MINISTRY OF EDUCATION AND SCIENCE OF THE KYRGYZ REPUBLIC

KYRGYZ-GERMAN INSTITUTE OF APPLIED INFORMATICS

**COURSEWORK**

Programming Languages 2

on the subject: « Employee relationship management system for KFC»

Supervisor:A.Kibekbaev(Senior Lecturer)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

English Language Adviser:M.K. Karagulova(Senior Lecturer)\_\_\_\_\_\_\_\_\_\_

English Language Adviser:A.S. Asanova(Senior Lecturer)\_\_\_\_\_\_\_\_\_\_\_\_\_

Monitoring Adviser:K.B.Turganov(Senior Lecturer)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Completed by WIN-1-21: Tashkulov N.B.\_\_\_\_\_\_\_\_\_\_\_

Maasaliev B.J.\_\_\_\_\_\_\_\_\_\_\_\_\_

Bishkek 2021

**Content**

[**Introduction** 3](#_Toc103127396)

[**Charter laws and rights:** 4](#_Toc103127397)

[**Team Roles:** 5](#_Toc103127398)

[**Software Requirements Specification** 6](#_Toc103127399)

[**Architectural Representation** 7](#_Toc103127400)

[**Main Part:** 9](#_Toc103127401)

[**Marketer account** 14](#_Toc103127402)

[**Director account** 19](#_Toc103127403)

[**Account of a worker** 29](#_Toc103127404)

[**Account of a manager** 32](#_Toc103127405)

[**Conclusion** 37](#_Toc103127406)

[**References:** 39](#_Toc103127407)

# **Introduction**

**Abstract:** There are a large number of fast-food firms at the moment. Therefore, there is a lot of competition in the market, and each brand wants to stand out from the rest and be the first. Today, KFC is the leader in terms of the number of restaurants in Kyrgyzstan. The brand intends to provide its franchisees with even more growth opportunities - through menu personalization and big data. We intend to help by creating an automated system to handle big data. This offers a huge advantage in terms of time as well as quality.

**Subject**: KFC is the most popular fast-food brand in Kyrgyzstan at the moment.

KFC - offers customers fast and inexpensive meals.

**Goals:** The main goal of our project is to create a user-friendly and efficient automated system and database for KFC

**Objectives:** Automation is an opportunity to speed up the processes of the service and to maintain them thoroughly. Our program gives you the opportunity to:

1. log in or create an account;
2. select from the menu based on the type of user account;
3. show your customer coverage by region;
4. show a list of all coverage areas;
5. show completed, in progress and active jobs;
6. modify and show the budget for salary and for marketing;
7. lower or raise the payroll;
8. show a list of equipment for the construction of facilities;
9. show a list of instructions to employees;
10. spend promotion budget, etc.

**Relevance:**  This topic is very relevant nowadays, as fast-food outlets are very popular, because most people don't have time to eat elsewhere it takes a long time. In addition, the prices in such outlets are low, making them a budget-friendly option for people. And for big brands, it is necessary to automate operations, as automation reduces human resources and significantly reduces the cost of their work.

**Target group:** The users of the program are a manager, an employee, a director and a marketer. Each user must have their own account to the system by login and password.

Depending on the account category, the corresponding menu opens to perform further actions.

# **Charter laws and rights:**

1. To receive, on request, limited medical leave for health reasons or for another valid reason.
2. The right to privacy and the non-disclosure of private information, provided that it is not prejudicial to the work.
3. Voluntarily physically attend our meetings in University.
4. Political, religious agitation within our team is prohibited.
5. It is prohibited to leave the team without raising the issue for general discussion.

# **Team Roles:**

Tashkulov Nurislam:

* Manager’s account developer;
* Employee’s account developer;
* Automation part developer;
* Documentation;
* Presentation.

Maasaliev Bektur:

* Main menu developer;
* Director’s account developer;
* Marketer’s account developer;
* Automation part developer;
* Documentation;
* Presentation.

# **Software Requirements Specification**

**Used tools**

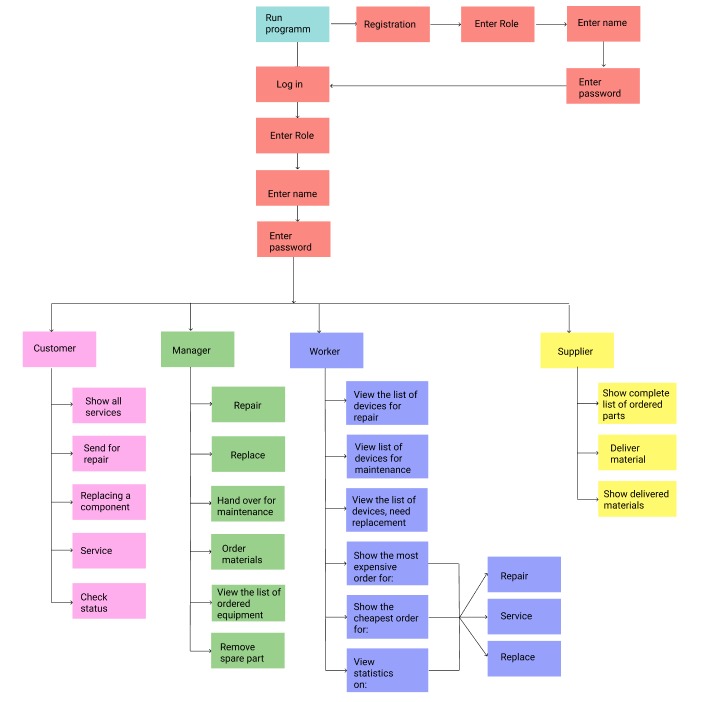
*Program language:* Java Version 8 Update 321

*IDE:* IntelliJ IDEA

*Database:* txt files

*GUI:* Terminal

# **Architectural Representation**



Pic.1.A Map-diagram

This diagram of our project shows the functions that the program performs. After launching the program, which is the very beginning of the system, it asks the user to log in or register. In case of registration, after creating your own account, the User returns to the entry point, where he needs to enter his data: account, login and password. Depending on the information entered into the system, the corresponding functions are opened:

1. Customer look at all services, give them for repair, give them for replacement of components, give them for maintenance, check the order readiness status by the serial number provided to him in the previous functions

2. Repairman: repairs, replace, hand over for maintenance, order the necessary materials, view the list of ordered equipment, remove the spare part from the list;

3. Worker: view the list of devices for repair, view the list of devices for maintenance, view the list of devices for replacement, show the expensive order, show the cheapest order, view statistics;

4. Supplier: show the complete list of ordered parts, deliver material, show delivered materials.

# **Main Part:**

package com.company;  
  
import java.io.IOException;  
import java.util.Scanner;  
  
public class Main {  
 static Scanner *input* = new Scanner(System.*in*);  
 public static void main(String[] args) throws IOException {  
 String[] result = Auth.*main*();  
 if("marketing".equals(result[0])){  
 Marketing.*main*(result[1],result[0]);  
 }  
 if("manager".equals(result[0])){  
 Manager.*main*(result[1],result[0]);  
 }  
 if("director".equals(result[0])){  
 Director.*main*(result[1],result[0]);  
 }  
 if("worker".equals(result[0])){  
 Worker.*main*(result[1],result[0]);  
 }  
 }  
 }

Pic.2. Main menu

Here’s given the main file of our program. All java files used during the process are imported here. Also, main file include Login and Registration. Everything starts from here.

Context file:

The Context file contains methods that I call in different files, mostly those that are repetitive and instead of writing the same thing all over again, I call and pass parameters to the method.

public static void end(){  
 System.*out*.println("Goodbye!");  
}  
public static void exit() throws IOException {  
 System.*out*.println("""  
 Log in to another account or register again for another account - 1  
 Exit - 2  
 """);  
 while (true){  
 int choice = *input*.nextInt();  
 String[] arr = new String[]{};  
 if (choice == 1){  
 *main*(arr);  
 }  
 else  
 {  
 *end*();  
 break;  
 }}  
}

Pic.3. Exit

This method is called when the user logs out, they are given a choice of whether they want to log out or just log in to another account. If he chooses to log out, the end() method is executed, but if he wants to log in to another account, the main() method is executed, i.e. everything is restarted.

public static String[] get\_lines(String file){  
 BufferedReader br = null;  
 ArrayList<String> array = new ArrayList<String>();  
 try {  
 Scanner sc = new Scanner(new File(file));  
 while (sc.hasNextLine()){  
 array.add(sc.nextLine());  
 }  
 }catch (IOException ex){  
 System.*out*.println("error");  
 }  
 return array.toArray(new String[0]);  
}

Pic.4. get\_lines method

This method reads all lines in the file and writes them into an array. The method takes as parameters the name of the file it will work with.

public static void delete\_line(String file,String line){  
 ArrayList<String> array = new ArrayList<String>();  
 try{  
 Scanner sc = new Scanner(new File(file));  
 while (sc.hasNextLine()){  
 array.add(sc.nextLine());  
 }  
 }  
 catch (IOException ex){  
 System.*out*.println(ex);  
 }  
 String[] arr = array.toArray(new String[0]);  
 try{  
 BufferedWriter writer = new BufferedWriter(new FileWriter(file));  
 for (String row : arr){  
 if (!row.strip().equals(line.strip())){  
 writer.write(row);  
 writer.newLine();  
 }}  
 writer.close();  
 }  
 catch (IOException ex){  
 System.*out*.println(ex);  
 }  
}

Pic.5. delete\_line method

This method deletes a line in a file, taking the file itself and the line to be deleted as parameters. Then it reads the file and overwrites all values except for the line that was passed to it.

public static void write(String file\_name,String param\_list) throws IOException{  
BufferedWriter br = new BufferedWriter(new FileWriter(file\_name,true));  
br.write( param\_list + "\n");  
br.close();  
}

Pic.6. write method

This method writes a value to a file, which it takes as a parameter file\_name, and the value param\_list itself as the parameter to write to the file.

Authorisation:

public static String[] main(){  
 Scanner input = new Scanner(System.*in*);  
 String[] account\_types = {"marketing","hr","manager","director","worker"};  
 int enter = 0;  
 System.*out*.println("1. Login | 2. Registration :");  
 enter = input.nextInt();  
  
 if (enter == 2){  
 *register*();  
 }  
  
 System.*out*.println("Enter to acc");  
 System.*out*.println("Choose type of acc");  
 while (true){  
 int count = 1;  
 for(String type : account\_types){  
 System.*out*.println(count + " : " + type);  
 count++;  
 }  
 int account\_type = input.nextInt();  
 String acc\_type = "";  
 if(account\_type ==1){  
 acc\_type = "marketing";  
 }  
 else if(account\_type == 2){  
 acc\_type = "manager";  
 }else if(account\_type == 3){  
 acc\_type = "director";  
 }else if(account\_type == 4){  
 acc\_type = "worker";  
 }else {  
 System.*out*.println("Sorry, but we did not find this type of account, please repeat");  
 continue;  
 }  
 System.*out*.println("Enter your login: ");  
 String password2 = input.nextLine();  
 String login = input.nextLine();  
 System.*out*.println("Enter your password: ");  
 String password = input.nextLine();  
  
  
 Boolean result = *auth*(login,password,acc\_type);  
  
  
 if(result){  
 String[] res = {acc\_type,login};  
 return res;  
 }else{  
 System.*out*.println("Invalid username or password, please try again");  
 }  
 }  
  
}

Pic.7. Login initiation

First of all, the system asks the User whether it wants to log in or register.



Pic.8. Run Program

You need to enter your user type. If you enter incorrect information, the system will ask you to re-enter your user type, it will continue to ask for re-entry until the user enters the correct information. After that, you will need to enter your username and password.

If the username and password match, then you will successfully log in, otherwise the system will display: "Invalid username or password, please try again" and the system will start working from the beginning, you will return to the account type selection point.

public static void register(){  
 Scanner input = new Scanner(System.*in*);  
 String[] account\_types = {"marketing","manager","director","worker"};  
 int count =1;  
 for( String type : account\_types){  
 System.*out*.println(count+" : "+type);  
 count++;  
 }  
 System.*out*.print("Enter Account Type - ");  
 int account\_type = input.nextInt();  
 String acc\_type = "";  
 if(account\_type ==1){  
 acc\_type = "marketing";  
 }else if(account\_type == 2){  
 acc\_type = "manager";  
 }else if(account\_type == 3){  
 acc\_type = "director";  
 }else if(account\_type == 4){  
 acc\_type = "worker";  
 }else {  
 System.*out*.println("Sorry, but we did not find this type of account, please repeat");  
 return;  
 }  
 System.*out*.println("Enter your login: ");  
 String password2 = input.nextLine();  
 String login = input.nextLine();  
 System.*out*.println("Enter your password: ");  
 String password = input.nextLine();  
 File myFile =new File("users.txt");  
 File salaryFile = new File("salary");  
 try{  
 BufferedWriter writer = new BufferedWriter(new FileWriter(myFile,true));  
 BufferedWriter writer2 = new BufferedWriter(new FileWriter(salaryFile,true));  
 writer.write(login + " ");  
 writer.write(password+ " ");  
 writer.write(acc\_type + " " +"\n");  
 if(acc\_type != "director"){  
 writer2.write(login+ " ");  
 writer2.write(acc\_type + " ");  
 writer2.write(0 + "\n");  
 writer2.close();  
 }  
 writer.close();  
 System.*out*.println("Register success");  
 }  
 catch (IOException ex){  
 ex.printStackTrace();  
 }  
}

Pic.9. Register

When selecting:

Registration:

When registering, the User must enter a category from the accounts listed to him, or the system will output: "Sorry, but we did not find this type of account, please repeat" and ask to re-enter the account type. Then the login and password are requested by which he will log in in the future. The account type, User login and its hashed password are stored in the file "users.txt ".

public static boolean auth(String login,String password,String acc\_type){  
 boolean res = false;  
 BufferedReader br = null;  
 try {  
 br = new BufferedReader(new FileReader("users.txt"));  
 String line;  
 while((line = br.readLine())!=null){  
 String[] arr = line.split(" ");  
 if(login.equals(arr[0]) && password.equals(arr[1]) && acc\_type.equals(arr[2])){  
 System.*out*.println(arr[1]);  
 System.*out*.println("Auth success");  
 res = true;  
 br.close();  
 break;  
 }else{  
 res = false;  
 }  
 }  
 }catch (IOException ex){  
 System.*out*.println("error");  
 }  
return res;  
 }  
}

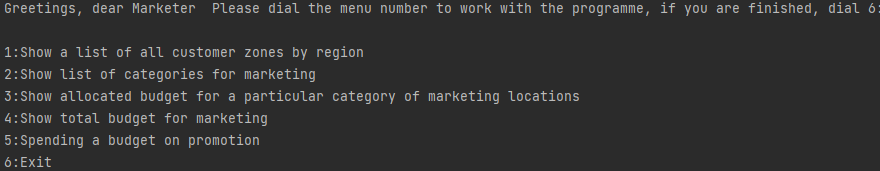
Pic.10. Authorisation

When selecting:

Authorisation:

When a user chooses to log in, they will be asked to enter their username and password, and if they are incorrect, they will be asked to re-enter them.

# **Marketer account**



Pic.11. Main menu of marketer’s account.

static Scanner *input* = new Scanner(System.*in*);  
public static void main(String name, String acc\_type) throws IOException {  
 System.*out*.println("Greetings, dear Marketer " +name+  
 " Please dial the menu number to work with the programme, if you are finished, dial 6:\n");  
 String[] to\_do = {"Show a list of all customer zones by region ",  
 "Show list of categories for marketing ",  
 "Show allocated budget for a particular category of marketing locations",  
 "Show total budget for marketing ",  
 "Spending a budget on promotion",  
 "Exit "};  
 while (true) {  
 int count = 1;  
 for (String work : to\_do) {  
 System.*out*.println(count + ":" + work);  
 count++;  
 }  
 int command = *input*.nextInt();  
 if(command == 1){  
 *show\_coverage\_areas*();  
 }  
 if(command == 2){  
 *show\_media*();  
 }  
 if(command == 3){  
 *allocated\_budget*();  
 }  
 if (command == 4){  
 *entire\_budget*();  
 }  
 if (command == 5){  
 *promotion*();  
 }  
 if (command == 6){  
 Main.*exit*();  
 break;  
 }  
 if(command>6 || command < 1 ){  
 System.*out*.println("Error, there is no such command here, please try again :-(");  
 continue;  
 }else if (command == 5){  
 Main.*end*();  
 break;  
 }  
 }  
}

Pic.12. Main menu of marketer’s account.

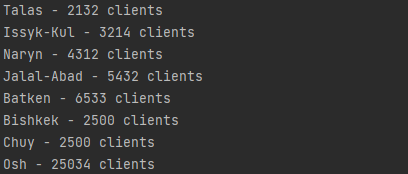
A marketer will be thrown to the main menu after authorization. The main menu of that account consist of 6 commands (functions). The user should enter a number from 1 to 6 to start the process. Unless the client will follow the conditions, he/she will be stuck in a loop.

As we have noticed earlier, the main menu includes 6 commands: 1) Show a list of all customer zones by region, 2) Show list of categories for marketing, 3) Show allocated budget for a particular category of marketing locations, 4) Show total budget for marketing,

5) Spending a budget on promotion, 6)Exit. Next we are going to parse all used functions and files below.

First we are going to show how it works, and then we’ll proceed to parsing of the code.(Pic.2.)

1. Show a list of all customer zones by region;



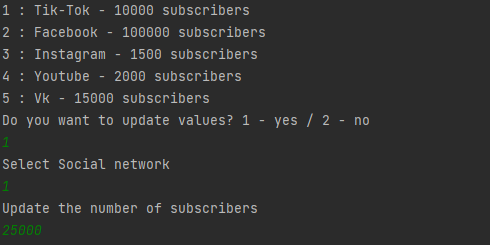
Pic.13. First command in the main menu

First command just shows all services the user can get in here.

private static void show\_coverage\_areas() throws IOException{  
 String[] areas = Context.*get\_lines*("areas");  
 for (String line : areas){  
 System.*out*.println(line);  
 }  
}

Pic.14. The function which shows all services.

Method simply reads and displays the data from the file.

1. Show list of categories for marketing;

Pic.15. Second command in the main menu

The second command shows the social media promotion, the social network itself and its subscribers. It is also possible to change the number of subscribers. When entering number 1, you must select the social network, then enter the number of subscribers for that social network.

private static void show\_media()throws IOException{  
 String[] areas = Context.*get\_lines*("media");  
 int count = 1;  
 for (String line : areas){  
 String[] arr = line.split(" ");  
 System.*out*.println(count +" : "+arr[0]+" " + arr[1]+" " + arr[2]+" " + arr[3]);  
 count++;  
 }  
 while (true){  
 System.*out*.println("Do you want to update values? 1 - yes / 2 - no");  
 int choice = *input*.nextInt();  
 if (choice == 1){  
 System.*out*.println("Select Social network");  
 int line = *input*.nextInt()-1;  
 System.*out*.println("Update the number of subscribers");  
 int client\_count = *input*.nextInt();  
 Context.*update\_clients\_value*("media",areas[line],client\_count,2);  
 }else {  
 break;  
 }}  
}

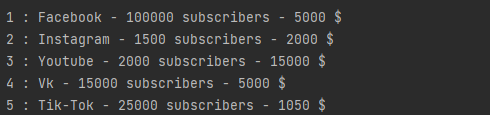
Pic.16. The function that is responsible for the second command in the main menu.

public static void update\_clients\_value(String filename, String line, int clients,int index){  
 String[] array\_line = line.split(" ");  
 array\_line[index] = String.*valueOf*(clients);  
 String new\_str = String  
 .*join*(" ",array\_line)  
 .replace("[","")  
 .replace("]","");  
 ArrayList<String> array = new ArrayList<String>();  
 try {  
 Scanner sc = new Scanner(new File(filename));  
 while (sc.hasNextLine()) {  
 array.add(sc.nextLine());  
 }  
 } catch (IOException ex) {  
 System.*out*.println(ex);  
 }  
 String[] arr = array.toArray(new String[0]);  
 try {  
 BufferedWriter writer = new BufferedWriter(new FileWriter(filename));  
 for (String row : arr) {  
 if (!row.strip().equals(line.strip())) {  
 writer.write(row);  
 writer.newLine();  
 }  
 }  
 writer.write(new\_str);  
 writer.close();  
 } catch (IOException ex) {  
 System.*out*.println(ex);  
 }  
}

Pic.17. update\_clients\_value method in Context file

To implement this method, use the method from the Context file, namely update\_clients\_value. As parameters I send the file to work, the social network itself in which to change the number of subscribers, the new number of clients, and the index to know which index in the file can be changed. In the update\_clients\_value method itself, I take the old value and overwrite it with the new one. Then I write everything back to the file with the new value.

1. Show allocated budget for a particular category of marketing locations;



Pic.18. The third command in the main menu

With the third command in the main menu, the marketer can find out: which social networks are currently being promoted, how many followers there are and how much money has been spent to promote them.

private static void allocated\_budget(){  
 String[] arr = Context.*get\_lines*("media");  
 int count = 1;  
 for (String line : arr){  
 System.*out*.println(count+" : "+ line);  
 count++;  
 }  
}

Pic.19. The function, responsible for the third command (allocated\_budget())

It's pretty easy here, the system calls the get\_lines method from the Context file we specified earlier (pic.3). It writes all the values into an array and then loops the system to output everything to the terminal. In the parameters the system sends the file to work.

1. Show total budget for marketing;



Pic.20. The command for services

The fourth command in the main menu shows the entire budget that has been allocated to the marketing department.

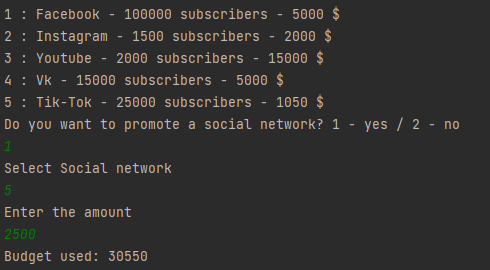
Here’s the function behind:

private static void entire\_budget(){  
 String[] arr = Context.*get\_lines*("marketing\_budget");  
 for (String line : arr){  
 System.*out*.println( line);  
 }  
}

Pic.21. The function responsible for the fourth command

This method is very similar to the previous one, it also calls the get\_line method from the Context file. Passes the file you want to work with as parameters. It then writes everything to an array and outputs it to the terminal.

1. Spending a budget on promotion;



Pic.21. Spending a budget on promotion

The fifth command shows the social media followers and how much money has been invested in promoting these social media. If you enter 1, you can change the budget for promoting one social network or another. First you need to choose a social network. Then you have to enter the amount of money to promote that social network. At the end you will also see how much money is used to promote all social networks.

private static void promotion() throws IOException{  
 String[] arr = Context.*get\_lines*("media");  
 int count = 1;  
 for (String line : arr){  
 System.*out*.println(count+" : "+ line);  
 count++;  
 }  
 while (true){  
 System.*out*.println("Do you want to promote a social network? 1 - yes / 2 - no");  
 int choice = *input*.nextInt();  
 if (choice == 1){  
 System.*out*.println("Select Social network");  
 int line = *input*.nextInt()-1;  
 System.*out*.println("Enter the amount ");  
 int sum = *input*.nextInt();  
 Context.*change\_budget*("marketing\_budget","media",arr[line],sum,"marketing","+",arr);  
 }else {  
 break;  
 }}  
 }  
}

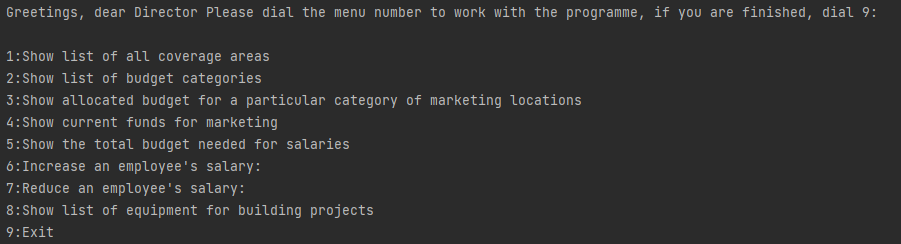
Pic.22. Method that are responsible for status checking

public static void change\_budget(String file,String filename, String line ,int sum, String type,String t, String[] array) throws IOException{  
 String[] budg = *get\_lines*(file);  
 String[] arr = budg[0].split(" ");  
 int sum\_of\_budg = 0;  
 if (type.equals("marketing")){  
 for(String li : array){  
 String[] hh = li.split(" ");  
 int f = Integer.*parseInt*(hh[5]);  
 sum\_of\_budg +=f;  
 }}  
 else{  
 for(String li : array){  
 String[] hh = li.split(" ");  
 int f = Integer.*parseInt*(hh[2]);  
 sum\_of\_budg +=f;  
 }  
 }  
 int b = Integer.*parseInt*(arr[9]);  
 int a = Integer.*parseInt*(arr[6]);  
 int new\_b = sum\_of\_budg;  
 if (t.equals("+")){  
 new\_b+=sum;}  
 else {  
 new\_b-=sum;  
 }  
 if (new\_b > a ){  
 System.*out*.println("The budget is tight");  
 }else  
 {  
 arr[9] = String.*valueOf*(new\_b);  
 BufferedWriter bw = new BufferedWriter(new FileWriter(file));  
 bw.write(arr[0]+" ");bw.write(arr[1]+" ");bw.write(arr[2]+" ");bw.write(arr[3]+" ");  
 bw.write(arr[4]+" ");bw.write(arr[5]+" ");bw.write(arr[6]+" ");bw.write(arr[7]+" ");bw.write(arr[8]+" ");  
 bw.write(arr[9]+" ");bw.write(arr[10]+" ");  
 bw.close();  
 if (type.equals("marketing")){  
 *update\_clients\_value*(filename,line,sum,5);}  
 else if (type.equals("salary")){  
 *change\_salary*(filename,line,sum,t);  
 }  
 }  
 System.*out*.println("Budget used: "+new\_b);  
}

Pic.23. change\_budget method in Context file

First, the system writes everything to an array via get\_line, then outputs it to the terminal. If the user has entered 1, the system will ask them to enter a social network and a new amount to promote the social network. After receiving this data, the change\_budget method is called in the Context file. This method first counts all the budget that is used and that the director has assigned. Then it compares it with the new budget and if it is higher than the one set by the director, the system will show that this budget is higher than the one assigned. If everything fits, the system changes the old budget to the new one in the file.

# **Director account**



Pic.24. Main menu of the director’s account

static Scanner *input* = new Scanner(System.*in*);  
public static void main(String name, String acc\_type) throws IOException{  
 System.*out*.println("Greetings, dear Director " + name +  
 " Please dial the menu number to work with the programme, if you are finished, dial 9:\n");  
 String[] to\_do = {"Show list of all coverage areas ",  
 "Show list of budget categories ",  
 "Show allocated budget for a particular category of marketing locations",  
 "Show current funds for marketing",  
 "Show the total budget needed for salaries",  
 "Increase an employee's salary:",  
 "Reduce an employee's salary:",  
 "Show list of equipment for building projects ",  
 "Exit"};  
 while (true) {  
 int count = 1;  
 for (String work : to\_do) {  
 System.*out*.println(count + ":" + work);  
 count++;  
 }  
 int command = *input*.nextInt();  
 if (command == 2){  
 *show\_budget\_category*();  
 }  
 if(command == 4){  
 *show\_market\_budget*();  
 }  
 if(command == 3){  
 *show\_market\_category*();  
 }  
 if(command == 1){  
 *show\_areas*();  
 }  
 if (command == 5){  
 *show\_salary\_category*();  
 }  
 if(command == 6){  
 *raise\_salary*("+");  
 }  
 if(command == 7){  
 *lower\_salary*();  
 }  
 if (command == 8){  
 *list\_of\_equipment*();  
 }  
 if (command == 9){  
 Main.*exit*();  
 }  
 }  
  
}

Pic.25. Main menu of the director’s account

A director will be thrown to the main menu after authorization. It consists of 9 commands (functions). The director should enter a number from 1 to 7 to start the process. Unless he will follow the conditions, he’ll get an error will be stuck in a loop.

As we have noticed earlier, the main menu includes 6 commands: 1) Show list of all coverage areas, 2) Show list of budget categories, 3) Show allocated budget for a particular category of marketing locations, 4) Show current funds for marketing, 5) Show the total budget needed for salaries, 6) Increase an employee's salary, 7) Reduce an employee's salary, 8) Show list of equipment for building projects, 9) Exit.

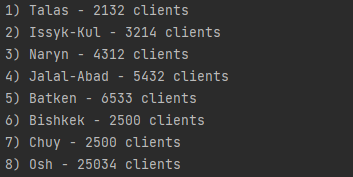
Here’s given the description of each command:

1. Show list of all coverage areas – shows areas with number of customers;
2. Show list of budget categories – shows budget for marketing and for salaries;
3. Show allocated budget for a particular category of marketing locations - shows social networks that are being promoted, how many followers they have and how much money is spent on their promotion;
4. Show current funds for marketing – shows the full marketing budget and how much is being used;
5. Show the total budget needed for salaries - shows all workers, how much they are getting, the total budget for salaries and how much is being used;
6. Increase an employee's salary – increases the employee's salary;
7. Reduce an employee's salary – decreases the employee's salary;
8. Show list of equipment for building projects – displays equipment from database;
9. Exit.

The director's account works closely with the other account types.

Next we are going to parse all used functions and files below. First we are going to show how it works, and then we’ll proceed to parsing of the code.

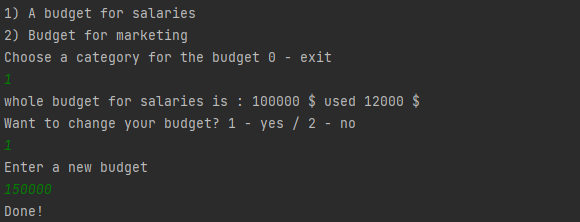
1. Show list of all coverage areas



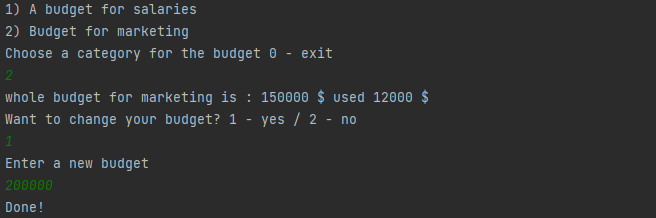
Pic.26. First command in the main menu

The first command shows the coverage by region, i.e. the number of clients in each region.

1. Show list of budget categories



Pic.27. Showing and changing the budget for salaries



Pic.28. Showing and changing the budget for marketing

The second command allows you to find out the budget for salaries as well as for marketing. In addition, the director has the possibility to change the budget for both categories.

private static void show\_budget\_category() throws IOException{  
 String[] categories = new String[]{"A budget for salaries","Budget for marketing"};  
 String[] filenames = new String[]{"salary\_budget","marketing\_budget"};  
 while (true) {  
 int count = 1;  
 for (String line : categories){  
 System.*out*.println(count + ") "+ line);  
 count++;  
 }  
 System.*out*.println("Choose a category for the budget 0 - exit");  
 int choice = *input*.nextInt();  
 if (choice == 1 || choice == 2) {  
 while (true) {  
 String[] arr = Context.*get\_lines*(filenames[choice -1]);  
 for (String line : arr) {  
 System.*out*.println(line);  
 }  
 System.*out*.println("Want to change your budget? 1 - yes / 2 - no");  
 String type = "add";  
 int ch = *input*.nextInt();  
 if (ch == 1) {  
 System.*out*.println("Enter a new budget");  
 String salary2 = *input*.nextLine();  
 int salary = *input*.nextInt();  
 System.*out*.println("Done!");  
 Context.*update\_budget*(arr[0], salary, filenames[choice-1]);  
 } else {  
 break;  
 }  
 }  
 }  
 else if (choice > 2 || choice < 0){  
 continue;  
 }  
 else break;  
 }  
}

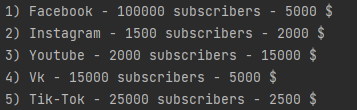
Pic.29. show\_budget\_category method

public static void update\_budget(String line, int budget\_sum, String file) throws IOException{  
 String[] arr = line.split(" ");  
 arr[6] = String.*valueOf*(budget\_sum);  
 BufferedWriter bw = new BufferedWriter(new FileWriter(file));  
 bw.write(arr[0]+" ");bw.write(arr[1]+" ");bw.write(arr[2]+" ");bw.write(arr[3]+" ");  
 bw.write(arr[4]+" ");bw.write(arr[5]+" ");bw.write(arr[6]+" ");bw.write(arr[7]+" ");bw.write(arr[8]+" ");  
 bw.write(arr[9]+" ");bw.write(arr[10]+" ");  
 bw.close();  
}

Pic.30. A method which updates the budget

First, the terminal will display which categories for the budget exist, selecting one of them will display the total budget and the budget used for that category. Then when entering 1, the director can change the budget. To do this, the system will request the amount of the new budget, after which the update\_budget method is called in the Context file, which takes the budget from the database, changes it to the new budget and re-writes it to the file.

1. Show allocated budget for a particular category of marketing locations



Pic.31. The third command in the main menu

The third command allows you to see all the social networks being promoted, how many subscribers they have and the budget being spent on their promotion.

1. Show current funds for marketing



Pic.32. Work of the fifth command

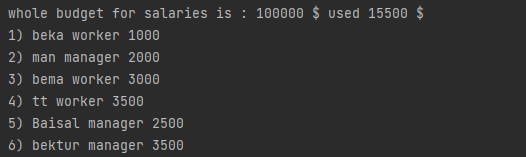
The fourth command allows you to see the entire budget allocated to the marketing department and how much is currently being used.

private static void show\_market\_budget(){  
 String[] areas = Context.*get\_lines*("marketing\_budget");  
 System.*out*.println(areas[0]);  
}

Pic.33. The function, responsible for the fourth command

Here, the get\_line method from the Context file is called, everything is written to an array, and then the system outputs this array to the terminal.

1. Show the total budget needed for salaries



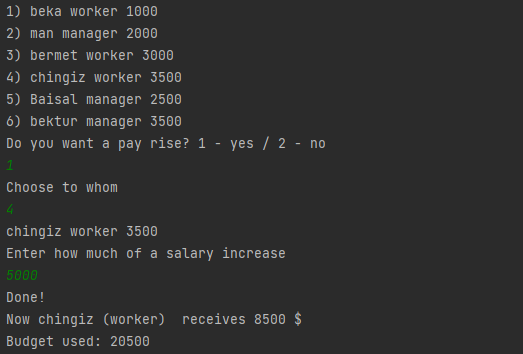
Pic.34. Work of the sixth command

private static void show\_salary\_category(){  
 String[] areas = Context.*get\_lines*("salary\_budget");  
 for (String line : areas){  
 System.*out*.println( line);  
 }  
 String[] areas2 = Context.*get\_lines*("salary");  
 int count2 = 1;  
 for (String line : areas2){  
 System.*out*.println(count2 + ") "+ line);  
 count2++;  
 }  
}

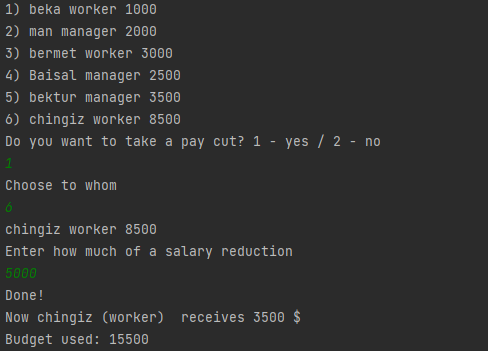
Pic.35. The function to delete an item from the list

The fifth command allows you to find out the entire budget for salaries and how much is used, it also displays all employees and how much they are paid.

6-7) Increase\ Reduce an employee's salary



Pic.36. Pay rises



Pic.37. A pay cut

The sixth and seventh teams are responsible for salaries, the sixth team raises the employee's salary and the seventh team lowers the employee's salary.

To do this, the system queries the employee and how much the salary will be increased or decreased. After the operation has been done, the terminal will show you how much the employee is getting now and the remaining budget.

private static void raise\_salary(String type\_of ) throws IOException {  
 String[] arr = Context.*get\_lines*("salary");  
 int count = 1;  
 for (String worker : arr){  
 System.*out*.println(count+") "+worker);  
 count++;  
 }  
 while (true){  
 if (type\_of.equals("+"))  
 System.*out*.println("Do you want a pay rise? 1 - yes / 2 - no");  
 else System.*out*.println("Do you want to take a pay cut? 1 - yes / 2 - no");  
 int choice = *input*.nextInt();  
 if (choice == 1){  
 System.*out*.println("Choose to whom");  
 int line = *input*.nextInt()-1;  
 System.*out*.println(arr[line]);  
 if (type\_of.equals("+"))  
 System.*out*.println("Enter how much of a salary increase");  
 else System.*out*.println("Enter how much of a salary reduction");  
 String salary2 = *input*.nextLine();  
 int salary = *input*.nextInt();  
 System.*out*.println("Done!");  
 Context.*change\_budget*("salary\_budget","salary",arr[line],salary,"salary",type\_of,arr);  
 }else  
 break;  
 }  
}

Pic.38. raise\_salary method

private static void lower\_salary() throws IOException {  
 *raise\_salary*("-");

}

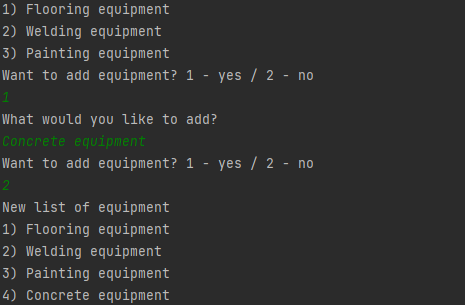
Pic.39. lower\_salary method

public static void change\_budget(String file,String filename, String line ,int sum, String type,String t, String[] array) throws IOException{  
 String[] budg = *get\_lines*(file);  
 String[] arr = budg[0].split(" ");  
 int sum\_of\_budg = 0;  
 if (type.equals("marketing")){  
 for(String li : array){  
 String[] hh = li.split(" ");  
 int f = Integer.*parseInt*(hh[5]);  
 sum\_of\_budg +=f;  
 }}  
 else{  
 for(String li : array){  
 String[] hh = li.split(" ");  
 int f = Integer.*parseInt*(hh[2]);  
 sum\_of\_budg +=f;  
 }  
 }  
 int b = Integer.*parseInt*(arr[9]);  
 int a = Integer.*parseInt*(arr[6]);  
 int new\_b = sum\_of\_budg;  
 if (t.equals("+")){  
 new\_b+=sum;}  
 else {  
 new\_b-=sum;  
 }  
 if (new\_b > a ){  
 System.*out*.println("The budget is tight");  
 }else  
 {  
 arr[9] = String.*valueOf*(new\_b);  
 BufferedWriter bw = new BufferedWriter(new FileWriter(file));  
 bw.write(arr[0]+" ");bw.write(arr[1]+" ");bw.write(arr[2]+" ");bw.write(arr[3]+" ");  
 bw.write(arr[4]+" ");bw.write(arr[5]+" ");bw.write(arr[6]+" ");bw.write(arr[7]+" ");bw.write(arr[8]+" ");  
 bw.write(arr[9]+" ");bw.write(arr[10]+" ");  
 bw.close();  
 if (type.equals("marketing")){  
 *update\_clients\_value*(filename,line,sum,5);}  
 else if (type.equals("salary")){  
 *change\_salary*(filename,line,sum,t);  
 }  
 }  
 System.*out*.println("Budget used: "+new\_b);  
}

Pic.40. change\_budget method in Context file

There is only one difference in ups and downs, either plus or minus. There is one method for these two functions, only one plus and minus change, when selecting downgrade this method takes - as parameters, and when selecting upgrade this method takes + as parameters. Also the change\_budget method is called from the Context file, this method immediately changes the budget for salaries depending on the parameter + or -.

1. Show list of equipment for building projects



Pic.41. Work of the eighth command

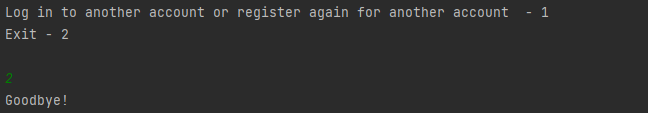
The eighth command allows you to see the available equipment for building facilities. You can also add equipment to the list immediately. Once added, you will be shown a new list of equipment.

private static void list\_of\_equipment() throws IOException{  
 String[] arr = Context.*get\_lines*("equipment");  
 int count = 1;  
 for (String worker : arr){  
 System.*out*.println(count+") "+worker);  
 count++;  
 }  
 while (true){  
 System.*out*.println("Want to add equipment? 1 - yes / 2 - no");  
 int choice = *input*.nextInt();  
 if (choice == 1){  
 System.*out*.println("What would you like to add?");  
 String equip1 = *input*.nextLine();  
 String equip = *input*.nextLine();  
 Context.*write*("equipment",equip);  
 }  
 else {  
 break;  
 }  
 }  
 String[] arr2 = Context.*get\_lines*("equipment");  
 System.*out*.println("New list of equipment");  
 int count2 = 1;  
 for (String worker : arr2){  
 System.*out*.println(count2+") "+worker);  
 count2++;  
 }  
}

Pic.42. Show equipment list

First, all file data is written to an array using the get\_line method in the Context file. Then when you enter 1, the director has the option to add the equipment to the list. To do this, the system asks for just the equipment and nothing extra. You will then be shown a new list of equipment.

1. Exit

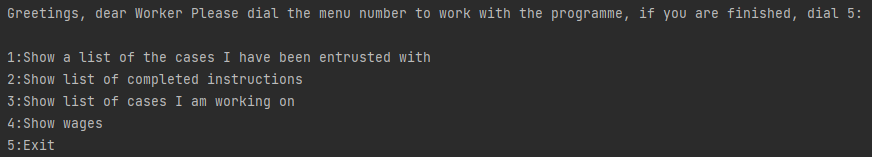


Pic.43. Work of the ninth command

When logging out you will be able to switch straight to another account without turning off the project by pressing 1, but if you want to log out you have to enter 2. The code is shown at picture 3.

# **Account of a worker**

We have logged into the worker’s account.



Pic.44. Worker’s menu

static Scanner *input* = new Scanner(System.*in*);  
public static void main(String name, String acc\_type) throws IOException {  
 System.*out*.println("Greetings, dear Worker " +  
 "Please dial the menu number to work with the programme, if you are finished, dial 5:\n");  
 String[] to\_do = {"Show a list of the cases I have been entrusted with",  
 "Show list of completed instructions",  
 "Show list of cases I am working on",  
 "Show wages",  
 "Exit"};  
 while (true) {  
 int count = 1;  
 for (String work : to\_do) {  
 System.*out*.println(count + ":" + work);  
 count++;  
 }  
 int command = *input*.nextInt();  
 if(command == 1){  
 *show\_work*("need\_to\_do");  
 }  
 if(command == 2){  
 *show\_finished\_work*();  
 }  
 if(command == 3){  
 *active\_work*();  
 }  
 if (command == 4){  
 *show\_salary*(name,acc\_type);  
 }  
 if(command>6 || command < 1 ){  
 System.*out*.println("Error, there is no such command here, please try again :-(");  
 continue;  
 }else if (command == 5){  
 Main.*exit*();  
 break;  
 }  
 }  
}

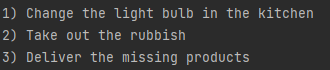
Pic.45. Worker’s menu

1-2. Show a list of the cases I have been entrusted with\ Show list of completed instructions.

private static void show\_work(String file) throws IOException {  
String[] arr = Context.*get\_lines*(file);  
 int count = 1;  
 for (int i = 0; i <= arr.length-1; i++){  
 System.*out*.println(count+") "+arr[i]);  
 count++;  
 }}  
 private static void show\_finished\_work() throws IOException{  
 *show\_work*("finished\_work");  
}

Pic.46 List of the cases worker have been entrusted with\ Show list of completed instructions

Here is the same as in the case of pay rises and downgrades. Only here only the name of the file changes. When any of these commands are called, the file parameter changes depending on the selection.

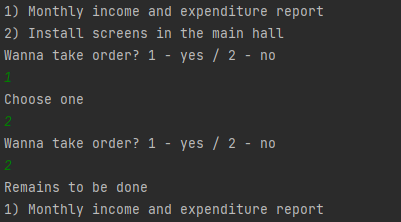


Pic.47 List of the cases worker have been entrusted with



Pic.48 Show list of completed instructions

1. Show list of cases I am working on - Shows the tasks he is already doing.



Pic.49. Shows the tasks he is already doing

private static void active\_work() throws IOException{  
 String[] arr = Context.*get\_lines*("active\_work");  
 int count = 1;  
 for (int i = 0; i <= arr.length-1; i++){  
 System.*out*.println(count+") "+arr[i]);  
 count++;  
 }  
 while (true){  
 System.*out*.println("Wanna take order? 1 - yes / 2 - no");  
 int choice = *input*.nextInt();  
 if (choice == 1){  
 System.*out*.println("Choose one");  
 int line = *input*.nextInt();  
 Context.*from\_to*("active\_work","finished\_work",arr[line-1]);  
 }else  
  
 break;  
 }  
 System.*out*.println("Remains to be done");  
 *show\_work*("active\_work");  
 }

Pic.50. active\_work method

The third command allows you to determine which tasks the worker is currently performing. The worker can also change this list by finishing the task, by pressing 1, the system will ask him for the number of the task he has completed. After the worker has specified the task, it will go to the finished\_work list where all finished tasks are stored.

1. Show wages - shows the salary of that particular worker.



Pic.51. The work of fourth command

private static void show\_salary(String name, String acc\_type) throws IOException{  
 String[] arr = Context.*show\_salary\_for\_user*("salary",name,acc\_type);  
 if (arr.length == 0){  
 System.*out*.println("We didn't find you");  
 }else {  
 System.*out*.println("Employee " + arr[0] + " receives - " + arr[2] + " $");  
 }  
}

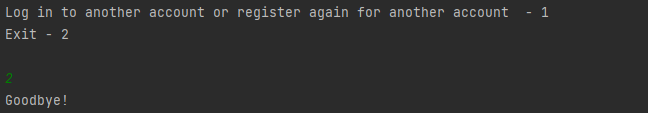
Pic.52. Showing the worker's salary

public static String[] show\_salary\_for\_user(String file, String name, String acc\_type) throws IOException {  
 BufferedReader br = null;  
 br = new BufferedReader(new FileReader(file));  
 ArrayList<String> array = new ArrayList<String>();  
 String line;  
 while ((line = br.readLine()) != null) {  
 String[] arr = line.split(" ");  
 if (name.equals(arr[0]) && acc\_type.equals(arr[1])) {  
 array.add(arr[0] + " ");  
 array.add(arr[1] + " ");  
 array.add(arr[2] + " ");  
 br.close();  
 break;  
 }  
 }  
 return array.toArray(new String[0]);  
}

Pic.53. show\_salary\_for\_user method in Context file

To implement this, the system calls the show\_salary\_for\_user method in the Context file. This method takes as parameters a file containing all salaries of the employees, name and type of account. Then in the method it reads the whole file with salaries and compares it to the account name and type that was passed as parameters. If it finds one, it saves it into an array and sends it back, if it doesn't, it sends an empty array. After this there is a condition that checks the length of the array, if its length is zero then there is no such user. And if he is, then displays in the terminal his salary.

1. Exit



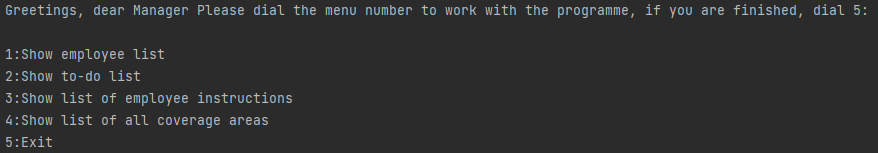
Pic.54. Work of the ninth command

When logging out you will be able to switch straight to another account without turning off the project by pressing 1, but if you want to log out you have to enter 2. The code is shown at picture 3.

# **Account of a manager**

Here we have entered the manager's account. The manager can select one of five preset commands from the main menu. He must enter a number between 1 and 5.

If the number entered is not 1 to 5, he will be stuck in a loop until he enters the correct number.

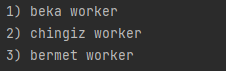


Pic.55. Manager’s menu

static Scanner *input* = new Scanner(System.*in*);  
public static void main(String name, String acc\_type) throws IOException{  
 System.*out*.println("Greetings, dear Manager " +  
 "Please dial the menu number to work with the programme, if you are finished, dial 5:\n");  
 String[] to\_do = {"Show employee list ",  
 "Show to-do list",  
 "Show list of employee instructions ",  
 "Show list of all coverage areas ",  
 "Exit"};  
 while (true) {  
 int count = 1;  
 for (String work : to\_do) {  
 System.*out*.println(count + ":" + work);  
 count++;  
 }  
 int command = *input*.nextInt();  
 if(command == 1){  
 *show\_workers*();  
 }  
 if(command == 2){  
 *show\_to\_do\_list*();  
 }  
 if (command == 3){  
 *workers\_instructions*();  
 }  
 if (command == 4){  
 *areas\_kg*();  
 }  
 if (command == 5){  
 Main.*exit*();  
 break;  
 }  
 }  
}

Pic.55. Manager’s menu

1. Show employee list– shows a list of all employees.



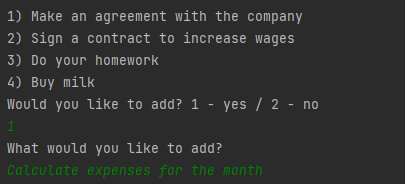
Pic.56. Show orders

private static void show\_workers(){  
 String[] arr = Context.*find\_workers*("users.txt");  
 int count = 1;  
 for (String line : arr){  
 String[] array = line.split(" ");  
 System.*out*.println(count + ") "+array[0] +" "+ array[2]);  
 count++;  
 }  
}

Pic.56. Show orders

The first command allows us to find out the list of all workers who are registered in our database.

1. Show to-do list – shows the tasks to be performed.



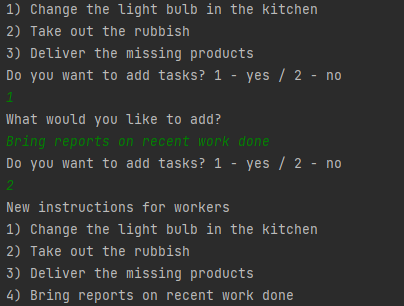
Pic.56. Deliver material

private static void show\_to\_do\_list() throws IOException {  
 String[] arr = Context.*get\_lines*("todo\_list\_manager");  
 if (arr.length == 0){  
 System.*out*.println("You have nothing.");  
 }else{  
 int count = 1;  
 for (int i = 0; i <= arr.length-1; i++){  
 System.*out*.println(count+") "+arr[i]);  
 count++;  
 }}  
 while (true){  
 System.*out*.println("Would you like to add? 1 - yes / 2 - no");  
 int ch = *input*.nextInt();  
 if (ch == 1){  
 System.*out*.println("What would you like to add?");  
 String addTask2 = *input*.nextLine();  
 String addTask = *input*.nextLine();  
 Context.*write*("todo\_list\_manager",addTask);  
 }else{  
 break;  
 }  
 }  
}

Pic.56. Deliver material

The second command works like a conspicuous book where the manager writes what he needs to do. Using the write (pic. 6) method in the Context. Passes as parameters the name of the file and what needs to be written to the file.

1. Show list of employee instructions – assigns tasks to workers.



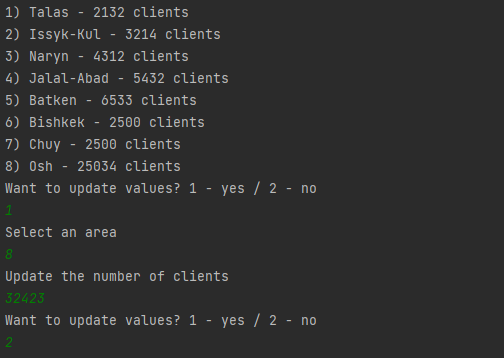
Pic.57. Delivered materials

private static void workers\_instructions() throws IOException{  
 String[] arr = Context.*get\_lines*("need\_to\_do");  
 int count = 1;  
 for (String line : arr){  
 System.*out*.println(count+") "+line);  
 count++;  
 }  
 while (true){  
 System.*out*.println("Do you want to add tasks? 1 - yes / 2 - no");  
 int choice = *input*.nextInt();  
 if (choice == 1){  
 System.*out*.println("What would you like to add?");  
 String line2 = *input*.nextLine();  
 String line = *input*.nextLine();  
  
 Context.*write*("need\_to\_do",line);  
 }else {  
 break;  
 }}  
 String[] arr2 = Context.*get\_lines*("need\_to\_do");  
 System.*out*.println("New instructions for workers");  
 int count2 = 1;  
 for (String line : arr2){  
 System.*out*.println(count2+") "+line);  
 count2++;  
 }  
}

Pic.57. Delivered materials

With the third command, the manager can give tasks to workers. By calling the write method (pic. 6) in the Context. The manager writes the job to a file, which the workers can then see.

1. Show list of all coverage areas – shows the number of customers by area.



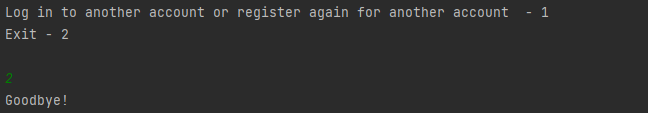
Pic.58. Exit from program

private static void areas\_kg() throws IOException{  
 String[] areas = Context.*get\_lines*("areas");  
 int count = 1;  
 for (String line : areas){  
 System.*out*.println(count + ") "+ line);  
 count++;  
 }  
 while (true){  
 System.*out*.println("Want to update values? 1 - yes / 2 - no");  
 int choice = *input*.nextInt();  
 if (choice == 1){  
 System.*out*.println("Select an area");  
 int line = *input*.nextInt()-1;  
 System.*out*.println("Update the number of clients");  
 int client\_count = *input*.nextInt();  
 Context.*update\_clients\_value*("areas",areas[line],client\_count,2);  
 }else {  
 break;  
 }  
 }  
}

Pic.58. Exit from program

The fourth command allows you to view the customer coverage by area. The manager can also change the number of clients in different areas by calling the update\_client\_value method (pic.17) in the Context file. The parameters passed are the file to work on, the area, the new number of clients, and access to the client cell itself. The method then changes the value in the file.

1. Exit



Pic.42. Work of the ninth command

When logging out you will be able to switch straight to another account without turning off the project by pressing 1, but if you want to log out you have to enter 2. The code is shown at picture 3

# **Conclusion**

By doing our project we achieved our goals.

Our goals were:

→ To create a program to facilitate process for people who work at KFC

→ To automate the work of program’s handling of big data

→ Control the work of all KFC employees

Our program works and can complete all of it’s functions that were given in the technical requirement. Also, we made some additional functions to improve the functionality of program. Despite the fact, there was not so much time, we made a demo version of a program, that can really be used in future, if it will be made more dynamic.

**We learned how to:**

1. work with different files in Java;

2. work with different types of functions in Java;

3. catch IO exceptions;

4. create the best possible algorithm.

# **References:**

1. INAI.kg Lectures
2. StackOverFlow - <https://ru.stackoverflow.com/>
3. Javarush - <https://javarush.ru/>
4. CodeRoad - <https://ru.stackoverflow.com/questions/tagged/java>