n){
        double numDAndU=0,numA=0;
        numA=numA+DataStructures[n][0];
        numDAndU=numDAndU+DataStructures[n][1]+DataStructures[n][2];
        if(numA==0 || numDAndU==0){
            return 0;
        }
        else{
            return numDAndU/numA;
        }
    }

    public int returnMaintenance(int n){
        return Operations[n][1]+Operations[n][2]+DataStructures[n][1]+DataStructures[n][2];
    }

   public double returnEmploymentRateDS(int current){
       double numDAndUAndA=DataStructures[current][0]+DataStructures[current]
[1]+DataStructures[current][2];
       if(current==0){
           return 0;
       }
       else{
           int year=Dates[current][2]-Dates[current-1][2];
           int months=year*12;
           months=months+Dates[current][1]-Dates[current-1][1];
           int days=months*30;
           days=days+Dates[current][0]-Dates[current-1][0];
           return numDAndUAndA/days;
       }
    }
}
```