МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения (ИКБСП)
Кафедра КБ-14 «Технологии искусственного интеллекта в безопасности»

Направление подготовки

09.03.02 Информационные системы и технологии

Работу выполнил:

Студент 2 курса группы БСБО-06-20

Халиков А.А.

Научный руководитель:

Ксенофонтов Н.В.

Оглавление

1.1 Задание 1

Написать программу, в которой обрабатываются личные данные сотрудников некой компании. Сотрудники получают зарплату в конце месяца с учётом выполненных заданий. Каждое задание занимает N часов работы. Соответственно гонорар за выполненный таск рассчитывается с учётом занимаемых часов. (формулу можете придумать сами).

Программа должна содержать следующие функции:

1. Ввод нового сотрудника вручную;
2. Добавление сотрудника из случайных данных:
- Отдельные три массива имён, фамилий и отчества откуда формируется ФИО;
- Дата рождения;
- Массив должностей (без генерального директора и т.п);
- Город проживания;
- Список заданий (тоже формируются случайным образом).
3. Добавление заданий;
4. Просмотр списка выполненных и открытых заданий;
5. Вывод топ-3 сотрудников по выполнению заданий в месяц;
6. Вывод задания с наибольшим гонораром за выполнение;
7. Формирование отчёта о выполненных заданиях за месяц и сохранение его в текстовый файл;
8. Сохранение списка сотрудников в файл;

● Желательно, чтобы текст не был однотонным (но в меру! “Вырвиглазное” оформление считается за невыполненную работу);

Для ясности, каждый сотрудник, помимо личной информации, содержит список задач (объектов класса Task), а его зарплата за месяц вычисляется из данных по выполненным задачам из этого списка. Сама задача состоит из полей: название, кол-во часов для выполнения, суть, гонорар за выполнение, статус (закрыта/открыта), кому назначена (объект класса Employer).

Общие требования для всех вариантов

Автологирование

В начале работы программы запускается отдельный поток для автоматической записи всех действий в программе с помощью классов FileWriter и BufferedWriter в текстовый файл, пример записи действия в файле:
[“НАЗВАНИЕ_КЛАССА”][“НАЗВАНИЕ_МЕТОДА”] Пользователь ввёл следующие данные:
[“НАЗВАНИЕ_КЛАССА”][“НАЗВАНИЕ_МЕТОДА”] Произошла ошибка, см. описание: (далее вывод stacktrace из блока catch в обработке исключений)
Очевидно, что логировать каждое действие пользователя не нужно.
Поток засыпает на 5 секунд и после чего производит перезапись файла новыми

данными.

Использование Optional<T> вместо обработки NullPointerException

При обработки функций, которые потенциально могут вернуть null (например, поиск конкретного объекта в массиве), получать данные через метод .get(), либо использовать .ifPresentOrElse() если требуется провести вывод/обработку.
Использование try-catch-finally для обработки NPE – запрещено

Выполнение задания 1

1. Для начала создаем класс работника (в нем основные поля характерезующие данный класс и два конструктора для ручного создания и рандомной генерации сотрудника)

   /Класс Employer/

```
package com.company;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.Random;

import static com.company.Data.*;

/*Сотрудник
* имён, фамилий и отчества;
○ Дата рождения;
○ Должность (без генерального директора и т.п);
○ Город проживания;
○ Список заданий (тоже формируются случайным образом).
* */
public class Employer {
    private String Name;
    private String Surname;
    private String secondName;
    private String dateOfBirth;
    private String Position;
    private String City;
    private List<Task> Tasks = new ArrayList<>();

    //Конструктор для ввода вручную
    public Employer(String name, String surname, String secondName, String
dateOfBirth, String position, String city) {
        Name = name;
        Surname = surname;
        this.secondName = secondName;
        this.dateOfBirth = dateOfBirth;
        Position = position;
        City = city;
        //Objects.requireNonNull(Tasks).add(tasks);
    }

    public Employer()
    {
        Name = manNames[new Random().nextInt(manNames.length)];
        Surname = Surnames[new Random().nextInt(Surnames.length)];
        secondName = secondNames[new Random().nextInt(secondNames.length)];
        dateOfBirth = String.valueOf(((int) (Math.random() * 30) + 1))
                + "." + ((int) (Math.random() * 12) + 1)
                + "." + (rnd(1980,2002));
        Position = jobs[new Random().nextInt(jobs.length)];
```

```
        City = Cities[new Random().nextInt(Cities.length)];
        new Task(this);
    }

    public String getName() {
        return Name;
    }

    public void setName(String name) {
        Name = name;
    }

    public String getSurname() {
        return Surname;
    }

    public void setSurname(String surname) {
        Surname = surname;
    }

    public String getSecondName() {
        return secondName;
    }

    public void setSecondName(String secondName) {
        this.secondName = secondName;
    }

    public String getDateOfBirth() {
        return dateOfBirth;
    }

    public void setDateOfBirth(String dateOfBirth) {
        this.dateOfBirth = dateOfBirth;
    }

    public String getPosition() {
        return Position;
    }

    public void setPosition(String position) {
        Position = position;
    }

    public String getCity() {
        return City;
    }

    public void setCity(String city) {
        City = city;
    }

    public List<Task> getTasks() {
        return Tasks;
    }

    public void setTasks(List<Task> tasks) {
        Tasks = tasks;
    }

    public int getSizeOfDoneTasks()
    {
        int i=0;
        for (Task t:Tasks) {
            if (!t.isStatus()) {
                i+=1;
            }
```

```
        }
        return i;
    }

    public void setTask(Task task) {
        Tasks.add(task);
    }

    public static int rnd(int min, int max)
    {
        max -= min;
        return (int) (Math.random() * ++max) + min;
    }

    public void printInfo()
    {
        System.out.println(Name + "\t" + secondName+ "\t" + Surname + "\t\t\t" +
Position);
    }

    public void printInfoWithTasks()
    {
        System.out.println(Name + "\t" + secondName+ "\t" + Surname + "\t\t\t" +
Position + "\t\t\tКоличесвто выполненныйх заданий: " + getSizeOfDoneTasks());
    }

    public String getAllInfo()
    {
        return Name + "\t" + secondName+ "\t" + Surname + "\tДата рождения: " +
dateOfBirth + "\tГород:"+City+"\tДолжность: " + Position + "\tВыполненные задания: "
+ getSizeOfDoneTasks() ;
    }

}
```

2. Откуда объект класса Employer будет рандомно брать информацию? Для этого создаем класс-хранилку в которм будут статические массивы данных.

/Класс Data/

```
package com.company;

public class Data {
    public static String[] Cities =
            {
                    "Москва",
                    "Санкт-Петербург",
                    "Кастрома",
                    "Ростов-На-Дону",
                    "Тула"
            };

    public static String[] jobs =
            {
                    "Разработчик сайта           ",
                    "Работник серверной          ",
                    "Информационная безопасность",
                    "Менеджер                    ",
                    "Менеджер по маркетингу      ",
                    "Охранник                    ",
                    "Секретарь                   ",
                    "Креативный директор         ",
                    "Уборщик                     ",
                    "Дизайнер                    "
            };
    public static String[] tasksForServer =
            {
```

```
                "Настроить коммутатор для отдела менеджеров                    ",
                "Настроить роутер                                              ",
                "Поставить новые коммутаторы                                   ",
                "Дать доступ директору до компьютеров менеджеров               "
        };
public static String[] tasksForInfobez =
        {
                "IDC/IPC система                                               ",
                "Просмотреть логи сети                                         ",
                "Провести встречу с работником серверной                       ",
                "Траблшутинг                                                   "
        };
public static String[] tasksForManager =
        {       "Определение перспективных и текущих целей и задач             ",
                "Проведение собраний совещаний                                 ",
                "Работа с документами                                          ",
                "Контроль, оценка и анализ деятельност группы в целом          "
        };
public static String[] tasksForMarketingManager =
        {
                "Изучение рынка и рыночных тенденций                           ",
                "Выбор целевого рынка                                          ",
                "Утверждение стратегии развития товара                         ",
                "Управление отношениями с клиентами                            "
        };
public static String[] taskOhrannik =
        {
                "Охранять                                                    "
        };
public static String[] tasksForSekretar =
        {
                "Участие в подготовке и проведении совещаний                   ",
                "Организация связи руководителя                                ",
                "Организация рабочего времени                                  ",
                "Поддержание деловых контактов                                 "
        };
public static String[] tasksForCreative =
        {
                "переговоры с ключевыми клиентами                              ",
                "презентация услуг компании и защита проектов перед заказчиками",
                "составление коммерческих предложений для бизнес-партнеров     ",
                "разработка концепций рекламных кампаний                       ",
                "подготовка и проведение презентаций, пресс-конференций        "
        };
public static String[] tasksForCleaner =
        {
                "Убрать зал                                                    ",
                "Убрать туалеты                                                ",
                "Убрать коридор                                                "
        };
public static String[] tasksForDesigner =
        {
                "Нарисовать новый логотип                                      ",
                "Нарисовать главный экран сайта                                ",
                "Нарисовать страницку товара                                   ",
                "Провести встречу с разработчиком сайта                        ",
                "Нарисовать дизайн мобильной версии сайта                      "
        };


public static String[] tasksForWeb =
        {
                "Сверстать страничку товара                                    ",
                "Выполнить работу по подготовке программ к отладке             ",
                "Сверстать раздел 'О нас'                                      ",
                "Провести совещание                                            ",
                "Начать верстку мобильной версии сайта                         "
```

```
        };

    public static String[] manNames =
        {
                "Александр ",
                "Алексей   ",
                "Иван      ",
                "Григорий  ",
                "Константин",
                "Илья      ",
                "Дмитрий   "
        };

    public static String[] secondNames =
        {
                "Александрович   ",
                "Алексеевич      ",
                "Иванович        ",
                "Григорьевич     ",
                "Константинович  ",
                "Николаевич      ",
                "Михайлович      "
        };

    public static String[] Surnames =
        {
                "Тарасов  ",
                "Жуков    ",
                "Баранов  ",
                "Филиппов ",
                "Комаров  ",
                "Давыдов  ",
                "Беляев   ",
                "Герасимов",
                "Богданов ",
                "Осипов   ",
                "Сидоров  ",
                "Матвеев  ",
                "Титов    ",
                "Марков   ",
                "Миронов  "
        };

}
```

3. Далее создаем класс в котором описываем задание для сотрудника. В нем имеется два конструктора для ручного и рандомного заполнения объекта.

/Класс Task/

```
package com.company;

import java.util.Random;

import static com.company.Data.jobs;
import static com.company.Data.*;
import static com.company.View.tasks;

/*
название, кол-во
часов для выполнения, суть, гонорар за выполнение, статус (закрыта/открыта), кому
назначена (объект класса Employer)
 */
public class Task {
    private String Name;
```

```
    private int Hours;
    private int addOn;
    private boolean Status;
    private Employer employer;

    public Task(String name, int hours, int addOn, boolean status, Employer
employer) {
        Name = name;
        Hours = hours;
        this.addOn = addOn;
        Status = status;
        this.employer = employer;
        employer.setTask(this); //записывает только что созданное задание сотруднику
передонному в конструктор
    }

    public Task(Employer employer)
    {
        if(employer.getPosition().equals("Разработчик сайта           "))
        {
            Name = tasksForWeb[new Random().nextInt(tasksForWeb.length)];
        }
        if(employer.getPosition().equals("Работник серверной          "))
        {
            Name = tasksForServer[new Random().nextInt(tasksForServer.length)];
        }
        if(employer.getPosition().equals("Информационная безопасность"))
        {
            Name = tasksForInfobez[new Random().nextInt(tasksForInfobez.length)];
        }
        if(employer.getPosition().equals("Менеджер                    "))
        {
            Name = tasksForManager[new Random().nextInt(tasksForManager.length)];
        }
        if(employer.getPosition().equals("Менеджер по маркетингу      "))
        {
            Name = tasksForMarketingManager[new
Random().nextInt(tasksForMarketingManager.length)];
        }
        if(employer.getPosition().equals("Охранник                    "))
        {
            Name = taskOhrannik[new Random().nextInt(taskOhrannik.length)];
        }
        if(employer.getPosition().equals("Секретарь                   "))
        {
            Name = tasksForSekretar[new Random().nextInt(tasksForSekretar.length)];
        }
        if(employer.getPosition().equals("Креативный директор         "))
        {
            Name = tasksForCreative[new Random().nextInt(tasksForCreative.length)];
        }
        if(employer.getPosition().equals("Уборщик                     "))
        {
            Name = tasksForCleaner[new Random().nextInt(tasksForCleaner.length)];
        }
        if(employer.getPosition().equals("Дизайнер                    "))
        {
            Name = tasksForDesigner[new Random().nextInt(tasksForDesigner.length)];
        }
        Hours = new Random().nextInt(30+1);
        this.addOn = Employer.rnd(1000,10000);
        Status = new Random().nextBoolean();
        this.employer = employer;
        employer.setTask(this);
        tasks.add(this);
    }
    public boolean isStatus() {
```

```
        return Status;
    }

    public void setStatus(boolean status) {
        Status = status;
    }

    public int getAddOn() {
        return addOn;
    }

    public void setAddOn(int addOn) {
        this.addOn = addOn;
    }

    public int getHours() {
        return Hours;
    }

    public void setHours(int hours) {
        Hours = hours;
    }

    public String getName() {
        return Name;
    }

    public void setName(String name) {
        Name = name;
    }

    public Employer getEmployer() {
        return employer;
    }

    public void setEmployer(Employer employer) {
        this.employer = employer;
    }

    public void printTask()
    {
        System.out.println(Name+ "\t|"+ Hours  + " ч.|\t\t\t|\t  "
                + addOn +" Руб.\t\t  |  \t\t" + Status + "\t\t|\t\t"
                + employer.getName() + " "+employer.getSurname() +"|");
    }
}
```

4. Создаем один из главных классов – **класс для взаимодействия с пользователем View**. В нем описываем методы для генерации текста и предоставления возможности ввода для пользователя. Все методы закольцовываем и делаем луп для возвращения в клавное меню, когда метод заканчивает работу.

   /Класс View/

```
package com.company;

import java.awt.image.AreaAveragingScaleFilter;
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;
import java.util.stream.Stream;
```

```
import static com.company.Report.writeInfo;

public class View {
    public String helloText = "\n\n" + "Выберете, пожалуйста что хотите сделать
(Введите цифру от 1 до 8): ";
    public String Menu = "\t1. Добавить нового сотрудника. \n " +
                         "\t2. Сгенерировать нового сотрудника \n" +
                         "\t3. Добавить задание. \n" +
                         "\t4. Просмотр списка выполненных и открытых заданий. \n" +
                         "\t5. Вывод топ-3 сотрудников по выполнению заданий в
месяц. \n" +
                         "\t6. Вывод задания с наибольшим гонораром за выполнение.
\n" +
                         "\t7. Сформировать отчёт о выполненных заданиях за месяц и
сохраненить его в текстовый файл.\n" +
                         "\t8. Сохраненить список сотрудников в файл. \n" +
                         "\nВведите цифру: ";




    public static ArrayList<Employer> employers = new ArrayList<>(); //Лист с
работниками
    public static ArrayList<Task> tasks = new ArrayList<>(); // Лист с тасками




    public void showEmployers() //вспомогательный метод для вывода сотрудников с их
номерами
    {
        System.out.println("\n\n№"+"    Имя" + "         Отчество    " + "
Фамилия      " + "         Должность  ");
        int i = 0;
        for (Employer e: employers) {
            System.out.print(i+ ". ");
            e.printInfo();
            i++;
        }
    }



    public void addEmployer()//метод для ручного добавления сотрудника
    {
        Scanner employerScanner = new Scanner(System.in);
        System.out.print("Введите имя сотрудника: " );
        String name = employerScanner.nextLine();
        System.out.print("Введите фамилию сотрудника: " );
        String Surname = employerScanner.nextLine();
        System.out.print("Введите отчество сотрудника: " );
        String secondName = employerScanner.nextLine();
        System.out.print("Введите дату рождения сотрудника(в формате 01.01.1999): "
);
        String dateOfBirth = employerScanner.nextLine();
        System.out.print("Введите должность сотрудника: " );
        String Position = employerScanner.nextLine();
        System.out.print("Введите город проживания сотрудника: " );
        String City = employerScanner.nextLine();

        Employer newEmployer = new
Employer(name,Surname,secondName,dateOfBirth,Position,City);
        employers.add(newEmployer);
```

```
        System.out.println("Сотрудник добавлен! Нажмите 1 для выхода в меню или 2
для завершения работы программы: ");
        writeInfo("["+this.getClass().getSimpleName()+"][“addEmployer”] Пользователь
ввёл" +
                "следующие данные:\n" + name + " "+Surname+ " "+secondName+"
"+dateOfBirth+ " "+Position+" "+ City+ "\n\n\n");
        Scanner sc = new Scanner(System.in);
        int answer = sc.nextInt();
        if(answer==1) getMenu();
        else System.exit(0);
    }



    private void generateEmloyer()//метод для рандомной генерации сотрудника (просто
вызываем пустой конструктор, выше описано как он работает)
    {
        Employer newEmployer = new Employer();
        employers.add(newEmployer);
        System.out.print("Сотрудник сгенерирован и добавлен! \nНажмите 1 для выхода
в меню или 2 для завершения работы программы: ");
       Scanner sc = new Scanner(System.in);
        int answer = sc.nextInt();
        if(answer==1) getMenu();
        else System.exit(0);
    }



    private void addTask()//метод для добавления заданий
    {
        Scanner employerScanner = new Scanner(System.in);
        System.out.print("Введите название задания: " );
        String name = employerScanner.nextLine();
        System.out.print("Введите количество часов на выполнение этого задания: " );
        int hours = employerScanner.nextInt();
        System.out.print("Введите сколько будет надбавка за выполненное задание: "
);
        int addOn = employerScanner.nextInt();
        System.out.print("Введите открыто ли это задание (false) или уже выполнено
(true): " );
        boolean status = employerScanner.nextBoolean();
        showEmployers();
        System.out.print("Введите какому сотруднику пренадлежит задание (введите его
номер): " );
        int addTaskToEmployer = employerScanner.nextInt();

        Task task = new Task(name,hours,addOn,status,
employers.get(addTaskToEmployer));
        tasks.add(task);

        writeInfo("["+this.getClass().getSimpleName()+"][“addTask”] Пользователь
ввёл " +
                "следующие данные:\n" + name  + " "+hours+ " "+addOn+" "+status+"
"+addTaskToEmployer + "\n\n\n");
        System.out.print("Задание добавлено! \nНажмите 1 для выхода в меню или 2 для
завершения работы программы: ");
        Scanner sc = new Scanner(System.in);
        int answer = sc.nextInt();

        if(answer==1) getMenu();
        else System.exit(0);
    }
```

```
    public void getTasks() //Просмотр списка выполненных и открытых заданий;
    {
        System.out.println("\t\t\t\tНазвание задания\t\t\t\t"+"\t\tСколько занимает
часов\t"
                +"Надбавка за выполнение\t"+"Открыто ли задание\t\t\t" + "   Кто
делает");
        Stream<Task> streamTasks = tasks.stream();
        streamTasks.forEach(Task::printTask);
        System.out.print("\nНажмите 1(для выхода в меню) или 2(для завершения работы
программы): ");
        Scanner sc = new Scanner(System.in);
        int answer = sc.nextInt();

        if(answer==1) getMenu();
        else System.exit(0);
    }



    public void getWorkersTop()//Вывод топ-3 сотрудников по выполнению заданий в
месяц
    {
        Stream<Employer> employerStream = employers.stream();

employerStream.sorted(Comparator.comparing(Employer::getSizeOfDoneTasks).reversed())
.limit(3).forEach(Employer::printInfoWithTasks);
        System.out.print("\nНажмите 1(для выхода в меню) или 2(для завершения работы
программы): ");
        Scanner sc = new Scanner(System.in);
        int answer = sc.nextInt();

        if(answer==1) getMenu();
        else System.exit(0);
    }



    public void getTopTask()//Вывод задания с наибольшим гонораром за выполнение;
    {
        System.out.println("\t\t\t\tНазвание задания\t\t\t\t"+"\t\tСколько занимает
часов\t"
                +"Надбавка за выполнение\t"+"Открыто ли задание\t\t\t" + "   Кто
делает");
        Stream<Task> streamTasks = tasks.stream();

streamTasks.sorted(Comparator.comparing(Task::getAddOn).reversed()).findFirst().get(
).printTask();
        System.out.print("\nНажмите 1(для выхода в меню) или 2(для завершения работы
программы): ");
        Scanner sc = new Scanner(System.in);
        int answer = sc.nextInt();

        if(answer==1) getMenu();
        else System.exit(0);
    }



    public void doneTasksToFile() //метод для Формирование отчёта о выполненных
заданиях за месяц и сохранение его в текстовый файл
    {
```

```
        try {
            OutputStream tasksFile = new FileOutputStream("doneTasks.txt");
            OutputStreamWriter donetasks = new OutputStreamWriter(tasksFile,
StandardCharsets.UTF_8);
            System.out.println("\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\tВЫПОЛНЕННЫЕ ЗАДАНИЯ
(ОТЧЕТ)");
            System.out.println("\t\t\t\tНазвание задания\t\t\t\t"+"\t\tСколько
заняло часов\t"
                    +"Надбавка за выполнение\t"+"Открыто ли задание\t\t\t" + "   Кто
сделал");
            for (Task t : tasks) {
                if(!t.isStatus())
                {
                    t.printTask();

writeInfo("["+this.getClass().getSimpleName()+"][“doneTasksToFile()”] Пользователь
получил " +
                            "следующие данные:\n" + t.getName() + " "+t.isStatus()+"
"+t.getAddOn()+ " "+t.getEmployer().getAllInfo().strip() + "\n\n\n");
                    donetasks.append(t.getName()).append("\n");
                }
            }
            donetasks.close();

            System.out.print("\u001b[38;5;10mВсе выполненные задания успешно
занесены в файл!\u001b[38;5;0m\nНажмите 1(для выхода в меню) или 2(для завершения
работы программы):");
            Scanner sc = new Scanner(System.in);
            int answer = sc.nextInt();

            if(answer==1) getMenu();
            else System.exit(0);
        } catch (IOException e) {
            System.out.println("\u001b[38;5;196mОшибка при добавлении задания в
файл! Задание НЕ были добавлены\u001b[38;5;0m");
            writeInfo("[“View”][“doneTasksToFile()”] Произошла ошибка, см.\n" +
                    "описание:"+ Arrays.toString(e.getStackTrace()) + "\n\n\n");
            e.printStackTrace();
        }

    }



    public void addWorkersToFile() { // Сохранение списка сотрудников в файл;
        try {
            OutputStream workersFile = new FileOutputStream("workersList.txt");
            OutputStreamWriter workersList = new OutputStreamWriter(workersFile,
StandardCharsets.UTF_8);
            for (Employer e:employers) {

writeInfo("["+this.getClass().getSimpleName()+"][“addWorkersToFile()”] Пользователь
получил" +
                        "следующие данные:\n" +e.getAllInfo() + "\n\n\n");
                workersList.append(e.getAllInfo()).append("\n");
            }
            workersList.close();
            writeInfo("["+this.getClass().getSimpleName()+"][“addWorkersToFile()”]
Пользователь получил" +
                    "следующие данные:\n"+"\u001b[38;5;10mРаботники добавлены в
файл!\u001b[38;5;0m\nНажмите 1(для выхода в меню) или 2(для завершения работы
программы): ");
            System.out.print("\u001b[38;5;10mРаботники добавлены в
файл!\u001b[38;5;0m\nНажмите 1(для выхода в меню) или 2(для завершения работы
программы): ");
            Scanner sc = new Scanner(System.in);
```

```
                int answer = sc.nextInt();

                if(answer==1) getMenu();
                else System.exit(0);
            } catch (IOException e) {
                writeInfo("["View"]["addWorkersToFile()"] Произошла ошибка, см.\n" +
                        "описание: "+ Arrays.toString(e.getStackTrace()) + "\n\n\n");
                System.out.println("\u001b[38;5;196mРаботники не были добавлены в файл!
    Произошла ошибка\u001b[38;5;0m");
                e.printStackTrace();
            }

        }




            public void getMenu() // метод для вывода главного меню
        {
            System.out.println(helloText + "\n\n" + Menu);
            Scanner console = new Scanner(System.in);
            int Variants = console.nextInt();
            switch(Variants) {
                case 1: addEmployer();
                case 2: generateEmloyer();
                case 3: addTask();
                case 4: getTasks();
                case 5: getWorkersTop();
                case 6: getTopTask();
                case 7: doneTasksToFile();
                case 8: addWorkersToFile();
            }

        }
    }
```

5. Автологирование

В начале работы программы запускается отдельный поток для автоматической записи всех действий в программе с помощью классов FileWriter и BufferedWriter в текстовый файл. Класс `Report` для хранения ArrayList'а с историей логов действий пользователя. Класс `AutoLogSaves` для автоматического сохранения в файл логов пользователя каждые 5 секунд.

/Класс Report/

```
package com.company;

import java.util.ArrayList;

public class Report {
    private static ArrayList<String> logHistory = new ArrayList<>(); //храним записи

    public static void writeInfo(String message) {
        logHistory.add(message);
    }

    public static ArrayList<String> getLogHistory() {
        return logHistory;

    }
}
```

/Класс AutoLogSaves/

```
package com.company;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.concurrent.TimeUnit;

public class AutoLogSaves extends Thread{
    @Override
    public void run() {
        while (!Thread.currentThread().isInterrupted()) {
            try (FileWriter logWriter = new FileWriter("log.txt");
                 BufferedWriter bufferedWriter = new BufferedWriter(logWriter)) {
                ArrayList<String> logBuffer = Report.getLogHistory();
                for (String s : logBuffer) {
                    s += "\n";
                    bufferedWriter.write(s);
                }
                bufferedWriter.flush();
                TimeUnit.SECONDS.sleep(5);
            } catch (IOException | InterruptedException e) {
                e.printStackTrace();
            }
        }

    }

}
```

6. В классе main просто создаем view и вспомогательный поток для логирования.

```
public class Main {

    public static void main(String[] args) {
   // write your code here
        AutoLogSaves LOG_SAVER = new AutoLogSaves();
        LOG_SAVER.start();
        View view = new View();
        view.getMenu();

    }
}
```

Вывод в консоль:

Главное меню при запуске программы:

```
Выберете, пожалуйста что хотите сделать (Введите цифру от 1 до 8):

    1. Добавить нового сотрудника.
    2. Сгенерировать нового сотрудника
    3. Добавить задание.
    4. Просмотр списка выполненных и открытых заданий.
    5. Вывод топ-3 сотрудников по выполнению заданий в месяц.
    6. Вывод задания с наибольшим гонораром за выполнение.
    7. Сформировать отчёт о выполненных заданиях за месяц и сохраненить его в текстовый файл.
    8. Сохраненить список сотрудников в файл.

Введите цифру:
```

Нажимаем 1 *(вводим информацию о новом сотруднике, далее выходим в главное меню нажимая цифру 1)*

```
Введите цифру:
1
Введите имя сотрудника: Иван
Введите фамилию сотрудника: Иванов
Введите отчество сотрудника: Иванович
Введите дату рождения сотрудника(в формате 01.01.1999): 19.03.1998
Введите должность сотрудника: Программист
Введите город проживания сотрудника: Москва
Сотрудник добавлен! Нажмите 1 для выхода в меню или 2 для завершения работы программы:
```

Нажимаем цифру 2

```
Выберете, пожалуйста что хотите сделать (Введите цифру от 1 до 8):

    1. Добавить нового сотрудника.
    2. Сгенерировать нового сотрудника
    3. Добавить задание.
    4. Просмотр списка выполненных и открытых заданий.
    5. Вывод топ-3 сотрудников по выполнению заданий в месяц.
    6. Вывод задания с наибольшим гонораром за выполнение.
    7. Сформировать отчёт о выполненных заданиях за месяц и сохраненить его в текстовый файл.
    8. Сохраненить список сотрудников в файл.

Введите цифру:
2
Сотрудник сгенерирован и добавлен!
Нажмите 1 для выхода в меню или 2 для завершения работы программы:
```

Вводим 3

```
Введите цифру:
3
Введите название задания: Доделать сайт
Введите количество часов на выполнение этого задания: 30
Введите сколько будет надбавка за выполненное задание: 12500
Введите открыто ли это задание (false) или уже выполнено (true): false


№    Имя          Отчество           Фамилия              Должность
0. Иван Иванович    Иванов           Программист
1. Дмитрий      Константинович      Марков               Уборщик
Введите какому сотруднику пренадлежит задание (введите его номер): 0
Задание добавлено!
Нажмите 1 для выхода в меню или 2 для завершения работы программы:
```

Вводим цифру 4

```
Введите цифру:
4
              Название задания                    Сколько занимает часов  Надбавка за выполнение  Открыто ли задание            Кто де
Убрать туалеты                                          |30 ч.|         |     6709 Руб.       |         true        |       Дмитрий
Доделать сайт                                           |30 ч.|         |     12500 Руб.      |         false       |       Иван

Нажмите 1(для выхода в меню) или 2(для завершения работы программы):
```

Вводим цифру 5 (так как мы ввели то что программист уже выполнил задание – у него оно уже есть в списке, у нас всего 2 сотрудника, поэтому в топ 3 выводится топ 2)

```
Введите цифру:
5
Иван    Иванович    Иванов          Программист         Количесвто выполненныйх заданий: 1
Дмитрий     Константинович      Марков              Уборщик                         Количесвто выполненныйх заданий: 0
```

Вводим цифру 6

```
Введите цифру:
6
              Название задания                    Сколько занимает часов  Надбавка за выполнение  Открыто ли задание            Кто дела
Доделать сайт                                           |30 ч.|         |     12500 Руб.          |         false       |       Иван Ива
```

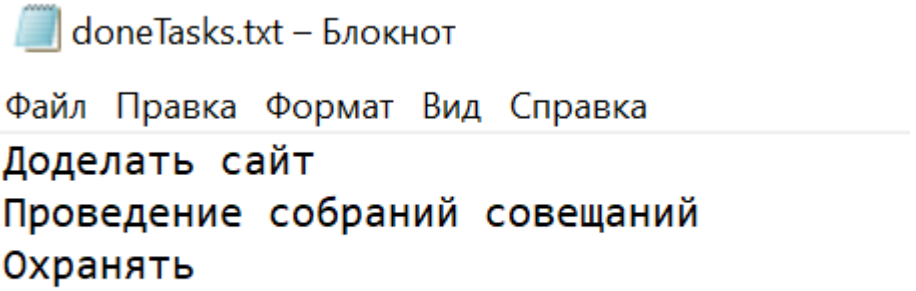
Вводим цифру 7 (до этого я сгенерировал еще 5 сотрудников, чтобы было побольше заданий)

```
Выберете, пожалуйста что хотите сделать (Введите цифру от 1 до 8):

    1. Добавить нового сотрудника.
    2. Сгенерировать нового сотрудника
    3. Добавить задание.
    4. Просмотр списка выполненных и открытых заданий.
    5. Вывод топ-3 сотрудников по выполнению заданий в месяц.
    6. Вывод задания с наибольшим гонораром за выполнение.
    7. Сформировать отчёт о выполненных заданиях за месяц и сохраненить его в текстовый файл.
    8. Сохраненить список сотрудников в файл.

Введите цифру:
7
                                                ВЫПОЛНЕННЫЕ ЗАДАНИЯ (ОТЧЕТ)
            Название задания                  Сколько заняло часов    Надбавка за выполнение  Открыто ли задание            Кто сдела
Доделать сайт                                       |30 ч.|         |     12500 Руб.          |         false       |       Иван Иванс
Проведение собраний совещаний                       |2 ч.|          |     3622 Руб.       |         false       |       Дмитрий    М
Охранять                                            |12 ч.|         |     7208 Руб.       |         false       |       Григорий   Б
Все выполненные задания успешно занесены в файл!
Нажмите 1(для выхода в меню) или 2(для завершения работы программы):
```

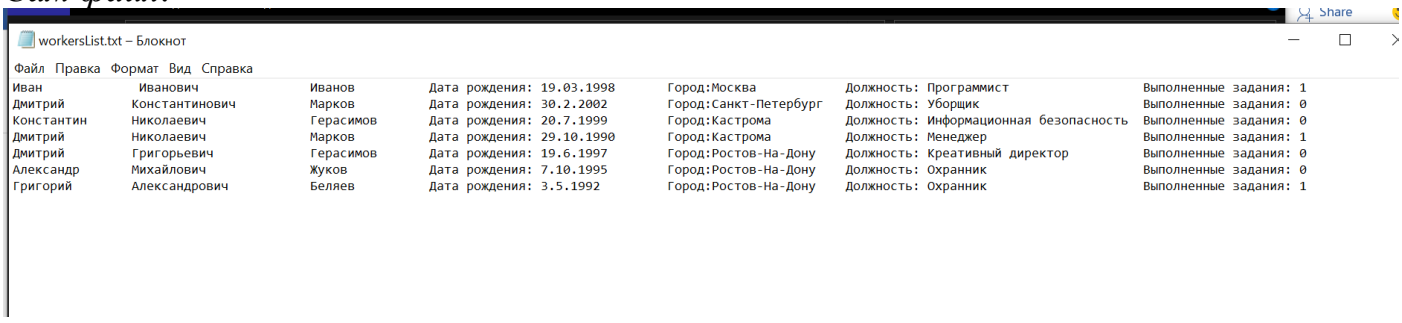
Сгенерированый файл:



Вводим цифру 8

```
Введите цифру:
8
Работники добавлены в файл!
Нажмите 1(для выхода в меню) или 2(для завершения работы программы):
```

Сам файл:



Во время работы программы был сгенерирован файл log.txt и туда записывалась информация в параллельном потоке. Давайте посмотрим что там есть.

/Содержание файла log.txt/

[View][“addEmployer”] Пользователь ввёлследующие данные:
Иван Иванов Иванович 19.03.1998 Программист Москва

[View][“addTask”] Пользователь ввёл следующие данные:
Доделать сайт 30 12500 false 0

[View][“doneTasksToFile()”] Пользователь получил следующие данные:
Доделать сайт false 12500 Иван Иванович Иванов Дата рождения: 19.03.1998 Город:Москва Должность: Программист Выполненные задания: 1

[View][“doneTasksToFile()”] Пользователь получил следующие данные:
Доделать сайт false 12500 Иван Иванович Иванов Дата рождения: 19.03.1998 Город:Москва Должность: Программист Выполненные задания: 1

[View][“doneTasksToFile()”] Пользователь получил следующие данные:
Проведение собраний совещаний false 3622 Дмитрий Николаевич Марков Дата рождения: 29.10.1990 Город:Кастрома Должность: Менеджер Выполненные задания: 1

[View][“doneTasksToFile()”] Пользователь получил следующие данные:
Охранять false 7208 Григорий Александрович Беляев Дата рождения: 3.5.1992 Город:Ростов-На-ДонуДолжность: Охранник Выполненные задания: 1

[View][“addWorkersToFile()”] Пользователь получилследующие данные:
Иван Иванович Иванов Дата рождения: 19.03.1998 Город:Москва Должность: Программист Выполненные задания: 1

[View][“addWorkersToFile()”] Пользователь получилследующие данные:
Дмитрий Константинович Марков Дата рождения: 30.2.2002 Город:Санкт-Петербург Должность: Уборщик Выполненные задания: 0

[View][“addWorkersToFile()”] Пользователь получилследующие данные:
Константин Николаевич Герасимов Дата рождения: 20.7.1999
Город:Кастрома Должность: Информационная безопасность Выполненные задания: 0

[View][“addWorkersToFile()”] Пользователь получилследующие данные:
Дмитрий Николаевич Марков Дата рождения: 29.10.1990
Город:Кастрома Должность: Менеджер Выполненные задания: 1

[View][“addWorkersToFile()”] Пользователь получилследующие данные:
Дмитрий Григорьевич Герасимов Дата рождения: 19.6.1997 Город:Ростов-На-Дону Должность: Креативный директор Выполненные задания: 0

[View][“addWorkersToFile()”] Пользователь получилследующие данные:
Александр Михайлович Жуков Дата рождения: 7.10.1995 Город:Ростов-На-Дону Должность: Охранник Выполненные задания: 0

[View][“addWorkersToFile()”] Пользователь получилследующие данные:
Григорий Александрович Беляев Дата рождения: 3.5.1992 Город:Ростов-На-Дону Должность: Охранник Выполненные задания: 1

[View][“addWorkersToFile()”] Пользователь получилследующие данные:
Работники добавлены в файл!
Нажмите 1(для выхода в меню) или 2(для завершения работы программы):