Data Structers Project 3

This project is a movie recomendation system. Processes user data and draws appropriate results.

There is a max heap class. This heap has a Node attribute. There is data and user id attiributes in Node class.

```
public class MaxHeap {

    Node[] heap;
    int size;

    public MaxHeap(int capacity) {
        heap = new Node[capacity];
        size = 0;
    }

    public boolean isEmpty() {
        return size == 0;
    }

    public void insert(double value,int id) {
        if (size == heap.length) {
            throw new IllegalStateException(s: "Heap is full. Cannot insert more elements.");
        }

        Node newNode = new Node(data:value, user_id: id);
        heap[size] = newNode;

        siftUp(index: size);
        size++;
    }
```

The insert method, sorts according to the data of the sent node. The user_id keeps the movieId or userID.

There is extractMax method in MaxHeap class. This method extracts Node that has max data from heap.

```
public Node extractMax() {
    if (isEmpty()) {
        throw new IllegalStateException(s: "Heap is empty. Cannot extract maximum element.");
    }

    Node max = heap[0];
    heap[0] = heap[size - 1];
    size--;
    siftDown(index: 0);

    return max;
}
```

There is a MovieRecomendationSystem class. This class has methods of transferring the sent file to the array.

```
public class MovieRecommendationSystem {



    public int[][] readUserMovieMatrixFromFile(String fileName) {
        int[][] kullaniciFilmMatrisi = null;
        try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
            List<int[]> rows = new ArrayList<>();
            String line;
            while ((line = br.readLine()) != null) {
                String[] values = line.split(regex: ",");

                int[] row = new int[values.length];
                for (int i = 0; i < values.length; i++) {

                    row[i] = Integer.parseInt(values[i]);
                }
                rows.add(e: row);
            }
            kullaniciFilmMatrisi = new int[rows.size()][];
            for (int i = 0; i < rows.size(); i++) {
                kullaniciFilmMatrisi[i] = rows.get(index: i);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return kullaniciFilmMatrisi;
    }
```

This method compares the target user with all the users in the table and assigns the cosine similarity value to the heap. Then, the cosine values in the heap are extracted with the extract method and the most matching users in the table with the target user are written to the String text variable.

```
private void brn_recommendationActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String text = "";

    heap = new MaxHeap(capacity:movieMatris.length);
    int x_gui = Integer.parseInt(s: label_x.getText());
    int k_gui = Integer.parseInt(s: label_k.getText());

    int[] targetUser = createUserArray(matris: targetUserMatris, cmb_users.getSelectedIndex() + 1);
    //heap.insertHeap(targetUser);
    for (int i = 1; i < movieMatris.length; i++) {
        double cosine = m.cosineSimilarity(kullanici1: targetUser, kullanici2: createUserArray(matris: movieMatris, row: i));
        heap.insert(value: cosine,id: i);
    }

    ArrayList<Integer> users_id = new ArrayList<>();
    for (int i = 0; i < x_gui; i++) {
        Node maxNode = heap.extractMax();
        double maxValue = maxNode.data;
        int maxID = maxNode.user_id;
        text +="Id = " + maxID + " Cosine : "+ maxValue + "\n";
        users_id.add(e: maxID);
    }

    text += "----------\n";
```

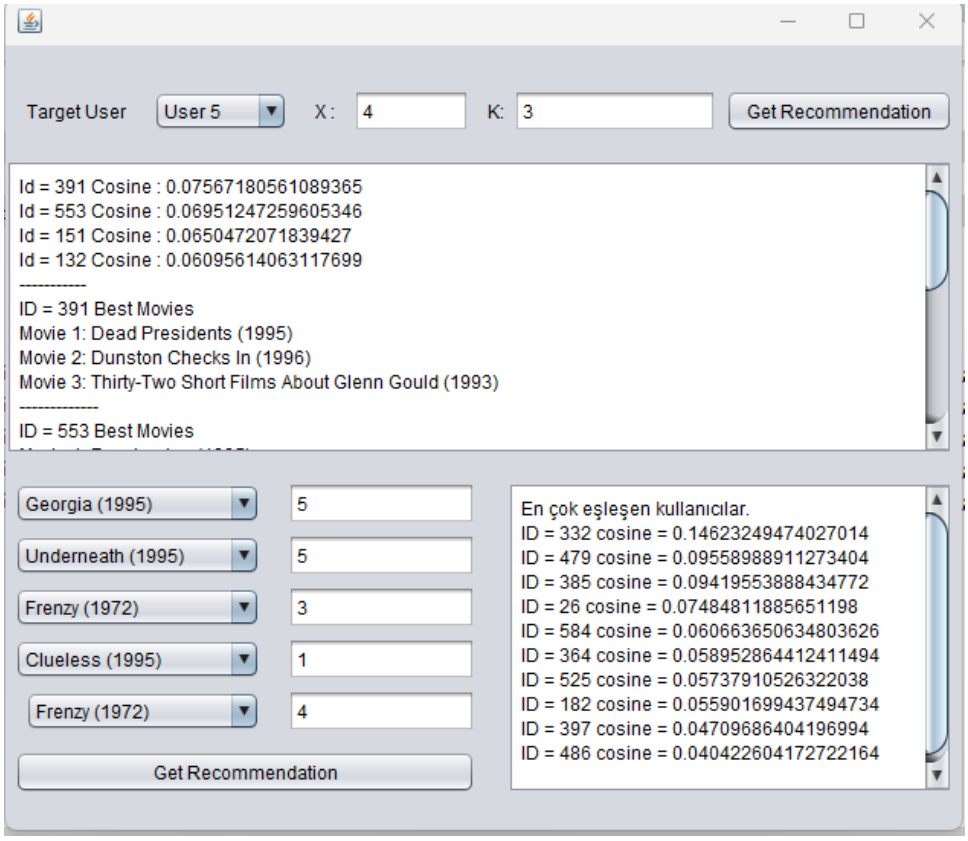
This loop sends an array list of the ids of the most matched users and this array sends all the movie scores and movie ids of each of the users in the list to the heap. Afterwards, the scores in this heap are extracted with the extract method and the favorite movies of the users are written to the text variable. Then it is transferred to the text area component with the set text method.

```
text += "-----------\n";
  for (int i = 0; i < users_id.size(); i++) {
    for (int j = 1; j < movieMatris.length; j++) {
        if(movieMatris[j][0] == users_id.get(index: i)){
            MaxHeap users_movies = new MaxHeap(capacity:movieMatris[0].length);
            for (int k = 1; k < movieMatris[j].length - 1; k++) {
                if (k == 0) {
                    System.out.println(x: movieMatris[j].length);
                }
                users_movies.insert(movieMatris[j][k], id: k);
            }
            text += "ID = " + users_id.get(index: i) + " Best Movies" + "\n";
            for (int k = 0; k < k_gui; k++) {
                int index = users_movies.extractMax().user_id;
                for (int l = 1; l < movies.length; l++) {
                    if(Integer.parseInt(movies[l][0]) == index){
                        text += "Movie " + (k+1) + ": " + movies[l][1] + "\n";
                    }
                }

            }
            break;
        }
    }
    text += "-------------\n";
}


text_area.setText(t: text);
```

In this method, movies are sent to a node array and the given scores are recorded. These scores are then recorded in an array. This array is compared with all users in the list and the users with the highest cosine value are printed on the screen.

```
// TODO add your handling code here:
String text = "";

Node[] movieID = new Node[5];
movieID[0] = new Node(data:Double.parseDouble(s: txt_movie1.getText()), user_id: switchCase(index: cmb_movie1.getSelectedIndex()));
movieID[1] = new Node(data:Double.parseDouble(s: txt_movie2.getText()), user_id: switchCase(index: cmb_movie2.getSelectedIndex()));
movieID[2] = new Node(data:Double.parseDouble(s: txt_movie3.getText()), user_id: switchCase(index: cmb_movie3.getSelectedIndex()));
movieID[3] = new Node(data:Double.parseDouble(s: txt_movie4.getText()), user_id: switchCase(index: cmb_movie4.getSelectedIndex()));
movieID[4] = new Node(data:Double.parseDouble(s: txt_movie5.getText()), user_id: switchCase(index: cmb_movie5.getSelectedIndex()));

int [] newUser = new int[movieMatris[1].length];

for (int i = 0; i < newUser.length; i++) {
    boolean bool = false;
    for (int j = 0; j < movieID.length; j++) {

        if(i == movieID[j].user_id){
            newUser[i] = (int) movieID[j].data;
            bool = true;
            break;
        }

    }
    if(!bool) {
            newUser[i] = 0;
        }
}


MaxHeap h = new MaxHeap(capacity:movieMatris.length);

 for (int i = 1; i < movieMatris.length; i++) {
    int[] arr = createUserArray(matris: movieMatris, row: i);

     double cosine = m.cosineSimilarity(kullanici1: newUser, kullanici2: arr);
     h.insert(value: cosine,id: i);
}
```

The output of the project is below.



ALİ HAYDAR OSMANOĞLU

2121221033