

## History.java

```

1 package DataModel;
2
3 import java.io.*;
4 import java.util.*;
5
6
7 public class History {
8     public String name;
9     public int InitialNumberOfOperations;
10    private int InitialNumberOfDataStructures;
11    private int Operations[] [];
12    private int DataStructures[] [];
13    private int Dates[][];
14    public int DiffDates[][];
15    File file;
16    private int counter=0;
17    ArrayList names;
18    public int idCounter=0;
19    private String line,lastline;
20    BufferedReader reader=null;
21    public Report report;
22    public History(File file,ArrayList names,int counter2){
23        this.names=names;
24        this.file=file;
25        report=new Report(counter2);
26        try{
27            reader=new BufferedReader(new FileReader(file));
28        }
29        catch(FileNotFoundException e){
30            System.err.print("error opening file");
31        }
32        try{
33            line=reader.readLine();
34            int counter=0;
35            while (counter!=2){
36                if(line==null){
37                    counter++;
38                    continue;
39                }
40                lastline=line;
41                line=reader.readLine();
42            }
43            StringTokenizer st=new StringTokenizer(lastline,"line: ");
44            String token=st.nextToken();
45            idCounter=Integer.parseInt(token);
46            Operations=new int[idCounter] [3];
47            DataStructures=new int[idCounter] [3];
48            Dates=new int[idCounter][3];
49            try{
50                reader=new BufferedReader(new FileReader(this.file));
51            }
52            catch(FileNotFoundException e){
53                System.err.print("error opening file");
54            }
55            try{
56                name=reader.readLine();
57                if(names.contains(name)){
58                    System.out.println("Έχει καταχωρηθεί ήδη το αρχείο ιστορικού.");
59                }
60                else{
61                    report.insertName(name);

```

# History.java

```

62         line=reader.readLine();
63         StringTokenizer st0=new StringTokenizer(line,"Initial Number of
Operations;");
64         InitialNumberOfOperations=Integer.parseInt(st0.nextToken());
65         line=reader.readLine();
66         StringTokenizer stDS=new StringTokenizer(line,"Initial Number of Data
Structures;");
67         InitialNumberOfDataStructures=Integer.parseInt(stDS.nextToken());
68         line=reader.readLine();
69         line=reader.readLine();
70         line=reader.readLine();
71         StringTokenizer stT=new StringTokenizer(line,"line: ; ");
72         String date=null;
73         String token2=stT.nextToken();
74         for(int i=0;i<idCounter;i++){
75             if(i!=0){
76                 token2=stT.nextToken();
77             }
78             date=stT.nextToken();
79             stT=new StringTokenizer(date,"/");
80             for(int j=0;j<3;j++){
81                 Dates[i][j]=Integer.parseInt(stT.nextToken());
82             }
83             stT=new StringTokenizer(line,"line: ; ");
84             int idFirst=Integer.parseInt(stT.nextToken());
85             date=stT.nextToken();
86             for(int j=0;j<3;j++){
87                 Operations[i][j]=Integer.parseInt(stT.nextToken());
88             }
89             for(int j=0;j<3;j++){
90                 DataStructures[i][j]=Integer.parseInt(stT.nextToken());
91             }
92             line=reader.readLine();
93             if(i!=idCounter-1){
94                 stT=new StringTokenizer(line,"line: ;");
95             }
96         }
97     }
98 }
99 catch(IOException e){
100     System.err.println("error reading line " + counter);
101 }
102 }
103 catch(IOException e){
104     System.err.println("error reading line " + counter);
105 }
106 }
107
108 public int returnNumberOfOperations(int n){
109     int numOfOper=InitialNumberOfOperations;
110     for(int i=0;i<n+1;i++){
111         numOfOper=numOfOper+Operations[i][0]-Operations[i][1];
112     }
113     return numOfOper;
114 }
115
116 public int returnGrowthOperations(int current){
117     if(current==0){
118         return 0;
119     }
120     return Operations[current][0]-Operations[current][1];

```

```

121     }
122
123     public int returnOperationChanges(int n){
124         return Operations[n][0]+Operations[n][1]+Operations[n][2];
125     }
126
127     public int returnDataStructuresChanges(int n){
128         return DataStructures[n][0]+DataStructures[n][1]+DataStructures[n][2];
129     }
130
131     public double returnComplexity0(int n){
132         double numDAndU=0,numA=0;
133         numA=numA+Operations[n][0];
134         numDAndU=numDAndU+Operations[n][1]+Operations[n][2];
135         if(numA==0 || numDAndU==0){
136             return 0;
137         }
138         else{
139             return numDAndU/numA;
140         }
141     }
142
143     public double returnEmploymentRate0(int current){
144         double numDAndUAndA=Operations[current][0]+Operations[current]
[1]+Operations[current][2];
145         if(current==0){
146             return 0;
147         }
148         else{
149             int year=Dates[current][2]-Dates[current-1][2];
150             int months=year*12;
151             months=months+Dates[current][1]-Dates[current-1][1];
152             int days=months*30;
153             days=days+Dates[current][0]-Dates[current-1][0];
154             return numDAndUAndA/days;
155         }
156     }
157
158     public int[][] retrunNumberOfOpPerYear(){
159         int c=0;
160         for(int i=0;i<Dates.length;i++){
161             if(i==Dates.length-1){
162                 if(Dates[i][2]!=Dates[i-1][2]){
163                     c++;
164                 }
165                 break;
166             }
167             if(Dates[i][2]!=Dates[i+1][2]){
168                 c++;
169             }
170         }
171         DiffDates=new int[c][2];
172         int s=0;
173         for(int i=0;i<Dates.length;i++){
174             if(i==Dates.length-1){
175                 if(Dates[i][2]!=Dates[i-1][2]){
176                     DiffDates[s][0]=Dates[i][2];
177                 }
178                 break;
179             }
180             if(Dates[i][2]!=Dates[i+1][2]){

```

# History.java

```

181         DiffDates[s++][0]=Dates[i][2];
182     }
183 }
184 int times=0;
185 for(int i=0;i<DiffDates.length;i++){
186     for(int j=0;j<Dates.length;j++){
187         if(DiffDates[i][0]==Dates[j][2]){
188             times++;
189         }
190     }
191     DiffDates[i][1]=times;
192     times=0;
193 }
194 return DiffDates;
195 }
196
197 public int returnNumberOfDataStructures(int n){
198     int numOfDataSt=InitialNumberOfDataStructures;
199     for(int i=0;i<n+1;i++){
200         numOfDataSt=numOfDataSt+DataStructures[i][0]-DataStructures[i][1];
201     }
202     return numOfDataSt;
203 }
204
205 public int returnGrowthDataStructures(int current){
206     if(current==0){
207         return 0;
208     }
209     return DataStructures[current][0]-DataStructures[current][1];
210 }
211
212 public double returnComplexityDS(int n){
213     double numDAndU=0,numA=0;
214     numA=numA+DataStructures[n][0];
215     numDAndU=numDAndU+DataStructures[n][1]+DataStructures[n][2];
216     if(numA==0 || numDAndU==0){
217         return 0;
218     }
219     else{
220         return numDAndU/numA;
221     }
222 }
223
224 public int returnMaintenance(int n){
225     return Operations[n][1]+Operations[n][2]+DataStructures[n][1]+DataStructures[n][2];
226 }
227
228 public double returnEmploymentRateDS(int current){
229     double numDAndUAndA=DataStructures[current][0]+DataStructures[current]
230 [1]+DataStructures[current][2];
231     if(current==0){
232         return 0;
233     }
234     else{
235         int year=Dates[current][2]-Dates[current-1][2];
236         int months=year*12;
237         months=months+Dates[current][1]-Dates[current-1][1];
238         int days=months*30;
239         days=days+Dates[current][0]-Dates[current-1][0];
240         return numDAndUAndA/days;
241     }

```

## History.java

```
241     }  
242 }
```