Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Институт информационных технологий и анализа данных

наименование института

ОТЧЕТ  
к лабораторной работе по дисциплине

Объектно-ориентированное программирование

Разработка классов с использованием механизмов наследования, полиморфизма и инкапсуляции

Наименование темы

Выполнил студент группы ИСТб 19–2 Первых А.А.

шифр Фамилия И.О.

Проверила Маланова Т.В.

Фамилия И.О.

Содержание отчета на 18 стр.

Иркутск 2020 г.

Оглавление

[1 Постановка задачи 3](#_Toc58184130)

[2 Структура меню 3](#_Toc58184131)

[3. Проектирование классов. 4](#_Toc58184132)

[4 Таблица спецификаций 5](#_Toc58184133)

[4.1 Описание класса Staff 5](#_Toc58184134)

[4.2 Описание класса Group 5](#_Toc58184135)

[4.3 Описание класса Administration. 6](#_Toc58184136)

[4.4 Описание класса Engineer. 6](#_Toc58184137)

[4.5 Описание класса Worker. 7](#_Toc58184138)

[5 Разработка тестов 8](#_Toc58184139)

[6 Результаты тестирования 9](#_Toc58184140)

[7. Листинг исходного кода 11](#_Toc58184141)

[Заключение 17](#_Toc58184142)

[Список литературы 18](#_Toc58184143)

1 Постановка задачи

В соответствии с индивидуальным заданием описать иерархию классов, для каждого класса описать поля и соответствующие методы доступа к ним.

В зависимости от задания некоторые из этих классов (как минимум один) являются абстрактными классами и служат для выделения общих данных и поведения для других классов. Абстрактный класс должен содержать как минимум один абстрактный метод, реализация которого у его наследников должна различаться.

Помимо этого в общую часть задания входит разработка класса, группирующего объекты описанных в соответствии с заданием классов. Для выполнения задания необходимо создать некоторое количество объектов, добавить их в группу используя предусмотренные методы класса «группы» и для каждого из них вызвать унаследованй метод. Индивидуальное задание: Реализовать следующие классы: Рабочий, Кадры, Инженер, Администратор

Разработать класс-группу. Описать иерархию данных классов.

2 Структура меню

Главное меню

new - добавить сотрудника

delete - удалить сотрудника

search - найти сотрудника

info - показать информацию о сотруднике

work - заставить сотрудника работать

exit - завершение программы

Профессия

1. Администратор

2. Инженер

3. Рабочий

Классы, реализующие пользовательское меню:

Класс Menu\_control (главное меню) – главный класс пользовательского меню, предоставляющий интерфейс взаимодействия с пользователем.

Класс Menu\_graphics – класс подменю, отвечающий за представления меню пользователю.

Класс Menu\_logic – класс подменю, отвечающий за обработку команд.

3. Проектирование классов.

На рисунке 1 представлена UML диаграмма классов.

Staff

Group

public static Random;

public static ArrayList staff;

public static void add;

public static void delete;

public static void search;

public static void info;

public static void work;

private String name;

private int age;

private String profession;

public Staff;

public abstract void work;

public void getInfo;

public String getName;

public void setName;

public int getAge;

public void setAge;

public String getProfession;

public void setProfession

Engineer

public Engineer;

public void work;

Worker

public Worker;

public void work;

Administration

public Administration;

public void work;

Класс Staff – абстрактный класс кадры служит для выделения общих данных и поведения для других классов.

Класс Group - группирует объекты.

Класс Administration – Класс администратора.

Класс Engineer – Класс инженера.

Класс Worker – Класс рабочего.

4 Таблица спецификаций

4.1 Описание класса Staff

Таблица 1 – Поля абстрактного класса Staff

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | name | Имя кадра | String | private |
| 2 | age | Возраст кадра | int | private |
| 3 | profession | Профессия кадра | String | private |

Таблица 2 – Конструкторы абстрактного класса Staff

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип доступа | Параметры |
| 1 | Staff | Конструктор задания полей. | public | name |
| age |
| profession |

Таблица 3 – Методы класса

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа | Абстрактный/ неабстрактный | Параметры |
| 1 | work | Метод работы | String | public | Абстрактный |  |
| 2 | getInfo | getter информации о кадре | String | public | Абстрактный |  |
| 3 | getName | getter имени | String | public | неабстрактный |  |
| 4 | setName | setter имени | String | public | неабстрактный | name |
| 5 | getAge | getter возраста | int | public | неабстрактный |  |
| 6 | setAge | setter возраста | int | public | неабстрактный | age |
| 7 | getProfession | getter профессии | String | public | неабстрактный |  |
| 8 | setProfession | setter профессии | String | public | неабстрактный | profession |

4.2 Описание класса Group

Таблица 4 – Поля абстрактного класса Group

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | ArrayList<Staff> staff | Лист объектов | List | public |
| 2 | Random rnd | Создания рандома |  | public |

Таблица 5 – Методы класса

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа | Абстрактный/ неабстрактный | Параметры |
| 1 | add | Добавление объекта | String | public static | неабстрактный | String[] list |
| 2 | delete | Удаление объекта | String | public static | неабстрактный | String name |
| 3 | search | Поиск объекта | String | public static | неабстрактный | String name |
| 4 | info | Информация о кадрах | String | public static | неабстрактный |  |
| 5 | work | Работа кадра | int | public static | неабстрактный | String name |

4.3 Описание класса Administration.

Таблица 6 – Поля абстрактного класса Administration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | Random rnd | Создания рандома |  | public |

Таблица 7 – Конструкторы абстрактного класса Administration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип доступа | Параметры |
| 1 | Administration | Конструктор объекта Administration | public | name |
| age |
|  |

Таблица 8 – Методы класса

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа | Абстрактный/ неабстрактный | Параметры |
| 1 | work | Метод работы | String | public | неабстрактный |  |

4.4 Описание класса Engineer.

Таблица 9 – Поля абстрактного класса Engineer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | Random rnd | Создания рандома |  | public |

Таблица 10 – Конструкторы абстрактного класса Engineer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип доступа | Параметры |
| 1 | Engineer | Конструктор объекта Engineer | public | name |
| age |
|  |

Таблица 11 – Методы класса

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа | Абстрактный/ неабстрактный | Параметры |
| 1 | work | Метод работы | String | public | неабстрактный |  |

4.5 Описание класса Worker.

Таблица 12 – Поля абстрактного класса Worker

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | Random rnd | Создания рандома |  | public |

Таблица 13 – Конструкторы абстрактного класса Worker

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип доступа | Параметры |
| 1 | Worker | Конструктор объекта Worker | public | name |
| age |
|  |

Таблица 14 – Методы класса

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа | Абстрактный/ неабстрактный | Параметры |
| 1 | work | Метод работы | String | public | неабстрактный |  |

5 Разработка тестов

Таблица 15 – Таблица тестов

|  |  |  |
| --- | --- | --- |
| № | Действие | Результат действия |
| 1 | Первоначальный запуск программы | Вывод меню действий |
| 2 | Создание нового кадра | Создания кадра, ввод имени, возвраста, и профессии. Команда new. |
| 3 | Поиск заданного кадра | Поиск кадра в списке по имени. Команда Search |
| 4 | Получение данных о кадре | Вывод информации о кадрах. Команда info. |
| 5 | Удаление кадра | Удаление кадра по имени. Команда delete. |
| 6 | Действие кадра | Вывод действия кадра. Команда work. |
| 7 | Завершение  работы программы | Выход из программы. Команда exit |
| 8 | Предупреждения при отсутсвие кадров | Вывод предупреждения |
| 9 | Неправильный ввод команды | Вывод предупреждения |

6 Результаты тестирования

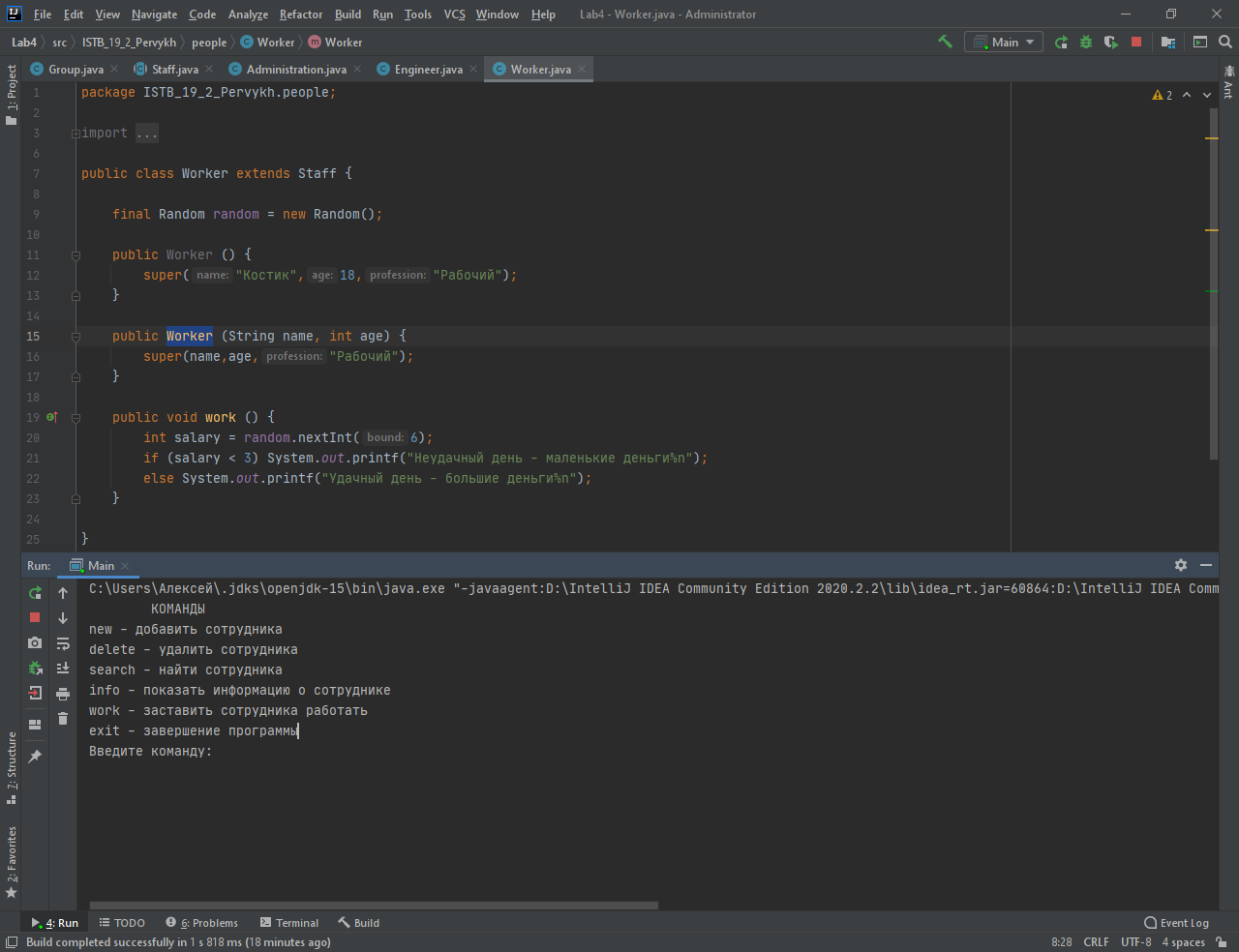


Рисунок 1 – Главное меню программы.

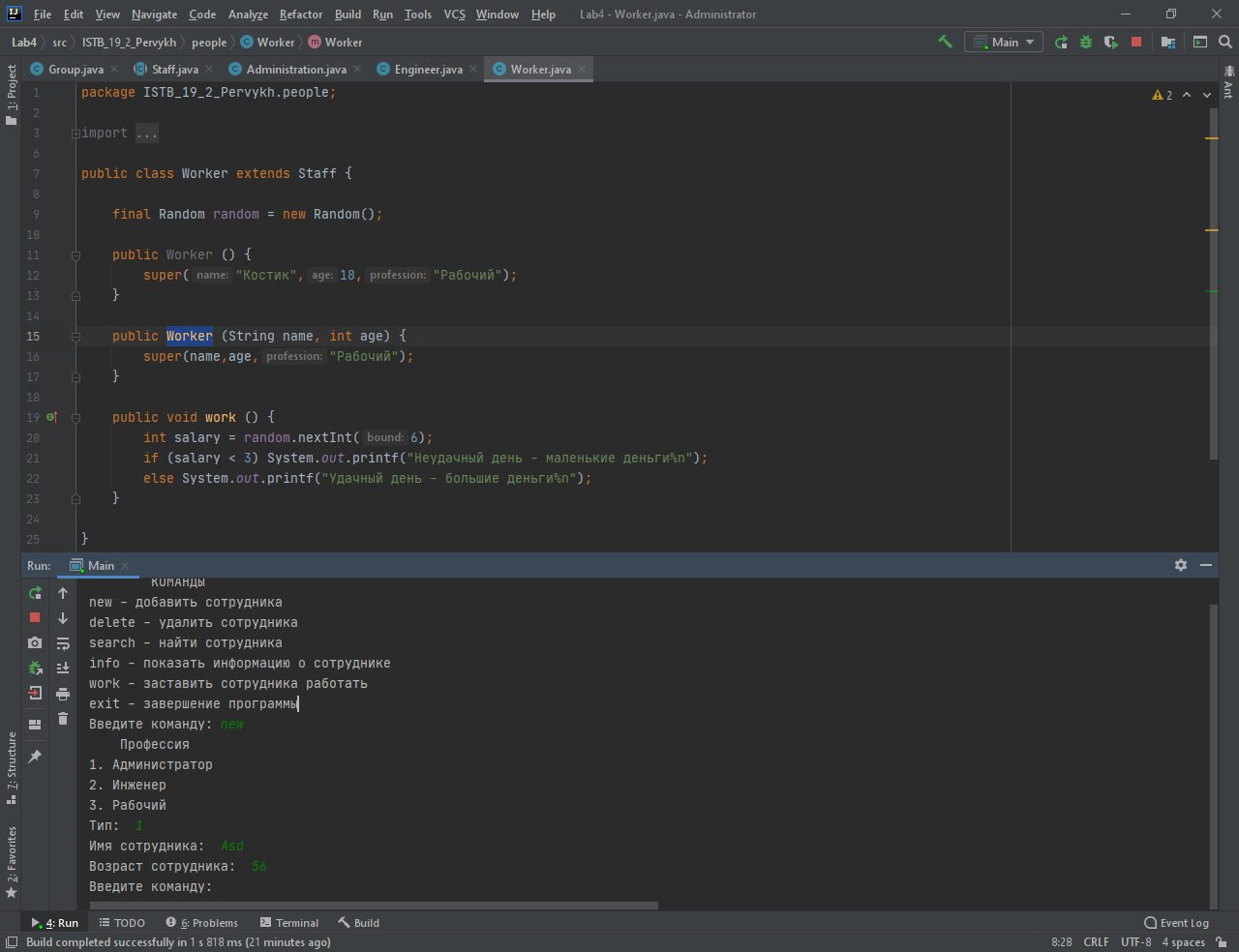


Рисунок 2 – Создание нового кадра.

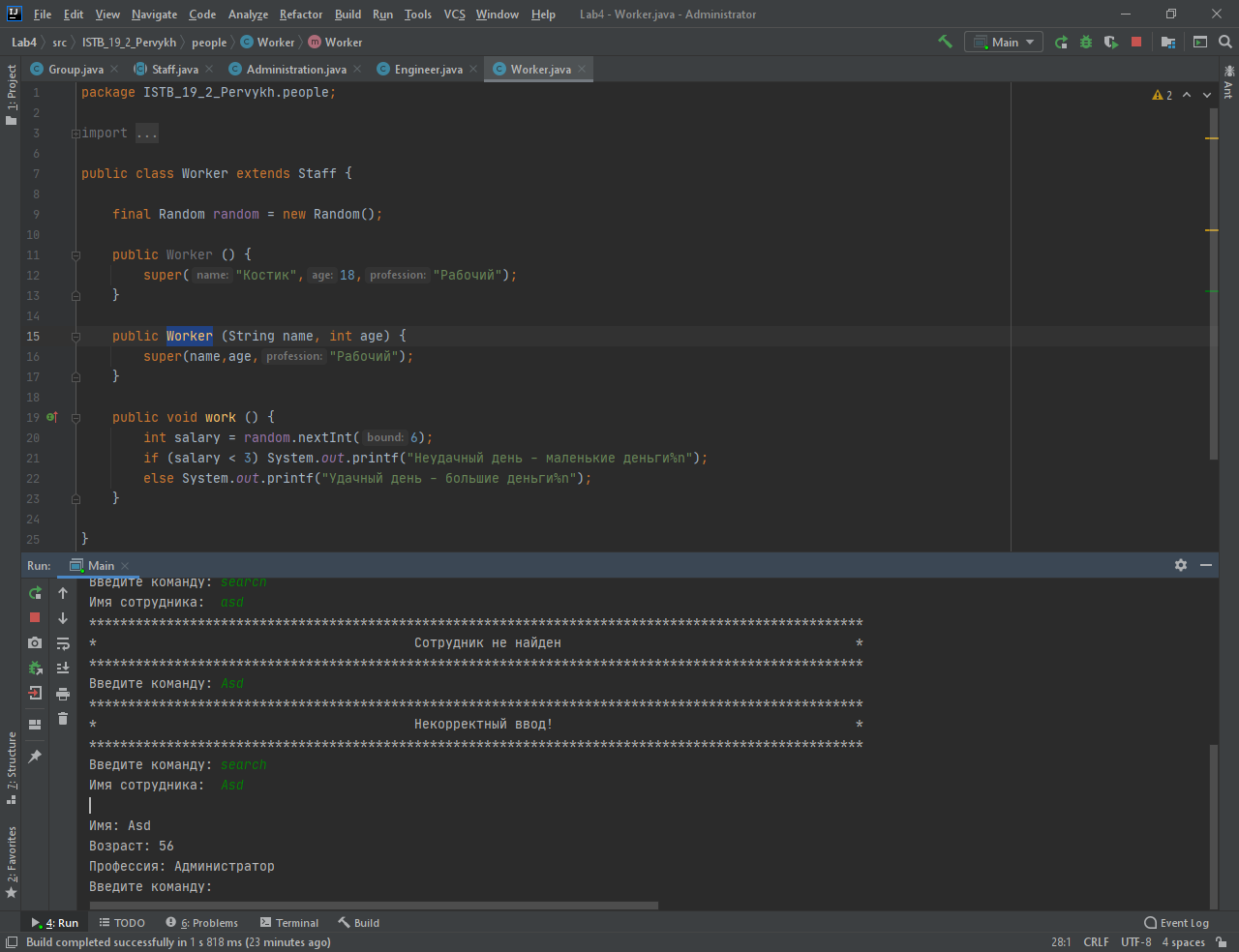


Рисунок 3 – Поиск заданного кадра.

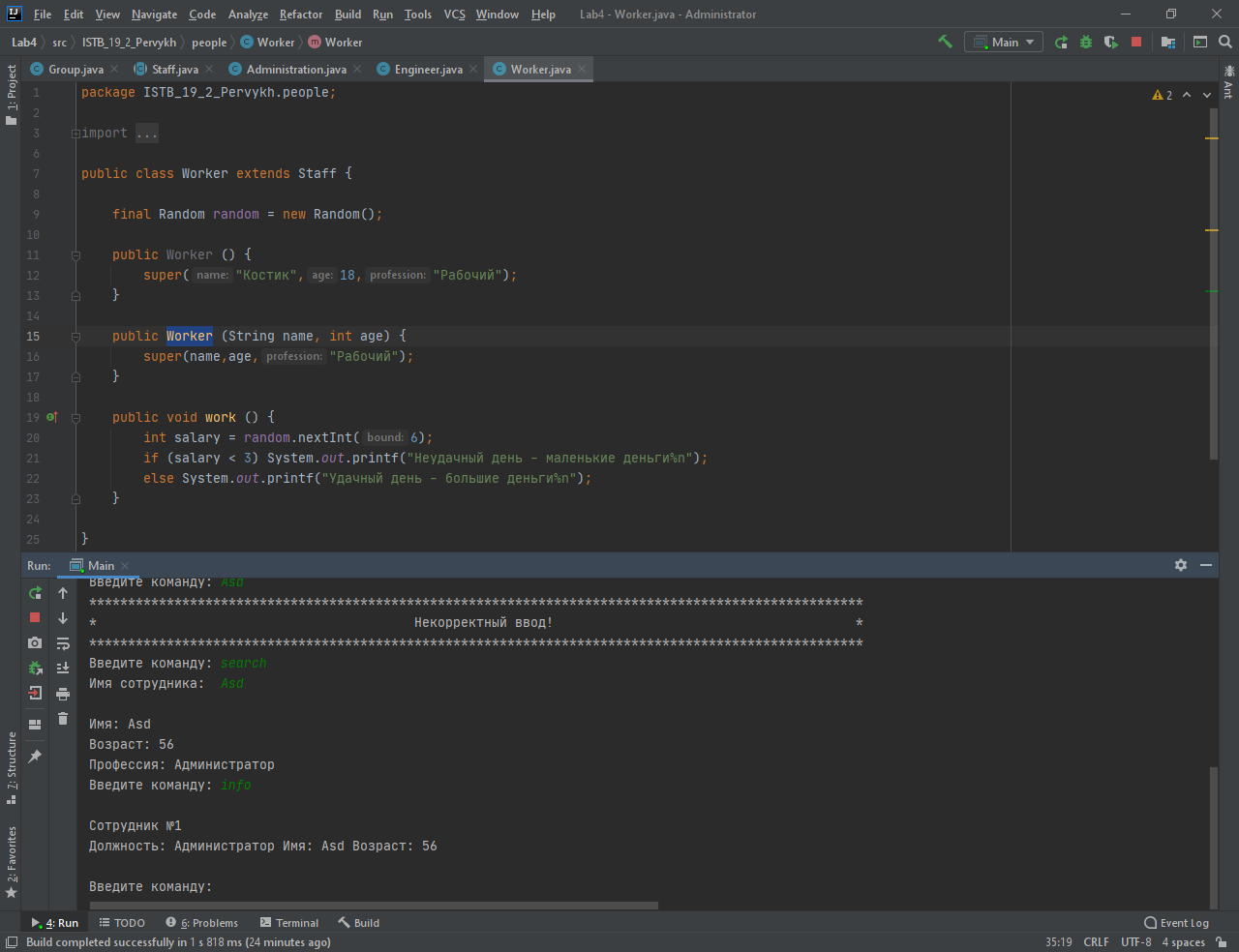


Рисунок 4 – Получение данных о кадре.

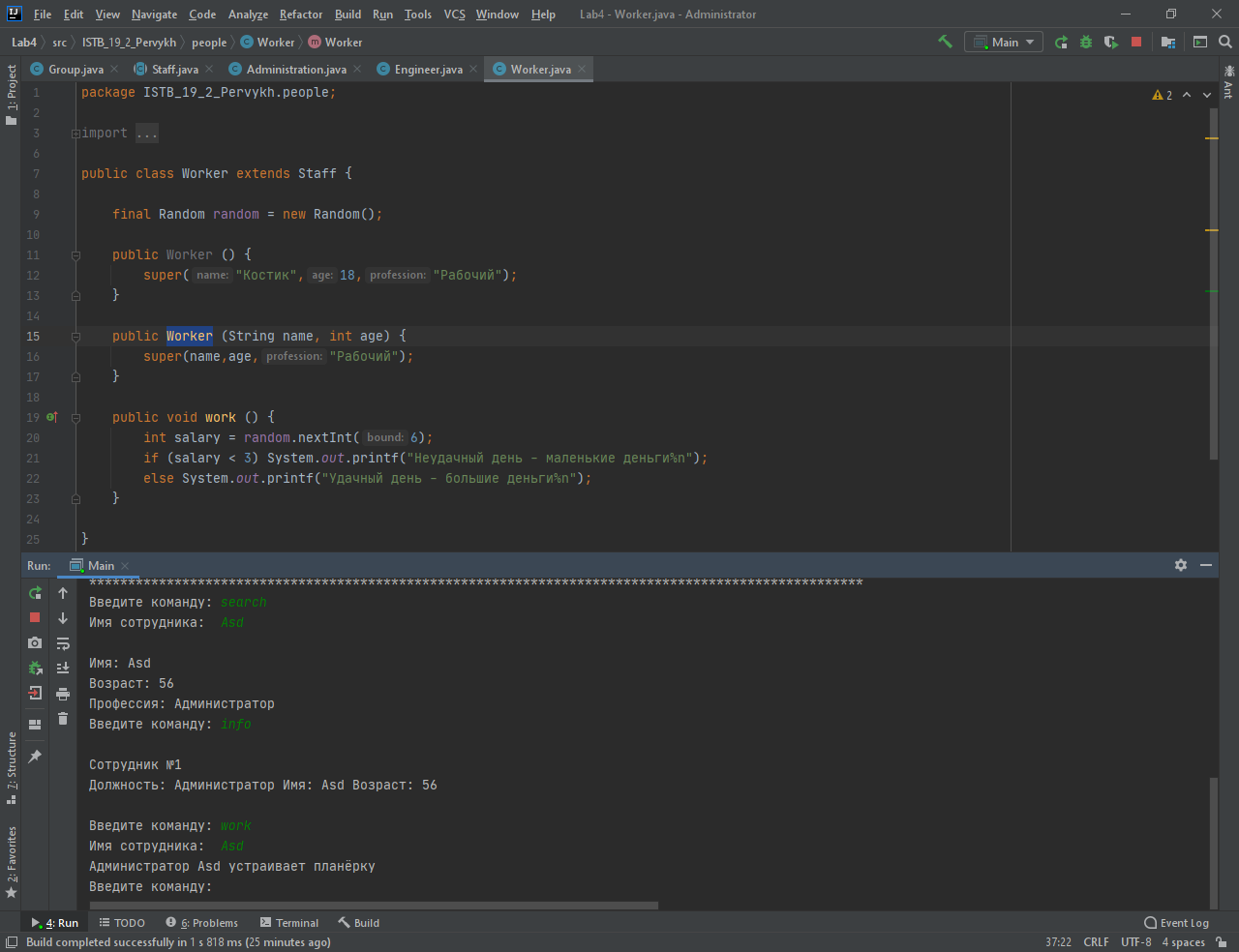


Рисунок 5 – Действие кадра.

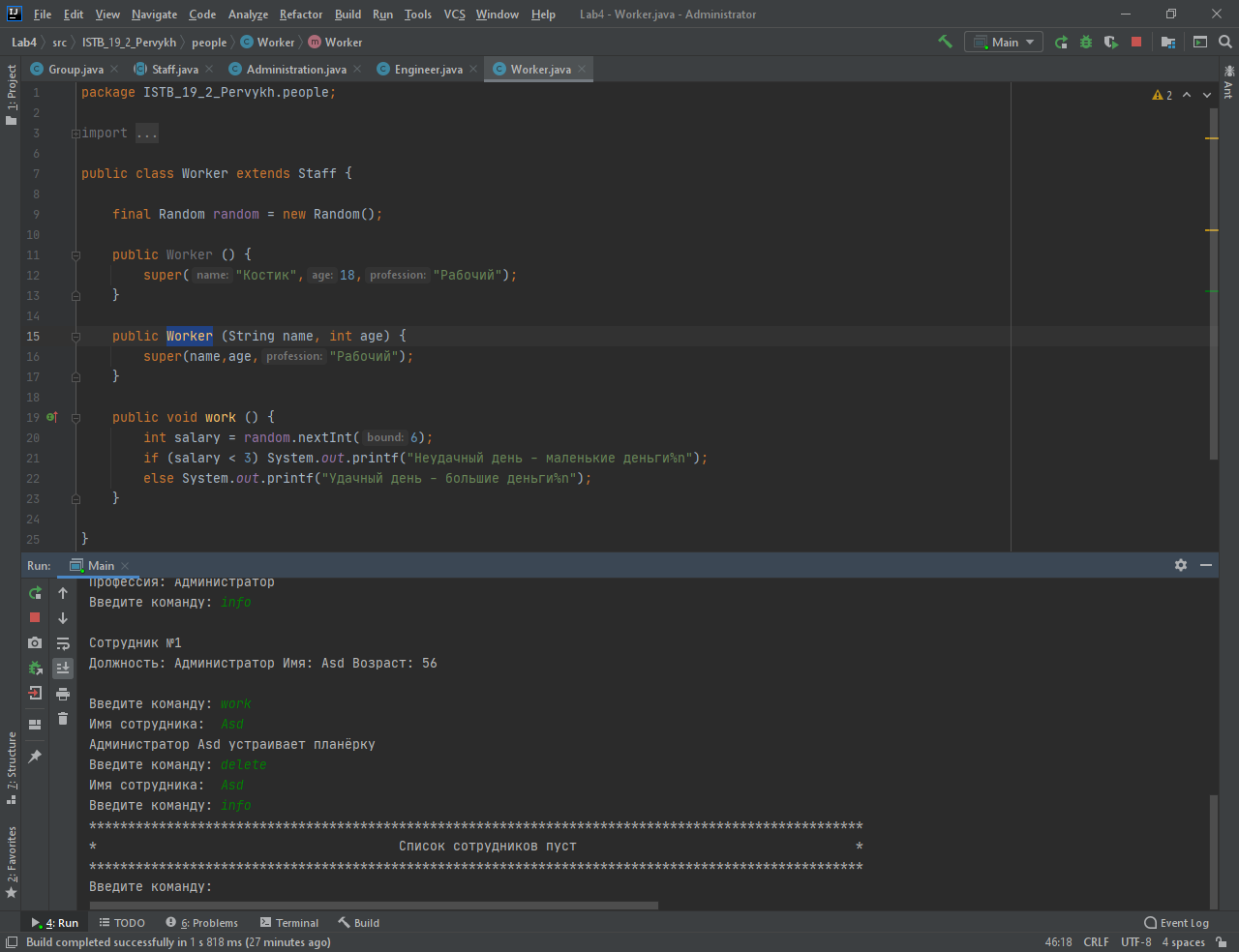


Рисунок 6 – Удаление кадра.

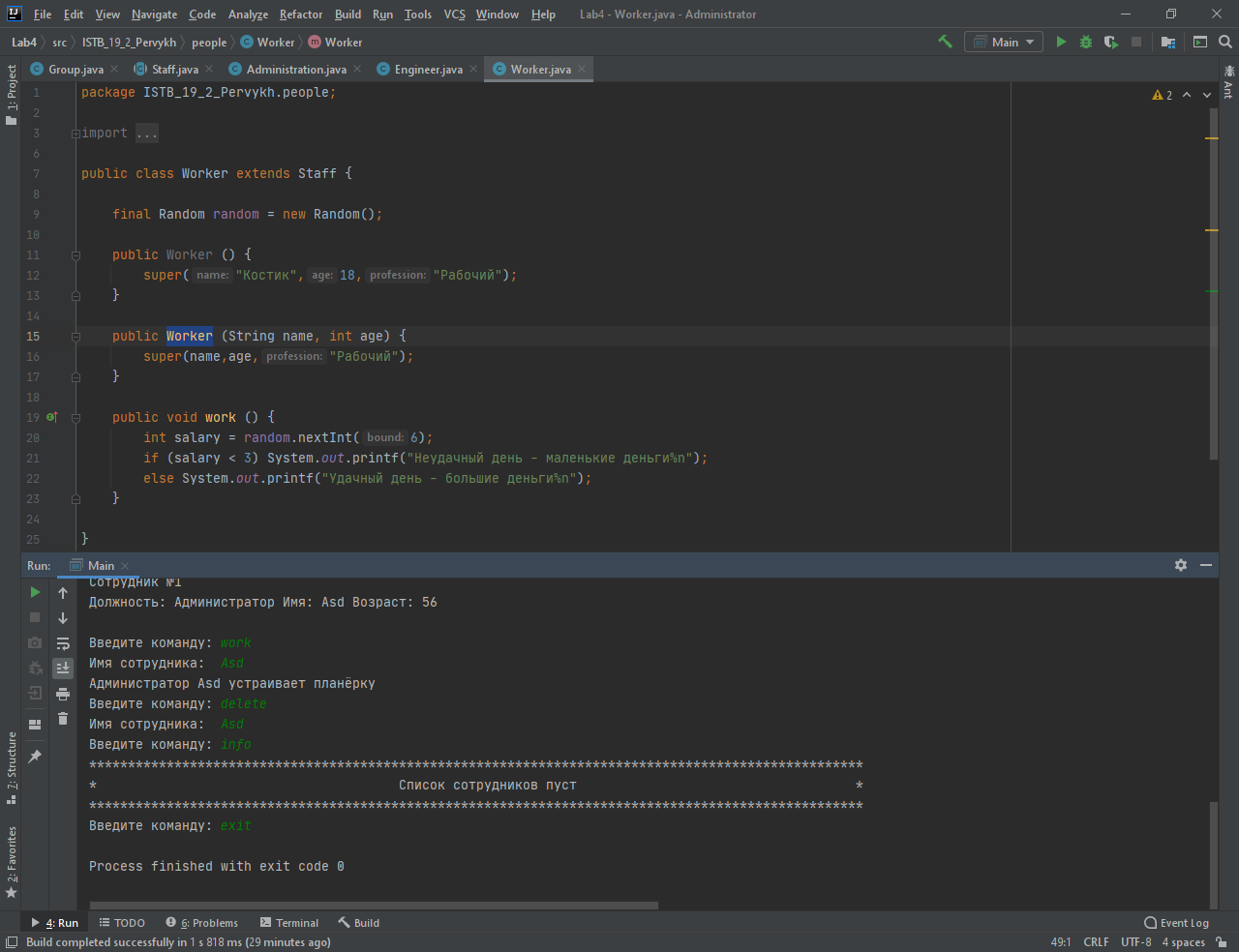


Рисунок 7 – Завершение работы программы.

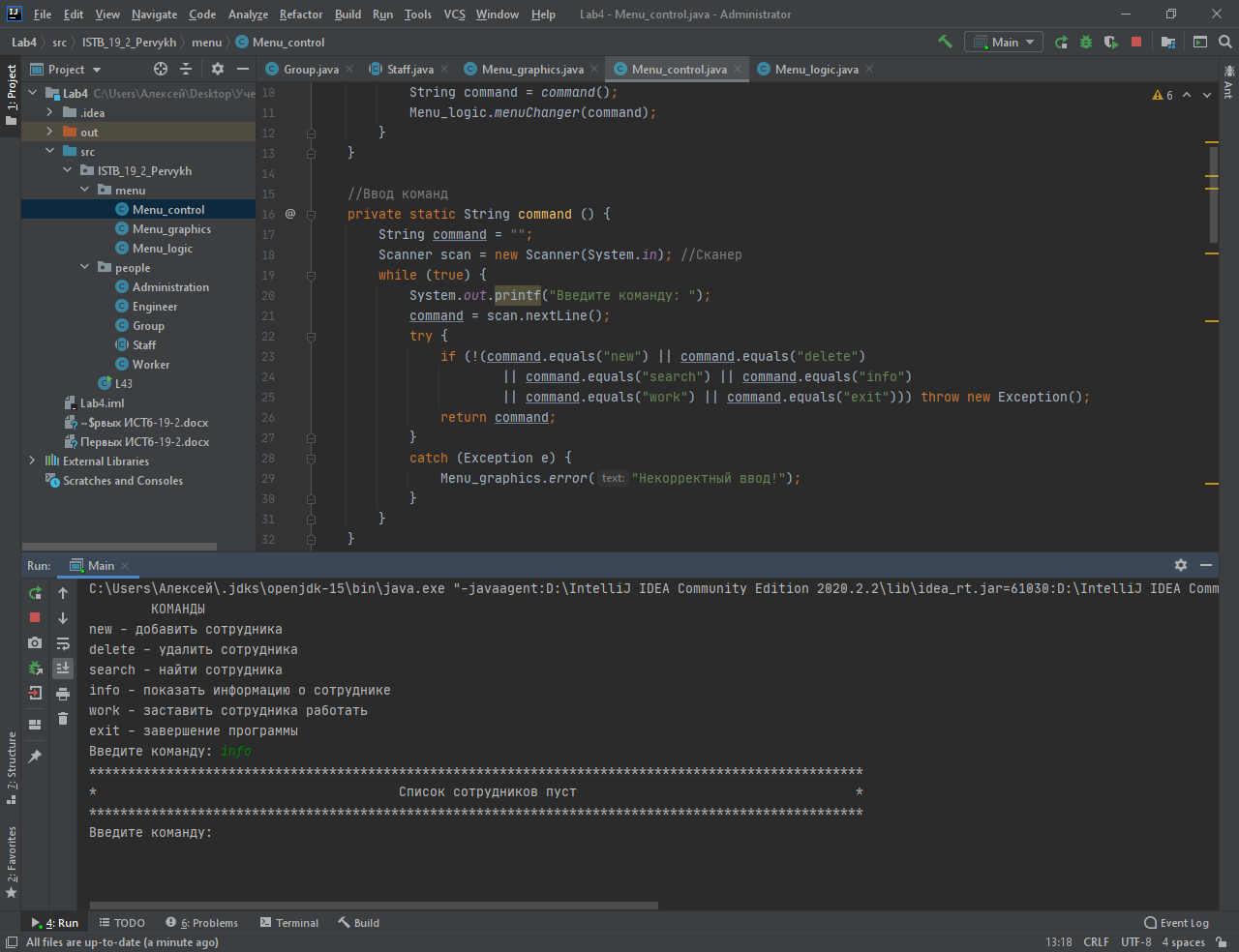


Рисунок 8 – Предупреждения при отсутсвие кадров.

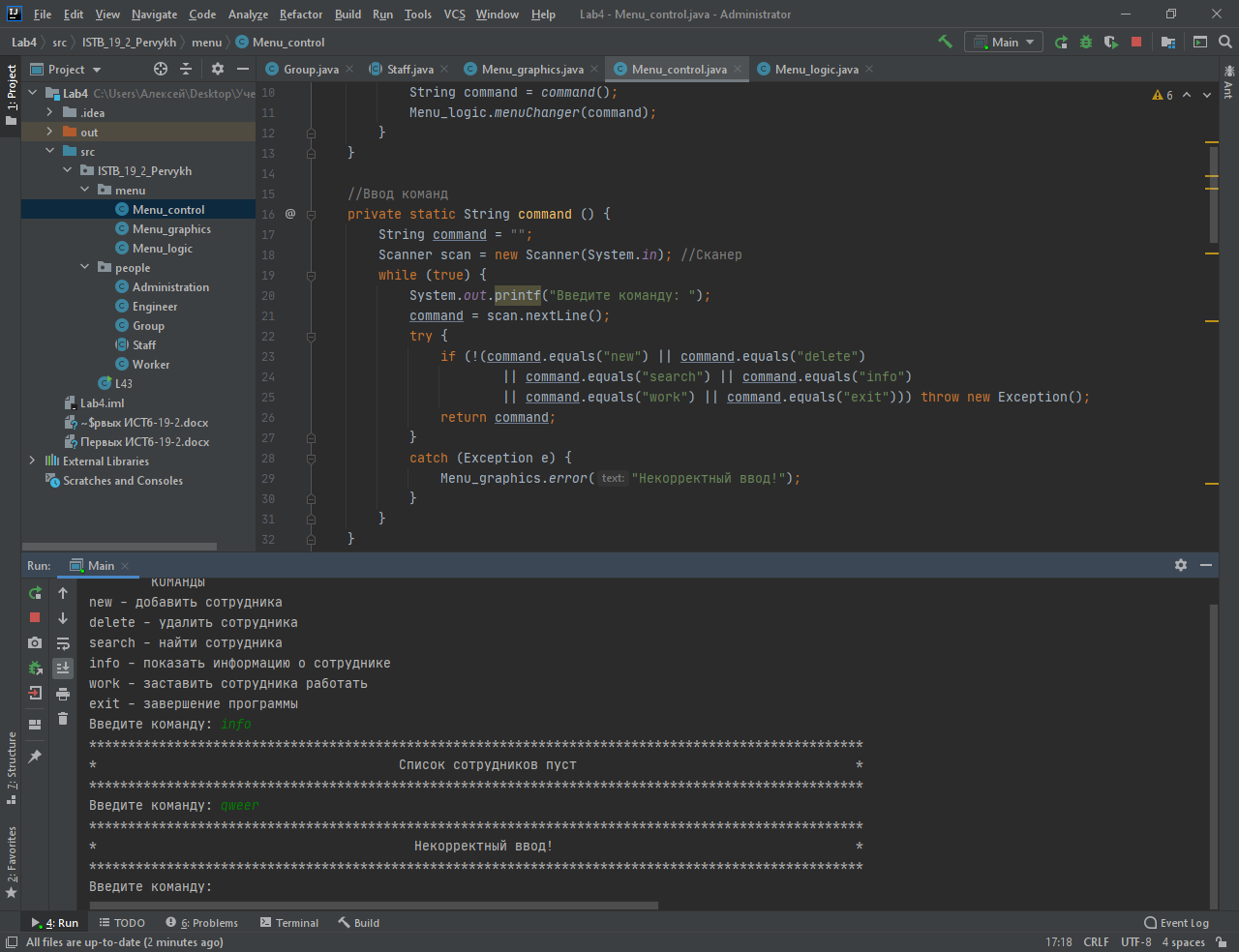


Рисунок 9 – Неправильный ввод команды.

7. Листинг исходного кода

**Класс Main**

Menu\_graphics.commands();

Menu\_control.menuCommand();

**Класс Menu\_graphics**

//Рисует меню команд

public static void commands (){

System.out.println(" КОМАНДЫ");

System.out.println("new - добавить сотрудника");

System.out.println("delete - удалить сотрудника");

System.out.println("search - найти сотрудника");

System.out.println("info - показать информацию о сотруднике");

System.out.println("work - заставить сотрудника работать");

System.out.println("exit - завершение программы");

}

//Оформленный вывод сообщения об ошибке

public static void error (String text) {

line("\*");

centralText("\*", text);

line("\*");

}

//Вывод текста при вводе

public static void inputText (String text) {

System.out.printf("%s ", text);

}

//Вывод оформленного текста в центр

public static void centralText(String symbol, String text) {

System.out.print(symbol);

for (int i = 0; i < 98-text.length(); i++) {

System.out.print(" ");

if (i == 49-text.length()/2)

System.out.print(text);

}

System.out.println(symbol);

}

//Вывод оформленного текста

public static void text (String symbol, String text) {

System.out.printf("%s %s", symbol, text);

for (int i = 0; i < 97-text.length(); i++) {

System.out.print(" ");

}

System.out.println(symbol);

}

//Вывод оформительной линии

public static void line(String symbol) {

for (int i = 0; i < 100; i++) {

System.out.print(symbol);

}

System.out.println();

}

**Класс Menu\_control**

//Управление действиями

public static void menuCommand () {

while (true) {

String command = command();

Menu\_logic.menuChanger(command);

}

}

//Ввод команд

private static String command () {

String command = "";

Scanner scan = new Scanner(System.in); //Сканер

while (true) {

System.out.printf("Введите команду: ");

command = scan.nextLine();

try {

if (!(command.equals("new") || command.equals("delete")

|| command.equals("search") || command.equals("info")

|| command.equals("work") || command.equals("exit"))) throw new Exception();

return command;

}

catch (Exception e) {

Menu\_graphics.error("Некорректный ввод!");

}

}

}

public static String[] add () {

String list [] = new String[3];

Scanner scan = new Scanner(System.in); //Сканер

boolean cycle = true;

System.out.println(" Профессия");

System.out.println("1. Администратор");

System.out.println("2. Инженер");

System.out.println("3. Рабочий");

while (cycle) {

Menu\_graphics.inputText("Тип: ");

int i = 0;

try {

i = scan.nextInt();

}

catch (Exception e) {

Menu\_graphics.error("Некорректный ввод");

}

switch (i) {

case 1:

list [0] = "Администратор";

cycle = false;

break;

case 2:

list [0] = "Инженер";

cycle = false;

break;

case 3:

list [0] = "Рабочий";

cycle = false;

break;

default:

Menu\_graphics.error("Некорректный ввод");

break;

}

}

list [1] = staffName();

list [2] = staffAge();

return list;

}

public static String staffName () {

Scanner scan = new Scanner(System.in);

while (true) {

Menu\_graphics.inputText("Имя сотрудника: ");

String name = scan.nextLine();

if (!name.equals("")) return name;

Menu\_graphics.error("Некорректный ввод");

}

}

public static String staffAge () {

Scanner scan = new Scanner(System.in);

while (true) {

try {

Menu\_graphics.inputText("Возраст сотрудника: ");

String age = Integer.toString(scan.nextInt());

return age;

}

catch (Exception e) {

Menu\_graphics.error("Некорректный ввод");

}

}

}

**Класс Menu\_logic**

//Статус пустоты списка (true = пустой)

public static boolean listNull = true;

//Обработка команд

public static void menuChanger (String command) {

switch (command) {

case "new":

Group.add(Menu\_control.add());

listNull = false;

break;

case "delete":

if (!listNull)

Group.delete(Menu\_control.staffName());

else

Menu\_graphics.error("Список сотрудников пуст");

break;

case "search":

if (!listNull)

Group.search(Menu\_control.staffName());

else

Menu\_graphics.error("Список сотрудников пуст");

break;

case "info":

if (!listNull)

Group.info();

else

Menu\_graphics.error("Список сотрудников пуст");

break;

case "work":

if (!listNull)

Group.work(Menu\_control.staffName());

else

Menu\_graphics.error("Список сотрудников пуст");

break;

case "exit":

System.exit(0);

break;

}

}

**Класс Staff**

private String name;

private int age;

private String profession;

public Staff(String name, int age, String profession) {

this.name = name;

this.age = age;

this.profession = profession;

}

public abstract void work ();

public void getInfo () {

System.out.printf("Имя: %s%n", name);

System.out.printf("Возраст: %d%n", age);

System.out.printf("Профессия: %s%n", profession);

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String getProfession() {

return profession;

}

public void setProfession(String profession) {

this.profession = profession;

}

**Класс Group**

public static Random rnd = new Random();

public static ArrayList<Staff> staff = new ArrayList<Staff>();

public static void add (String[] list){

switch (list[0]) {

case "Рабочий":

staff.add(new Worker(list[1], Integer.parseInt(list[2])));

break;

case "Инженер":

staff.add(new Engineer(list[1], Integer.parseInt(list[2])));

break;

case "Администратор":

staff.add(new Administration(list[1], Integer.parseInt(list[2])));

break;

}

}

public static void delete (String name) {

int i = -1;

for (Staff org: staff) {

if (name.equals(org.getName())) i = staff.indexOf(org);

}

if (i != -1) {

staff.remove(i);

if (staff.size() == 0) Menu\_logic.listNull = true;

}

else Menu\_graphics.error("Сотрудник не найден");

}

public static void search (String name) {

int check = 0;

for (Staff stf : staff) {

if (name.equals(stf.getName())) {

System.out.println();

stf.getInfo();

check++;

}

}

if (check == 0) Menu\_graphics.error("Сотрудник не найден");

}

public static void info () {

for (Staff stf : staff) {

System.out.println();

System.out.println("Сотрудник №" + (staff.indexOf(stf) + 1));

System.out.println("Должность: " + stf.getProfession() + " Имя: " + stf.getName() + " Возраст: " + stf.getAge());

System.out.println();

}

}

public static void work (String name) {

int check = 0;

for (Staff stf : staff) {

if (name.equals(stf.getName())) {

stf.work();

check++;

}

}

if (check == 0) Menu\_graphics.error("Сотрудник не найден");

}

**Класс Administration**

Random random = new Random();

public Administration () {

super("Костик",18,"Администратор");

}

public Administration (String name, int age) {

super( name, age, "Администратор");

}

public void work () {

int workType = random.nextInt(3);

switch (workType) {

case 0:

System.out.printf("%s %s устраивает планёрку%n", getProfession(), getName());

break;

case 1:

System.out.printf("%s %s раздаёт задания%n", getProfession(), getName());

break;

case 2:

System.out.printf("%s %s придумывает стратегию развития%n", getProfession(), getName());

break;

}

}

**Класс Engineer**

public Engineer () {

super("Костик",18,"Инженер");

}

public Engineer (String name, int age) {

super(name,age,"Инженер");

}

public void work () {

System.out.printf("Новый день - новый чертёж!");

}

**Класс Worker**

Random random = new Random();

public Worker () {

super("Костик",18,"Рабочий");

}

public Worker (String name, int age) {

super(name,age,"Рабочий");

}

public void work () {

int salary = random.nextInt(6);

if (salary < 3) System.out.printf("Неудачный день - маленькие деньги%n");

else System.out.printf("Удачный день - большие деньги%n");

}

Заключение

В процессе выполнения лабораторной работы 4: “Разработка консольного приложения” были разработаны алгоритмы решения индивидуального задания. Было сформировано 9 тестов, покрывающих все ситуации. Составленные на языке Java программы, реализующие разработанные алгоритмы, прошла все тесты успешно.

В ходе лабораторной работы были получены практические навыки использования конструкций языка Java, а также знакомство с платформой.

Все цели практики достигнуты.

Список литературы

1. Java. Экспресс-курс [электронный ресурс] // Сайт Александра Климова [сайт], URL: http://developer.alexanderklimov.ru/android/java/java.php

2. API Specification for the Java 7 SE. [официальный сайт] URL: http://docs.oracle.com/javase/7/docs/api/

3. The Java Tutorials. SE [электронный ресурс], URL: http://docs.oracle.com/javase/tutorial/index.html

4. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированны анализ и проек-тирование с примерами приложений. Третье издание. М.: "Вильямс", 2010.

5. Хабибуллин И.Ш. Java 7: для программистов / И. Ш. Хабибуллин. – Санкт-Петербург : БХВ–Петербург, 2014.