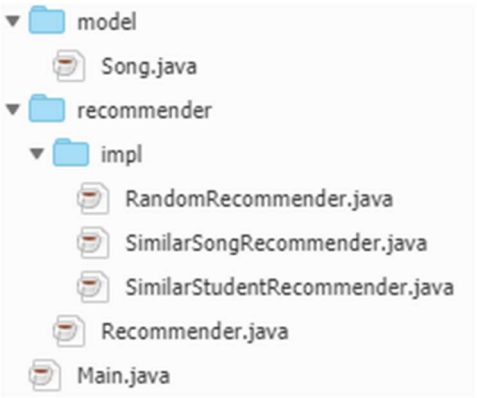
객체지향 프로그래밍 project Report

user-based recommend system

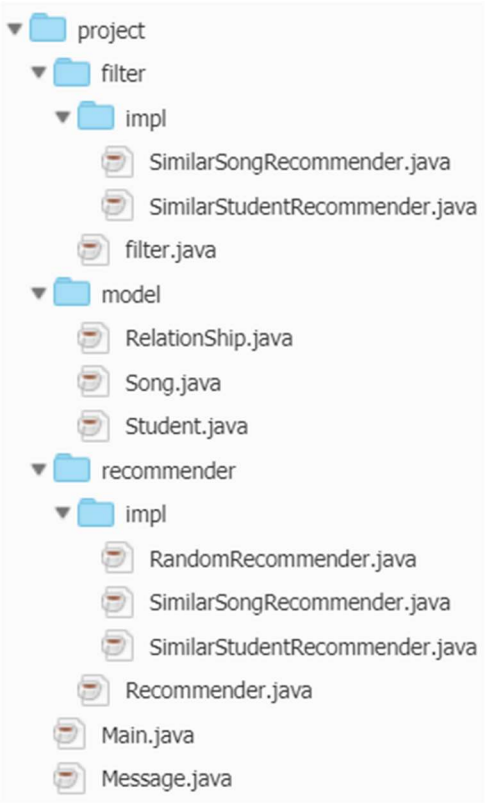
20182175 양민주

<목차>

1. 프로그램 구조



<기존구조>



<바뀐구조>

- 바뀐 부분

기존의 구조와 차이점은 유사도를 검사하여 일정 유사도 아래의 데이터들을 걸러주는 filter, 유저의 정보를 담은 객체를 생성하는 Student 클래스, 이와 Song 의 관계를 나타내는 RelationShip, 결과를 출력해주는 Message 클래스를 추가하였다.

- 작동방식

Main 을 실행하면 SetupMain 에서 파일을 읽어 들인 이후, 유저 ID 와 노래의 정보를 분리해서 Song 클래스의 객체들과 Student 객체들을 생성한후, 이들의 관계를 RelationShip 에서 노래와 노래와의 관계 또는 유저와 유저와의 관계를 벡터 값으로 수치화 한 후, filter 에서 코사인 유사도를 이용하여 유사도가 80%이상인 song 들만 걸러낸 후, 걸러낸 정보들을 가지고 recommender 에서 유사도정보를 담은 통계와 추천된 노래를 출력한다.

2. 구현해야 하는 부분(평가 기준)

- 생성자

```
33      public Main(String filePath) {
34          setUpMain(filePath);
35          message = new Message();
36          relationShip = RelationShip.getInstance();
37      }
```

생성자에서는 Main 에서 필요한 멤버변수들의 값을 할당한다.

- setupMain

```
39      private static void setUpMain(String filePath){
40          File inputFile = new File(filePath);
41
42          HashMap<String,Student> studentsIds = new HashMap<>();
43          List<Song> songs = new ArrayList<>();
44
45          try (BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(inputFile), "UTF-8"))) {
46              for (String line; (line = br.readLine()) != null;) {
47                  String[] fieldValues = line.split(",(?=([^\"]*\"[^\"]*\")*[^\"]*$)", -1);
48
49                  Song song = extractSongs(Arrays.copyOfRange(fieldValues, SONG_INFO_START_IDX, fieldValues.length));
50                  songs.add(song);
51                  extractStudentsIds(fieldValues[STUDENT_ID_IDX],studentsIds, song);
52              }
53          } catch (Exception e) {
54              e.printStackTrace();
55          }
56          makeRelationShip(studentsIds, songs);
57      }
58      /*
59       * From the input file path that contains information about the song
60       * preference list, it should extract unique students IDs and it should
61       * return it as a String array. There are 36 unique student IDs and you
62       * should return only the unique IDs.
63       */
64      private static void makeRelationShip(HashMap<String,Student> studentsIds, List<Song> songs){
65          RelationShip relationShip =RelationShip.getInstance();
66          Song[] arrSongs = songs.toArray(new Song[songs.size()]);
67          Student[] arrStudents = studentsIds.values().toArray(new Student[studentsIds.size()]);
68          relationShip.init(arrSongs,arrStudents);
69      }
70
71      private static void extractStudentsIds(String id, HashMap<String,Student> studentsIds, Song song) {
72          if(studentsIds.containsKey(id)){
73              studentsIds.get(id).getMySongs().add(song);
74          }else{
75              studentsIds.put(id, new Student(id, song));
76          }
77      }
78
79      private static Song extractSongs(String[] songInfo){
80          Song song = new Song(songInfo);
81          return song;
82      }
```

SetupMain 에서는 파일을 읽어서 song 객체와 Student 객체를 생성하고, 이 객체들 간의 관계를 형성, 관리하는 Relation 객체를 생성해준다. extractStudentsIds()는 고유한 유저의 ID 를 추출하는 메소드인데, HashMap 의 Key 값은 고유하다는 점을 이용하여 유저의 ID 를 추출함과 동시에 학생 객체를 만들어 주었다. 기존의 구조는 recommender 의 메소드인 buildSongObjects()에서 다시한번 파일을 읽어서 song 객체를 생성하지만, Main 에서 파일을 한 번 읽고 동시에 학생객체와 song 객체를 만들어 냄으로써, 파일을 여러 번 읽지

않게 하였다. 또한 모든 song 객체를 가지고 있는 RelationShip 객체를 만들어 recommender 에서 buildSongObjects()를 대체하였다.

- Song 클래스 구현

```java
public class Song {


    private static String[] fieldNames = {"title", "singer", "genre", "releaseYear",
            "albumName", "songWriter", "producer", "issuingCountry", "language", "playingTime"};

    private String title;
    private HashMap<String, String> songInfo;

    public Song(String[] fieldValues) {

        title = fieldValues[0];
        songInfo = new HashMap<>();

        for(int i=0; i< fieldValues.length; i++)
            songInfo.put(fieldNames[i], fieldValues[i]);
    }

    public String getTitle() {
        return title;
    }

    public HashMap<String, String> getSongInfo() {
        return songInfo;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null || getClass() != obj.getClass()) return false; // 널값이거나 song 클래스가 아니면 false.

        Song song = (Song) obj;
        for(int i=0; i< fieldNames.length; i++)
            if(!songInfo.get(fieldNames[i])
                    .equals(song.getSongInfo().get(fieldNames[i])))
                return false;
        return true;
    }

    @Override
    public int hashCode() {
        return songInfo.get("title").hashCode()+songInfo.get("singer").hashCode();
    }

    @Override
    public String toString() {
        StringBuilder stringBuilder = new StringBuilder();
        for(String key : fieldNames){
            stringBuilder.append(key+":" +songInfo.get(key)+" ");
        }
        return stringBuilder.toString();
    }
}

```

노래의 정보가 많기 때문에 가장 중요한 정보인 title 은 하나의 필드로 만들어주었고, 나머지 정보들은 HashMap 인 songinfo 를 선언해서 관리할 수 있게 하였다. 그리고 songinfo 의 Key 값을 정적 멤버로 만들어서, 생성자에 들어오는 필드 value 와 Key 값을 연결하여 주었다.

노래정보에 대한 수정이 불가능하게 하기 위해서 setter 는 만들지 않았고, 노래의 제목과 노래의 정보들은 가져올 수 있게 하기위해 getter 는 만들어주었다.

equals 메소드를 오버라이딩 하여 모든 노래의 정보가 같다면 동일한 노래인지 확인을 할 수 있게 해주었다.

- buildSongObjects() 구현

RelationShip 으로 대체하였기 때문에 구현하지 않았다.

```
6  public class RelationShip {
7      /*
8       *  고전적방식의 singleton class
9       *  songs, students Main Class 에서 파일을 읽어온 후 1회 초기화 이후 read only
10      *  filter 에게 추천을 위한 적절한 관계를 제공합니다.
11      */
12
13
14     private static final String[] SONGS_VECTOR_COL_NAMES = {"singer", "genre", "releaseYear", "issuingCountry", "language"};
15     private static final String[] STUDENT_VECTOR_COL_NAMES = {"singer", "genre", "language"};
16
17     private static RelationShip instance;
18
19     private Song[] songs = null; // 모든 노래
20     private Student[] students = null; // userID : 해당 유저가 들은 노래.
21
22     private RelationShip(){
23     }
24
25     public static synchronized RelationShip getInstance(){
26         if (instance == null) {
27             instance = new RelationShip();
28         }
29         return instance;
30     }
31
32     public void init(Song[] songs, Student[] students){
33         if(this.songs == null && this.students == null ){
34             this.songs = songs;
35             this.students = students;
36         }
37     }
```

RelationShip 객체는 프로그램에 있어서 자주 쓰이며, 학생과 노래 간의 관계를 관리하기 때문에 단 하나의 인스턴스만 생성하여 데이터를 저장해야 된다고 생각하였다. 따라서 Singletone 패턴의 클래스로 만들어주었다. Main 에서 모든 학생과 노래들을 한 번 초기화 할 수 있다.

Singletone 패턴이기 때문에 정적 멤버로서 자기 자신을 가지며, 추천을 위해 필요한 노래와 노래, 학생과 학생 간의 관계들의 항목을 정적멤버로 가진다.

멤버변수로서는 모든 학생과 노래들을 가지고 있다.

- similarSongVectorMap

```
37      public HashMap<String, HashMap<CharSequence, Integer>> similarSongsVectorMap(Song targetSong){
38          HashMap<String, HashMap<CharSequence, Integer>> vectorMap
39                  = new HashMap<String, HashMap<CharSequence, Integer>>();
40
41          for(Song song : songs) {
42              HashMap<CharSequence, Integer> songVector = createSongVector(targetSong, song);
43              vectorMap.put(song.getTitle(), songVector);
44          }
45          return vectorMap;
46      }
```

	singer	genre	releaseYear	issuingCountry	language
song1	1	1	0	0	0
song2	1	0	0	1	1
song3	1	1	0	1	1
…					
song375	1	1	0	1	1

노래와 노래를 이용하여 유사한 노래를 선정하는 기준을 바탕으로 위와 같은 표를 생성해준다. 이 메소드의 결과값은 나중에 노래와 노래의 유사성을 바탕으로 노래를 추천하는데 사용된다.

- similarStudentVectorMap

```
68      public HashMap<String, HashMap<CharSequence, Integer>> similarStudentVectorMap(Student targetStudent){
69          HashMap<String, HashMap<CharSequence, Integer>> vectorMap
70                  = new HashMap<String, HashMap<CharSequence, Integer>>();
71
72          for(Student student : students) {
73              HashMap<CharSequence, Integer> studentVector = createStudentVector(targetStudent, student);
74              vectorMap.put(student.getUserId(), studentVector);
75          }
76          return vectorMap;
77      }
```

	singer	genre	language
student1	1	1	0
student2	1	0	1
student3	1	1	1
…			
Student37	1	1	1

학생과 학생 간의 선호도 관계를 이용하여 유사한 노래를 선정하는 기준을 바탕으로 위와 같은 표를 생성해준다. 이 메소드의 결과값은 나중에 학생과 학생의 유사성을 바탕으로 노래를 추천하는데 사용된다.

- 기타 메소드

```
143     public Song[] getSongs() {
144         return songs;
145     }
146
147     public Student[] getStudents() {
148         return students;
149     }
150
151     public Song searchSongByTitle(String title){
152         for(Song song : songs) {
153             if (song.getTitle().equals(title))
154                 return song;
155         }
156         return null;
157     }
158
159     public Student searchStudentById(String id){
160         for(Student student : students) {
161             if (student.getUserId().equals(id))
162                 return student;
163         }
164         return null;
165     }
```

전체 노래, 전체 학생을 반환해주는 getter 와 제목을 기반으로 노래를 찾을 수 있는 메소드, ID 를 기반으로 학생을 찾을 수 있는 메소드를 구현하였다.

- Filter

```
8   public interface Filter<T> {
9
10      public double THRESHOLD = 0.8;
11      public CosineSimilarity cosineSimilarity = new CosineSimilarity();
12
13      public HashMap<T, Double> filter(String id, RelationShip relationShip);
14
15  }
```

Filter 는 인터페이스로서 80%이상 유사한 노래를 선정하기 위한 상수인 THRESHOLD 를 가지고 있고, 필수적으로 이 인터페이스를 구현해야 되는 클래스들이 가져야 하는 필터의 기능을 filter 메소드로 가지고 있다. 그리고 학생과 노래의 벡터를 필터링을 해야 하기 때문에 제네릭을 이용하였다. 다음은 필터 인터페이스를 구현해서 song 벡터를 필터링하는 클래스다.

```java
public class SimilarSongFilter implements Filter<Song> {


    @Override
    public HashMap<Song, Double> filter(String id, RelationShip relationShip){

        HashMap<Song, Double> songSimilarMap = new HashMap<>();
        Student student = relationShip.searchStudentById(id);
        List<Song> songList = student.getMySongs();

        for(Song song : songList) {
            HashMap<String, HashMap<CharSequence, Integer>> vectorMap
                    = relationShip.similarSongsVectorMap(song);
            Iterator<String> iterator = vectorMap.keySet().iterator();
            while (iterator.hasNext()) {
                String key = iterator.next();
                double similarity = cosineSimilarity.cosineSimilarity(vectorMap.get(song.getTitle()), vectorMap.get(key));
                if (similarity > THRESHOLD && !song.getTitle().equals(key)) {
                    songSimilarMap.put(relationShip.searchSongByTitle(key), similarity);
                }
            }
        }
        return songSimilarMap;
    }


}
```

유사도의 기준은 코사인 유사도를 사용하였다. song list 파일이 텍스트 파일이기 때문에, 텍스트와 텍스트의 유사도를 구하는 방식으로 코사인 유사도를 사용하기 적합하다고 생각하였다.

Student 를 필터링하는 클래스도 동일하게 코사인 유사도를 사용하여 구현하였다.

- recommender

```java
public interface Recommender {

    /*
     * A function to print statistics
     * have to implement it
     */


    public void printStatistics();

    public void recommend();

}
```

Recommender 는 통계 값(유사도)을 출력하는 기능과 노래를 추천하는 기능을 수행해야 하기 때문에 추상클래스보다 인터페이스로 만들어, 3 개의 추천 방식이 담긴 클래스에서 각각 구현하는 것이 적합하다고 생각하였다. 기존의 Recommender 추상 클래스에서 생성자를 통해서 전체 노래를 만들어서 멤버변수로 가졌던 것을 이제는 Relationship 클래스에서 가져올 수 있기 때문에 Can-do 형태에 적합한 인터페이스로 변경하였다.

<RandomRecommender>

```java
public class RandomRecommender implements Recommender{

    private List<Song> randomSongList;

    public RandomRecommender() {
        randomSongList = new ArrayList<>();
    }

    @Override
    public void printStatistics(){
        Song[] songs =  RelationShip.getInstance().getSongs();
        int bound = songs.length;
        Random random = new Random();
        System.out.println("======무작위로 선정된 index=======");
        for(int i = 0; i < 30; i++) {
            int idx = random.nextInt(bound);
            System.out.println(idx);
            randomSongList.add(songs[idx]);
        }
    }

    @Override
    public void recommend() {
        System.out.println("==========추천 노래 목록==========");
        for(Song song : randomSongList)
            System.out.println(song);
    }
}

```

Random 클래스를 이용하여 전체 노래 1 번부터 375 번 중에서 랜덤한 값을 30 개 입력 받아서 printStatistics 에서 랜덤한 30 개의 값을 출력하고, recommend 에서는 선정된 노래를 추천해준다.

<SimilarSongRecommender>

```java
public class SimilarSongRecommender implements Recommender {


    private List<Song> studentSongList;
    private Filter<Song> filter = new SimilarSongFilter();
    private HashMap<Song, Double> cosineSimilarityMap;

    public SimilarSongRecommender(String studentId) {
        RelationShip relationShip = RelationShip.getInstance();
        Student student = relationShip.searchStudentById(studentId);
        studentSongList = student.getMySongs();
        cosineSimilarityMap = filter.filter(studentId, relationShip);
    }

    @Override
    public void printStatistics(){
        System.out.println("======학생이 들은 노래와 유사한 노래(중복 포함)=======");
        Iterator<Song> iterator = cosineSimilarityMap.keySet().iterator();
        while(iterator.hasNext()){
            Song key = iterator.next();
            System.out.println(key);
            System.out.println("유사도: "+cosineSimilarityMap.get(key) );
        }
    }

    @Override
    public void recommend() {
        Set<Song> set =  cosineSimilarityMap.keySet();
        for(Song song : studentSongList)
            set.remove(song);
        System.out.println("==========추천 노래 목록==========");
        Iterator it = set.iterator();
        while(it.hasNext()){
            System.out.println(it.next());
        }
    }
}
```

학생의 ID 를 생성자에서 입력 받아 그 학생이 들은 노래목록을 가져와서 각 노래마다 필터에서 정한 기준 이상의 노래와, 그 유사도를 멤버변수인 cosineSimilarlityMap 에 저장한다. 그리고 그 cosineSimilarlityMap 을 이용하여 printStatistics 에서 출력한다. recommend 에서는 학생이 기존에 들었던 노래를 제외하고 추천해준다.

<SimilarStudentRecommender>

```java
public class SimilarStudentRecommender implements Recommender{

    private Set<Song> similarStudentsSongList;
    private Filter<Student> filter;
    private HashMap<Student, Double> cosineSimilarityMap;

    public SimilarStudentRecommender(String id) {
        filter = new SimilarStudentFilter();
        cosineSimilarityMap = filter.filter(id, RelationShip.getInstance());
        similarStudentsSongList = new HashSet<>();

    }

    @Override
    public void printStatistics(){
        System.out.println("======현재 유저와 비슷한 성향의 학생들=======");
        Iterator<Student> iterator = cosineSimilarityMap.keySet().iterator();
        while(iterator.hasNext()){
            Student key = iterator.next();
            System.out.println(key);
            System.out.println("유사도: "+ cosineSimilarityMap.get(key) );
        }
    }

    @Override
    public void recommend() {
        System.out.println("==========추천 노래 목록==========");
        Set<Student> similarStudentSet = cosineSimilarityMap.keySet();
        Iterator<Student> iterator = similarStudentSet.iterator();
        while (iterator.hasNext()) {
            Student student = iterator.next();
            similarStudentsSongList.addAll(student.getMySongs());
        }

        for(Song song : similarStudentsSongList){
            System.out.println(song);
        }
    }


}
```

학생의 ID 를 생성자에서 입력 받아 그 학생과 유사한 학생들을 필터에서 정한 기준 이상의 학생과, 그 유사도를 멤버변수인 cosineSimilarlityMap 에 저장한다. 그리고 그 cosineSimilarlityMap 을 이용하여 유사한 학생들과 유사도를 printStatistics 에서 출력한다. recommend 에서는 유사한 학생이 들었던 노래를 중복을 제외하고 추천해준다.

- 이상한 값 입력 시 에러 핸들링

```
95      public boolean idValidation(String id){
96          id.trim(); // 공백 제거.
97          if(isProperId(id) && isExistedId(id)) {
98              message.modeSelectMessage();
99              return true;
100         }
101         return false;
102     }
103
104     public boolean isProperId(String id){
105         if(Pattern.matches("^user[0-9]+$", id))
106             return true;
107         message.errorMessage(Message.NOT_PROPER_ID);
108         return false;
109     }
110
111     public boolean isExistedId(String id){
112         for(Student student : relationShip.getStudents()){
113             if(student.getUserId().equals(id))
114                 return true;
115         }
116         message.errorMessage(Message.NOT_EXIST_ID);
117         return false;
118     }
119
120     public boolean modeValidation(String mode){
121         if(Pattern.matches("^[1-3]$", mode))
122             return true;
123         message.errorMessage(Message.NOT_PROPER_MODE);
124         return false;
125     }
```

사용자가 잘못된 값을 입력하는 경우는 다음 3 종류가 있을 것이라고 생각하였다.

1. 잘못된 ID 형식

   User 의 ID 는 “User+숫자”의 형태로 고정이 되어있기 때문에 이것을 정규식으로 만들어서 잘못된 값이 입력되면 더 이상 프로그램을 진행시키지 않고 메시지를 띄우고, 사용자가 올바른 ID 형식을 입력할 때까지 반복한다.

2. 존재하지 않는 ID

   “User+숫자”의 형태가 맞더라도 존재하지 않는 유저 ID 를 입력한다면 사용자가 존재하는 ID 를 입력할 때까지 반복한다. 이때, ID 의 존재여부는 relationShip 이 모든 학생객체를 가지고 있기 때문에 쉽게 확인할 수 있다.

3. 존재하지 않는 추천모드

   User 의 ID 는 잘 입력했으나 추천 모드를 (1)(2)(3)이 아닌 다른 이상한 숫자 또는 문자를 입력하였을 경우 또한 정규식을 만들어서 더 이상 프로그램을 진행시키지 않고 메시지를 띄워, 사용자가 올바른 모드를 입력할 때까지 반복한다.

- 기타 클래스

```java
public class Message {

    private static final String INPUT_ID_COMMENT = "학생의 ID를 입력해주세요 >> ";

    private static final String INPUT_MODE_MESSAGE = "노래를 추천받을 모드 입력해주세요 >> ";

    private static final String MODE_EXPLANATION = "(1) 랜덤추천 (2) 유사한 학생들이 들은 노래 (3) 들었던 노래와 유사한 노래";

    private static final String[] ERROR_MESSAGES = {
            "정상적인 ID가 아닙니다.(예시: user+숫자)",
            "존재 하지 않는 Id 입니다.",
            "추천 모드는 [1  2  3] 중에 선택해주세요."};

    public static final int  NOT_PROPER_ID = 0;
    public static final int  NOT_EXIST_ID = 1;
    public static final int  NOT_PROPER_MODE = 2;

    private String message;

    public Message(){
        startMessage();
    }

    public void startMessage(){
        System.out.println(INPUT_ID_COMMENT);
    }

    public void modeSelectMessage(){
        System.out.println(INPUT_MODE_MESSAGE);
        System.out.println(MODE_EXPLANATION);
    }

    public void errorMessage(int err){
        switch (err) {
            case NOT_PROPER_ID: System.out.println(ERROR_MESSAGES[0]); break;
            case NOT_EXIST_ID: System.out.println(ERROR_MESSAGES[1]); break;
            case NOT_PROPER_MODE: System.out.println(ERROR_MESSAGES[2]); break;
        }
    }
```

메시지 클래스는 Main 프로그램이 진행될 때 필요한 메시지를 정적 멤버로 가지고 있다. 이 클래스를 만듦으로써 추후에 메시지를 변경하기 용이하고, Main 코드를 깔끔하게 할 수 있다.

코사인 유사도 클래스는 구글링을 통해서 Apache.common.text.similarlity 패키지에 있는 코사인 유사도 클래스를 가져와서 사용했다.

- 실행한 화면

1. 랜덤 추천

```
20182175:~/environment/oop-leeky-project $ java -cp target/oop-leeky-project-1.0-SNAPSHOT.jar kr.ac.kookmin.cs.oop.project.Main
학생의 ID를 입력해주세요 >>
user2
노래를 추천받을 모드 입력해주세요 >>
(1) 랜덤추천 (2) 유사한 학생들이 들은 노래 (3) 들었던 노래와 유사한 노래
1
======무작위로 선정된 index=======
147
253
52
135
264
369
33
31
249
```

...

```
==========추천 노래 목록==========
title:해에게서 소년에게 singer:NEXT genre:ROCK releaseYear:1997 albumName:Lazenca: A Space Rock Opera songWriter:신해철 producer:crom issuingCountry:대한민국 language:한국어 p
layingTime:5:03
title:휘파람 singer:블랙핑크 genre:랩/힙합 releaseYear:2016 albumName:SQUARE ONE songWriter:teddy producer:양현석 issuingCountry:대한민국 language:"한국어,영어" playingTime:3:
31
title:rain singer:태연 genre:발라드 releaseYear:2016 albumName:Rain songWriter:Matthew Tishier producer: issuingCountry:대한민국 language:한국어 playingTime:3:49
title:녹은 가슴 singer:Monday Kiz genre:발라드 releaseYear:2007 albumName:Incompletion songWriter:박해문 producer: issuingCountry:대한민국 language:한국어 playingTime:4:37
title:Demons singer:Imagine Dragons genre:락 releaseYear:2012 albumName:Night Visions songWriter: producer: issuingCountry:미국 language:영어 playingTime:2:56
title:Beggin singer:Madcon genre:"R&B, neo soul, hip hop" releaseYear:2004 albumName:So Dark the Con of Man songWriter:Bob Gaudio-Peggy Farina producer:3Elementz issuingCountr
y:USA language:English playingTime:3:38
title:Oh Boy singer:레드벨벳 genre:댄스 releaseYear:2015 albumName:The Red-The 1st Album songWriter:Herbie Crichlow producer:Herbie Crichlow issuingCountry:대한민국 language:한
국어 playingTime:3:08 8
```

2. 유사한 학생들이 들은 노래

```
20182175:~/environment/oop-leeky-project $ java -cp target/oop-leeky-project-1.0-SNAPSHOT.jar kr.ac.kookmin.cs.oop.project.Main
학생의 ID를 입력해주세요 >>
user20
노래를 추천받을 모드 입력해주세요 >>
(1) 랜덤추천 (2) 유사한 학생들이 들은 노래 (3) 들었던 노래와 유사한 노래
2
======학생이 들은 노래와 유사한 노래(중복 포함)=======
title:just singer:마마무 genre:알앤비 releaseYear:2016 albumName:Melting songWriter:박우상 producer:박우상 issuingCountry:대한민국 language:"한국어,영어" playingTime:3:41
유사도: 0.8164965809277259
title:오래전 그날 singer:"윤종신,이적" genre:발라드 releaseYear:2013 albumName:행보 2013윤종신 songWriter:박주연 producer: issuingCountry:대한민국 language:한국어 playingTime:
5:16
유사도: 0.8164965809277259
title:Always singer:윤미래 genre:드라마음악 releaseYear:2016 albumName:태양의후예OST Part1 songWriter:로코베리 producer:"지훈, 개미" issuingCountry:대한민국 language:한국어 pl
ayingTime:3:29
유사도: 1.0000000000000002
title:오래된 노래  singer:스탠딩 에그 genre:발라드 releaseYear:2012 albumName:오래된 노래 songWriter:스탠딩에그  producer:스탠딩에그 issuingCountry:대한민국 language:한국어 p
layingTime:4:33
유사도: 0.8164965809277259
title:사랑이 아프다 singer:환희 genre:드라마음악 releaseYear:2016 albumName:함부로 애틋하게 OST Part 10 songWriter:박곰 producer:박곰 issuingCountry:대한민국 language:한국어 p
layingTime:4:02
유사도: 1.0000000000000002
title:행복하지 말아요 singer:M.C.the Max genre:Ballad releaseYear:2014 albumName:Solitude Love… songWriter:Kohmi Hrose producer: issuingCountry:대한민국 language:한국어 playin
gTime:5:56
유사도: 0.8164965809277259
title:연애 singer:프리스타일 genre:랩 releaseYear:2009 albumName:Dry&Wet songWriter:지오 producer:미노 issuingCountry:대한민국 language:한국어 playingTime:3:43
유사도: 0.8660254037844387
title:겨울을 걷는다 singer:윤딴딴 genre:락 releaseYear:2014 albumName:반오십 songWriter:윤딴딴 producer:윤딴딴 issuingCountry:대한민국 language:한국어 playingTime:4:38
유사도: 0.8164965809277259
title:영원한 십자가 singer:하명지 genre:CCM releaseYear:2016 albumName:영원한 십자가 songWriter:하명지 producer:하명지 issuingCountry:대한민국 language:한국어 playingTime:4:04

유사도: 1.0000000000000002
title:네 생각 singer:존박 genre:발라드 releaseYear:2016 albumName:네 생각 songWriter:존 박 producer:적재 issuingCountry:대한민국 language:한국어 playingTime:3:50
유사도: 1.0000000000000002
title:사랑은 Move singer:시크릿 genre:댄스 releaseYear: albumName: songWriter: producer: issuingCountry:대한민국 language:한국어 playingTime:3:20
유사도: 0.8164965809277259
title:짧은 한마디 singer:민경훈 genre:발라드 releaseYear:2011 albumName:2집 소풍 songWriter:김세준 producer: issuingCountry:대한민국 language:한국어 playingTime:4:09
유사도: 0.8164965809277259
title:거북이 singer:다비치 genre:발라드 releaseYear:2013 albumName: MYSTIC BALLAD Part 2 songWriter:이단옆차기 producer: issuingCountry:대한민국 language:한국어 playingTime:3:
43
유사도: 0.8164965809277259
title:Why singer:태연(TAEYEON) genre:Dance releaseYear:2016 albumName:Why- The 2nd Mini Album songWriter:"LDN Noise, Lauren Dyson, Rodnae `Chikk` Bell" producer: issuingCountr
y:대한민국 language:한국어 playingTime:3:27
유사도: 0.8944271909999159
title:잘미꽃 singer:개코 genre:랩/힙합 releaseYear:2014 albumName:Redingray songWriter:개코 producer: issuingCountry:대한민국 language:한국어 playingTime:
유사도: 0.8164965809277259
title:사랑해요 singer:김범수 genre:드라마음악 releaseYear:2016 albumName:함부로 애틋하게 OST Part 9 songWriter:"롤이랑 제리, 전우성" producer:"롤이랑 제리, 전우성" issuingCoun
try:대한민국 language:한국어 playingTime:4:08
```

3. 들었던 노래와 유사한 노래

```
20182175:~/environment/oop-leeky-project $ java -cp target/oop-leeky-project-1.0-SNAPSHOT.jar kr.ac.kookmin.cs.oop.project.Main
학생의 ID를 입력해주세요 >>
user5
노래를 추천받을 모드 입력해주세요 >>
(1) 랜덤추천 (2) 유사한 학생들이 들은 노래 (3) 들었던 노래와 유사한 노래
3
======현재 유저와 비슷한 성향의 학생들=======
id: user23 mySongs: [title:봄을 노래하다 singer:40 genre:R&B / 어반 releaseYear:2013 albumName:April's Spring songWriter:40 producer: issuingCountry:대한민국 language:한국어 playingTime:3:44 , title:오늘은 가지마 singer:임세준&벤 genre:발라드 releaseYear:2016 albumName:오늘은 가지마 songWriter:임세준 producer: issuingCountry:대한민국 language:한국어 playingTime:4:03 , title:웅크려 앉아 singer:페이퍼컷 프로젝트 genre:"인디뮤직, 팝락" releaseYear:2014 albumName:1892014 songWriter: producer: issuingCountry:대한민국 language:한국어 playingTime:4:44 , title:뻐에로는 우릴 보고 웃지 singer:아이유 genre:발라드 releaseYear:2014 albumName:꽃갈피 songWriter: producer: issuingCountry:대한민국 language:한국어 playingTime:3:53 , title:함께 singer:노을 genre:발라드 releaseYear:2015 albumName:응답하라 1988 Part.7 songWriter: producer: issuingCountry:대한민국 language:한국어 playingTime:4:32 , title:내가 저지른 사랑 singer:임창정 genre:발라드 releaseYear:2016 albumName:I'M songWriter: producer: issuingCountry:대한민국 language:한국어 playingTime:4:01 , title:슬픈 인연 singer:박윤하 genre:발라드 releaseYear:2014 albumName:K팝 스타 시즌4 songWriter: producer: issuingCountry:대한민국 language:한국어 playingTime:2:26 , title:If Only singer:나윤건 genre:발라드 releaseYear:2014 albumName:If only songWriter: producer: issuingCountry:대한민국 language:한국어 playingTime:4:02 , title:I Want You Back singer:허각 genre:발라드 releaseYear:2015 albumName:사월의 눈 songWriter: producer: issuingCountry:대한민국 language:한국어 playingTime:3:30 , title:가을비 내리는 밤에 singer:11시 11분 genre:인디뮤직 releaseYear:2014 albumName:가을비 songWriter: producer: issuingCountry:대한민국 language:한국어 playingTime:4:27 ]
유사도: 0.8164965809277259
id: user14 mySongs: [title:piano singer:December genre:발라드 releaseYear:2015 albumName:Reason songWriter:조영수 producer:조영수 issuingCountry:대한민국 language:한국어 playingTime:3:03 , title:그림자 singer:Monday Kiz genre:드라마 ost releaseYear:2012 albumName:해를 품은 달 songWriter:"한상원, 반형문" producer: issuingCountry:대한민국 language:한국어 playingTime:3:56 , title:녹슨 가슴 singer:Monday Kiz genre:발라드 releaseYear:2007 albumName:Incompletion songWriter:박해문 producer: issuingCountry:대한민국 language:한국어 playingTime:4:37 , title:유리 심장 singer:Monday Kiz genre:발라드 releaseYear:2010 albumName:ru:t; songWriter:김원 producer: issuingCountry:대한민국 language:한국어 playingTime:3:37 , title:친구라서 끝내 참는 말 singer:Monday Kiz genre:발라드 releaseYear:2012 albumName:Healing activity songWriter:"이진성, 멜로딘 준영" producer: issuingCountry:대한민국 language:한국어 playingTime:3:38 , title:She's gone singer:December genre:발라드 releaseYear:2012 albumName:She's gone songWriter:조영수 producer:조영수 issuingCountry:대한민국 language:한국어 playingTime:3:51 , title:사랑을 고백합니다 singer:December genre:발라드 releaseYear:2009 albumName:Dear My Lover songWriter:박정욱 producer: issuingCountry:대한민국 language:한국어 playingTime:3:59 , title:혼자됐어요 singer:December genre:발라드 releaseYear:2010 albumName:A Story To The Sky songWriter:오성훈 producer: issuingCountry:대한민국 language:한국어 playingTime:3:42 , title:You are Mine singer:SG Wannabe genre:발라드 releaseYear:2015 albumName:THE VOICE songWriter:김진호 producer: issuingCountry:대한민국 language:한국어 playingTime:4:20 , title:So Sick singer:Ne-Yo genre:R&B releaseYear:2006 albumName:In My Own Words songWriter:Ne-Yo producer: issuingCountry:USA language:English playingTime:3:27 ]
유사도: 0.8164965809277259
```