Державна служба України з надзвичайних ситуацій

Львівський державний університет безпеки життєдіяльності

Кафедра УПІТТ

Курсова робота

з дисципліни: «Об’єктно-орієнтоване програмування»

на тему:

«Базові принципи об’єктно-орієнтованого програмування»

Індивідуальне завдання: «Навчальний заклад»

Виконав:

Студент групи КН-21

Кушка Роман

Прийняв:

Олександр ПРИДАТКО

Львів-2020

АНОТАЦІЯ

Тема моєї курсової роботи – «Навчальний заклад». Перед тим як приступити до її виконання, я ознайомився з діяльністю різних навчальних закладів, розглянув різноманітні інтернет-ресурси(див. Список використаних джерел), після чого продумав план розробки програмного продукту, тобто функціоналу.

Він буде побудований на паттерні «Команда», який дає значні переваги для написання складних систем та подальшого їх вдосконалення.

Програма буде написана мовою програмування JAVA, оскільки вона високошвидкісна, та зручна у використанні.

Курсова робота займає 45 сторінок, з яких код джерела містить 26 сторінок.

ЗМІСТ

ВСТУП

Перед виконанням цієї курсової роботи, постало завдання продумати як організувати таку систему як «Навчальний заклад», оскільки є декілька варіантів її реалізації, таких як формування навчального плану, чи розкладу, облік зарплат викладачам чи персоналу, система оцінювання студентів та ін.. З безлічі можливих для реалізації систем, мною було вирішено розробити систему для опрацювання навчальних груп, студентів, які в них входять, а також викладачів які навчають ці групи.

Для релазації такої системи потрібно продумати ієархію класів освітчан, а також продумати функціонал, який буде виконувати така система. Отже функціонал системи буде такий:

1. Додати студента/викладача з клавіатури (при тому групи формуються автоматично)
2. Записати всіх студентів/викладачів з файлу (при тому всі студенти будуть додані у вказану групу, якщо її не існує, така буде створена, те ж саме з викладачами)
3. Видалити зі списку студента або викладача за його прізвищем (при тому викладач буде видалений з усіх груп, де він записаний, а також буде врахована можливість появи однофамільців)
4. Вивести список студентів або виклдачів певної групи.
5. Сортувати студентів за прізвищем, або датою народження.
6. Фільтрувати студентів за одним з параметрів: лише хлопці, лише дівчата, лише пільговики, або в проміжку від до дати народження.
7. Шукати студента або викладача у всіх списках, і вивести інформацію про нього.

1. РОЗРОБКА ТА ОПИС АЛГОРИТМУ

У вступі була зазначена ідея та підхід до реалізації системи, у цьому розділі буде коротко описано алгоритм роботи програми. Отже, алгоритм виглядає так:

1) Викликати основне меню з вибором виконання функції програми, на вибір користувача.

   Меню виглядає так:

   1. Вивести список груп
   2. Додати освітчанина (студента або викладача) з клавіатури
   3. Прочитати файл з освітчанами.
   4. Видалити освітчанина.
   5. Вивести список студентів або викладачів певної групи.
   6. Сортувати список студентів певної групи
   7. Фільтрувати список студентів певної групи
   8. Шукати студента/викладача.
   9. Вийти з програми.

2) Якщо користувач вводить число 3 йому пропонується ввести шлях до файлу, після чого будуть виконані дії запису/читання даних з файлу.
3) 2 – користувач вводить всі дані вручну з клавіатури
4) 4 , 5, 8 або 9 – виконуються зазначені дії (не потребується додаткового опису)
5) 6 або 7 – буде можливість вибрати параметр сортування (за прізвищами або за датою народження) або параметр фільтру: лише хлопці, лише дівчата, лише пільговики, або вивести студентів з датою народження в заданому діапазоні.
6) Якщо число менше 1, або більше 9 – повернутись до пункту 1.

2. ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ

Main.java

```
/**
 * Курсова робота з ООП. Індивідуальне завдання: "Навчальний заклад"
 * @author Kushka Roman
*/

import controller.Controller;

/**
 * Клас запускає функцію main() і починається виконання програми
 * @see Controller
 */
public class Main {
    public static void main(String[] args) {
        new Controller().userStart();
    }
}
```

Actions.java

```
package actions;

import model.Educator;
import model.Group;
import model.educatorTypes.Student;
import model.educatorTypes.Teacher;
import scanner.Scan;

import java.io.BufferedReader;
import java.io.FileReader;
import java.text.ParseException;
import java.util.ArrayList;


/**
 * Клас для функціонування системи "Навчальний заклад", тут
 * зберігаються усі функції системи, які викликає користувач,
 * за допомогою інтерфейсу Команда
 * @see com.kushka.menu.Command
 * @see Group
 * @see Student
 * @see Teacher
 */
public class Actions {
    ArrayList<Group> groups = new ArrayList<>(); // список груп
    Scan in = new Scan(); // сканер з безпечним вводом даних

    /**
     * вивести список груп
```

```
     * @see Group#toString()
     */
    public void showGroups() {
        if(groups.size() == 0)
        {
            System.out.println("Groups list is empty! U dont add any
student or teacher");
            return;
        }

        System.out.println("List of groups: ");
        for(int i = 0; i < groups.size(); i++)
        {
            System.out.print(i+1+". ");
            System.out.println(groups.get(i));
        }
    }

    /**
     * Додати студента/викладача
     * група створиться автоматично, якщо такої немає в списку
     */
    public void addEducatorWithKeyboard()
    {
        String name, surname, fathername, date;
        Boolean isMale;
        System.out.println("Add: 1 - student || 2 - teacher");
        int choice = in.scanInt();
        System.out.print("Name: ");
        name = in.scanString();
        System.out.print("Surname: ");
        surname = in.scanString();
        System.out.print("Fathername: ");
        fathername = in.scanString();
        System.out.print("Date of birth *format(dd.MM.yyyy): ");
        date = in.scanString();
        System.out.println("Sex: 1 - male, 2 - female");
        int chooseSex = in.scanInt();
        if(chooseSex==1)
        {
            isMale = true;
        }
        else
        {
            isMale = false;
        }

        if (choice == 1) // student
        {
            System.out.print("Group: ");
            String group = in.scanString();
```

```
            System.out.println("Beneficiary: 1 - yes, 2 -no");
            int chooseBeneficiary = in.scanInt();
            Boolean isBeneficiary;
            if(chooseBeneficiary==1)
            {
                isBeneficiary = true;
            }
            else
            {
                isBeneficiary = false;
            }
            Student student = new Student(name, surname, fathername,
date, isMale, group, isBeneficiary);
            Group currentGroup = groupIsExist(group);
            if(currentGroup != null){
                currentGroup.addStudentToGroup(student);
            }
            else {
                currentGroup = createNewGroup(group);
                currentGroup.addStudentToGroup(student);
            }
        }
        else { // teacher
            System.out.print("Profile subject: ");
            String profileSubject = in.scanString();
            System.out.println("Enter list of teacher group, submit it
with 'Enter', to stop - submit empty string:");
            String group = "none";
            ArrayList<String> teacherGroups= new ArrayList<>();
            Teacher teacher = new Teacher();
            teacher.setName(name);
            teacher.setProfileSubject(profileSubject);
            teacher.setSurName(surname);
            teacher.setFatherName(fathername);
            teacher.setDateOfBirth(date);
            teacher.setMale(isMale);

            while (!group.equals("\0")){
                group = in.scanString();
                teacherGroups.add(group);

                Group currentGroup = groupIsExist(group);
                if(currentGroup != null){
                    currentGroup.addTeacherToGroup(teacher);
                }
                else {
                    currentGroup = createNewGroup(group);
                    currentGroup.addTeacherToGroup(teacher);
                }
            }
            teacher.setGroups(teacherGroups);
```

```
132         }
133 
134         System.out.println("One more? 1 - Yes. 2 - No.");
135         choice = in.scanInt();
136         while(choice < 1 || choice > 2)
137         {
138             System.out.println("Retry: ");
139             choice = in.scanInt();
140         }
141         if(choice == 1)
142             addEducatorWithKeyboard();
143         else return;
144     }
145 
146     /**
147      * Перевірити чи така група існує за назвою
148      * @param name назва групи
149      */
150     private Group groupIsExist(String name)
151     {
152         if (groups.size() == 0){
153             return null;
154         }
155 
156         for (Group group : groups)
157         {
158             if(name.equals(group.getName()))
159             {
160                 return group;
161             }
162         }
163         return null;
164     }
165 
166     /**
167      * Створити групу за назвою
168      * @param name назва групи
169      */
170     private Group createNewGroup(String name)
171     {
172         Group group = new Group();
173         group.setName(name);
174         groups.add(group);
175         return group;
176     }
177 
178     /**
179      * Додати студентів|викладачів з файлу
180      * групи сформуються автоматично
181      */
182     public void addEducatorFromFile()
```

```
    {
        try {
            System.out.println("File: ");
            String file = in.scanString();
            FileReader reader = new FileReader(file);
            System.out.println("Reading data from file...");
            BufferedReader bf = new BufferedReader(reader);
            String line;
            while(( line =  bf.readLine())!= null) {
                String typeOfEducator = line;
                String name, surname, fathername, date;
                Boolean isMale;
                name = bf.readLine();
                surname = bf.readLine();
                fathername = bf.readLine();
                date = bf.readLine();
                String chooseSex = bf.readLine();
                if(chooseSex.equals("чол"))
                {
                    isMale = true;
                }
                else
                {
                    isMale = false;
                }

                if (typeOfEducator.equals("студент")) // student
                {
                    String group = bf.readLine();
                    String chooseBeneficiary = bf.readLine();
                    Boolean isBeneficiary;
                    if(chooseBeneficiary.equals("пільговик"))
                    {
                        isBeneficiary = true;
                    }
                    else
                    {
                        isBeneficiary = false;
                    }
                    Student student = new Student(name, surname,
fathername, date, isMale, group, isBeneficiary);
                    Group currentGroup = groupIsExist(group);
                    if(currentGroup != null){
                        currentGroup.addStudentToGroup(student);
                    }
                    else {
                        currentGroup = createNewGroup(group);
                        currentGroup.addStudentToGroup(student);
                    }
                }
                else { // teacher
```

```
                    String profileSubject = bf.readLine();
                   // System.out.println("Enter list of teacher group,
submit it with 'Enter', to stop - submit empty string:");

                    ArrayList<String> teacherGroups = new ArrayList<>();
                    Teacher teacher = new Teacher();
                    teacher.setName(name);
                    teacher.setProfileSubject(profileSubject);
                    teacher.setSurName(surname);
                    teacher.setFatherName(fathername);
                    teacher.setDateOfBirth(date);
                    teacher.setMale(isMale);
                    String group = bf.readLine();
                    while (!group.equals("!")){
                        teacherGroups.add(group);

                        Group currentGroup = groupIsExist(group);
                        if(currentGroup != null){
                            currentGroup.addTeacherToGroup(teacher);
                        }
                        else {
                            currentGroup = createNewGroup(group);
                            currentGroup.addTeacherToGroup(teacher);
                        }
                        group = bf.readLine();
                    }
                    teacher.setGroups(teacherGroups);
                }
            }
            bf.close();
        }catch (Exception e){
            System.out.println("Critical error. Exit program");
            System.out.println(e);
            System.exit(-1);
        }
    }

    /**
     * Видалити студента/викладача за прізвищем
     */
    public void deleteEducator()
    {
        System.out.print("Enter surname: ");
        String surname = in.scanString();
        ArrayList<Educator> educators =
searchEducatorThroughout(surname);
        int choice = 0;
        if (educators == null)
        {
            System.out.println("cant find this educator!");
            return;
```

```
        }
        if (educators.size() > 1)
        {
            for (int i =0; i < educators.size(); i++)
            {
                System.out.println("Matches: ");
                System.out.print(i+1 + ". ");
                System.out.println(educators.get(i));
            }

            System.out.println("Who are u looking for? Enter number from
list");
            choice = in.scanInt() - 1;
        }

        if (educators.get(choice).getClass().equals(Student.class))
        {
            Student stud = (Student)educators.get(choice);
            Group gr = groupIsExist(stud.getGroup());
            for(int i = 0; i < gr.getStudents().size(); i++)
            {
                if (gr.getStudents().get(i) == stud)
                {
                    gr.getStudents().remove(i);
                }
            }
        }

        if (educators.get(choice).getClass().equals(Teacher.class))
        {
            Teacher teacher = (Teacher)educators.get(choice);
            ArrayList<String> groups = teacher.getGroups();
            for (String group : groups)
            {
                Group gr = groupIsExist(group);
                for(int i = 0; i < gr.getAmountTeachers(); i++)
                {
                    if (gr.getTeachers().get(i) == teacher)
                    {
                        gr.getTeachers().remove(i);
                    }
                }
            }
        }
        System.out.println("Educator was deleted success.");
    }

    /**
     * Вивести список студентів/викладачів певної групи
     */
    public void showListOfEducators()
```

```
    {
        Group gr = chooseGroup();
        System.out.println("1-show student, 2-show teachers, 3-show
all");
        System.out.print("Choose: ");
        int choice = in.scanInt();
        while (choice > 3 || choice < 1)
        {
            System.out.print("Retry: ");
            choice = in.scanInt();
        }
        if (choice==1)
        {
            gr.showStudentsList();
        }
        else if (choice==2)
            gr.showTeachersList();
        else {
            gr.showStudentsList();
            gr.showTeachersList();
        }
    }

    /**
     * Вибрати групу зі списку
     * @return об'єкт класу група - вибрана користувачем група
     */
    private Group chooseGroup()
    {
        System.out.println("1-Put name of group; 2-Choose from list of
groups");
        System.out.print("Your choice: ");
        int choice = in.scanInt();
        while (choice > 2 || choice < 1)
        {
            System.out.print("Retry: ");
            choice = in.scanInt();
        }
        if (choice == 1)
        {
            System.out.print("Group: ");
            String group = in.scanString();
            Group gr = groupIsExist(group);
            while(gr == null)
            {
                System.out.print("error. Try again: ");
                group = in.scanString();
                gr = groupIsExist(group);
            }
            return gr;
        }
```

```
        else{
            showGroups();
            System.out.print("Your choose: ");
            int select = in.scanInt() - 1;
            while(select > groups.size() || select < 0)
            {
                System.out.print("Try again. Your choose: ");
                select = in.scanInt();
            }
            return groups.get(select);
        }
    }

    /**
     * Сортувати список студентів певної групи за одним з параметрів: за
прізвищем, за датою народження
     */
    public void sortStudents()
    {
        Group gr = chooseGroup();

        if(gr.getAmountStudents() == 0)
        {
            System.out.println("students list is empty. nothing to
sort!");
            return;
        }

        System.out.println("Choose sort param: 1 - sort by surname ");
        System.out.println("                   2 - sort by date of
birth");
        int choice = in.scanInt();
        while(choice < 1 || choice > 2)
        {
            System.out.print("Retry: ");
            choice = in.scanInt();
        }

        if(choice == 1)
            gr.sortStudentsForSurname();
        else gr.sortStudentsForDateOfBirth();
        System.out.println("Students was sorted success!");
    }

    /**
     * Фільтрувати студентів певної групи за одним з параметрів: показати
лише хлопців/дівчат,
     * показати лише пільговиків
     * показати студентів с певною датою народження: від до.
     */
    public void filterStudents()
```

```
    {
        Group gr = chooseGroup();

        if(gr.getAmountStudents() == 0)
        {
            System.out.println("students list is empty. nothing to
filter!");
            return;
        }

        System.out.println("Choose filter param: 1 - show only boys ");
        System.out.println("                     2 - show only girls");
        System.out.println("                     3 - show only
beneficiary");
        System.out.println("                     4 - filter by date of
birth (from to)");
        int choice = in.scanInt();
        while(choice < 1 || choice > 4)
        {
            System.out.print("Retry: ");
            choice = in.scanInt();
        }

        if(choice == 1)
            gr.showOnlyMale();
        else if(choice==2)
        {
            gr.showOnlyFemale();
        }
        else if(choice==3)
        {
            gr.showOnlyBeneficiary();
        }
        else {
            try {
                gr.filterByDateOfBirth();
            }
            catch (ParseException e)
            {
                System.out.println(e);
                System.out.println("Critical error. Exit programm...");
                System.exit(-1);
            }
        }
        System.out.println("Students was filtered success!");
    }

    /**
     * Шукати студента/викладача за прізвищем
     */
    public void searchForEducator()
```

```
    {
        if(groups.size() == 0)
        {
            System.out.println("Groups list is empty! U dont add any
student or teacher");
            return;
        }
        System.out.print("Enter surname: ");
        String surname = in.scanString();
        ArrayList<Educator> educators =
searchEducatorThroughout(surname);
        if(educators == null)
        {
            System.out.println("No matches.");
        }
        else if(educators.size() == 1)
        {
            System.out.println(educators.get(0));
        }
        else {
            System.out.println("Matches: ");
            for (int i = 0; i < educators.size(); i++) {
                System.out.print(i+1+". ");
                System.out.println(educators.get(i));
            }
        }
    }

    private ArrayList<Educator> searchEducatorThroughout(String surname)
    {
        ArrayList<Educator> searched = new ArrayList<>();
        if(groups.size() == 0)
        {
            System.out.println("Groups list is empty! U dont add any
student or teacher");
            return null;
        }

        for (Group group : groups)
        {
            for (int i = 0; i < group.getAmountStudents(); i++)
            {

if(surname.equals(group.getStudents().get(i).getSurName()))
                {
                    searched.add(group.getStudents().get(i));
                }
            }

            for (int i = 0; i < group.getAmountTeachers(); i++)
            {
```

```
540
541 if(surname.equals(group.getTeachers().get(i).getSurName()))
542                 {
543                     searched.add(group.getTeachers().get(i));
544                     return searched;
545                 }
546             }
547         }
548 
549         if(searched.size() >= 1)
550         {
551             return searched;
552         }
553 
554         return null;
555     }
556 
557 }
```

Controller.java

```
1  package controller;
2  
3  import actions.Actions;
4  import menu.Command;
5  import menu.CommandExecutor;
6  import menu.commands.*;
7  
8  import java.util.Scanner;
9  
10 /**
11  * Клас Controller потрібен для використання команд програми користувачем
12  * @see Command
13  * @see Actions
14  */
15 public class Controller {
16     private Actions actions = new Actions();
17     private CommandExecutor executor = new CommandExecutor();
18     private Command command;
19 
20 
21     /**
22      * Почати виконання програми
23      */
24     public void userStart()
25     {
26         int amountCommands = ShowCommandList();
27         int choice = UserChoose(amountCommands);
28         while(true)
29         {
30             switch (choice) {
```

```
                case 1: command = new showGroupsCommand(actions);
                    break;
                case 2: command = new
addEducatorWithKeyboardCommand(actions);
                    break;
                case 3: command = new
addEducatorFromFileCommand(actions);
                    break;
                case 4: command = new deleteEducatorCommand(actions);
                    break;
                case 5: command = new
showListOfEducatorsCommand(actions);
                    break;
                case 6: command = new sortStudentsCommand(actions);
                    break;
                case 7: command = new filterStudentsCommand(actions);
                    break;
                case 8: command = new searchForEducatorCommand(actions);
                    break;
                case 9: System.out.println("Exit from program...");
                    System.exit(0);
                    break;
                default:
                    System.out.println("Critical error. Program is
closing...");
                    System.exit(-1);
            }
            executor.setCommand(command);
            executor.execute();
            ShowCommandList();
            choice = UserChoose(amountCommands);
        }
    }

    /**
     * Показати список команд
     * @return ціле число - кількість команд
     */
    private int ShowCommandList() {
        System.out.println("\n");
        System.out.println("****************************");
        System.out.println(" EDUCATION INSTITUTION MENU ");
        System.out.println("****************************");
        System.out.println("1. Show list of groups.");
        System.out.println("2. Add educator with keyboard...");
        System.out.println("3. Add educators from file...");
        System.out.println("4. Delete educator...");
        System.out.println("5. Show list of students/teachers of certain
group.");
        System.out.println("6. Sort students of certain group...");
        System.out.println("7. Filter students of certain group...");
```

```
        System.out.println("8. Search for student/teacher.");
        System.out.println("9. Exit from program. ");
        System.out.println("*****************************");
        return 9;
    }

    /**
     * Вибір наступної команди користувачем
     * @param amountCommands кількість команд
     * @return ціле число - вибрана команда
     */
    private int UserChoose(int amountCommands) {
        System.out.print("Your choose: ");
        Scanner in = new Scanner(System.in);
        while (!in.hasNextInt()) {
            System.out.print("Error. Try again: ");
        }
        int choose = in.nextInt();
        while (choose > amountCommands && choose < 0) {
            System.out.print("Unknown command. Try again: ");
            while (!in.hasNextInt()) {
                System.out.print("Error. Try again: ");
            }
            choose = in.nextInt();
        }
        return choose;
    }
}
```

Command.java

```
package menu;

/**
 * Інтерфейс команда
 * потрібен для взаємодія юзера та функціоналу програми
 */
public interface Command {
    void execute();
}
```

CommandExecutor.java

```
package menu;

/**
 * Клас інвокер, виконує команди
 * @see Command
 */
public class CommandExecutor {
    private Command command;
```

```
9
10      public void setCommand(Command c)
11      {
12          this.command = c;
13      }
14
15      public void execute() {
16          command.execute();
17      }
18  }
```

addEducatorFromFileCommand.java

```
1   package menu.commands;
2
3   import actions.Actions;
4   import menu.Command;
5
6   public class AddEducatorFromFileCommand implements Command {
7       Actions actions;
8
9       public addEducatorFromFileCommand(Actions actions) {
10          this.actions = actions;
11      }
12
13      public void execute() {
14          actions.addEducatorFromFile();
15      }
16  }
```

addEducatorWithKeyboardCommand.java

```
1   package menu.commands;
2
3   import actions.Actions;
4   import menu.Command;
5
6   public class AddEducatorWithKeyboardCommand implements Command {
7       Actions actions;
8
9       public AddEducatorWithKeyboardCommand(Actions actions) {
10          this.actions = actions;
11      }
12
13      public void execute() {
14          actions.addEducatorWithKeyboard();
15      }
16  }
```

deleteEducatorCommand.java

```java
package menu.commands;

import actions.Actions;
import menu.Command;

public class DeleteEducatorCommand implements Command {
    Actions actions;

    public deleteEducatorCommand(Actions actions) {
        this.actions = actions;
    }

    public void execute() {
        actions.deleteEducator();
    }
}
```

filterStudentsCommand.java

```java
package menu.commands;

import actions.Actions;
import menu.Command;

public class FilterStudentsCommand implements Command {
    Actions actions;

    public FilterStudentsCommand(Actions actions) {
        this.actions = actions;
    }

    public void execute() {
        actions.filterStudents();
    }
}
```

searchForEducatorCommand.java

```java
package menu.commands;

import actions.Actions;
import menu.Command;

public class searchForEducatorCommand implements Command {
    Actions actions;

    public searchForEducatorCommand(Actions actions) {
        this.actions = actions;
    }
```

```
12
13      public void execute() {
14          actions.searchForEducator();
15      }
16  }
```

showGroupsCommand.java

```
1   package menu.commands;
2
3   import actions.Actions;
4   import menu.Command;
5
6   public class ShowGroupsCommand implements Command {
7       Actions actions;
8
9       public ShowGroupsCommand(Actions actions) {
10          this.actions = actions;
11      }
12
13      public void execute() {
14          actions.showGroups();
15      }
16  }
```

showListOfEducatorsCommand.java

```
1   package menu.commands;
2
3   import actions.Actions;
4   import menu.Command;
5
6   public class ShowListOfEducatorsCommand implements Command {
7       Actions actions;
8
9       public ShowListOfEducatorsCommand(Actions actions) {
10          this.actions = actions;
11      }
12
13      public void execute() {
14          actions.showListOfEducators();
15      }
16  }
```

sortStudentsCommand.java

```
1   package menu.commands;
2
3   import actions.Actions;
4   import menu.Command;
```

```
5
6  public class SortStudentsCommand implements Command {
7      Actions actions;
8
9      public SortStudentsCommand(Actions actions) {
10         this.actions = actions;
11     }
12
13     public void execute() {
14         actions.sortStudents();
15     }
16 }
```

Educator.java

```
1  package model;
2
3  import java.text.ParseException;
4  import java.text.SimpleDateFormat;
5  import java.util.Date;
6
7  /**
8   * Клас Освітчанин(Educator) є батьківським класом для класів Студент та
9  Викладач
10  * Зберігає основний список полів та методи геттери та сеттери
11  * @see com.kushka.model.educatorTypes.Student
12  * @see com.kushka.model.educatorTypes.Teacher
13  */
14 public class Educator {
15     private String name; // ім'я
16     private String surName; // прізвище
17     private String fatherName; // по батькові
18     private String date; // дата народження у стрінгу
19     private Date dateOfBirth; // дата народження у класі Date
20     private boolean isMale; // стать (1-Ч, 0-Ж)
21
22     public Educator(){
23         this.name = "unknown";
24         this.surName = "unknown";
25         this.fatherName = "unknown";
26         this.date = "00.00.0000";
27         try {
28             this.dateOfBirth = new
29 SimpleDateFormat("dd.MM.yyyy").parse(date);
30         }
31         catch(ParseException e)
32         {
33             System.out.println("Parse Error exception. Critical error");
34             System.exit(-1);
35         }
36         this.isMale = false;
```

```
    }

    public Educator(String name, String surName, String fatherName,
String date, boolean isMale) {
        this.name = name;
        this.surName = surName;
        this.fatherName = fatherName;
        this.date = date;
        try {
            this.dateOfBirth = new
SimpleDateFormat("dd.MM.yyyy").parse(date);
        }
        catch(ParseException e)
        {
            System.out.println("Parse Error exception. Critical error");
            System.exit(-1);
        }
        this.isMale = isMale;
    }

    public String getName() {
        return name;
    }

    public String getSurName() {
        return surName;
    }

    public String getFatherName() {
        return fatherName;
    }

    public boolean isMale() {
        return isMale;
    }

    public Date getDateOfBirth() {
        return dateOfBirth;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setSurName(String surName) {
        this.surName = surName;
    }

    public void setFatherName(String fatherName) {
        this.fatherName = fatherName;
    }
```

```
88
89      public void setDateOfBirth(String date) {
90          this.date = date;
91          try {
92              this.dateOfBirth = new
93  SimpleDateFormat("dd.MM.yyyy").parse(date);
94          }
95          catch(ParseException e)
96          {
97              System.out.println("Parse Error exception. Critical error");
98              System.exit(-1);
99          }
100     }
101
102     public String getDate() {
103         return date;
104     }
105
106     public void setMale(boolean male) {
107         isMale = male;
108     }
109 }
```

Group.java

```
1   package model;
2
3   import model.educatorTypes.Student;
4   import model.educatorTypes.Teacher;
5   import scanner.Scan;
6
7   import java.text.ParseException;
8   import java.text.SimpleDateFormat;
9   import java.util.ArrayList;
10  import java.util.Collections;
11  import java.util.Comparator;
12  import java.util.Date;
13
14  /**
15   * Клас Група зберігає список студентів певної групи а також їхніх
16  викладачів
17   * Виконуються деякі функції зі списками студентів та викладачів
18   * @see Student
19   * @see Teacher
20   * @see Educator
21   * @see com.kushka.actions.Actions
22   */
23  public class Group {
24      private String name; // назва групи
25      private ArrayList<Student> students; // студенти групи
26      private ArrayList<Teacher> teachers;  // викладачі групи
```

```
27
28      public Group(String name, ArrayList<Student> students,
29  ArrayList<Teacher> teachers) {
30          this.name = name;
31          this.students = students;
32          this.teachers = teachers;
33      }
34
35      public Group() {
36          this.name = "unknown";
37          this.students = new ArrayList<>();
38          this.teachers = new ArrayList<>();
39      }
40
41      public String getName() {
42          return name;
43      }
44
45      public ArrayList<Student> getStudents() {
46          return students;
47      }
48
49      public ArrayList<Teacher> getTeachers() {
50          return teachers;
51      }
52
53      public void setName(String name) {
54          this.name = name;
55      }
56
57      public void setStudents(ArrayList<Student> students) {
58          this.students = students;
59      }
60
61      public void setTeachers(ArrayList<Teacher> teachers) {
62          this.teachers = teachers;
63      }
64
65      public void addStudentToGroup(Student student) {
66          students.add(student);
67      }
68
69      public void addTeacherToGroup(Teacher teacher) {
70          teachers.add(teacher);
71      }
72
73
74      public int getAmountStudents()
75      {
76          return students.size();
77      }
```

```
78
79      public int getAmountTeachers()
80      {
81          return teachers.size();
82      }
83
84      /**
85       * Показати список студентів
86       */
87      public void showStudentsList()
88      {
89          System.out.println("Students list of group "+this.getName());
90          for (int i = 0; i < this.getAmountStudents(); i++) {
91              System.out.print(i+1 + ". ");
92              System.out.println(this.getStudents().get(i));
93          }
94      }
95
96      /**
97       * Показати список викладачів
98       */
99      public void showTeachersList()
100     {
101         System.out.println("Teachers list:");
102         for (int i = 0; i < this.getAmountTeachers(); i++) {
103             System.out.print(i+1 + ". ");
104             System.out.println(this.getTeachers().get(i));
105         }
106     }
107
108     /**
109      * Сортувати студентів за прізвищем
110      */
111     public void sortStudentsForSurname()
112     {
113         Comparator<Student> comparator = Comparator.comparing(obj ->
114 obj.getSurName());
115         Collections.sort(students, comparator);
116     }
117
118     /**
119      * Сортувати студентів за датою народження
120      */
121     public void sortStudentsForDateOfBirth()
122     {
123         Comparator<Student> comparator = Comparator.comparing(obj ->
124 obj.getDateOfBirth());
125         Collections.sort(students, comparator.reversed());
126     }
127
128
```

```
129     /**
130      * Фільтр: показати лише пільговиків
131      */
132     public void showOnlyBeneficiary(){
133         for (int i = 0; i < students.size(); i++) {
134             if(students.get(i).isBeneficiary())
135             {
136                 System.out.print(i+1 + ". ");
137                 System.out.println(students.get(i));
138             }
139         }
140     }
141 
142     /**
143      * Показати лише хлопців
144      */
145     public void showOnlyMale(){
146         for (int i = 0; i < students.size(); i++) {
147             if(students.get(i).isMale())
148             {
149                 System.out.print(i+1 + ". ");
150                 System.out.println(students.get(i));
151             }
152         }
153     }
154 
155     /**
156      * Показати лише дівчат
157      */
158     public void showOnlyFemale()
159     {
160         for (int i = 0; i < students.size(); i++) {
161             if(!students.get(i).isMale())
162             {
163                 System.out.print(i+1 + ". ");
164                 System.out.println(students.get(i));
165             }
166         }
167     }
168 
169     /**
170      * Відфільтрувати учнів за датою народження
171      */
172     public void filterByDateOfBirth() throws ParseException {
173         Scan in = new Scan();
174         Date interval_start;
175         Date interval_end;
176         System.out.println("Filter from(dd.MM.yyyy): ");
177         interval_start = new
178 SimpleDateFormat("dd.MM.yyyy").parse(in.scanString());
179         System.out.println("To (dd.MM.yyyy): ");
```

```
        interval_end = new
SimpleDateFormat("dd.MM.yyyy").parse(in.scanString());
        System.out.println("Filtered list: ");
        int counter = 1;
        for (Student st : students)
        {

            if ((st.getDateOfBirth().compareTo(interval_start) >= 0) &&
(st.getDateOfBirth().compareTo(interval_end) <= 0)) {
                System.out.println(counter + ". " + st);
                counter++;
            }

        }
    }

    /**
     * Метод ту стрінг для групи
     * @return рядок з назвою групи і кількістю студентів і викладачів
     */
    @Override
    public String toString() {
        return name + " | Студентів: " + getAmountStudents() + " |
Викладачів: " + getAmountTeachers();
    }
}
```

Student.java

```
package model.educatorTypes;

import com.kushka.model.Educator;

/**
 * Клас Студент - чайлд клас класу Educator(освітчанин)
 * зберігає додаткое поле назва групи, та чи пільговик
 * @see Educator
 */
public class Student extends Educator {
    private String group; // група
    private boolean isBeneficiary; // пільговик (1-так,0-ні)

    public Student() {
        this.group = "unknown";
        this.isBeneficiary = false;
    }

    public Student(String name, String surName, String fatherName, String
date, boolean isMale, String group, boolean isBeneficiary) {
        super(name, surName, fatherName, date, isMale);
        this.group = group;
```

```
        this.isBeneficiary = isBeneficiary;
    }

    public String getGroup() {
        return group;
    }

    public boolean isBeneficiary() {
        return isBeneficiary;
    }

    public void setGroup(String group) {
        this.group = group;
    }

    public void setBeneficiary(boolean beneficiary) {
        isBeneficiary = beneficiary;
    }


    @Override
    public String toString() {
        String fullname = this.getSurName() + " " + this.getName() + " "
+ this.getFatherName();
        if (this.isMale()) {
            String he = "Студент " + fullname +
                    "\n" + "Народився " + this.getDate() +
                    "\n Група '" + group + '\'';
                    if(this.isBeneficiary())
                    {
                        return he + " , пільговик";
                    } else
                        return he;
        }
        else{
            String she = "Студентка " + fullname +
                    "\n" + "Народилася " + this.getDate() +
                    "\nГрупа '" + group + '\'';
            if(this.isBeneficiary())
            {
                return she + " , пільговик";
            } else
                return she;
        }
    }
}
```

Teacher.java

```
package model.educatorTypes;

```

```
import model.Educator;

import java.util.ArrayList;


/**
 * Клас Викладач - чайл клас класу Освітчанин (Едукейтор)
 * Зберігає профільний предмет викладача, та перелік груп, в яких він викладає
 */
public class Teacher extends Educator {
    private String profileSubject;
    private ArrayList<String> groups;

    public Teacher(String profileSubject, ArrayList<String> groups) {
        this.profileSubject = profileSubject;
        this.groups = groups;
    }

    public Teacher(String name, String surName, String fatherName, String date, boolean isMale, String profileSubject, ArrayList<String> groups) {
        super(name, surName, fatherName, date, isMale);
        this.profileSubject = profileSubject;
        this.groups = groups;
    }

    public Teacher()
    {
        this.profileSubject = "none";
        this.groups = new ArrayList<String>();
    }

    public String getProfileSubject() {
        return profileSubject;
    }

    public ArrayList<String> getGroups() {
        return groups;
    }

    public void setProfileSubject(String profileSubject) {
        this.profileSubject = profileSubject;
    }

    public void setGroups(ArrayList<String> groups) {
        this.groups = groups;
    }

    @Override
    public String toString() {
```

```
        String fullname = this.getSurName() + " " + this.getName() + " "
+ this.getFatherName();
        if (this.isMale()) {
            return "Викладач " + fullname +
                    "\n" + "Народився " + this.getDate() +
                    "\nВеде предмет " + profileSubject +
                    " у групах: " + groups;
        }
        else{
            return "Викладачка " + fullname +
                    "\n" + "Народилася " + this.getDate() +
                    "\nВеде предмет " + profileSubject +
                    " у групах: " + groups;
        }
    }
}
```

Scan.java

```
package scanner;

import java.util.Scanner;

public class Scan {
    private Scanner in;

    /**
     * Прочитати з клавіатури ціле число
     * @return ціле число, введене користувачем
     */
    public int scanInt()
    {
        int value;
        in = new Scanner(System.in);
        while(!in.hasNextInt())
        {
            in.next();
            System.out.print("Retry: ");
        }
        value = in.nextInt();
        return value;
    }

    /**
     * Прочитати рядок введений юзером
     * @return рядок, введений з клавіатури
     */
    public String scanString()
    {
        in = new Scanner(System.in);
        String line;
```

```
        line = in.nextLine();
        return line;
    }

    /**
     * Прочитати дійсне число з клавіатури
     * @return дійсне число
     */
    public double scanDouble()
    {
        double value;
        in = new Scanner(System.in);
        while(!in.hasNextDouble())
        {
            in.next();
            System.out.print("Retry: ");
        }
        value = in.nextDouble();
        return value;
    }
}
```

3. ОПИС РЕАЛІЗОВАНИХ МЕТОДІВ

В даному розділі будуть описані основні методи програмного продукту, а також результати їх реалізації. Додатково, як можна було замітити вище, кожен метод та клас програми має документацію (Javadoc).

Оскільки в програмі використовується паттерн Команда, окрім деяких методів будуть наведені основні команди, їх опис, та результат реалізації (скріншот консольного вікна).

1.ShowCommandList()

Цей метод з класу `com.kushka.controller.Controller`

Суть проста – відобразити меню для подальших дій від користувача, активно використовується контролером, кожного разу для відображення списку меню.

Результат роботи:

```
****************************
 EDUCATION INSTITUTION MENU
****************************
1. Show list of groups.
2. Add educator with keyboard...
3. Add educators from file...
4. Delete educator...
5. Show list of students/teachers of certain group.
6. Sort students of certain group...
7. Filter students of certain group...
8. Search for student/teacher.
9. Exit from program.
****************************
```

Рис.3.0. Результат ShowCommandList()

2.AddEducatorFromFileCommand

Цей клас запускає метод `addEducatorFromFile()` з класу `com.kushka.actions.Actions.`

Він відкриває файл за вказаною адресою і читає з нього дані, які повинні бути записані за спеціальною формою користувачем.

Наприклад, існує файл educators.txt у якому записані студенти різних груп та виклдачі:

Рис.3.1. Файл educators.txt (не повністю)

Результат виконання даної команди:

```
Your choose: 3
File:
educators.txt
Reading data from file...

Your choose: 1
List of groups:
1. КН-21 | Студентів: 6 | Викладачів: 4
2. КН-20 | Студентів: 3 | Викладачів: 2
3. КН-22 | Студентів: 4 | Викладачів: 2

Group: КН-21
1-show student, 2-show teachers, 3-show all
Choose: 1
Students list of group КН-21
1. Студент Микола Шевченко Іванович
Народився 20.12.2000
 Група 'КН-21'
2. Студент Бурий Петро Карпович
Народився 05.03.2000
 Група 'КН-21' , пільговик
3. Студент Олійник Степан Денисович
Народився 20.03.2000
 Група 'КН-21'
4. Студент Кузьменко Микола Богданович
Народився 20.12.2000
 Група 'КН-21'
5. Студент Кравченко Захар Захарович
Народився 01.01.2001
 Група 'КН-21'
6. Студентка Клименко Тетяна Батьківна
Народилася 01.03.2000
Група 'КН-21'
```

Рис. 3.2. Результат виконання методу addEducatorFromFile()

3. addEducatorWithKeyboardCommand

Цей клас запускає метод addEducatorWithKeyboard() з класу com.kushka.actions.Actions. Він дає змогу ввести дані про студента вручну користувачем.

Результат:

```
Your choose: 2
Add: 1 - student || 2 - teacher
1
Name: Петро
Surname: Петрівський
Fathername: Петрович
Date of birth *format(dd.MM.yyyy): 10.10.2000
Sex: 1 - male, 2 - female
1
Group: КН-19
Beneficiary: 1 - yes, 2 -no
1
One more? 1 - Yes. 2 - No.
2

Your choose: 1
List of groups:
1. КН-21 | Студентів: 6 | Викладачів: 4
2. КН-20 | Студентів: 3 | Викладачів: 2
3. КН-22 | Студентів: 4 | Викладачів: 2
4. КН-19 | Студентів: 1 | Викладачів: 0

Group: КН-19
1-show student, 2-show teachers, 3-show all
Choose: 1
Students list of group КН-19
1. Студент Петівський Петро Петрович
Народився 10.10.2000
 Група 'КН-19' , пільговик
```

Рис.3.3. Результат виконання методу addEducatorWithKeyboard()

4. deleteEducatorCommand

Цей клас запускає метод deleteEducator() з класу com.kushka.actions.Actions. Він дає змогу видаляти любого зі студентів або викладачів за прізвищем. При тому враховано можливість появи однофамільців а також викладач видаляється зі списків усіх груп, де він викладав. Спробуємо видалити раніше доданого з файлу студента Бурого Петра з групи КН-21

Результат:

```
Your choose: 4
Enter surname: Бурий
Educator was deleted success.
Students list of group KH-21
1. Студент Микола Шевченко Іванович
Народився 20.12.2000
 Група 'KH-21'
2. Студент Олійник Степан Денисович
Народився 20.03.2000
 Група 'KH-21'
3. Студент Кузьменко Микола Богданович
Народився 20.12.2000
 Група 'KH-21'
4. Студент Кравченко Захар Захарович
Народився 01.01.2001
 Група 'KH-21'
5. Студентка Клименко Тетяна Батьківна
Народилася 01.03.2000
Група 'KH-21'
```

Рис.3.4. Результат виконання методу DeleteEducator()

5. showGroupsCommand

Ця команда запускає метод `showGroups()` з класу `com.kushka.actions.Actions` Суть проста – надрукувати список груп та кількість студентів в них і викладачів, які навчають ці групи. Результат роботи цього методу можна було побачити на Рис.3.2 або Рис.3.3.

6. showListOfEducatorsCommand

Ця команда запускає метод `showListOfEducators()` з класу `com.kushka.actions.Actions`. Суть така: роздрукувати студентів та/або викладачів вказаної групи. Групу можна ввести за назвою, або вибрати її зі списку. Результат роботи методу:

```
Your choose: 5
1-Put name of group; 2-Choose from list of groups
Your choice: 1
Group: KH-21
1-show student, 2-show teachers, 3-show all
Choose: 3
Students list of group KH-21
1. Студент Микола Шевченко Іванович
Народився 20.12.2000
 Група 'KH-21'
2. Студент Олійник Степан Денисович
Народився 20.03.2000
 Група 'KH-21'
3. Студент Кузьменко Микола Богданович
Народився 20.12.2000
 Група 'KH-21'
4. Студент Кравченко Захар Захарович
Народився 01.01.2001
 Група 'KH-21'
```

```
5. Студентка Клименко Тетяна Батьківна
Народилася 01.03.2000
Група 'КН-21'
Teachers list:
1. Викладач Іванов Петро Петрович
Народився 10.10.1972
Веде предмет Мат.аналіз у групах: [КН-20, КН-21, КН-22]
2. Викладачка Ткаченко Іванна Борисівна
Народилася 04.09.1980
Веде предмет Системний аналіз у групах: [КН-21]
3. Викладач Марченко Василь Олександрович
Народився 25.11.1970
Веде предмет Схемотехніка у групах: [КН-21, КН-22]
4. Викладачка Поліщук Христина Романівна
Народилася 05.10.1969
Веде предмет Філософія у групах: [КН-20, КН-21]
```

Рис. 3.5. Результат роботи методу showListOfEducators()

7. sortStudentsCommand

Ця команда запускає метод `sortStudents()` з класу `com. kushka.actions.Actions`

Суть методу – посортувати студентів вибраної групи за певним параметром:

1) в алфавітному порядку;

2) за датою народження:

Нехай записано дані з файлу educators.txt (рис. 3.1.) . До виконання команди група КН-21 виглядала так як зображено на рисунку 3.2. Після виконання сортування в алфавітному порядку:

```
Your choose: 6
1-Put name of group; 2-Choose from list of groups
Your choice: 1
Group: KH-21
Choose sort param: 1 - sort by surname
                   2 - sort by date of birth
1
Students was sorted success!

     Students list of group KH-21
     1. Студент Бурий Петро Карпович
     Народився 05.03.2000
      Група 'КН-21' , пільговик
     2. Студентка Клименко Тетяна Батьківна
     Народилася 01.03.2000
     Група 'КН-21'
     3. Студент Кравченко Захар Захарович
     Народився 01.01.2001
      Група 'КН-21'
     4. Студент Кузьменко Микола Богданович
     Народився 20.12.2000
      Група 'КН-21'
     5. Студент Микола Шевченко Іванович
     Народився 20.12.2000
      Група 'КН-21'
     6. Студент Олійник Степан Денисович
     Народився 20.03.2000
      Група 'КН-21'
```

Рис. 3.6. Результат sortStudents()

8. filterStudentsCommand

Ця команда запускає метод `filterStudents()` з класу `com.kushka.actions.Actions`

Суть методу відфільтрувати студентів певної групи за одним з поданих параметрів:

1) тільки хлопці;

2) тільки дівчата;

3) тільки пільговики;

4) студенти з певною датою народження з вказаного діапазону

Записано дані з рис.3.2 група КН-21.. Дивимось результат роботи:

```
Your choose: 7
1-Put name of group; 2-Choose from list of groups
Your choice: 1
Group: KH-21
Choose filter param: 1 - show only boys
                     2 - show only girls
                     3 - show only beneficiary
                     4 - filter by date of birth (from to)
4
Filter from(dd.MM.yyyy):
01.03.2000
To (dd.MM.yyyy):
21.12.2000

        Filtered list:
        1. Студент Бурий Петро Карпович
        Народився 05.03.2000
         Група 'КН-21' , пільговик
        2. Студентка Клименко Тетяна Батьківна
        Народилася 01.03.2000
        Група 'КН-21'
        3. Студент Кузьменко Микола Богданович
        Народився 20.12.2000
         Група 'КН-21'
        4. Студент Микола Шевченко Іванович
        Народився 20.12.2000
         Група 'КН-21'
        5. Студент Олійник Степан Денисович
        Народився 20.03.2000
         Група 'КН-21'
        Students was filtered success!
```

Рис.3.8. Результат FilterStudents()

9. searchForEducatorCommand

Ця команда запускає метод `searchForEducator()` з класу `com.kushka.actions.Actions`.

Суть методу – за введеним прізвищем освітчанина знайти і вивести інформацію про нього. Результат роботи:

```
Your choose: 8
Enter surname: Кузьменко
Студент Кузьменко Микола Богданович
Народився 20.12.2000
 Група 'КН-21'
```

Рис.3.9. Результат роботи searchForEducator()

4. ПРЕДСТАВЛЕННЯ ДІАГРАМИ КЛАСІВ

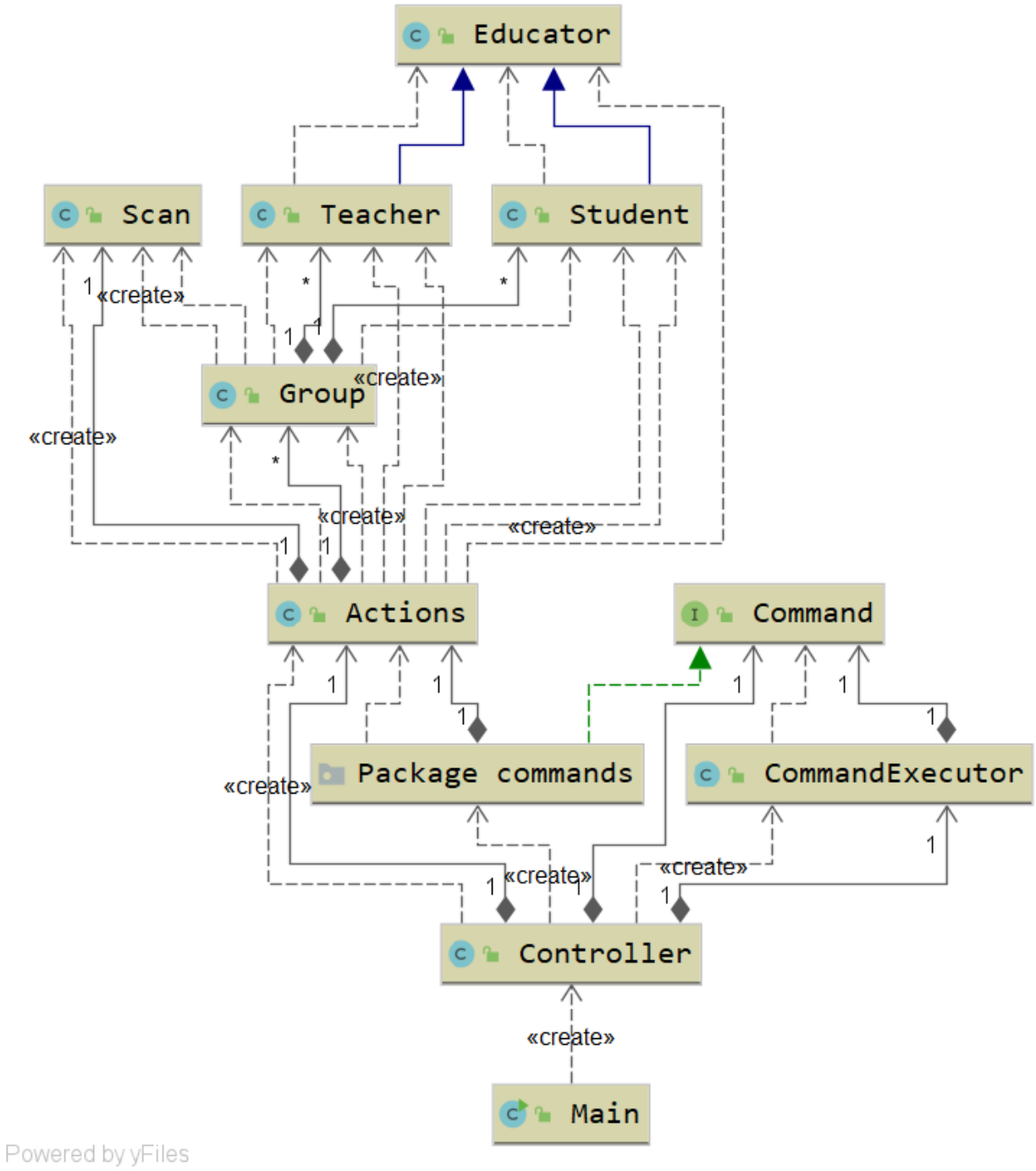

Рис. 4.1. Діаграма класів

ВИСНОВКИ

Після виконання даної курсової роботи я засвоїв основні підходи та методи побудови алгоритмів, закріпив навики програмної реалізації написаних алгоритмів із використанням принципів об’єктно-орієнтованого підходу, а також побудови об’єктно-орієнтованих структур за допомогою уніфікованої мови моделювання (Unified Modeling Language).

Додатково я навчився реалізовувати паттерн проектування Команда, також навчився працювати з файлами, зокрема читати дані з файлу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Брюс Еккель. Философия Java. 4-е полное изд. – СПБ.: Питер, 2015 – 1168 с.: ил. – (Серия «Классика computer science»). підручник / Брюс Еккель
2. Команда шаблон проектування // електронне джерело. Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%B0_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D1%94%D0%BA%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)
3. Навчальний заклад – Вікіпедія // електронне джерело. Режим доступу: https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B2%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%B7%D0%B0%D0%BA%D0%BB%D0%B0%D0%B4
4. Університет – Вікіпедія // електронне джерело. Режим доступу: https://uk.wikipedia.org/wiki/%D0%A3%D0%BD%D1%96%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82