Friend Recommendation System for the Social Network based on User Behavior

Authors

Mohammed Mehedi Hasan   Student Id: 104416

Noor Hussain Shaon   Student Id: 104442

Supervisor

Md. Kamrul Hasan, PhD

Systems & Software Lab (SSL)
Assistant Professor
Department of Computer Science & Engineering (CSE)
Islamic University of Technology (IUT)

A thesis submitted to the Department of CSE in partial fulfilment of the requirements for the degree of B.Sc. Engineering in CSE

Academic Year: 2013-2014

Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT)
Gazipur, Dhaka, Bangladesh

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and investigation carried out by Mohammed Mehedi Hasan and Noor Hussain Shaon under the supervision of Md. Kamrul Hasan in the Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Gazipur, Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Mohammed Mehedi Hasan
Student ID: 104416

Noor Hussain Shaon
Student ID: 104442

Supervisor:

Md. Kamrul Hasan, Phd
Assistant Professor,
Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT)

Abstract

Social network sites have connected millions of users creating the social revolution in Web 2.0 now-a-days. If the group of people or organizations have the common interest then, a social network is constituted. In the present world, the most visited sites in the Internet are Twitter, Facebook, Orkut, Google plus etc. which is actually Online social networking sites. In the social network sites, a user makes friends with the other users and enjoy the communication with them. However, the large amount of online users and their diverse and dynamic interests possess great challenges to support such a novel feature in online social networks. In this thesis, we design a general friend recommendation framework based on user behavior. The main idea of the proposed method is consisted of the following stages-measuring the frequency of the activities and updating the dataset according to the activities, applying FP-Growth algorithm to find out the user behavior, then finding out the uncommon behavior containing the common behavior.

Contents

Chapter 1

Introduction

1.1 Overview

Now-a-days online social networks such as Facebook, Twitter, Google+, LinkedIn, and others have become significantly popular all over the world and people are using it throughout their daily lives. The number of users in the social networks is increasing day by day. Besides traditional desktop PCs and laptops, new emerging mobile devices makes it easier to make social networking. In online social network user behavior means various social activities that users can do online, such as friendship creation, content publishing, profile browsing, messaging, and commenting, liking, sharing and so on. So friend recommendation in online social network is very rich research area to recommend friends in the social network sites.

1.2 Problem Statement

There are so many users in the social network. We cannot be able to recommend every one of them as friend. There should be some criteria to recommend one a friend. So a friend recommendation system based on user behavior for the social network can be very helpful for recommending friends for these social networking sites. At first behavior has to be defined. From the behavior definition, common and uncommon behavior of the users can be identified. Users have many behaviors in social networking site. Some has common and some has uncommon behavior between themselves. Friends will be recommended with the help of the combination of these uncommon and common behavior of the users.

1.3 Research Challenge

Recommending people on social networking sites is worth studying because it is different from traditional recommendations of books, movies, restaurants, etc. due to the social implications of "friending". For example, before adding a friend, one has to consider a lot of things, whether the he or she know the person personally or his or her likings, activities etc. match with the person he or she wants add as a friend. Furthermore, the most challenging part in designing a recommendation system for a social network is the privacy issue of the users. With the ever increasing web crimes and identity theft, people are becoming more and more careful in sharing their personal information. Hence, unless a user can trust the system with their data, the system cannot stand and it will be valueless. Exploitation of social network data is the fragmentation of the population of social network users into numerous proprietary and closed social networks. This issue is compounded by the fact that each new game or media application tends to build its own social network around it rather than building upon the rich data available about existing social relationships. So it is difficult to find out real data of user in case of research. Spam detection and advertisement detection are research challenges that need extra attention from the research community. Since users and data production increase, spam (irrelevant in-formation) and advertisements will continue growing. As social networks will continue to evolve, discovering communities and constructing specific social graphs from large scale social networks will continue to be a dynamic research challenge. Also, online social communities face critical social and ethical issues that need special care and delicate handling. So it is very important to generate a recommending system where the users can trust the system and ensuring their privacy issue.

1.4 Motivation

The motivation behind our work is to recommend friends from the uncommon behavior. Here friends are not only recommended based on their uncommon behavior, but also first the common behaviors are be a part of recommendation. If we recommend friends only based on their common behavior, it will not be effective. Suppose one may like rock songs. So people who like rock songs should be friends. But this thing makes the recommendation complex. Rock songs may be a common behavior but they may come from different communities, they works in different places, their other activities may not be the same. So if we use some common behavior as the parameter and based on that parameter if we find the common behaviors among those uncommon behaviors we can recommend friend very effectively.

1.5 Scope

For recommending friends there may be a lot of systems. But considering the common and uncommon behavior there are nothing. So using this common and uncommon behavior it will be a milestone for the online social networking sites. Merging the common and uncommon behavior together it is possible to create a new system for recommending friends in the online social networking sites.

1.6 Research Contribution

With the help of this research online social network sites will get a better recommendation system for their sites. If a huge amount of users are under consideration there will be many things there where people have common likings, they do similar kinds of activities. But only few activities in common should not be appropriate for recommending a friend. Some activities are common to some set of users, apart from that there are also some uncommon behaviors. Among those uncommon behavior there are possibly some common attributes. Mixing the common and uncommon behavior friend recommendation will be more effective and efficient.

Chapter 2

Literature Review

2.1 Social Networking

With the advent of Web 2.0, social computing has emerged as one of the hot research topics recently. It involves the collecting, extracting, accessing, processing, computing and visualizing of social signals and information. SNSs are an online phenomenon which provides social network based services to support easy message posting, information sharing and inter-friend communication. A social network is a set of people or groups of people with some pattern of contacts or interactions between them. The patterns of friendships between individuals, business relationships between companies, and intermarriages between families are all examples of networks that have been studied in the past. Social Network Sites are defined as web-based services that allow individuals to (1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system [1]. Social Networking sites (SNS's) provide users with opportunity to connect with their offline friends as well as making new friends with latent ties who otherwise would never have met them. They also supplement their relationships with close relations and help to maintain the social capital [2]. People tend to trust the opinions of friends they know rather than the opinions of strangers.

2.2 Recommendation System

Over the last decade, Recommendation Systems became an important research area to find out new approaches of recommendation both in industry and academia. Recommendation systems can be traced back to the extensive work in the cognitive science, approximation theory, information retrieval, forecasting theories, and also have links to management science, and also to the consumer choice modeling in marketing. Recommendation systems are a subclass of information filtering system that seek to predict the 'rating' or 'preference' that a user would give to an item (such as music, books, or movies) or social element (e.g. people or groups) they had not yet considered, using a model built from the characteristics of an item or the user's social environment. In recommender systems the utility of an item is usually represented by a rating, which indicates how a particular user liked a particular item [3].

Generally Recommender systems are divided into two categories.

- Content-based recommendations: The user is recommended items similar to the ones the user preferred in the past.
  Info finder [4] and News weeder [5] are some examples of content-based model.
- Collaborative recommendations: The user is recommended items that people with similar tastes and preferences liked in the past. In other word, recommend new items based on previous transactions as well as preference of similar users [6].

There are some hybrid models also where content-based and collaborative-based models have been unified to compromise their shortcomings [7]. They use components like linear combination of predicted ratings, various voting schemes, incorporating one component as a part of the heuristic for the other. For example, Billsus & Pazzani 2000 uses hybrid recommendation system.

2.3 Friend Recommendation System

With the rapid growth of social networks, users of SNSs may easily get overwhelmed by the excessive volume of information. Therefore, the recommendation of better friend is the essential factor of social network sites to find truly valuable information. There are many friend recommendation systems for social networking sites. For understanding different behavior in the social network different approach may be taken. *Connectivity and interaction, traffic activity, malicious behavior, mobile social behavior* are needed for understanding user behavior in social network [8]. Some people use the photo comment and wall post as interaction to determine the behavior [9]. If the users are denoted as node and the friendship between different users are denoted as edge we can make a graph for recommending the friends. This graph may be formed as a directed graph or an undirected graph. There may be different types of graphs for the social graph, friendship graph where friendship between users are the edges, interaction graph where visible interaction, such as posting on a wall are the edges, latent graph Latent interaction, such as browsing profile are the edges, following graph subscribe to receive all messages are the edges [8]. Cluster analysis performed on the feature vectors may be used for identifying the user behaviors. From these cluster analysis dominant user behavior may be found using a threshold value [9]. If we can identify the user behaviors we can manage to suggest one person to another person to be his friend. By leveraging interest-based features, we can design a general friend recommendation framework by which we can recommend one person to be friend of another person [11]. Characterizing the user interest by two dimension, *content and context* we can build these framework [11]. Friend recommendation can be done using the clustered method [12]. Using the clustered method based on some factors the friends can be recommended. Twelve factors are used for defining the behavior. Among a lot of users first of all from this twelve features users are divided in five different clusters. This five clusters are formed by using K-means algorithm. Now if we select a focused user and plot it into the five clusters, and this user will match any of the five clusters. Then we can recommended this user into the users of that cluster. A cohesion based friend recommendation system are used for the friend recommendation system [13]. Cohesion based friend recommendation can be recommended by using some steps, extracting sub network from the whole network. Then measuring the link strength among the networks. After that a threshold value can be introduced for augmenting the network. Then we can recommend friends using community detection. Friend recommendation is based on the user social relations and personal information profiles or the “friend of friend” method [14]. Friends in common, similar age, geographic location are used to recommend friends. Users from same age same geographic location can be friends. User profiling system can be used for recommending friends [15]. Interpolation of profile data by the data of other users who are linked with the focused user. Friends are recommended based on clustering method [16]. Users browsing records and the content users interested in. Used clustering method for reducing system overhead while dealing with a large amount of data. It will make the friend recommendation much faster. Yu Zheng et al [17] proposed a system of friend recommendation based on location.-GeoLife2.0. It is a GPS-data-driven social networking service where people can share life experiences and connect to each other with their location histories. By mining people‘s location history that can measure the similarity between users and perform personalized friend recommendation for an individual.

However, the previous approaches did not consider user behavior for friend recommendation in social networking sites. Activities of the users are the behavior of the users. From the behavior common and uncommon behavior can be defined. Based on this common and uncommon behavior friends can be recommended.

Chapter 3

Proposed Method

In the previous chapter, we have extensively discussed about the existing friend recommendation systems. We have tried to find out the problems of that system and gain a lot of information about social networking and friend recommendation. After analyzing those, we also try to make a new system for suggesting friends in social networking sites. In this section, we present our proposed friendship algorithm based on user behavior.

3.1 User Behavior in Social Network

User behavior is the most important thing for this approach in friend recommendation system. For this, the behavior in the social network must be well defined. User behavior is basically the activities of the users in the social network. From the user behavior common and uncommon behavior need to be formed. From this common and uncommon behavior the desired recommendation will come.

3.1.1 Behavior Definition

U = {u | users in Facebook}

A = {a | activities of the users in Facebook}

R = {r | considered activities of the users in Facebook}

R = P(A)

So that,

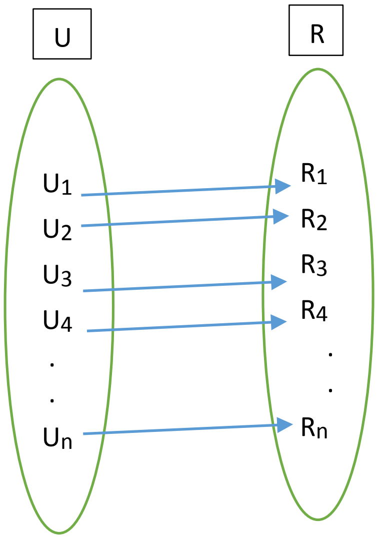
$U = \{ U_1, U_2, U_3, \ldots\ldots , U_n\}$

$A = \{ a_1, a_2, a_3, \ldots\ldots , a_n\}$

$R = \{\{a_1\},\{a_2\},\{a_3\},\ldots,\{a_n\},\{a_1,a_2\},\{a_1,a_3\},\ldots,\{a_1, a_2, a_3,\ldots, a_n\}\}$

The behavior of the user is completely related to the activities of the users. Users can do different activities. But the behavior will be those activities which are considered in our related activity set(R). We can give the relation like this

$\mathbf{B : U \rightarrow R}$



B_i = behavior of the U_i

$B_1(U_1, R_1)$, $B_2(U_2, R2)$, $B_3(U_3, R_3)$, $B_4(U_4, R_4)$,…………, $B_n(U_n, R_n)$

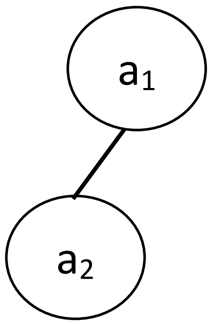
Here,

$R_1 = \{a_1\}$, $R_2 = \{a_2, a_3\}$, $R_3 = \{a_1, a_2, a_3\}$, $R_4 = \{a_1, a_3\}$,…, $R_n = \{a_1, a_2,\ldots\ldots, a_n\}$

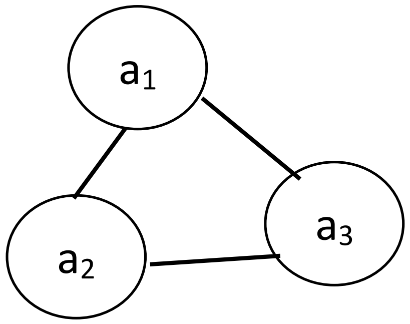
So, R is a proper subset of A. ($R \subseteq A$)

This relation can be defined as many formulation as it is defined in the discussion.

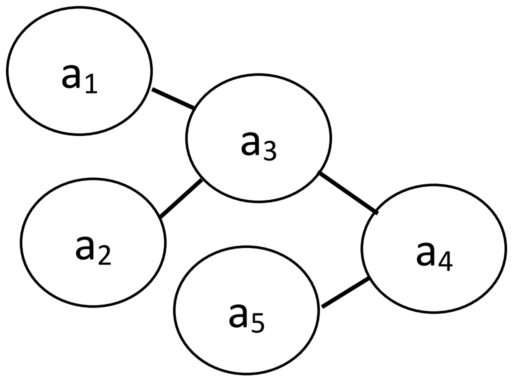
The user behavior can be represented as a complex graph where activities will be the node and relation of the activities as edges of the graph G, then



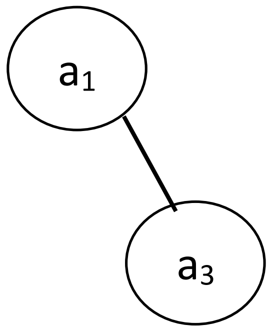
G_1



G_2



G_3



G_4

$B_1(U_1, R_1)$, $B_2(U_2, R2)$, $B_3(U_3, R_3)$, $B_4(U_4, R_4)$

Here,

$R_1 = G_1 = [\{a_1, a_2\}, \{(a_1,a_2)\}]$, $R_2 = G_2 = [\{a_1,a_2, a_3\}, \{(a_1,a_2), (a_1,a_3), (a_3,a_2)\}]$,
$R_3 = G_3 = [\{a_1, a_2, a_3, a_4, a_5\}, \{(a_1,a_3), (a_3,a_2), (a_3,a_4), (a_4,a_5)\}]$,

$R_4 = G_4 = [\{a_1, a_3\}, \{(a_1,a_3)\}]$

Again sequence of the activity can also be the behavior of the user

$S1 = a1 \rightarrow a2 \rightarrow a3 \rightarrow a5 \rightarrow a6 \rightarrow a3 \rightarrow a4$

$S2 = a1 \rightarrow a2 \rightarrow a5 \rightarrow a7 \rightarrow a8$

These sequence of the activities S1, S2, S3, …., Sn can be the behavior of the user u1.

So, b1 = {S1, S2, S3, …., Sn}

From this behavior, it can be divided into parts.

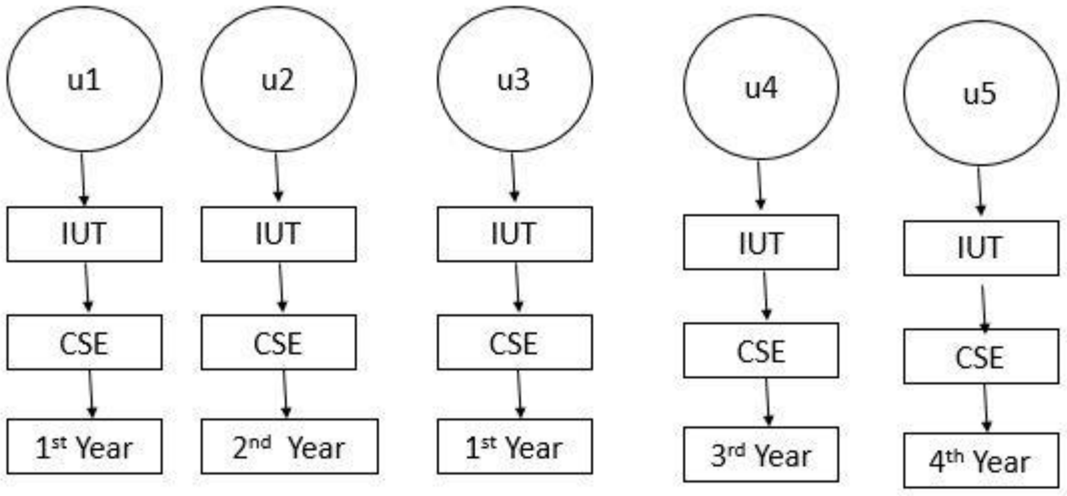
(i) Common Behavior

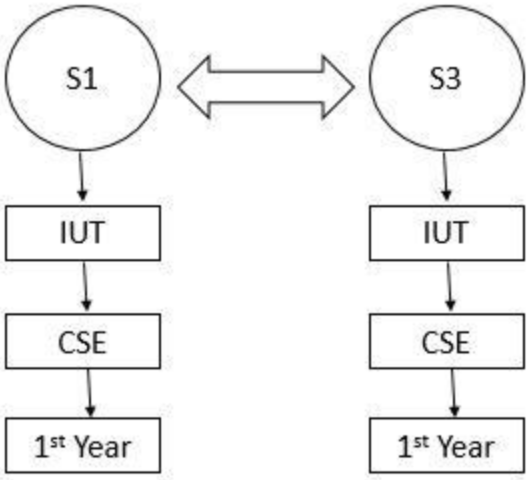
(ii) Uncommon Behavior

- **Common behavior:** Common behavior means the common activities of the users. This common behavior is not fixed or pre-defined. For different data set the common behavior will be different. Common behavior will not be only one activity. Two or more activity can make a common behavior. In our methodology the common behavior is the max frequency of the any activity in the dataset. Formally, we can define common behavior as like, B_1 and B_2 has a common behavior of U_1 and U_2, if and only if activities R_1 & R_2 has some common activities. Mathematically,
  $\max(\text{Common}(B_1,B_2,\ldots,B_3,\ldots,B_n))\text{iff}R_1 \wedge R_2 \wedge R_3 \wedge \ldots\ldots \wedge R_n \neq \phi$

- **Uncommon Behavior:** Uncommon behavior are the uncommon activities of the user apart from the common behavior. Any activity of a user will be considered as an uncommon behavior that are not in the common behavior. For different data set the uncommon behavior will be different. Uncommon behavior maybe one activity or more than one activities.

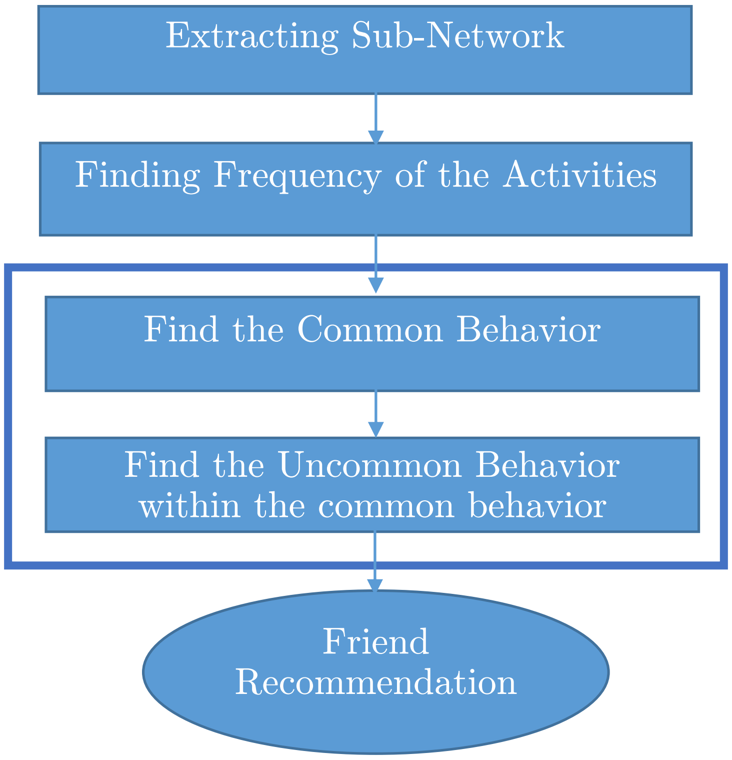
Suppose for the students of IUT, students of department of CSE can be friends. If we consider five users, all from IUT and reading in different years. Then the students of same year will get priority to be friends. So, friends will be recommended if two student are from same year in other words batch mates.





Suppose we are considering a fixed amount of time. By this time users are doing different types of activities. Some of the users used to chat with his friends, some are surfing different groups, some of them are listening songs, some of them watch movies, and some of them are playing different types of games. Among those activities, users usually do this activities one after another. Here we consider one users activities. After logged in he used to check his friend request then see his unread message then see his notification, then play a game. Some other users used to do the same activities but in different sequence. Here the activities are same but the sequences are different. The sequence of activities are the behavior of the system. So, from here we can determine the common and uncommon behavior and these common and uncommon behavior friends can be recommended.

3.2 Framework of the Proposed Approach


Extracting Sub-Network
Finding Frequency of the Activities
Find the Common Behavior
Find the Uncommon Behavior within the common behavior
Friend Recommendation

3.3 Detail Explanation of Proposed System

3.3.1 Extracting Sub-network

Social Networking sites are very large entity with its size. Day by day the size of the network is increasing and as the people are joining there is huge number of information overload happens on those sites. For experiment of our proposed system, we take the whole network of a random individual. After getting the whole network of a client for who are going to suggest friends, we extract the sub-network of 'x' people from the visualized graph.

3.3.2 Finding Frequency of the Activities

Activities are the main category of the thesis. This activities will provide us the behavior of the user. There are so many activities in the social network site. We just take three activities.

- Which type of song they like to hear?
- Which type of movie they like to watch?
- Which type of game they like to play?

We categorizes each activity within three types each.

- Song:
  three types of song. They are
  - Rock song
  - Melody song
  - Folk song
- Movie:
  three types of movie. They are
  - Horror movie
  - Thriller movie
  - Action movie

- Game:
  three types of game. They are
  - Strategy game
  - Puzzle game
  - Real-life game

For each type of activities we find out the maximum frequency of the different kind of activities. We make the network scrutinized by using this frequency where the maximum frequency related activities are there. So the total nine activities come down to three activities where the every categorized activities contains one activity with the maximum frequency.

3.3.3 The user behavior:

To find the desired user behavior we use FP-growth algorithm in our modified dataset. FP-growth algorithm gives us the pattern from the dataset. Among this pattern the desired behavior will be found. From this behavior we can recommend friends.

FP-Growth Algorithm:

FP-Growth works in a divide and conquer way. It requires two scans on the database. FP-Growth first computes a list of frequent items sorted by frequency in descending order (F-List) during its first database scan. In its second scan, the database is compressed into a FP-tree. Then FP-Growth starts to mine the FP-tree for each item whose support is larger than α by recursively building its conditional FP-tree. The algorithm performs mining recursively on FP-tree. The problem of finding frequent item sets is converted to searching and constructing trees recursively.

Common Behavior:

We will get the user behaviors by the FP-growth algorithm. From the acquired pattern we will find out the common behavior by using the maximum number of frequency for any pattern with the single activities of the user behavior. Naturally single activities will give the highest frequency value. This common behavior will help us to find the uncommon behavior.

Uncommon Behavior:

After finding the common behavior our next goal is to find the uncommon behavior. If we recommend friend only using the common behavior it will just like the natural recommendation process using only one feature like fof (friend of friend). So for more integrity we use uncommon behavior. This uncommon activities is different from the natural uncommon process. We actually find out the less no of frequency of other two activities from the user behavior containing the common user behavior. We call it uncommon because it appears less among the user whose have similar interest early. Actually this behavior can be considered as their unique behavior.

3.3.4 Friend Recommendation:

At the last step we recommend the users with the user behavior found by us. We can take any random user from any other sub network and recommend them as friend. In this sub network we can also recommend them as friend.

Chapter 4
Conclusion

4.1 Contribution

The major contribution of this thesis that is we define the user behavior in online social network. From the user behavior we find the common and uncommon behaviors among a set of users. This is a unique idea to create a system that recommendation system based on user behavior. On the other hand this system will be more efficient from the other system. Here we find the frequency of the activities of some users. Then we define the common and uncommon behavior. After finding the common behavior we find the common behavior among the uncommon behavior of the user. Based on that we recommend the friends. Suppose one person is recommended 8-10 person to be his or her friend by any friend recommendation system. But in our recommendation system in this case the less number will be recommended and their activities will be very much similar to that person. So this will be more efficient way to recommend friends in online social network.

4.2 Future Work

In this thesis we try to define the user behavior in online social network. From that we find the common and uncommon behavior. Here only three activities are used. So we try to use more activities to make this research more efficient. Apart from that we will try to use research in the original dataset from the online social network sites. And the most important thing we will do that we have to compare our recommendation system with the existing systems.

References

---

[1] Boyd, Danah; Ellison, Nicole, ”Social Network Sites: Definition, History, and Scholarship”, Journal of Computer- Mediated Communication, Vol.13, No.1, 2007

[2] Catanese, Salvatore Meo, Pasquale De Ferrara, Emilio Fiumara, Giacomo. ”Analyzing the Facebook Friendship Graph.”

[3] Gediminas Adomavicius and Alexander Tuzhilin. “Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.”

[4] Krulwich, B., and Burkey, C., “Learning user information interests through hextraction of semantically significant phrases,” In Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access, Stanford, Calif., March 1996.

[5] Lang, K., “Newsweeder: Learning to filter netnews,” In Proceedings of the 12th International Conference on Machine Learning, Tahoe City, Calif., USA, 1995.

[6] Herlocker, J. L., Konstan, J. A., and Riedl, J, ”Explaining Collaborative Filtering Recommendations,” In Proceeding of ACM 2000 Conference on Computer Supported Cooperative Work, 2000.

[7] Melville, Prem Mooney, Raymond J Nagarajan, Ramadass. “Content-Boosted Collaborative Filtering for Improved Recommendations”. Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002), pp. 187-192, Edmonton, Canada, July 2002.

[8] Long Jin et al (2013), “Understanding User Behavior in Online Social Networks: A Survey”

[9] C. Wilson et al (2009), “User Interactions in Social Networks and Their Implications,”

[10] Marcelo Maia et al, **“**Identifying User Behavior in Online Social Networks”

[11] Xing Xie (2010), “Potential Friend Recommendation in Online Social Network”

[12] Francis T. O'Donovan et al (2013), "Characterizing user behavior and information propagation on a social multimedia network"

[13] Md. Nafiz Hamid (2014), "A cohesion-based friend-recommendation system"

[14] Micheal Moricz et al (2010), "PYMK: friend recommendation at myspace"

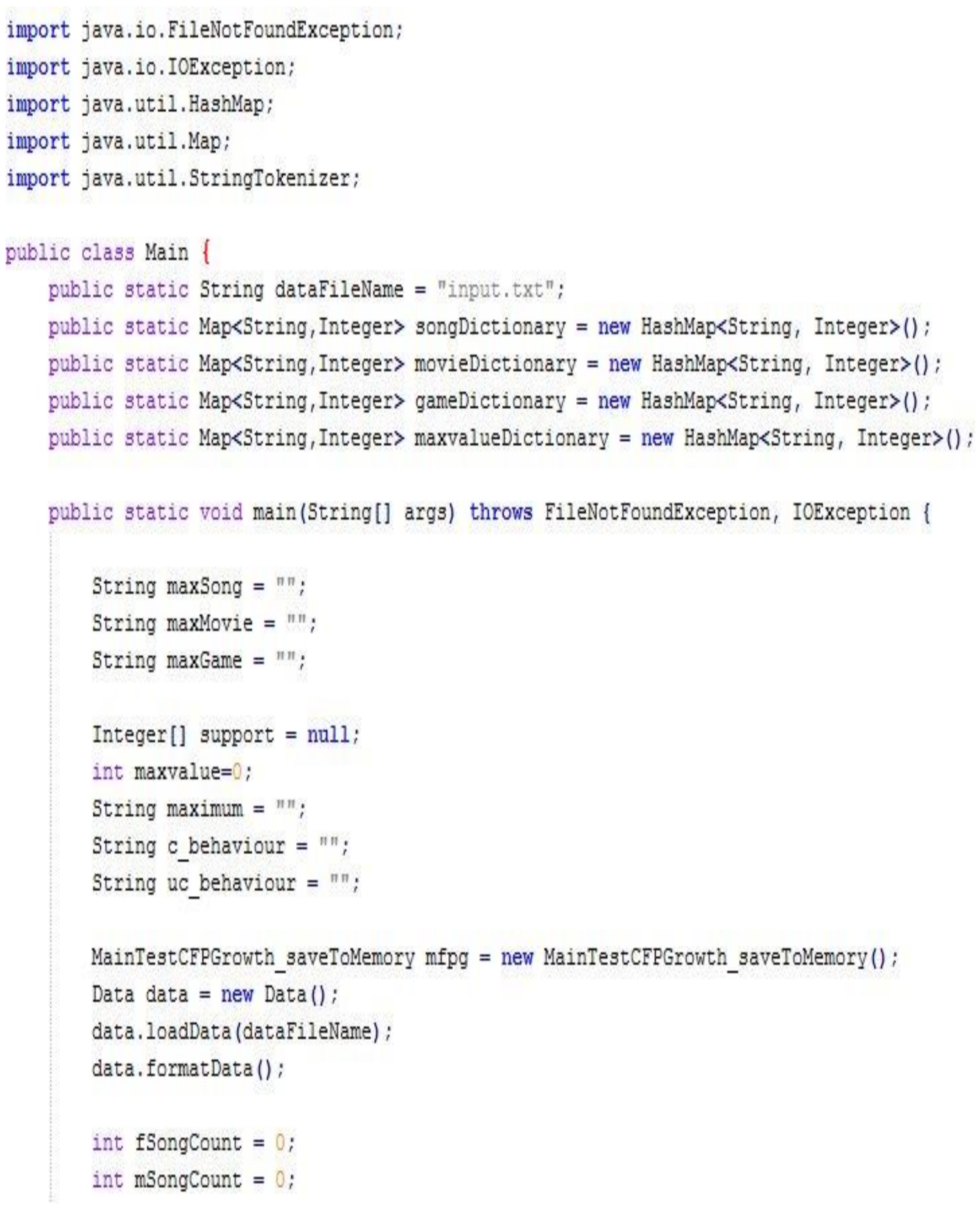
[15] Hajime Hotta et al (2007), "User Profiling System Using Social Networks for Recommendation"

[16] Zhiwei Deng et al (2012), "Personalized friend recommendation in social network based on clustering method"

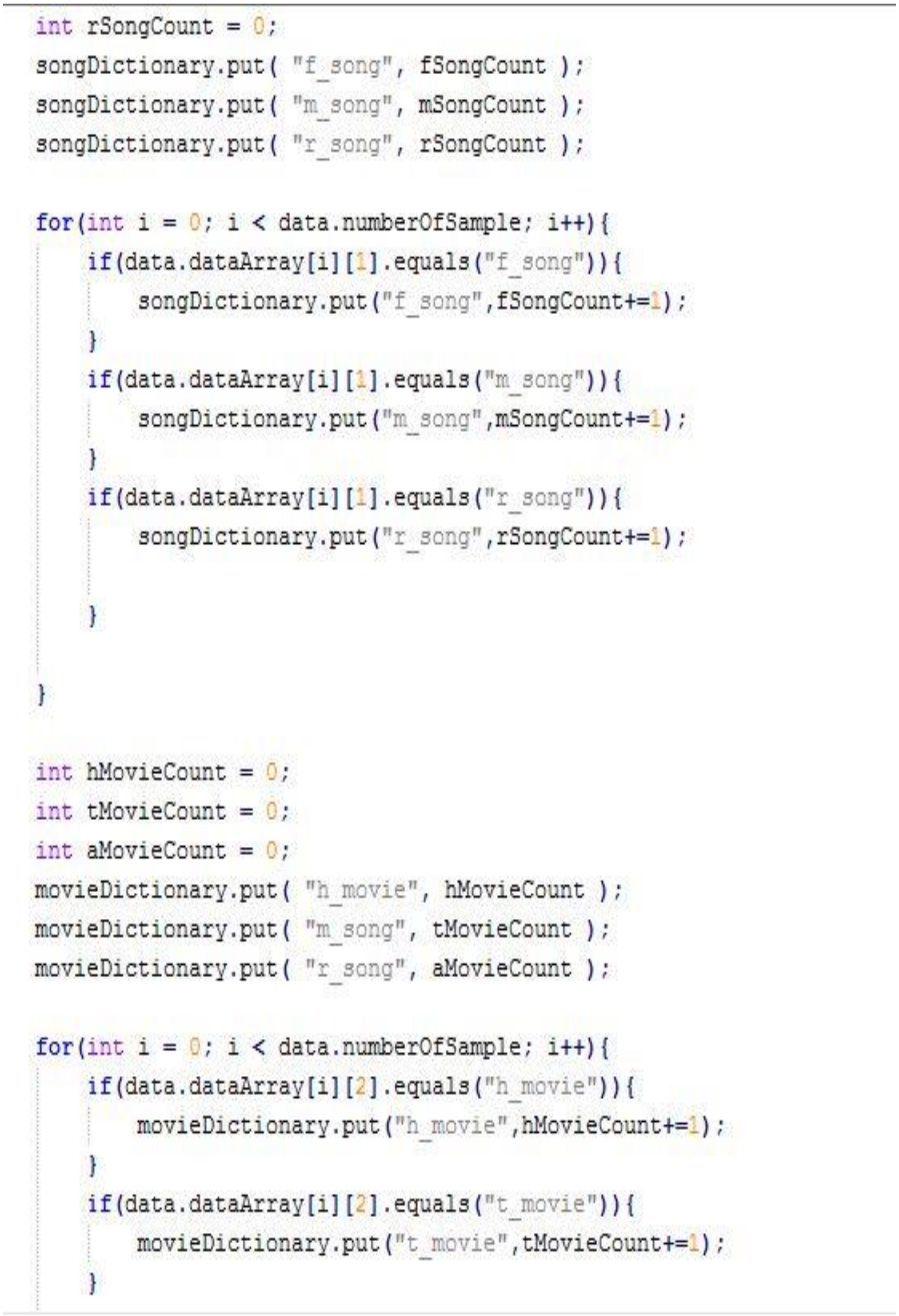
[17] Yu Zheng, Yukun Chen, Xing Xie, Wei-Ying Ma "GeoLife2.0: A Location-Based Social Networking Service‖."

Code

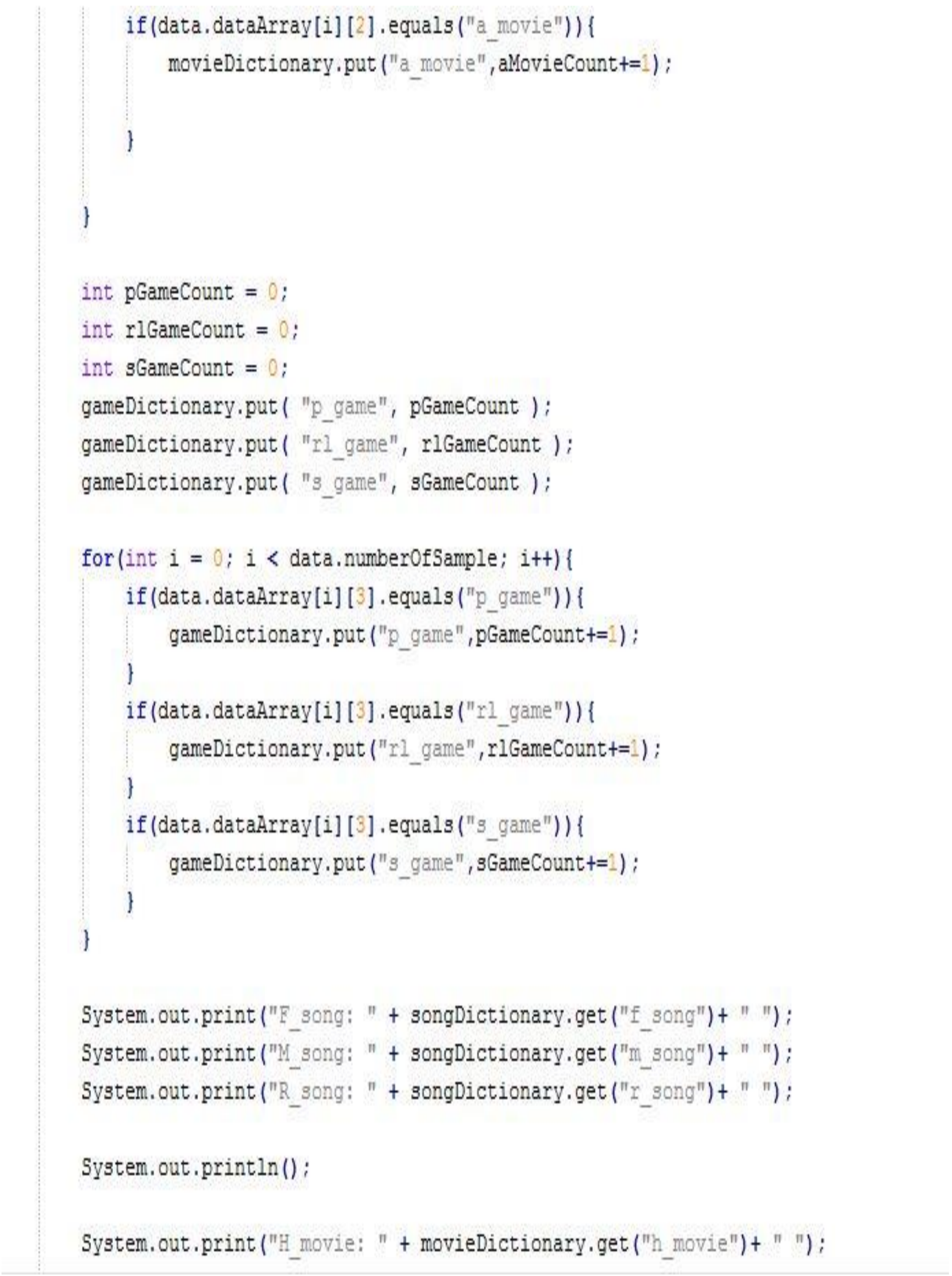
Main.java



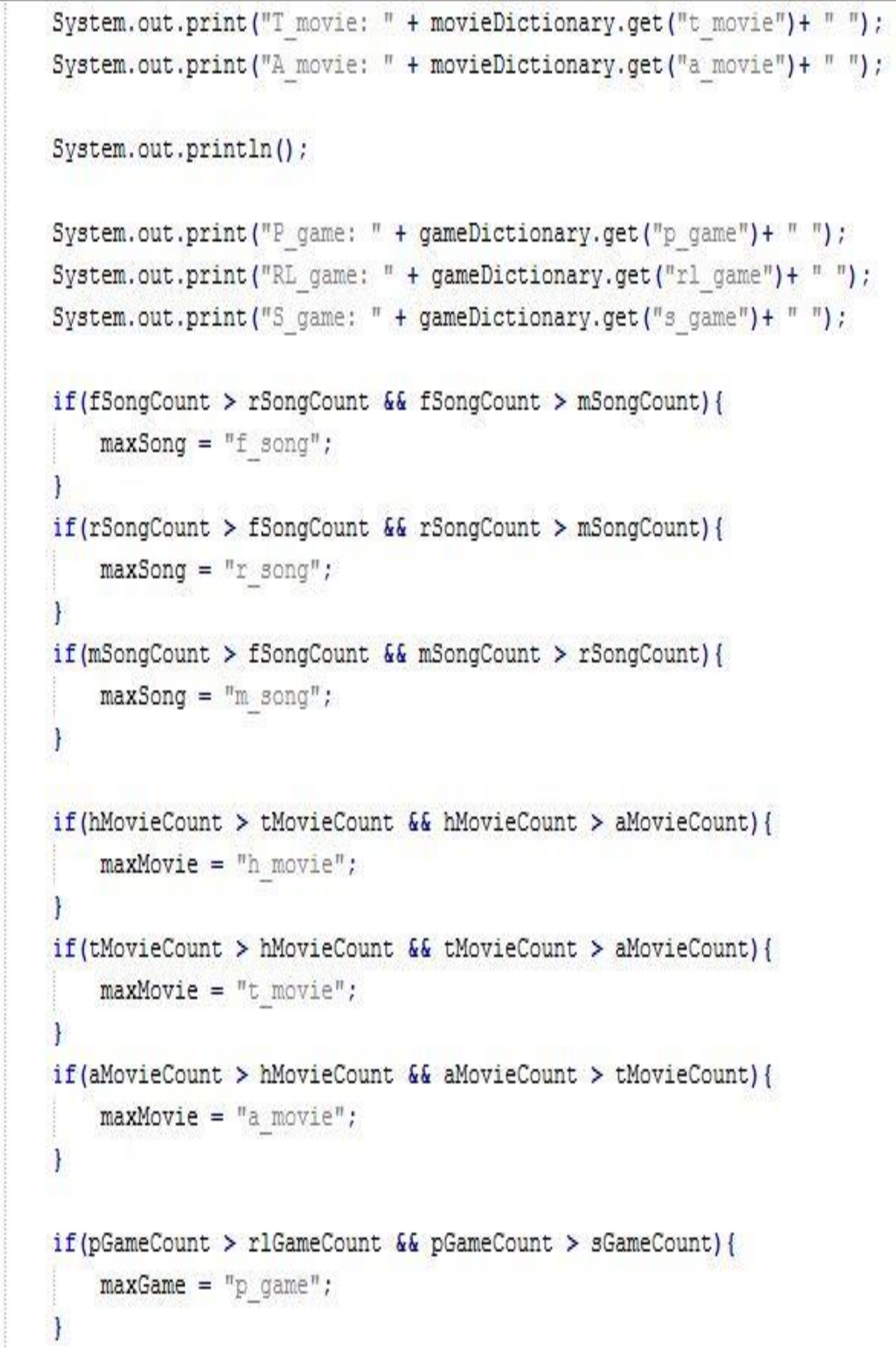
```java
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.StringTokenizer;

public class Main {
    public static String dataFileName = "input.txt";
    public static Map<String,Integer> songDictionary = new HashMap<String, Integer>();
    public static Map<String,Integer> movieDictionary = new HashMap<String, Integer>();
    public static Map<String,Integer> gameDictionary = new HashMap<String, Integer>();
    public static Map<String,Integer> maxvalueDictionary = new HashMap<String, Integer>();

    public static void main(String[] args) throws FileNotFoundException, IOException {

        String maxSong = "";
        String maxMovie = "";
        String maxGame = "";

        Integer[] support = null;
        int maxvalue=0;
        String maximum = "";
        String c_behaviour = "";
        String uc_behaviour = "";

        MainTestCFPGrowth_saveToMemory mfpg = new MainTestCFPGrowth_saveToMemory();
        Data data = new Data();
        data.loadData(dataFileName);
        data.formatData();

        int fSongCount = 0;
        int mSongCount = 0;
```



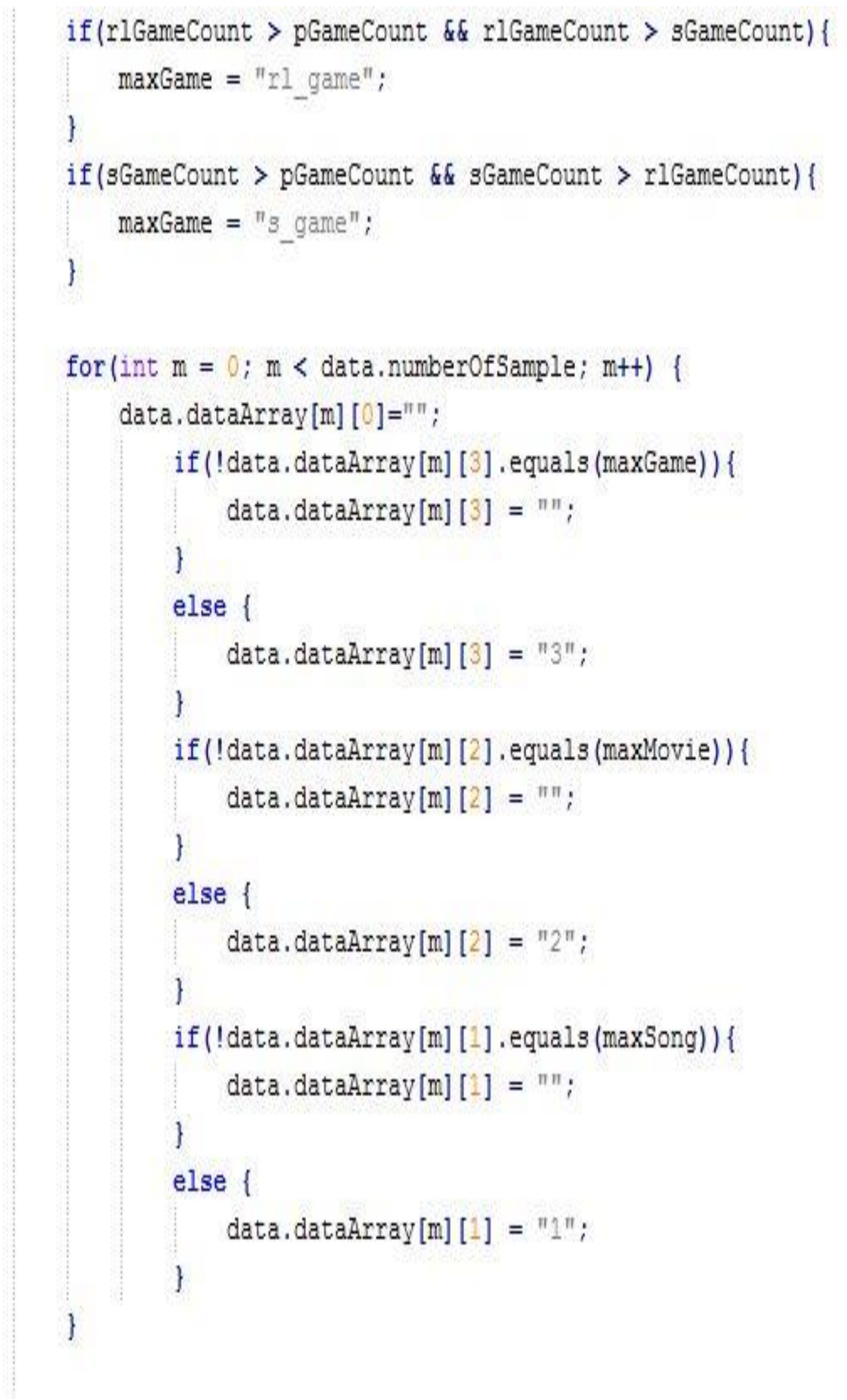
```
int rSongCount = 0;
songDictionary.put( "f_song", fSongCount );
songDictionary.put( "m_song", mSongCount );
songDictionary.put( "r_song", rSongCount );

for(int i = 0; i < data.numberOfSample; i++){
    if(data.dataArray[i][1].equals("f_song")){
        songDictionary.put("f_song",fSongCount+=1);
    }
    if(data.dataArray[i][1].equals("m_song")){
        songDictionary.put("m_song",mSongCount+=1);
    }
    if(data.dataArray[i][1].equals("r_song")){
        songDictionary.put("r_song",rSongCount+=1);

    }

}

int hMovieCount = 0;
int tMovieCount = 0;
int aMovieCount = 0;
movieDictionary.put( "h_movie", hMovieCount );
movieDictionary.put( "m_song", tMovieCount );
movieDictionary.put( "r_song", aMovieCount );

for(int i = 0; i < data.numberOfSample; i++){
    if(data.dataArray[i][2].equals("h_movie")){
        movieDictionary.put("h_movie",hMovieCount+=1);
    }
    if(data.dataArray[i][2].equals("t_movie")){
        movieDictionary.put("t_movie",tMovieCount+=1);
    }
```



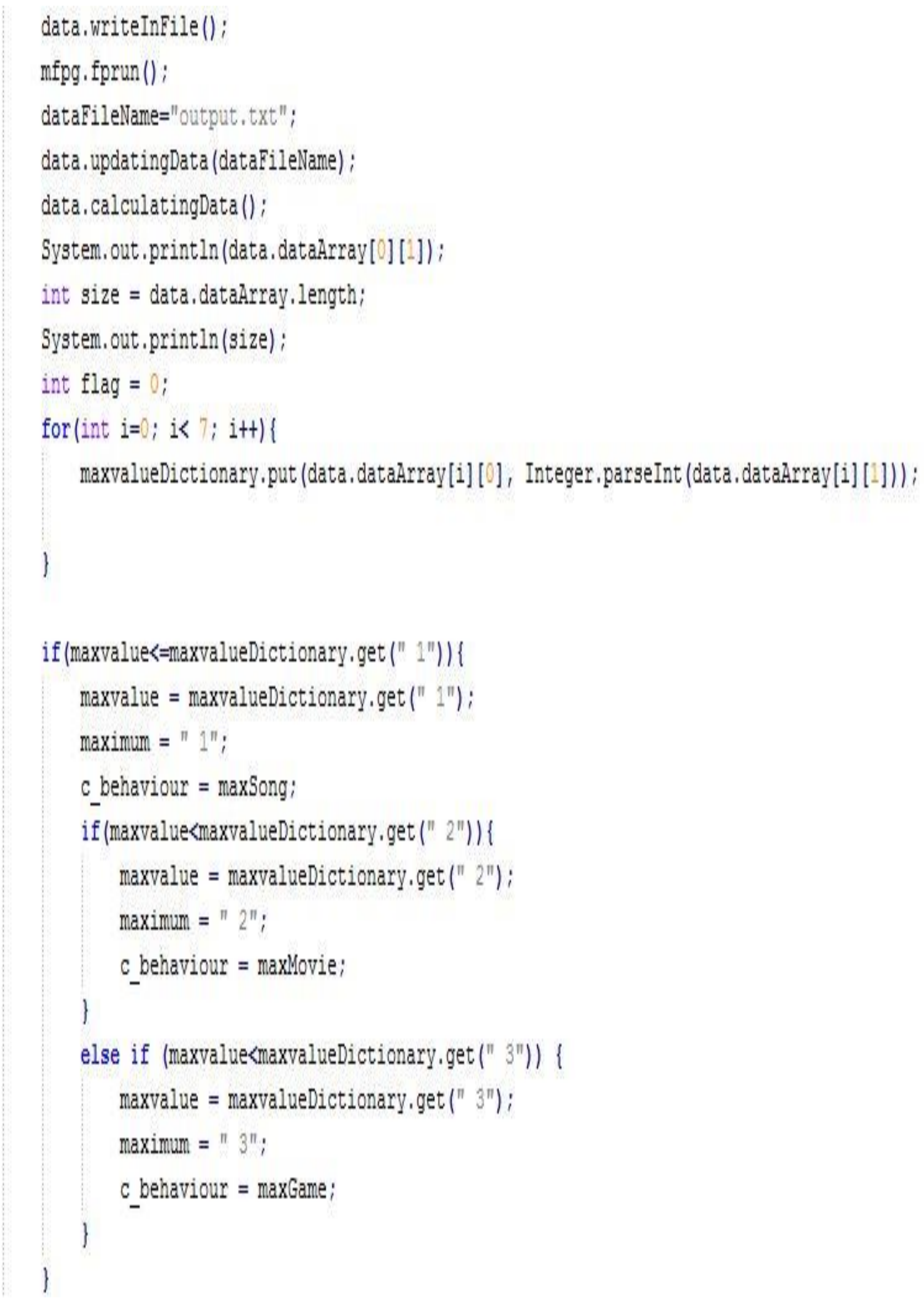
```
        if(data.dataArray[i][2].equals("a_movie")){
            movieDictionary.put("a_movie",aMovieCount+=1);


        }


    }

    int pGameCount = 0;
    int rlGameCount = 0;
    int sGameCount = 0;
    gameDictionary.put( "p_game", pGameCount );
    gameDictionary.put( "rl_game", rlGameCount );
    gameDictionary.put( "s_game", sGameCount );

    for(int i = 0; i < data.numberOfSample; i++){
        if(data.dataArray[i][3].equals("p_game")){
            gameDictionary.put("p_game",pGameCount+=1);
        }
        if(data.dataArray[i][3].equals("rl_game")){
            gameDictionary.put("rl_game",rlGameCount+=1);
        }
        if(data.dataArray[i][3].equals("s_game")){
            gameDictionary.put("s_game",sGameCount+=1);
        }
    }

    System.out.print("F_song: " + songDictionary.get("f_song")+ " ");
    System.out.print("M_song: " + songDictionary.get("m_song")+ " ");
    System.out.print("R_song: " + songDictionary.get("r_song")+ " ");

    System.out.println();

    System.out.print("H_movie: " + movieDictionary.get("h_movie")+ " ");
```



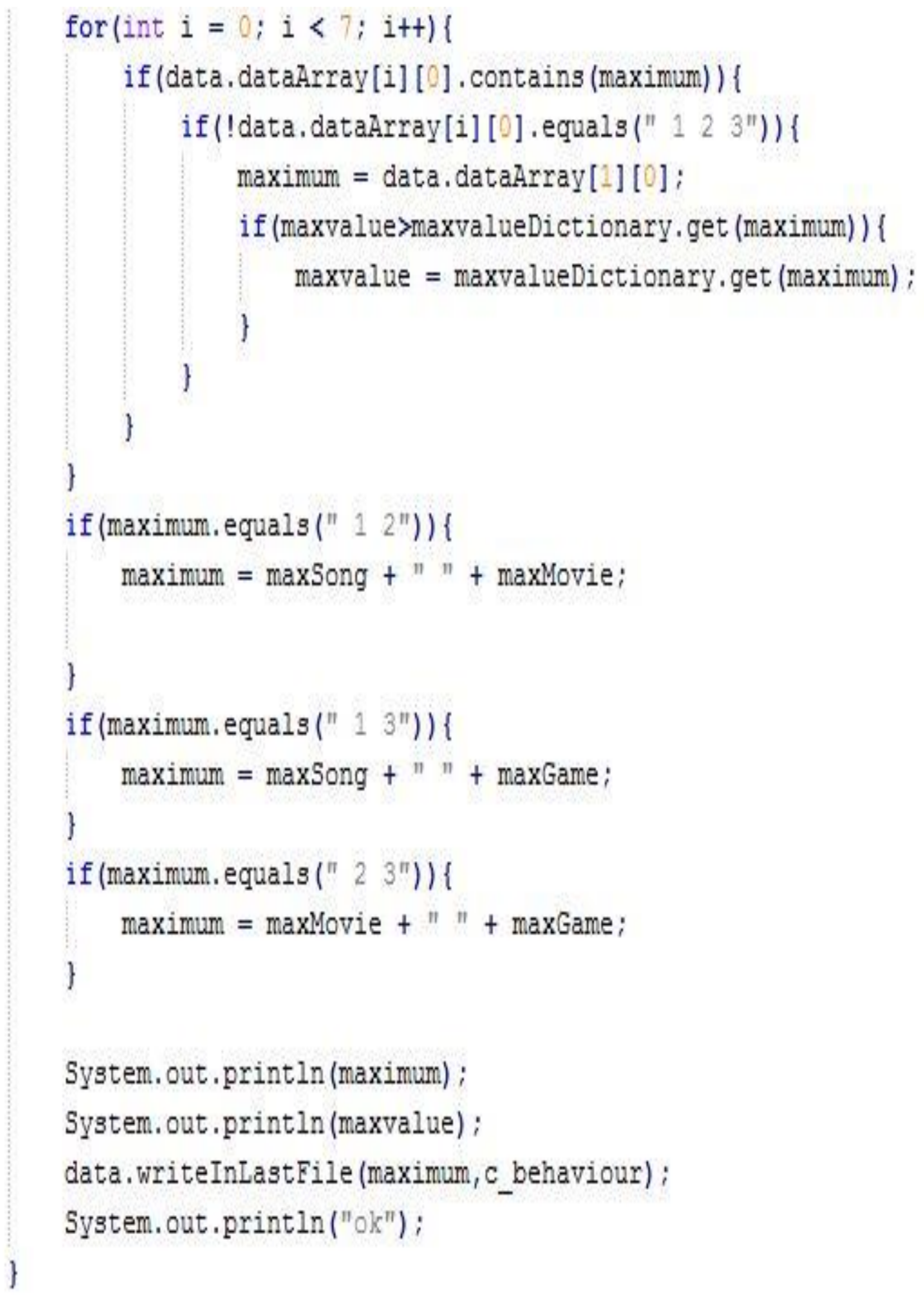
```
System.out.print("T_movie: " + movieDictionary.get("t_movie")+ " ");
System.out.print("A_movie: " + movieDictionary.get("a_movie")+ " ");

System.out.println();

System.out.print("P_game: " + gameDictionary.get("p_game")+ " ");
System.out.print("RL_game: " + gameDictionary.get("rl_game")+ " ");
System.out.print("S_game: " + gameDictionary.get("s_game")+ " ");

if(fSongCount > rSongCount && fSongCount > mSongCount){
    maxSong = "f_song";
}
if(rSongCount > fSongCount && rSongCount > mSongCount){
    maxSong = "r_song";
}
if(mSongCount > fSongCount && mSongCount > rSongCount){
    maxSong = "m_song";
}

if(hMovieCount > tMovieCount && hMovieCount > aMovieCount){
    maxMovie = "h_movie";
}
if(tMovieCount > hMovieCount && tMovieCount > aMovieCount){
    maxMovie = "t_movie";
}
if(aMovieCount > hMovieCount && aMovieCount > tMovieCount){
    maxMovie = "a_movie";
}

if(pGameCount > rlGameCount && pGameCount > sGameCount){
    maxGame = "p_game";
}
```



```
if(rlGameCount > pGameCount && rlGameCount > sGameCount){
    maxGame = "rl_game";
}
if(sGameCount > pGameCount && sGameCount > rlGameCount){
    maxGame = "s_game";
}

for(int m = 0; m < data.numberOfSample; m++) {
    data.dataArray[m][0]="";
        if(!data.dataArray[m][3].equals(maxGame)){
            data.dataArray[m][3] = "";
        }
        else {
            data.dataArray[m][3] = "3";
        }
        if(!data.dataArray[m][2].equals(maxMovie)){
            data.dataArray[m][2] = "";
        }
        else {
            data.dataArray[m][2] = "2";
        }
        if(!data.dataArray[m][1].equals(maxSong)){
            data.dataArray[m][1] = "";
        }
        else {
            data.dataArray[m][1] = "1";
        }
}
```

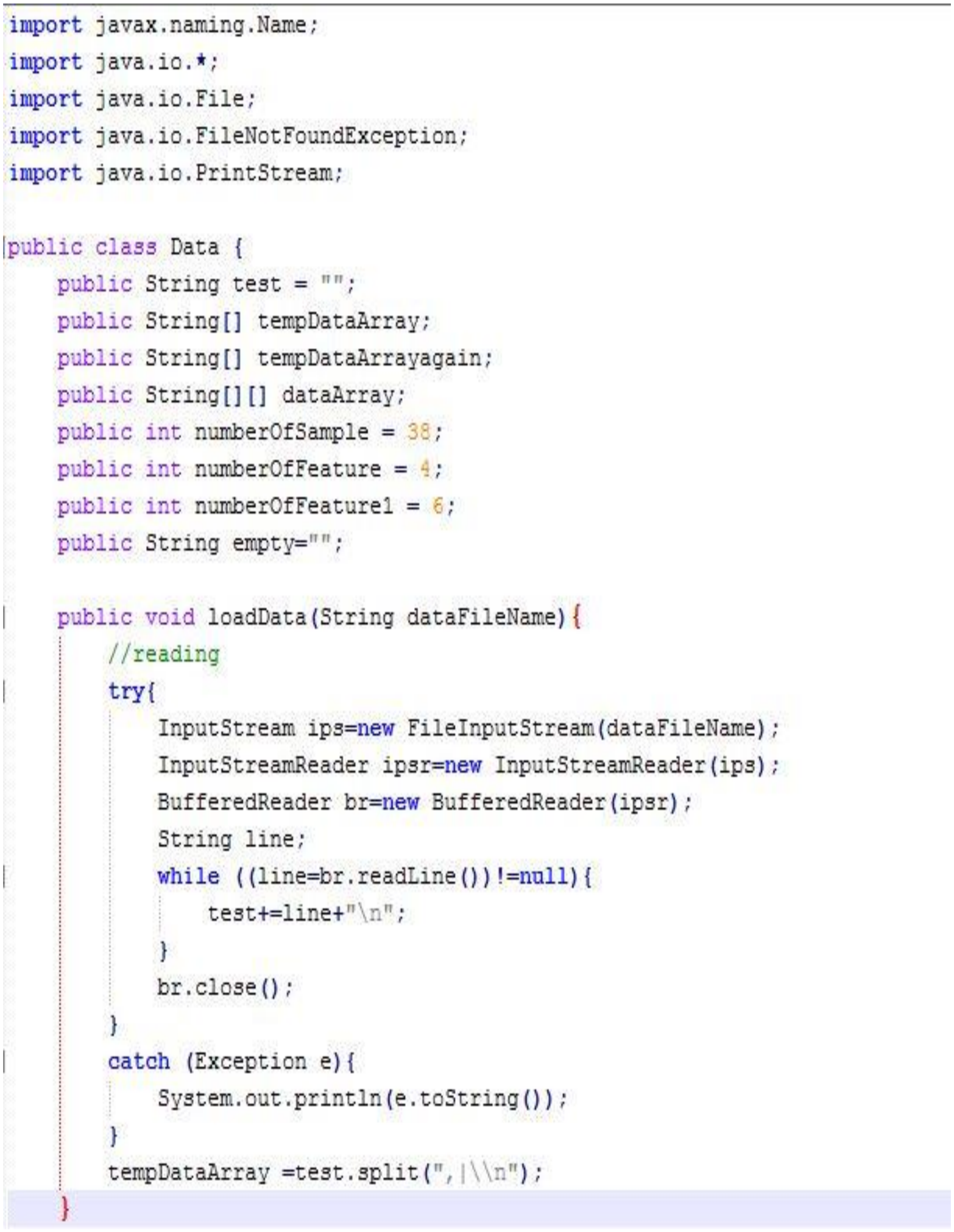


```java
data.writeInFile();
mfpg.fprun();
dataFileName="output.txt";
data.updatingData(dataFileName);
data.calculatingData();
System.out.println(data.dataArray[0][1]);
int size = data.dataArray.length;
System.out.println(size);
int flag = 0;
for(int i=0; i< 7; i++){
    maxvalueDictionary.put(data.dataArray[i][0], Integer.parseInt(data.dataArray[i][1]));


}


if(maxvalue<=maxvalueDictionary.get(" 1")){
    maxvalue = maxvalueDictionary.get(" 1");
    maximum = " 1";
    c_behaviour = maxSong;
    if(maxvalue<maxvalueDictionary.get(" 2")){
        maxvalue = maxvalueDictionary.get(" 2");
        maximum = " 2";
        c_behaviour = maxMovie;
    }
    else if (maxvalue<maxvalueDictionary.get(" 3")) {
        maxvalue = maxvalueDictionary.get(" 3");
        maximum = " 3";
        c_behaviour = maxGame;
    }
}
```

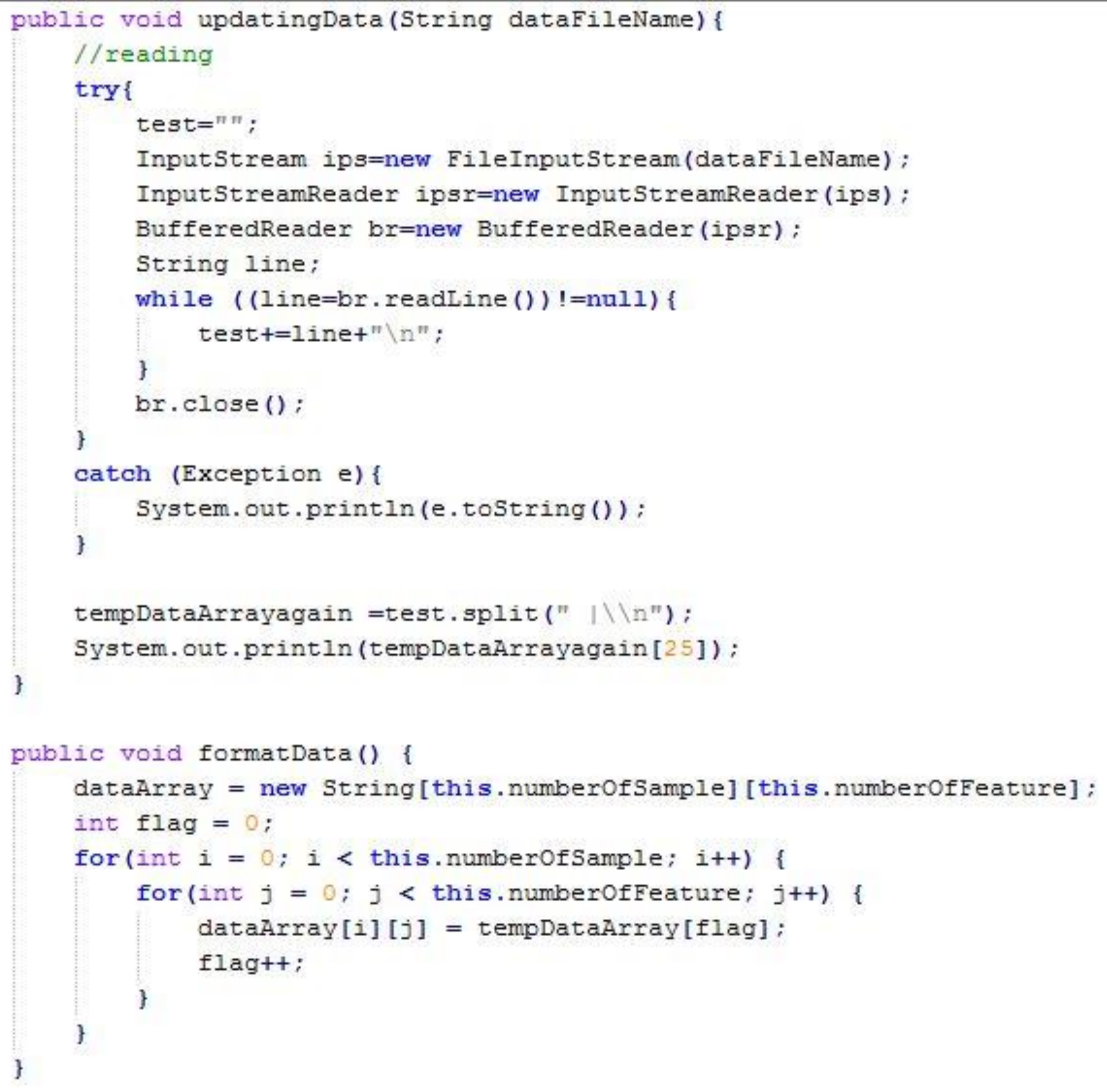


```java
    for(int i = 0; i < 7; i++){
        if(data.dataArray[i][0].contains(maximum)){
            if(!data.dataArray[i][0].equals(" 1 2 3")){
                maximum = data.dataArray[1][0];
                if(maxvalue>maxvalueDictionary.get(maximum)){
                    maxvalue = maxvalueDictionary.get(maximum);
                }
            }
        }
    }
    if(maximum.equals(" 1 2")){
        maximum = maxSong + " " + maxMovie;

    }
    if(maximum.equals(" 1 3")){
        maximum = maxSong + " " + maxGame;
    }
    if(maximum.equals(" 2 3")){
        maximum = maxMovie + " " + maxGame;
    }

    System.out.println(maximum);
    System.out.println(maxvalue);
    data.writeInLastFile(maximum,c_behaviour);
    System.out.println("ok");
}
```

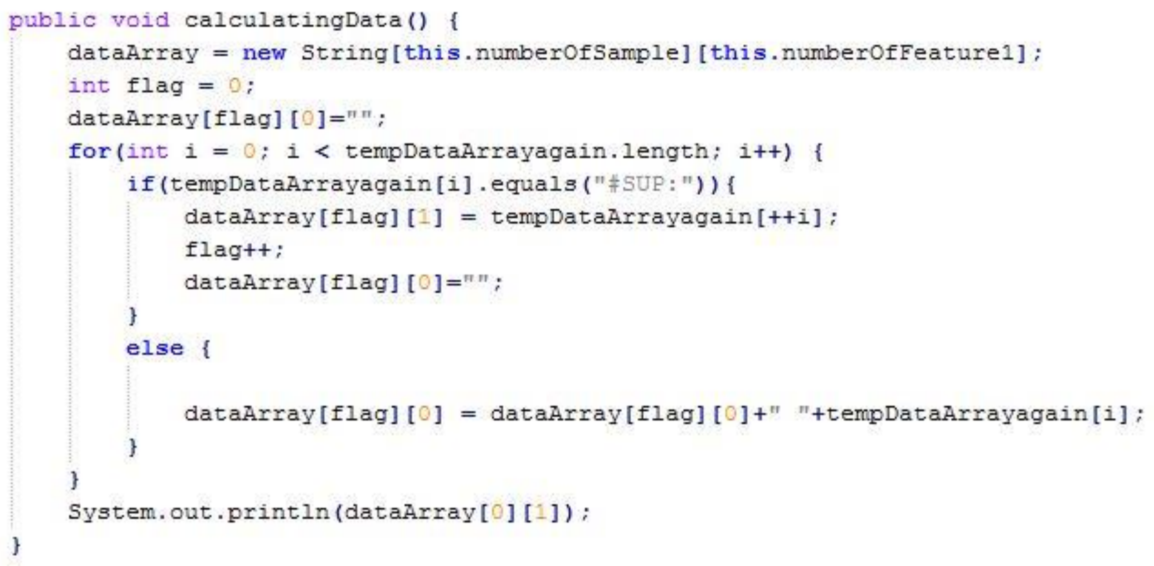
Data.java



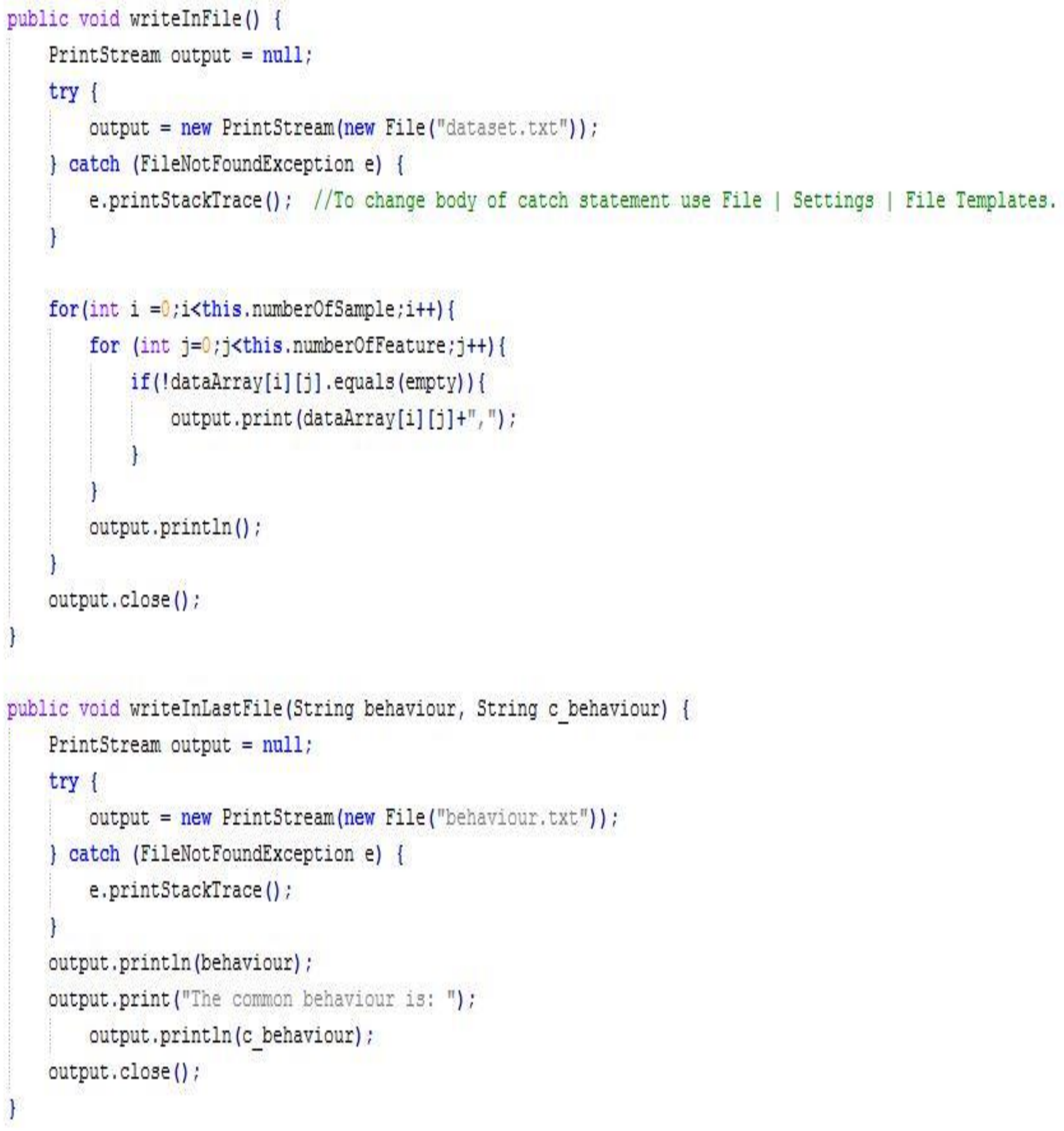
```java
import javax.naming.Name;
import java.io.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintStream;

public class Data {
    public String test = "";
    public String[] tempDataArray;
    public String[] tempDataArrayagain;
    public String[][] dataArray;
    public int numberOfSample = 38;
    public int numberOfFeature = 4;
    public int numberOfFeature1 = 6;
    public String empty="";

    public void loadData(String dataFileName){
        //reading
        try{
            InputStream ips=new FileInputStream(dataFileName);
            InputStreamReader ipsr=new InputStreamReader(ips);
            BufferedReader br=new BufferedReader(ipsr);
            String line;
            while ((line=br.readLine())!=null){
                test+=line+"\n";
            }
            br.close();
        }
        catch (Exception e){
            System.out.println(e.toString());
        }
        tempDataArray =test.split(",|\\n");
    }
```



```
public void updatingData(String dataFileName){
    //reading
    try{
        test="";
        InputStream ips=new FileInputStream(dataFileName);
        InputStreamReader ipsr=new InputStreamReader(ips);
        BufferedReader br=new BufferedReader(ipsr);
        String line;
        while ((line=br.readLine())!=null){
            test+=line+"\n";
        }
        br.close();
    }
    catch (Exception e){
        System.out.println(e.toString());
    }

    tempDataArrayagain =test.split(" |\\n");
    System.out.println(tempDataArrayagain[25]);
}

public void formatData() {
    dataArray = new String[this.numberOfSample][this.numberOfFeature];
    int flag = 0;
    for(int i = 0; i < this.numberOfSample; i++) {
        for(int j = 0; j < this.numberOfFeature; j++) {
            dataArray[i][j] = tempDataArray[flag];
            flag++;
        }
    }
}
```

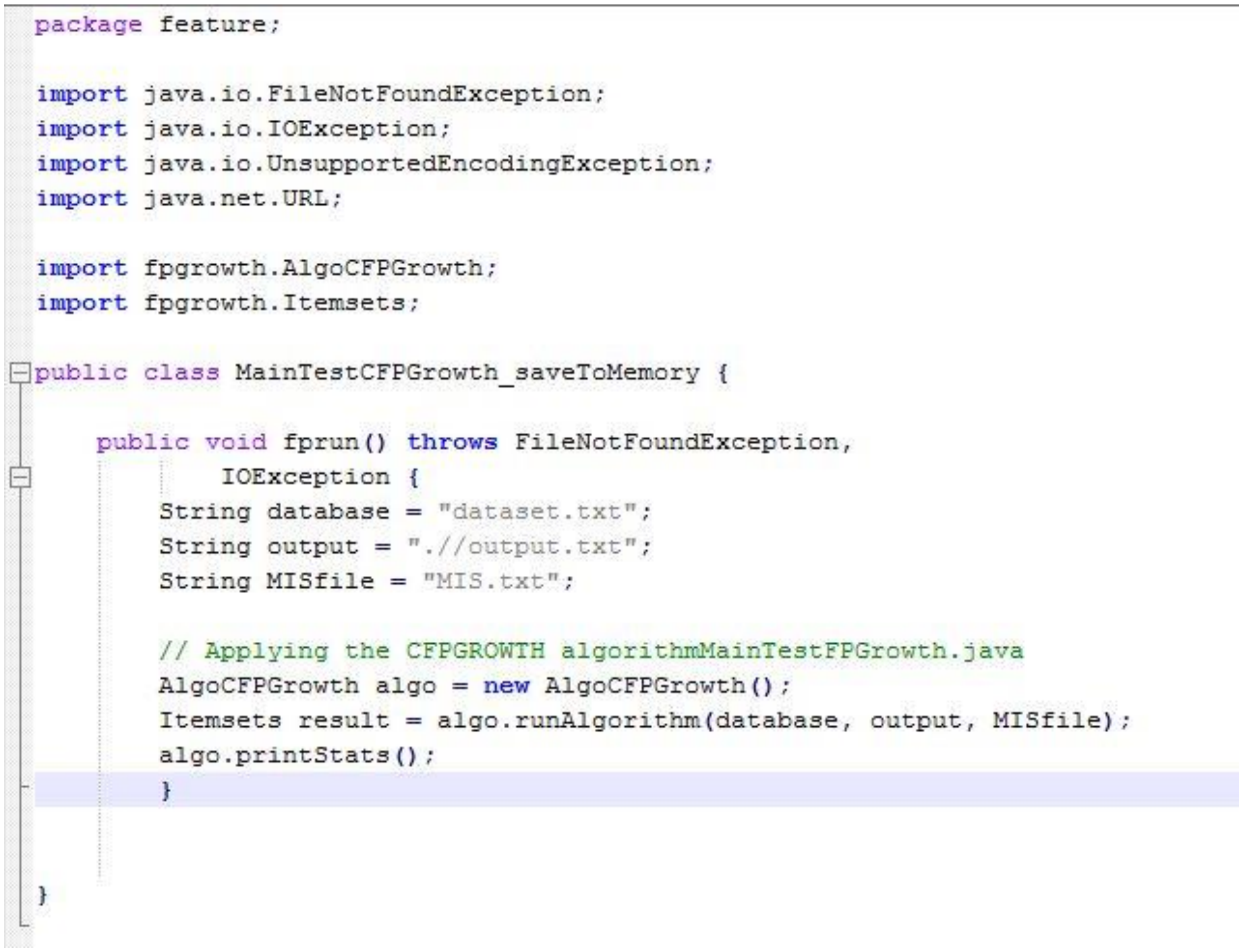


```
public void calculatingData() {
    dataArray = new String[this.numberOfSample][this.numberOfFeature1];
    int flag = 0;
    dataArray[flag][0]="";
    for(int i = 0; i < tempDataArrayagain.length; i++) {
        if(tempDataArrayagain[i].equals("#SUP:")){
            dataArray[flag][1] = tempDataArrayagain[++i];
            flag++;
            dataArray[flag][0]="";
        }
        else {

            dataArray[flag][0] = dataArray[flag][0]+" "+tempDataArrayagain[i];
        }
    }
    System.out.println(dataArray[0][1]);
}
```



```
public void writeInFile() {
    PrintStream output = null;
    try {
        output = new PrintStream(new File("dataset.txt"));
    } catch (FileNotFoundException e) {
        e.printStackTrace();  //To change body of catch statement use File | Settings | File Templates.
    }

    for(int i =0;i<this.numberOfSample;i++){
        for (int j=0;j<this.numberOfFeature;j++){
            if(!dataArray[i][j].equals(empty)){
                output.print(dataArray[i][j]+",");
            }
        }
        output.println();
    }
    output.close();
}

public void writeInLastFile(String behaviour, String c_behaviour) {
    PrintStream output = null;
    try {
        output = new PrintStream(new File("behaviour.txt"));
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
    output.println(behaviour);
    output.print("The common behaviour is: ");
        output.println(c_behaviour);
    output.close();
}
```

MainTestCFPGrowth˙saveToMemory.java



```
package feature;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URL;

import fpgrowth.AlgoCFPGrowth;
import fpgrowth.Itemsets;

public class MainTestCFPGrowth_saveToMemory {

    public void fprun() throws FileNotFoundException,
            IOException {
        String database = "dataset.txt";
        String output = ".//output.txt";
        String MISfile = "MIS.txt";

        // Applying the CFPGROWTH algorithmMainTestFPGrowth.java
        AlgoCFPGrowth algo = new AlgoCFPGrowth();
        Itemsets result = algo.runAlgorithm(database, output, MISfile);
        algo.printStats();
        }


}
```