Cosine Similarity Code

```
#CosineSimilarity

import java.util.Scanner;
import java.util.HashMap;
import java.util.Map;

class similarityEstimation {
    Map<String, Integer> wordFreqEstimation(String str) {
        Map<String, Integer> freqMap = new HashMap<String, Integer>();
        Integer count = null;

        String delim = " ";
        String[] token = str.split(delim);
        String word;

        for (int i = 0; i < token.length; i++) {
            word = token[i];
            count = freqMap.get(word);

            if (count == null) {
                count = 1;
            } else {
                count++;
            }
            freqMap.put(word, count);
        }
        return freqMap;
    }

    Double cosSim(String str1, String str2) {
        Map<String, Integer> docfreq1 = wordFreqEstimation(str1);
        Map<String, Integer> docfreq2 = wordFreqEstimation(str2);
        Double cosine_similarity;
        double mul = 0.0f;
        double fr1 = 0.0f;
        double fr2 = 0.0f;

        for (String key1 : docfreq1.keySet()) {
            fr1 += Math.pow(docfreq1.get(key1), 2);
        }
        for (String key2 : docfreq2.keySet()) {
            fr2 += Math.pow(docfreq2.get(key2), 2);
        }

        for (String key1 : docfreq1.keySet()) {
            if (docfreq2.containsKey(key1)) {
                mul += docfreq1.get(key1) * docfreq2.get(key1);
            }
        }
        cosine_similarity = (Double) (mul / Math.sqrt(fr1 * fr2));
        return cosine_similarity;
    }
}

public class cosineSimilarity {
    public static void main(String[] args) {
        similarityEstimation similarityEstimationObj = new similarityEstimation();
        Map<String, Integer> fMap = similarityEstimationObj.wordFreqEstimation(
            "the best data science course in the best university");
        Map<String, Integer> fMap2 = similarityEstimationObj.wordFreqEstimation(
            "the best course in university of science");
        System.out.println(fMap);
        System.out.println(fMap2);
        Double x = similarityEstimationObj.cosSim(
            "the best data science course in the best university",
```

```
            "the best course in university of science");
        System.out.println(x);
    }
}
```

Listing 1: Cosine Similarity Implementation

Jaccard Similarity Code

```
#JaccardSimilarity

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Scanner;
import java.util.Set;

public class jaccardSimilarity {

    class wordFrequency {
        Map<String, Integer> calculate(String str) {
            Map<String, Integer> freqMap = new HashMap<String, Integer>();
            String[] token1 = str.split("\\s+");
            String word;
            Integer count = null;
            for (int i = 0; i < token1.length; i++) {
                word = token1[i];
                count = freqMap.get(word);

                if (count == null) {
                    count = 1;
                } else {
                    count++;
                }
                freqMap.put(word, count);
            }
            return freqMap;
        }
    }

    class jacSim {
        Map<String, Integer> freqMap1 = new HashMap<String, Integer>();
        Map<String, Integer> freqMap2 = new HashMap<String, Integer>();

        jacSim(String str1, String str2) {
            wordFrequency wFObj = new wordFrequency();
            freqMap1 = wFObj.calculate(str1);
            freqMap2 = wFObj.calculate(str2);
        }

        double calculate() {
            Set<String> set1 = new HashSet<String>();
            Set<String> set2 = new HashSet<String>();
            for (String key1 : freqMap1.keySet()) {
```

Listing 2: Jaccard Similarity Implementation

SMC

```
#SMC
import java.util.Scanner;
public class SMC {
```

```
        public static void main(String[] args) {
                Scanner myInput = new Scanner(System.in);
                String x = myInput.nextLine();
                String y = myInput.nextLine();
                int f01=0, f10=0, f00=0, f11=0;
                double SMC=0;
                for(int i = 0; i<x.length(); i++) {
                        if(x.charAt(i) == '0' && y.charAt(i) == '1') {
                                f01++;
                        }
                        if(x.charAt(i) == '1' && y.charAt(i) == '0') {
                                f10++;
                        }
                        if(x.charAt(i) == '0' && y.charAt(i) == '0') {
                                f00++;
                        }
                        if(x.charAt(i) == '1' && y.charAt(i) == '1') {
                                f11++;
                        }
                }
                SMC = (double)(f11+f00)*(f01+f10+f11+f00)/100;
                System.out.println(SMC);
        }
}
```

Listing 3: Cosine Similarity Implementation

FileProblem

```
import java.io.*;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
public class FileProblem {
        public static void main(String[] args) {
                String filePath1 = "D:\\Academic-Coding\\2nd-Semester\\OOP\\eclipse-
                    workplace\\LabExamPractice\\src\\oldmast.txt";
                String filePath2 = "D:\\Academic-Coding\\2nd-Semester\\OOP\\eclipse-
                    workplace\\LabExamPractice\\src\\trans.txt";
                FileProblem f = new FileProblem();
                FileMatch fileMatchObj = f.new FileMatch(filePath1, filePath2);
                fileMatchObj.read();
                String outPath = "D:\\Academic-Coding\\2nd-Semester\\OOP\\eclipse-workplace
                    \\LabExamPractice\\src\\newmast.txt";
                String errorPath = "D:\\Academic-Coding\\2nd-Semester\\OOP\\eclipse-
                    workplace\\LabExamPractice\\src\\log.txt";;
                fileMatchObj.newRecord(outPath, errorPath);
        }
        class FileMatch{
                String filePath1, filePath2;
                Map<String, Double> map = new HashMap<String, Double>();
                Map<String, Double> map2 =  new HashMap<String, Double>();
                FileMatch(String filePath1, String filePath2){
                        this.filePath1 = filePath1;
                        this.filePath2 = filePath2;
                        this.map = map;
                        this.map2 = map2;
                }
                void read(){
                        System.out.println("oldmast.txt");
                        try(BufferedReader reader = new BufferedReader(new FileReader(
                            filePath1))){
                                String line;
                                while((line = reader.readLine())!= null) {
                                        System.out.println(line);
```

```
32                              }
33                      }
34                      catch(IOException e){
35                              e.printStackTrace();
36                      }
37                      System.out.println("trans.txt");
38                      try(BufferedReader reader = new BufferedReader(new FileReader(
                            filePath2))){
39                              String line;
40                              while((line = reader.readLine())!= null) {
41                                      System.out.println(line);
42                              }
43                      }
44                      catch(IOException e){
45                              e.printStackTrace();
46                      }
47              }
48              void newRecord(String outPath, String errorPath){
49                      map = new HashMap<String, Double>();
50                      try(BufferedReader reader = new BufferedReader(new FileReader(
                            filePath1))){
51                              String line;
52                              while((line = reader.readLine())!= null) {
53                                      String[] words;
54                                      words = line.split("\\s+");
55                                      map.put(words[0], Double.parseDouble(words[2]));

56                              }
57                      }
58                      catch(IOException e){
59                              e.printStackTrace();
60                      }
61                      map2 = new HashMap<String, Double>();
62                      Map<String, Double> mapOut = new HashMap<String, Double>();
63                      Map<String, Double> mapLog = new HashMap<String, Double>();
64                      try(BufferedReader reader = new BufferedReader(new FileReader(
                            filePath2))){
65                              String line;
66                              while((line = reader.readLine())!= null) {
67                                      String[] words;
68                                      words = line.split("\\s+");
69                                      if(!map2.containsKey(words[0])) {
70                                              map2.put(words[0], Double.parseDouble(words
                                                    [1]));
71                                      }
72                                      else {
73                                              map2.put(words[0], map2.get(words[0])+Double
                                                    .parseDouble(words[1]));
74                                      }
75                              }
76                      }
77                      catch(IOException e){
78                              e.printStackTrace();
79                      }
80                      for(String id: map2.keySet()) {
81                              if(map.containsKey(id)) {
82                                      Double balance1 = map.get(id);
83                                      Double balance2 = map2.get(id);
84                                      Double balance = balance1 + balance2;
85                                      mapOut.put(id,balance);
86                              }
87                              else if(!map.containsKey(id)) {
88                                      mapLog.put(id, map2.get(id));
89                              }
90                      }
91                      for(String id: map.keySet()) {
92                              if(!map2.containsKey(id)) {
93                                      mapOut.put(id, map.get(id));
```

```
                                }
                        }
                        try(BufferedWriter br = new BufferedWriter(new FileWriter(outPath))) {
                                for(String id: mapOut.keySet()) {
                                        br.write(id+" "+mapOut.get(id));
                                        br.newLine();
                                }
                        }
                        catch(IOException e){
                                e.printStackTrace();
                        }
                        try(BufferedWriter br = new BufferedWriter(new FileWriter(errorPath))){
                                for(String id: mapLog.keySet()) {
                                        br.write("Unmatched transaction for A/C no. "+id);
                                        br.newLine();
                                }
                        }
                        catch(IOException e){
                                e.printStackTrace();
                        }
                }
        }
}
```

Listing 4: Cosine Similarity Implementation

V1OOP:One(CourseStudents)

```
package V1OOPProgrammingLab;
import java.io.*;
import java.util.*;
public class OneV1 {
        class course{
                String courseName;
                String[] stdIDs;
                Map<String, String[]> courseMap = new HashMap<String, String[]>();
                course() {
                        try(BufferedReader reader = new BufferedReader(new FileReader("D:\\
                            Academic-Coding\\2nd-Semester\\OOP\\eclipse-workplace\\
                            LabExamPractice\\src\\V1OOPProgrammingLab\\course.txt"))){
                                String line;
                                int i = 0;
                                while((line=reader.readLine())!= null) {
                                        String[] elements = line.split("\t");
                                        this.courseName = elements[0];
                                        this.stdIDs = elements[1].split(", ");
                                        i++;
                                }
                                courseMap.put(courseName, stdIDs);
                        }
                        catch(IOException e){
                                e.printStackTrace();
                        }
                }
                void readStds(String courseName) {
                        String[] stds = courseMap.get(courseName);
                        for(int i = 0; i < stds.length; i++) {
                                System.out.println(stds[i]);
                        }
                }
                void readAll() {
                        for(String cN: courseMap.keySet()) {
                                String[] stds = courseMap.get(cN);
```

```
                                System.out.print(cN + " : ");
                                for(int i = 0; i < stds.length; i++) {
                                        System.out.print(stds[i]);
                                        if(i < stds.length - 1) {
                                                System.out.print(", ");
                                        }
                                }
                        }
                }
        }
        public static void main(String[] args) {
                OneV1 oneObj = new OneV1();
                course cObj = oneObj.new course();
                cObj.readStds("CSE312");
                cObj.readAll();
        }
}
```

Listing 5: Cosine Similarity Implementation

Cosine Similarity Code

Listing 6: Cosine Similarity Implementation

Cosine Similarity Code

Listing 7: Cosine Similarity Implementation

Cosine Similarity Code

Listing 8: Cosine Similarity Implementation

Cosine Similarity Code

Listing 9: Cosine Similarity Implementation

Cosine Similarity Code

Listing 10: Cosine Similarity Implementation

Cosine Similarity Code

Listing 11: Cosine Similarity Implementation

Cosine Similarity Code

Listing 12: Cosine Similarity Implementation

Cosine Similarity Code

Listing 13: Cosine Similarity Implementation

Cosine Similarity Code

Listing 14: Cosine Similarity Implementation

Cosine Similarity Code

Listing 15: Cosine Similarity Implementation

Cosine Similarity Code

Listing 16: Cosine Similarity Implementation

Cosine Similarity Code

Listing 17: Cosine Similarity Implementation

Cosine Similarity Code

Listing 18: Cosine Similarity Implementation

Cosine Similarity Code

Listing 19: Cosine Similarity Implementation

Cosine Similarity Code

Listing 20: Cosine Similarity Implementation