

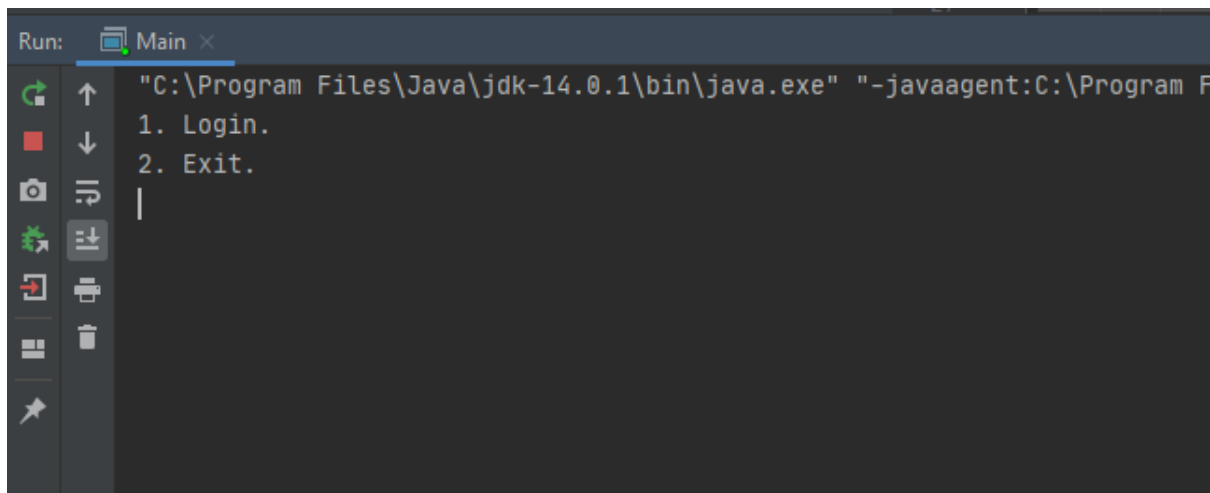
Company Information System

This Project Developed by
Ozan Yücel and Cem Ceylan

20190602043 - Ozan Yücel

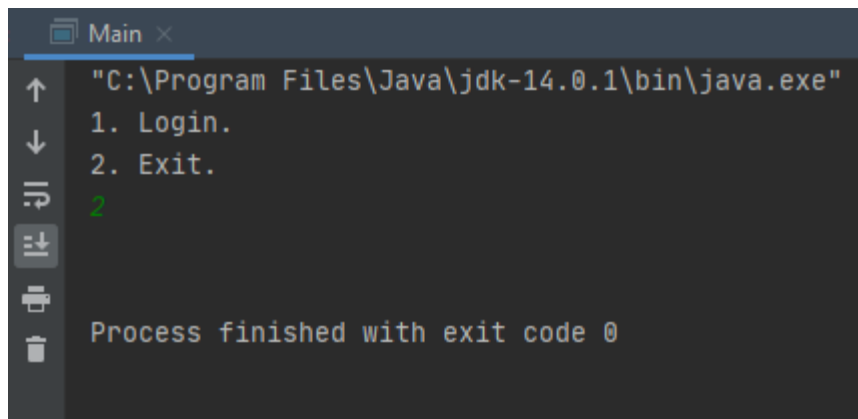
20190602011 - Cem Ceylan

Our software is a console-based company information system. There are 4 user roles which are Administrator, Manager, Doctor and Worker. At the start program welcomes you with a short menu.



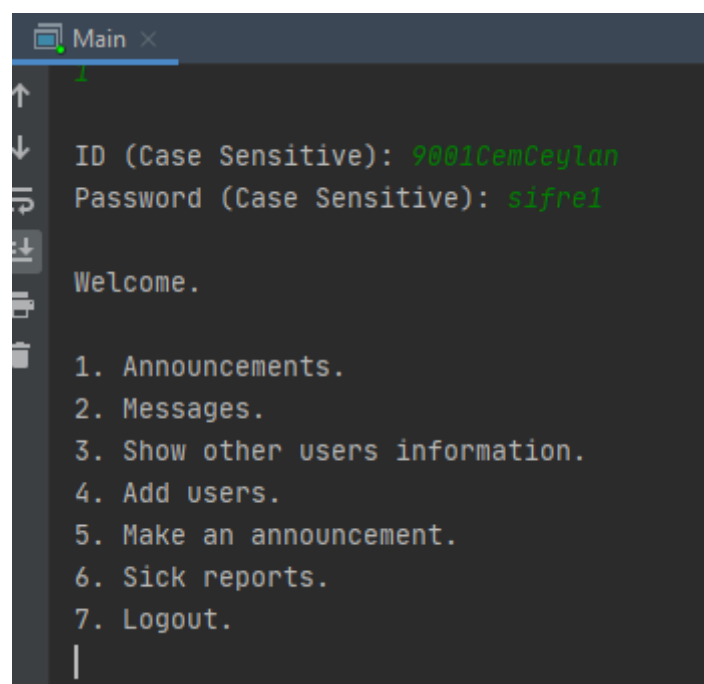
```
Run: Main x
"C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:C:\Program F
1. Login.
2. Exit.
|
```

If you click to exit program will be terminated.



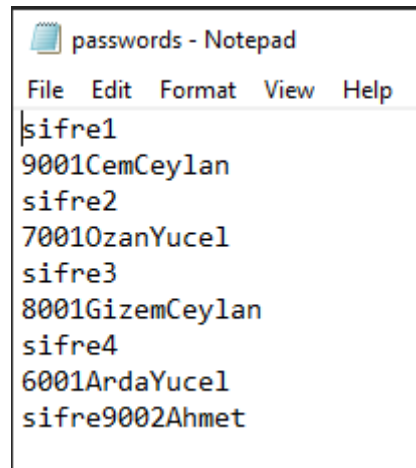
```
Main x
"C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "
1. Login.
2. Exit.
2
Process finished with exit code 0
```

If you click to Login a simple login screen welcomes you. An example for Administrator Menu;



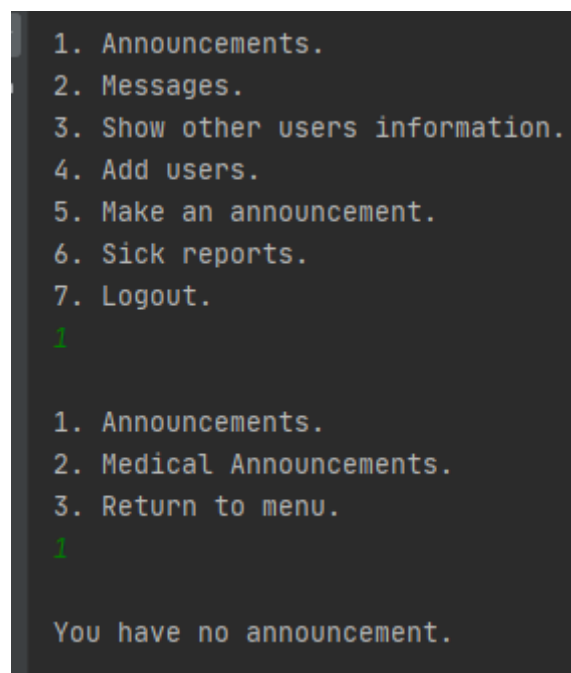
```
Main x
1
ID (Case Sensitive): 9001CemCeylan
Password (Case Sensitive): sifre1
Welcome.
1. Announcements.
2. Messages.
3. Show other users information.
4. Add users.
5. Make an announcement.
6. Sick reports.
7. Logout.
|
```

We have created 4 different members with 4 different roles for the test. Every user's information gets stored in text files so when program gets restarted, no data gets lost.



```
File Edit Format View Help
sifre1
9001CemCeylan
sifre2
70010zanYucel
sifre3
8001GizemCeylan
sifre4
6001ArdaYucel
sifre9002Ahmet
```

When we enter an input for announcement menu, we will see a sub-menu for announcements. Because of no announcement have been set before, we will get a message like: "You have no announcement."

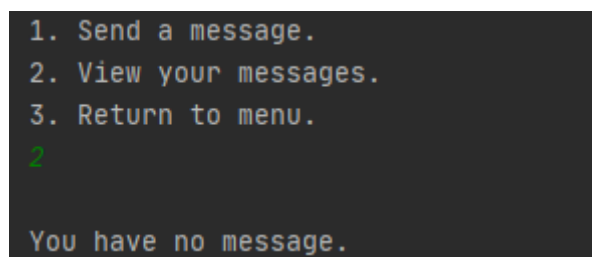


```
1. Announcements.
2. Messages.
3. Show other users information.
4. Add users.
5. Make an announcement.
6. Sick reports.
7. Logout.
1

1. Announcements.
2. Medical Announcements.
3. Return to menu.
1

You have no announcement.
```

If we enter an input for messages, again a little sub-menu will come out for us.



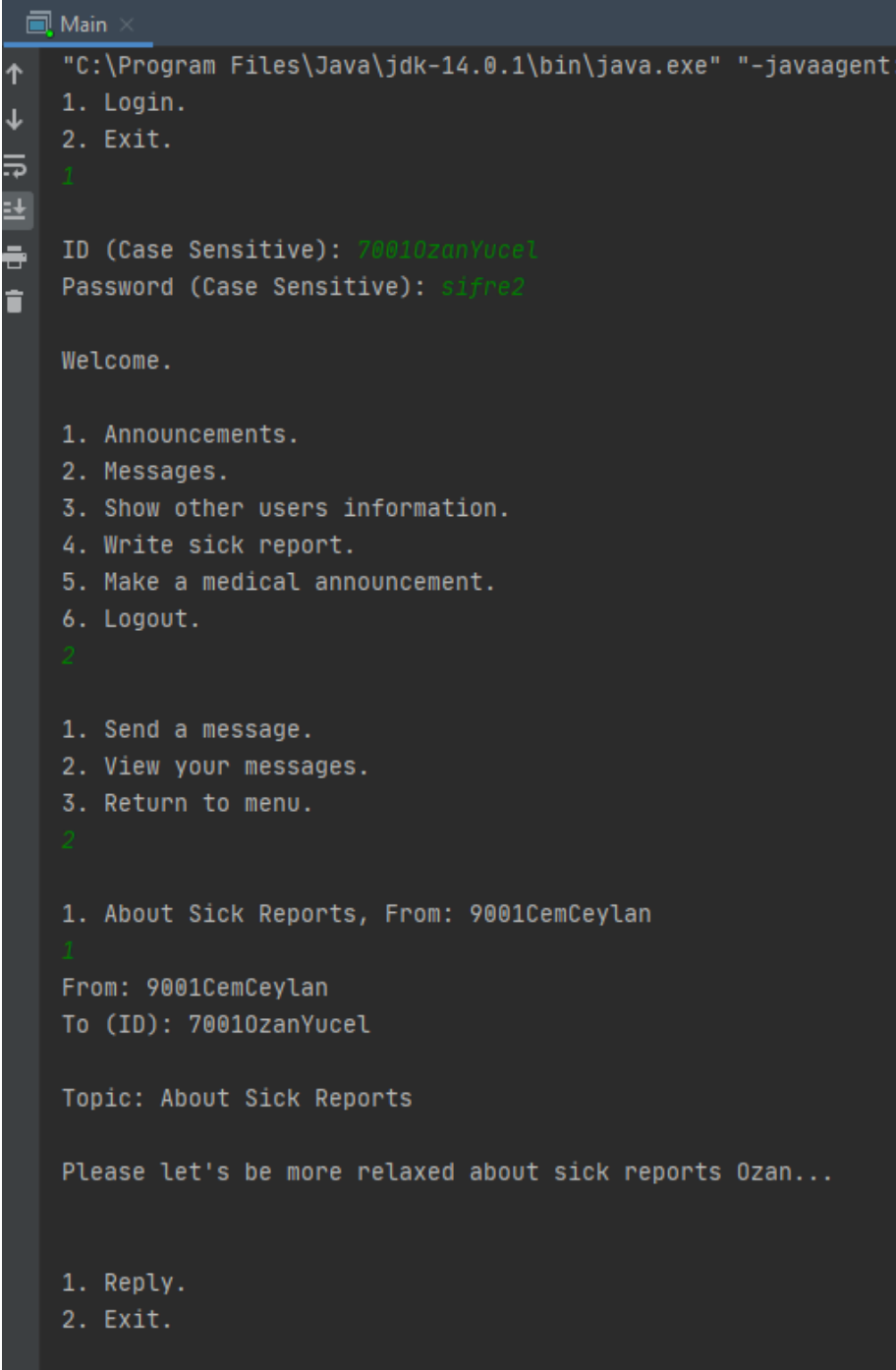
```
1. Send a message.
2. View your messages.
3. Return to menu.
2

You have no message.
```

We don't have a message of course so let's try to send a message.

```
To (ID): 70010zanYucel
Topic: About Sick Reports
Your Message: Please let's be more relaxed about sick reports Ozan...
Your message has been sent.
```

From Cem's account we have sent a message to Ozan's account. Every user can see other user's ID in show other user's information. That is required for sending message.



```
"C:\Program Files\Java\jdk-14.0.1\bin\java.exe" "-javaagent:
1. Login.
2. Exit.
1

ID (Case Sensitive): 70010zanYucel
Password (Case Sensitive): sifre2

Welcome.

1. Announcements.
2. Messages.
3. Show other users information.
4. Write sick report.
5. Make a medical announcement.
6. Logout.
2

1. Send a message.
2. View your messages.
3. Return to menu.
2

1. About Sick Reports, From: 9001CemCeylan
1
From: 9001CemCeylan
To (ID): 70010zanYucel

Topic: About Sick Reports

Please let's be more relaxed about sick reports Ozan...

1. Reply.
2. Exit.
```

In Ozan's message box, we can see messages as a list. When we select a message, we will be able to see everything about the message.

As you can see, we are able to reply a message. Let's see how it works.

```
1. About Sick Reports, From: 9001CemCeylan
```

```
1
```

```
From: 9001CemCeylan
```

```
To (ID): 70010zanYucel
```

```
Topic: About Sick Reports
```

```
Please let's be more relaxed about sick reports Ozan...
```

```
1. Reply.
```

```
2. Exit.
```

```
1
```

```
Topic: Reply to your message About Sick Reports
```

```
Your Message: Okay. I will try to be more relaxed about sick reports.
```

```
ID (Case Sensitive): 9001CemCeylan
```

```
Password (Case Sensitive): sifre1
```

```
Welcome.
```

```
1. Announcements.
```

```
2. Messages.
```

```
3. Show other users information.
```

```
4. Add users.
```

```
5. Make an announcement.
```

```
6. Sick reports.
```

```
7. Logout.
```

```
2
```

```
1. Send a message.
```

```
2. View your messages.
```

```
3. Return to menu.
```

```
2
```

```
1. Reply to your message About Sick Reports, From: 70010zanYucel
```

```
1
```

```
From: 70010zanYucel
```

```
To (ID): 9001CemCeylan
```

```
Topic: Reply to your message About Sick Reports
```

```
Okay. I will try to be more relaxed about sick reports.
```

```
1. Reply.
```

```
2. Exit.
```

Let's select show other user's information. We are showing an image from administrator's account because administrator can see every information about the others.

```
Please choose the user type which you are looking for:
1. Administrator.
2. Manager.
3. Worker.
4. Doctor.
5. Return to menu.
1

Please choose the user which you are looking for:
1. Cem Ceylan
2. Ahmet Celik
1
ID: 9001CemCeylan
Name: Cem Ceylan
Title: General Manager
Work Hours: 15.00-23.00
Age: 45
Height: 1.8
Weight: 80.5
Born Date: 04.03.1975
Salary: 7800.0
Office Location: A Block Floor 1
Office Hours: 17.00-20.00
```

If we want to add a new doctor the path, we will follow will be like this:

```
What type of user you wish to add?
1. Administrator. (ID has to be like 90xx-Name!)
2. Manager. (ID has to be like 80xx-Name!)
3. Doctor. (ID has to be like 70xx-Name!)
4. Worker. (ID has to be like 60xx-Name!)
3
ID:
8004
ID has to be like 70xx!
ID:
7006CerenTurkben
Password:
cerenpassword
```

Name:
Ceren Turkben
Work Hours:
14:00-17:00
Date of Born (DD/MM/YY):
01/05/1994
Title:
Doctor
Height:
56
Weight:
1.61
Salary:
7000
Age:
26
Office Location:
A Block Floor 03
Office Hours:
16:00-17:00

passwords - Notepad

File	Edit	Format	View	Help
sifre1				
9001CemCeylan				
sifre2				
70010zanYucel				
sifre3				
8001GizemCeylan				
sifre4				
6001ArdaYucel				
cerenpassword				
7006CerenTurkben				

information - Notepad

File	Edit	Format	View	Help
9001CemCeylan,Cem Ceylan,15.00-23.00,04.03.1975,General Manager,1.80,80.5,7800.0,45,A Block Floor 1,17.00-20.00				
70010zanYucel,Ozan Yucel,15.00-23.00,04.03.1975,General Manager,1.80,80.5,7800.0,45,A Block Floor 1,17.00-20.00				
8001GizemCeylan,Gizem Ceylan,15.00-23.00,04.03.1975,General Manager,1.80,80.5,7800.0,45,A Block Floor 1,17.00-20.00				
6001ArdaYucel,Arda Yucel,15.00-23.00,04.03.1975,General Manager,1.80,80.5,7800.0,45,Chemistry				
9002Ahmet,Ahmet Celik,9.00-19.00,06.06.1988,Officer,1.75,78.9,5000.0,31,B Block Floor 2,13.00-15.00				
7006CerenTurkben,Ceren Turkben,14:00-17:00,01/05/1994,Doctor,56.0,1.61,7000.0,26,A Block Floor 03,16:00-17:00				

When Ceren logs into the account, she can use all the features of the doctor menu.

Let's make a medical announcement from Ceren's account.

```

ID (Case Sensitive): 7006CerenTurkben
Password (Case Sensitive): cerenpassword

Welcome.

1. Announcements.
2. Messages.
3. Show other users information.
4. Write sick report.
5. Make a medical announcement.
6. Logout.
5

Enter -1 to return menu.
Please enter your announcement's title: About masks.
Please enter your announcement's body.
Please let's be more sensitive about masks. Mask are so important on these days.
Your announcement has been set.

```

Every user can see these announcements. If there will be more than 1 announcement, they will be listed under: "1. About masks." Program let's you choose which announcement you want to read.

```

Welcome.

1. Announcements.
2. Messages.
3. Show other users information.
4. Write sick report.
5. Make a medical announcement.
6. Logout.
1

1. Announcements.
2. Medical Announcements.
3. Return to menu.
2

1. About masks.
1
About masks.
Please let's be more sensitive about masks. Mask are so important on these days.

```


Doctors are able to write sick reports to every employee.

If there is no sick report user sees a message like this:

```
ID (Case Sensitive): 6001ArdaYucel
Password (Case Sensitive): sifre4

Welcome.

1. Announcements.
2. Messages.
3. Show other users information.
4. My sick reports.
5. Logout.
4

No sick report found.
```

Let's create a sick report with Ceren's account.

```
ID (Case Sensitive): 7006CerenTurkben
Password (Case Sensitive): cerenpassword

Welcome.

1. Announcements.
2. Messages.
3. Show other users information.
4. Write sick report.
5. Make a medical announcement.
6. Logout.
4

For: 6001ArdaYucel
Date range: 20/05/2020-21/05/2020
Sick report body:
As I have informed by him, he has a terrible headache. Also he has all symptoms of common cold.
```

When Arda wants to check the program, he is able to see the sick report for himself.

```
ID (Case Sensitive): 6001ArdaYucel
Password (Case Sensitive): sifre4

Welcome.

1. Announcements.
2. Messages.
3. Show other users information.
4. My sick reports.
5. Logout.
4

1. Dr. Ceren Turkben Date Range: 20/05/2020-21/05/2020
1

For: 6001ArdaYucel
Date: 20/05/2020-21/05/2020

As I have informed by him, he has a terrible headache. Also he has all symptoms of common cold.

Dr. Ceren Turkben
```

Also, admins or managers are able to see other employees' sick reports.

If Gizem will try to check someone's sick report with her manager account, she will see something like this on her console:

```
ID (Case Sensitive): 0001GizemCeylan
Password (Case Sensitive): sifre3

Welcome.

1. Announcements.
2. Messages.
3. Show other users information.
4. Make an announcement.
5. Sick reports
6. Logout.
5

1. My sick reports.
2. Other employee's sick reports.
2

1. Dr. Ceren Turkben Date Range: 20/05/2020-21/05/2020 For: 6001ArdaYucel
1

For: 6001ArdaYucel
Date: 20/05/2020-21/05/2020

As I have informed by him, he has a terrible headache. Also he has all symptoms of common cold.

Dr. Ceren Turkben
```

If user tries to enter with wrong password or any other people's password program won't let him login.

```
1. Login.
2. Exit.
1

ID (Case Sensitive): 6001ArdaYuce1
Password (Case Sensitive): thispasswordiswrong

ID or Password not found.

ID (Case Sensitive): 6001ArdaYuce1
Password (Case Sensitive): cerenpassword

ID or Password not found.

ID (Case Sensitive): wrongusername
Password (Case Sensitive): s1fre3

ID or Password not found.

ID (Case Sensitive): wrongusername
Password (Case Sensitive): wrongpassword

ID or Password not found.
```

We also handled with exceptions. If user enters a String instead of integer while program wants an integer from him, program won't give an error and won't be closed.

```
1. Login.
2. Exit.
i'm going to enter String instead of integer
Invalid input.
```

Also, a user can log out from the system. If user logs out, program won't be terminated. Log in screen will show up again.

```
ID (Case Sensitive): 9001CemCeylan
Password (Case Sensitive): sifre1

Welcome.

1. Announcements.
2. Messages.
3. Show other users information.
4. Add users.
5. Make an announcement.
6. Sick reports.
7. Logout.
7

ID (Case Sensitive): 8001GizemCeylan
Password (Case Sensitive): sifre3

Welcome.

1. Announcements.
2. Messages.
3. Show other users information.
4. Make an announcement.
5. Sick reports
6. Logout.
```

IO Class

IO class is the class which we are using for all our I/O operations and methods.

Public void addDoctor(String ID,String name,String workHour,String bornDate,String title,double height,double weight,double salary,int age,String officeLocation,String officeHours) is a method that using for add an doctor to system. It creates a new doctor object and adds it to allDoctors ArrayList. It also writes the given information as a line and separates by comma into the information.txt text file.

Public void addManager(String ID,String name,String workHour,String bornDate,String title,double height,double weight,double salary,int age,String officeLocation,String officeHours) is a method using for add an manager to system. It creates a new manager object and adds it to allManagers ArrayList. It also writes the given information as a line and separates by comma into the information.txt text file.

Public void addAdministrator(String ID,String name,String workHour,String bornDate,String title,double height,double weight,double salary,int age,String officeLocation,String officeHours) is a method that using for add an administrator to system. It creates a new administrator object and adds it to allAdministrators ArrayList. It also writes the given information as a line and separates by comma into the information.txt text file.

Public void addWorker(String ID,String name,String workHour,String bornDate,String title,double height,double weight,double salary,int age,String field) is a method that using for add an worker to system. It creates a new worker object and adds it to allWorkers ArrayList. It also writes the given informations as a line and seperates by comma into the information.txt text file.

Public ArrayList<ArrayList> getAllInformation() is a method that using for getting all user information from the information.txt, create objects according the coming information and store them in the 4 different arrayList(allDoctors, allAdministrator, allWorkers, allManagers). Then adds these 4 arrayList into one arrayList of ArrayLists(allArrayList) and returns it to use in main class.

Public void readID() is using for getting IDs from the password.txt file and store them in the ID arrayList.

Public boolean checkID(String ID) is search for given ID in the IDs ArrayList. If given ID exist in the IDs ArrayList the method returns true otherwise it is returns false.

Public boolean checkPassword(String password) using for check existence of given password in the passwords ArrayList. If password exist method returns true otherwise it returns false.

Public boolean matchPasswordID(string password, String ID) controls given password and ID existnce in the passwords arrayList and IDs arrayList.

Public passwordController(String password, String userID) does control process for given password and userID. It is searchs them in password.txt and if finds its returns false. We are using this method for avoiding the users who have same ID or same password.

Public void savePasswords(String password, String userID) is using during the adding new user process. It is writes given information into password.txt file.

Public void readPasswords() is using for give all passwords from the passwords.txt and store them in the passwords ArrayList.

Public void makeAnnouncement() is the method that using for create new announcement. It takes console inputs which are entered by the user (such as: title and body) and writes them into the announcements.txt file.

Public void readAnnouncement() reads announcements and titles from the announcements.txt and stores them in the annBodyArrayL and annTitleArrayL.

Public void makeAnnouncement() is the method that using for create new medical announcement. It takes console inputs which are entered by user (such as: title and body) and writes them into the medicalAnnouncements.txt file.

Public void readmedicalAnnouncement() reads medical announcements and titles from the medicalAnnouncements.txt and stores them in the medAnnBodyArrayL and medAnnTitleArrayL.

Public void readMessages() reads messages, message topics, ToIDs and FromIDs from the sentMessages.txt. It also stores them in messagesArrayList, messageTopicsArrayList, messageToIDArrayList and messageFromIDsArrayList.

Public boolean checkAnnouncement() controls annTittleArrayL's size. If annTitleArrayL's size not equals zero its returns true otherwise returns false.

Public boolean checkMedAnnouncement() controls medAnnTitleArrayL's size. If medAnnTitleArrayL's size not equals zero it returns true otherwise returns false.

Public boolean checkMessages(String ID) searches for given ID in the messageToIDArrayList. If it finds the ID returns true otherwise returns false.

Public boolean checkSickReport(String ID) searches for given ID in the sickReportForArray. If it finds the ID returns true otherwise returns false.

Public void readSickReports() reads sick reports from the sickReports.txt and stores the information in the sickReportForArray, sickReportDateArray sickReportBodyArray and sickReportDoctorArray.

Public void createSickReport(String ID) is the method that using for creating new sick report. It takes console inputs which are entered by the user (such as date range) and stores them in the sickReports.txt file.

Public void addUserMenu() is the a method that using in the adding new user process. It takes console inputs which are entered by the user (such as name,ID,password etc) and sends them to related method.

IMenu Interface

We have used interface and abstract class feature for overriding sub-menus.

Employee Class

Public void printInfoForWorker() is a method that using for show the employee's information according to Worker Class's permission. Workers cannot see some selected information such as salary, born date etc.

Public void printInfoForManager() is a method that using for show the employee's information according to Manager Class's permission. Managers cannot see some selected information such as height,weight etc.

Public void printInfoForDoctor() is a method that using for show the employee's information according to Doctor Class's permission. Doctors cannot see some selected information such as Salary.

Public void printInfoForAdministrator() is a method that using for show the employee's information according to Administrator Class's permission. Administrators can see all information in the Employee Class.

We used encapsulation for variables on this class. Also, this is a super class. A part of inheritance is here.

Doctor Class

Public void Menu () is a method that using for show doctor class's menu. Because of the Doctor Class gets implemented to IMenu, sub-menu for doctors gets overridden from IMenu.

Public printInfoForWorkers() is basically overrides the same method from Employee class. It calls super class's printInfoForWorkers() method and prints Office hours and office location information.

Public printInfoForManagers() is basically overrides the same method from Employee class. It calls super class's printInfoForManagers() method and prints Office hours and Office location information.

Public printInfoForDoctors() is basically overrides the same method from Employee class. It calls super class's printInfoForDoctors() method and prints Office hours and Office location information.

Public printInfoForAdministrators() is basically overrides the same method from Employee class. It calls super class's printInfoForAdministrators() method and prints Office hours and Office location information.

Doctor class is a sub-class of Employee. We used "inheritance" for carrying variables. Also, with overriding printInfo... methods, we used "run-time polymorphism" here.

Worker Class

Public void Menu () is a method that using for show doctor class's menu. Because of the Worker Class gets implemented to IMenu, sub-menu for workers gets overridden from IMenu.

Public printInfoForWorkers() is basically overrides the same method from Employee class. It calls super class's printInfoForWorkers() method and prints field information.

Public printInfoForManagers() is basically overrides the same method from Employee class. It calls super class's printInfoForManagers() method and prints field information.

Public printInfoForDoctors() is basically overrides the same method from Employee class. It calls super class's printInfoForDoctors() method and prints field information.

Public printInfoForAdministrators() is basically overrides the same method from Employee class. It calls super class's printInfoForAdministrators() method and prints field information.

Worker class is a sub-class of Employee. We used "inheritance" for carrying variables. Also, with overriding printInfo... methods, we used "run-time polymorphism" here.

Manager Class

Public void Menu () is a method that using for show manager class's menu. Because of the Manager Class gets implemented to IMenu, sub-menu for workers gets overridden from IMenu.

Public printInfoForWorkers() is basically overrides the same method from Employee class. It calls super class's printInfoForWorkers() method and prints Office hours and office location information.

Public printInfoForManagers() is basically overrides the same method from Employee class. It calls super class's printInfoForManagers() method and prints office hours and office location information.

Public printInfoForDoctors() is basically overrides the same method from Employee class. It calls super class's printInfoForDoctors() method and prints office hours and office location information.

Public printInfoForAdministrators() is basically overrides the same method from Employee class. It calls super class's printInfoForAdministrators() method and prints office hours and office location information.

Manager class is a sub-class of Employee. We used "inheritance" for carrying variables. Also, with overriding printInfo... methods, we used "run-time polymorphism" here.

Administrator Class

public void Menu() is a method that using for show Administrator class's menu. Because of the Administrator Class gets implemented to IMenu, sub-menu for workers gets overridden from IMenu.

Public printInfoForManagers() is basically overrides the same method from Employee class. It calls super class's printInfoForManagers() method and prints Office hours and Office location information.

Public printInfoForDoctors() is basically overrides the same method from Employee class. It calls super class's printInfoForDoctors() method and prints Office hours and Office location information.

Public printInfoForAdministrators() is basically overrides the same method from Employee class. It calls super class's printInfoForAdministrators() method and prints Office hours and Office location information.

Administrator class is a sub-class of Employee. We used "inheritance" for carrying variables. Also, with overriding printInfo... methods, we used "run-time polymorphism" here.

Office Class

The Office class has constructors, get and set methods. We are using Office with "has-a" relationship.

Main Class

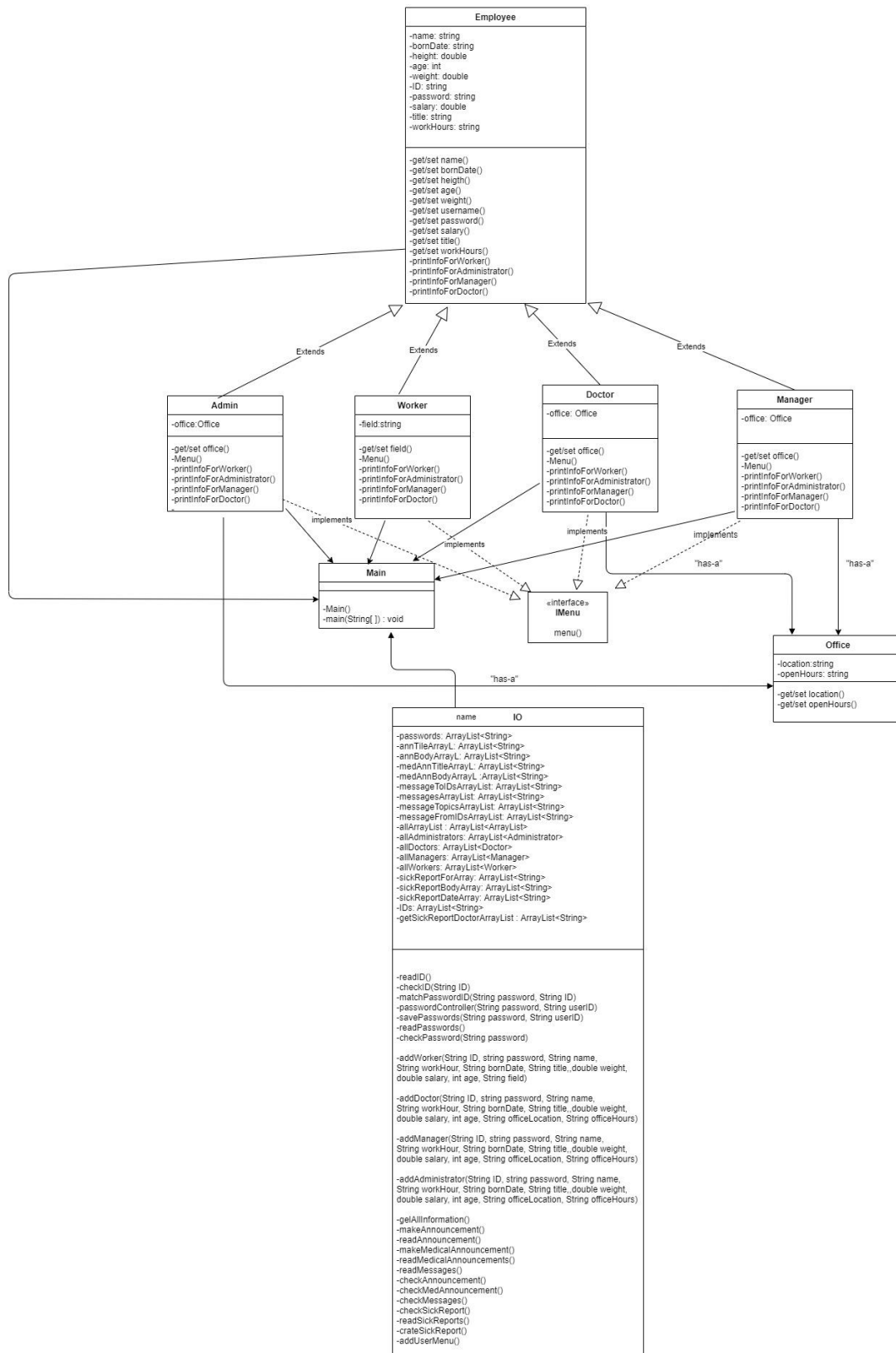
This class the place the all methods and menus become together. We have 2 methods. One of them is mainMenu() which includes our main menu. The other method is main(String[] args) which all methods and functionalities gets combined.

Notes

- We removed the diseases ArrayList from the Employee class.
- We removed the speciality variable from the Doctor class.
- We added a class called as IO. The IO is the class which we are using for all our I/O operations and methods.
- We removed menu methods from the classes and we added an interface called by IMenu and implement the following classes: Doctor, Administrator, Manager, Worker.
- We removed the Workers ArrayList from the Manager class.

UML Class Diagram

Company Information System Class Diagram



PSUEDOCODES

IMENU

Create an interface called IMenu{

 Create an abstract class called "Menu" for sub-menus;

}

EMPLOYEE

Create a super class called Employee {

 Create these following private variables;

 String name, workHour, bornDate, title, ID;

 double height, weight, salary;

 int age;

 Create get and set methods for these variables;

 Add parametrized and non-parametrized constructors for Employee class;

 Create a method called printInfoForWorker{

 print variables that you want to let any worker will be able see;

 }

 Create a method called printInfoForAdmin{

 print variables that you want to let any admin will be able see;

 }

 Create a method called printInfoForDoctor{

 print variables that you want to let any doctor will be able see;

 }

 Create a method called printInfoForManager{

 print variables that you want to let any manager will be able see;

 }

}

OFFICE

Create a class called Office{

 Add those following private variables;

 String location, openHours;

 Add get and set methods for those variables;

 Add parametrized and non-parametrized constructors for Office class;

}

ADMINISTRATOR

Create a sub-class called Administrator which extends to Employee and implements to IMenu{

 Administrator “has-a” Office so add this variable as private;

 Office office;

 Add get and set methods for office;

 Add parametrized and non-parametrized constructors for Administrator class;

 @Override Menu method from IMenu interface {

 Add sub-menu outputs for Administrator;

 }

 @Override printInfoMethods from Employee class {

 Print Office variables with get methods;

 }

DOCTOR

Create a class called Doctor which extends to Employee and implements to IMenu{

Doctor "has-a" Office so add this variable as private;

Office office;

Add get and set methods for office;

Add parametrized and non-parametrized constructors for Doctor class;

@Override Menu method from IMenu interface {

Add sub-menu outputs for Doctor;

}

@Override printInfoMethods from Employee class {

Print Office variables with get methods;

}

}

MANAGER

Create a class called Manager which extends to Employee and implements to IMenu{

Manager "has-a" Office so add this variable as private;

Office office;

Add get and set methods for office;

Add parametrized and non-parametrized constructors for Manager class;

@Override Menu method from IMenu interface {

Add sub-menu outputs for Manager;

}

@Override printInfoMethods from Employee class {

Print Office variables with get methods;

}

}

WORKER

Create a class called Manager which extends to Employee and implements to IMenu{

Add this following private variable;

String field;

Add get and set methods for field;

Add parametrized and non-parametrized constructors for Manager class;

@Override Menu method from IMenu interface {

Add sub-menu outputs for Manager;

}

@Override printInfoMethods from Employee class {

Print field variables with get methods;

}

}

IO

Create a class called IO {

Add these following new ArrayLists as String type;

Private passwords;

Public IDs, annTitleArrayL, annBodyArrayL, medAnnTitleArrayL, medAnnBodyArrayL,
messageToIDsArrayList, messagesArrayList, messageTopicsArrayList,
messageFromIDsArrayList, sickReportForArray, sickReportBodyArray, sickReportDateArray,
getSickReportsDoctorArrayList;

Add an ArrayList as Doctor type with an allDoctors reference;

Add an ArrayList as Manager type with an allManagers reference;

Add an ArrayList as Administrator type with an allAdministrators reference;

Add an ArrayList as Worker type with an allWorkers reference;

Add a new File and text files for I/O feature that we will be using in this class;

//DON'T FORGET TO HANDLE WITH IOExceptions !

Create a method with passwordController reference (take password and ID) {

Search those 2 variables in passwords ArrayList and IDs ArrayList;

```
        If(exist) {Return true;}

        Else {return false;}

    }
```

Create a method with savePasswords reference (take password and ID) {

```
    If (passwordController (password, ID) returns true) {

        Write to text file;

    }

    Else {

        Print error;

    }

}
```

Create a boolean method with matchPasswordID reference (take password and ID) {

```
    Check password and ID is written together in text the file;

    If (written together) {return true;}

    Else {return false;}

}
```

Create a method with checkPassword reference (take String password) {

```
    Search passwords ArrayList specifically for taken password;

    If (exist) {return true;}

    Else {return false;}

}
```

Create a boolean method with checkID reference (take String ID) {

```
    Search IDs ArrayList specifically for taken ID;

    If (exist) {return true;}

    Else {return false;}

}
```

Create a boolean method checkAnnouncement {

```
    Control annTitleArrayL;

    If (empty) {return false;}
```

```

        Else {return true;}
    }

    Create a boolean method checkMedAnnouncement {
        Control medAnnTitleArrayL;
        If (empty) {return false;}
        Else {return true;}
    }

    Create a boolean method checkSickReport (take String ID) {
        If (sickReportFor is empty) {return false;}
        Else {
            Search sickReportForArray for taken ID;
            If (exist) {return true;}
            Else {return false;}
        }
    }

    Create a boolean method checkMessages (take String ID) {
        If (messageToIDsArrayList is empty) {return false;}
        Else {
            Search messageToIDsArrayList for taken ID;
            If (exist) {return true;}
            Else {return false;}
        }
    }

    Create a method with readPasswords reference {
        Read passwords from text file and add to specified ArrayList;
    }

    Create a method with readAnnouncement reference {
        Read announcements from text file and add to specified ArrayList;
    }

```


Create a method with readID reference {

Read IDs from text file and add to specified ArrayList;

}

Create a method with readMedicalAnnouncements reference {

Read medical announcements from text file and add to specified ArrayList;

}

Create a method with readSickReports reference {

Read sick reports from text file and add to specified ArrayList;

}

Create a method with makeAnnouncement reference {

Print needed inputs which are title and body;

Read inputs;

Write inputs to text file;

Clear annTitleArrayL, annBodyArrayL;

Call readAnnouncement();

}

Create a method with makeMedicalAnnouncement reference {

Print needed inputs which are title and body;

Read inputs;

Write inputs to text file;

Clear medAnnTitleArrayL, medAnnBodyArrayL;

Call readMedicalAnnouncement();

}

```

Create a method with createSickReport reference (take String ID) {
    Print needed inputs which are for who, date range and report body;
    Read inputs;
    Write inputs to text file;
    Search for Doctor's ID in allDoctors ArrayList;
    If (exist) {
        Write doctor's ID to text file;
    }
    Else {Print error;}
    Clear sickReportDateArray, sickReportForArray, sickReportBodyArray;
    Call readSickReport();
}

```

```

Add addUserMenu and handle with IOException {
    Print a menu for user to select which type of user that will be added;
    Read the choice and connect to sub-addMenus with if and else;
    Read the specified inputs for user types and give them to addAdministrator
    method;
}
}

```

MAIN

Create a class called Main {

 Create a method with mainMenu reference {

 Print main menu;

 }

 Create a main method {

 Call all read... methods for filling information to ArrayLists from text files;

 Call mainMenu();

 Read mainMenu choice;

 If (choice is login) {

 Read ID and password from user;

 Use checkPassword, checkID, matchPassword methods for checking password and id;

 Print out the sub-menus with the help of substring method;

 //If password starts with 90, user is an admin;

 //If password starts with 80, user is a manager;

 //If password starts with 70, user is a doctor;

 //If password starts with 60, user is a worker;

 For sending message, first take the needed inputs which are to who (as ID), topic and message;

 Write them and the sender's ID to text file;

 For reading a message, first use checkMessages with ID to check are there any message;

 If (exist) {List the messages;}

 Else {Print "You have no message."}

 //Same method can be used for reading message and announcements.

 //Use other methods which are stored in IO for sub-class options;

 }

 Else {Close the program;}