



## ЗМІСТ

|  |    |
|--|----|
| ВСТУП .....  | 3  |
| 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ .....                   | 4  |
| 1.1. Постановка завдання.....                      | 4  |
| 1.2. Опис використаних методів програмування ..... | 4  |
| 1.3. Текст програми з коментарями.....             | 5  |
| 1.4. Результати роботи програми.....               | 5  |
| ВИСНОВКИ.....                                      | 12 |
| СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ .....               | 14 |
| ДОДАТОК А. КОД ПРОГРАМИ .....                      | 15 |
| Person.java .....                                  | 15 |
| Contact.java.....                                  | 16 |
| Student.java.....                                  | 18 |
| Order.java.....                                    | 20 |
| SorterByID.java.....                               | 23 |
| SorterByDate.java .....                            | 23 |
| SorterByLastName.java .....                        | 23 |
| SorterByMark.java .....                            | 23 |
| MenuForm.java .....                                | 24 |
| ContactForm.java .....                             | 25 |
| StudentForm.java .....                             | 29 |
| OrderForm.java .....                               | 35 |
| Form.java.....                                     | 39 |
| Main.java.....                                     | 39 |

## ВСТУП

Практика «Об'єктно-орієнтоване програмування» призначена для розвитку професійних навичок, навичок самостійного прийняття рішень у певних сферах роботи в реальних виробничих умовах шляхом виконання певних функцій та завдань, які є специфічними для майбутньої професії.

Мета практики – засвоєння базових знань з основ об'єктно-орієнтованого програмування, включаючи основні поняття, парадигми та принципи об'єктно-орієнтованого програмування. Оволодіння базовими навичками проектування програмних систем, набуття навичок об'єктно-орієнтованого програмування та оволодіння мовою програмування Java.

Завданням практики з «Об'єктно-орієнтованого програмування» є набуття знань, умінь та навичок (компетенцій) на рівні новітніх досягнень у програмуванні. Зокрема, розвивати:

- здатність проектувати та розробляти програмне забезпечення із об'єктно-орієнтованої парадигми програмування з відповідними моделями, методами та алгоритмами обчислень;
- застосування знань у практичних ситуаціях;
- знання та розуміння предметної області та розуміння професійної діяльності;
- розвинули здатність до алгоритмічного та логічного мислення;
- здатність розробляти архітектури, модулі та компоненти програмних систем;
- здатність приймати участь у проектуванні програмного забезпечення, включаючи моделювання його структури, поведінки та процесів функціонування;
- ознайомлення з життєвим циклом розробки програмного продукту;
- створення програмного продукту.

# 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1. Постановка завдання

Даний продукт призначений для заповнення даних про контакти, студентів та замовлення.

Даний продукт розроблявся для виконання наступних операцій.

З боку можливостей користувача:

- додавання інформації до списку;
- запис інформації у файл;
- зчитування і перегляд інформації з файлу;
- сортування інформації різними способами.

Для реалізації даного продукту використовується мова Java.

Для організації діалогу системи з користувачем повинен використовуватися графічний інтерфейс користувача або консоль.

## 1.2. Опис використаних методів програмування

При виконанні даного завдання використовувалися такі методи та конструкції:

- `Collections.sort(comparator)` – сортування масиву;
- `compare(T t, T t1)` інтерфейсу `Comparator<T extends Object>`;
- `ActionListener` та `ActionEvent`;
- `FileWriter()` та `FileReader()`;
- методи класів `JFrame`, `JButton`, `JTextField`, `JLabel`, `JOptionPane`;
- цикли `for` та `foreach`;
- лямбда-вирази;
- `try-catch`.

### 1.3. Текст програми з коментарями

Текст програми розміщений у Додатку А.

### 1.4. Результати роботи програми

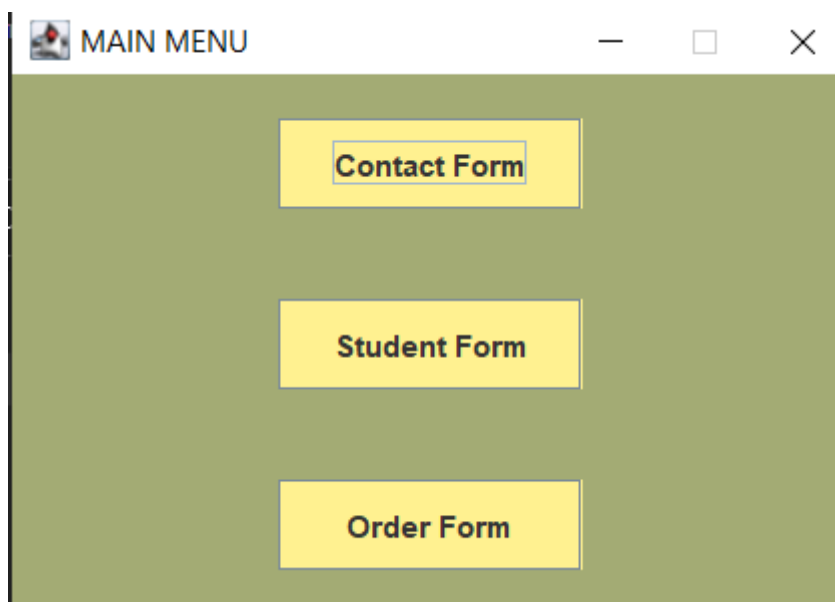


Рисунок 1.1 – Головне меню

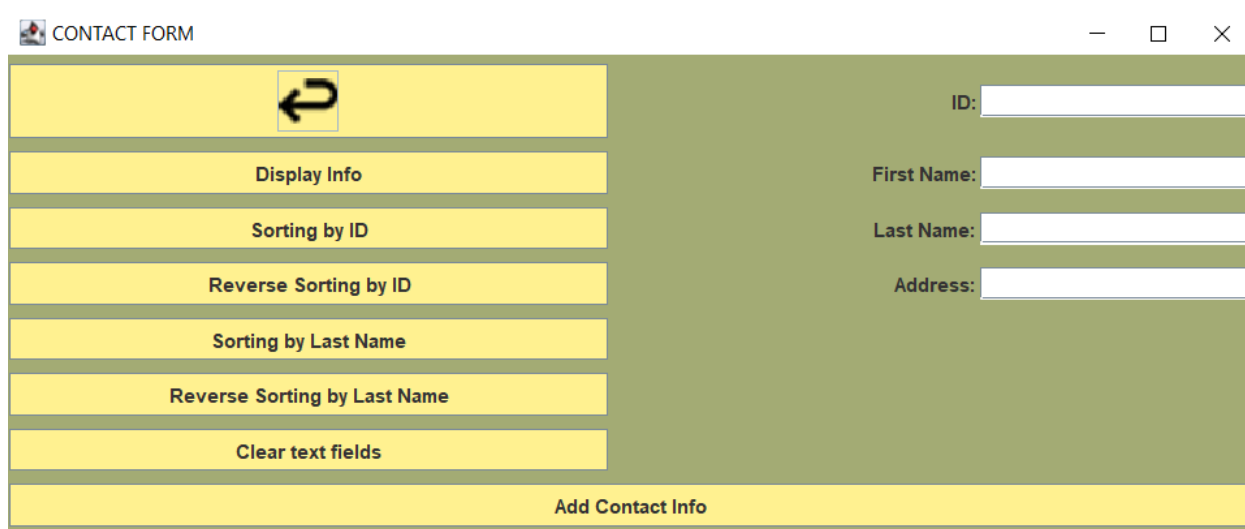
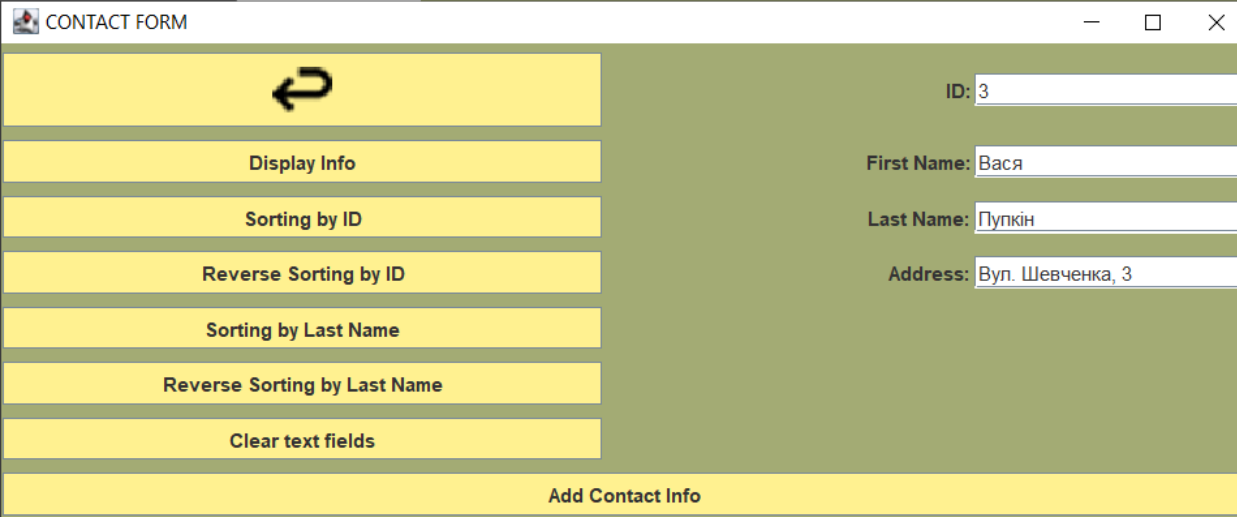
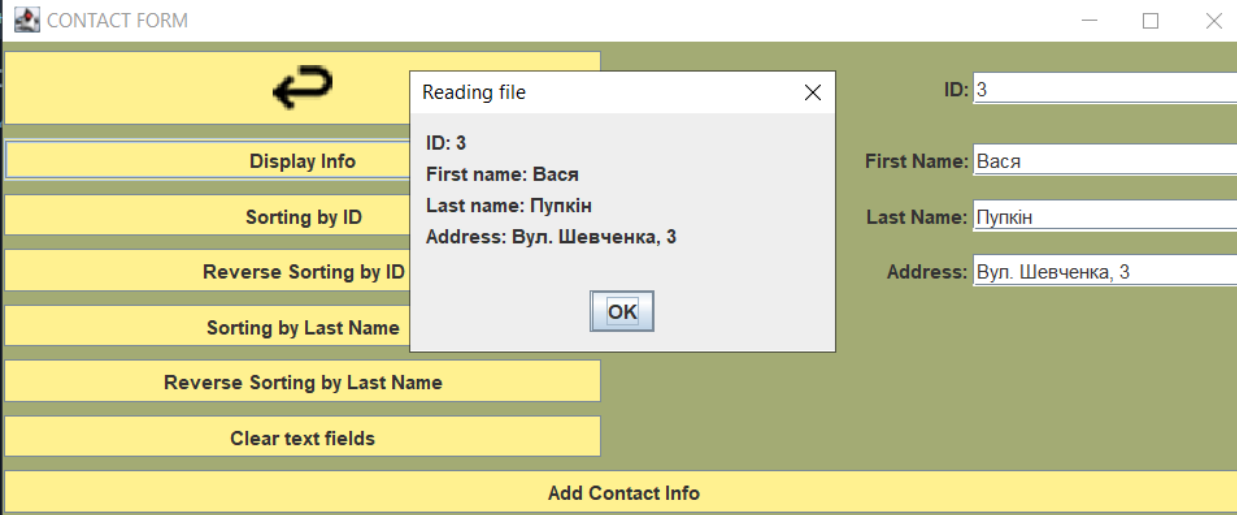


Рисунок 2.1 – Вікно форми для заповнення даних про контакт



The screenshot shows a window titled "CONTACT FORM" with a standard Windows-style title bar (minimize, maximize, close buttons). The window has a green background. On the left, there is a vertical stack of yellow buttons: a large button with a circular arrow icon, followed by "Display Info", "Sorting by ID", "Reverse Sorting by ID", "Sorting by Last Name", "Reverse Sorting by Last Name", "Clear text fields", and a wide "Add Contact Info" button at the bottom. On the right, there are input fields for "ID: 3", "First Name: Вася", "Last Name: Пупкін", and "Address: Вул. Шевченка, 3".

Рисунок 2.2 – Введення даних у поля і додавання даних до списку



This screenshot shows the same "CONTACT FORM" window as in Figure 2.2, but with a "Reading file" dialog box open in the center. The dialog box has a title bar with a close button and contains the following text: "ID: 3", "First name: Вася", "Last name: Пупкін", and "Address: Вул. Шевченка, 3". At the bottom of the dialog box is an "OK" button. The background application window is partially obscured by the dialog box.

Рисунок 2.3 – Зчитування інформації з файлу і її виведення на екран

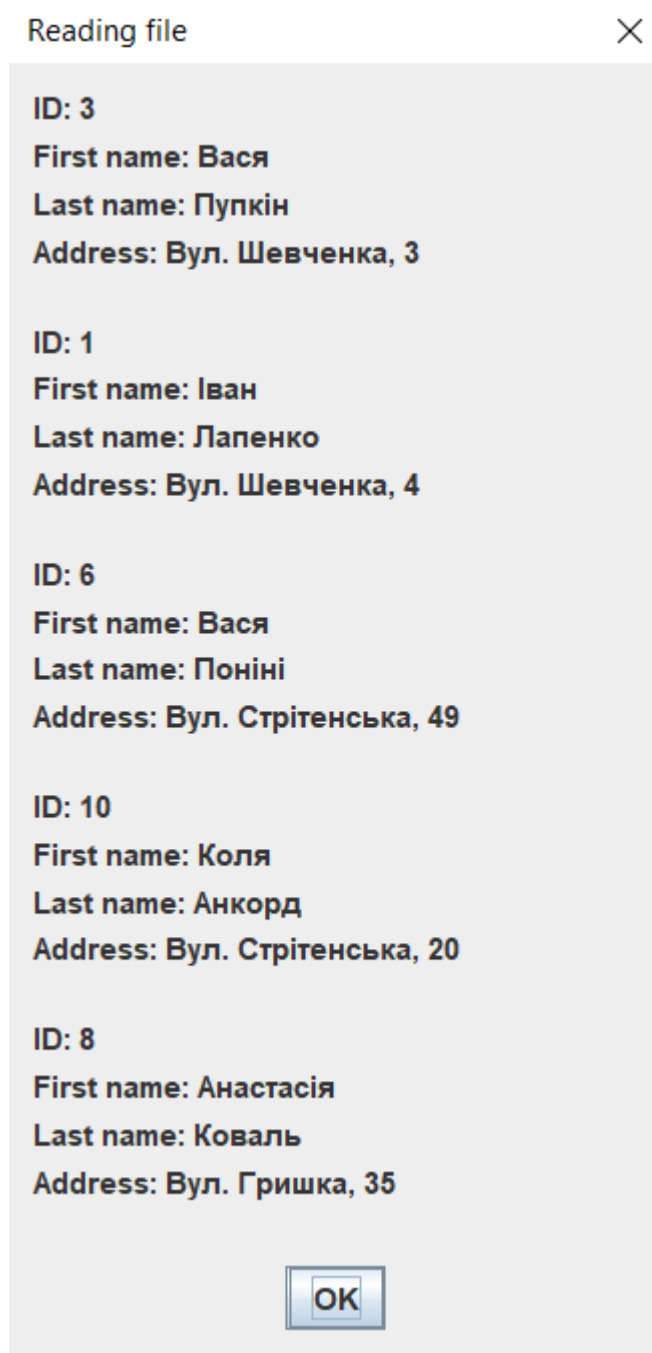


Рисунок 3.1 – Список до сортування

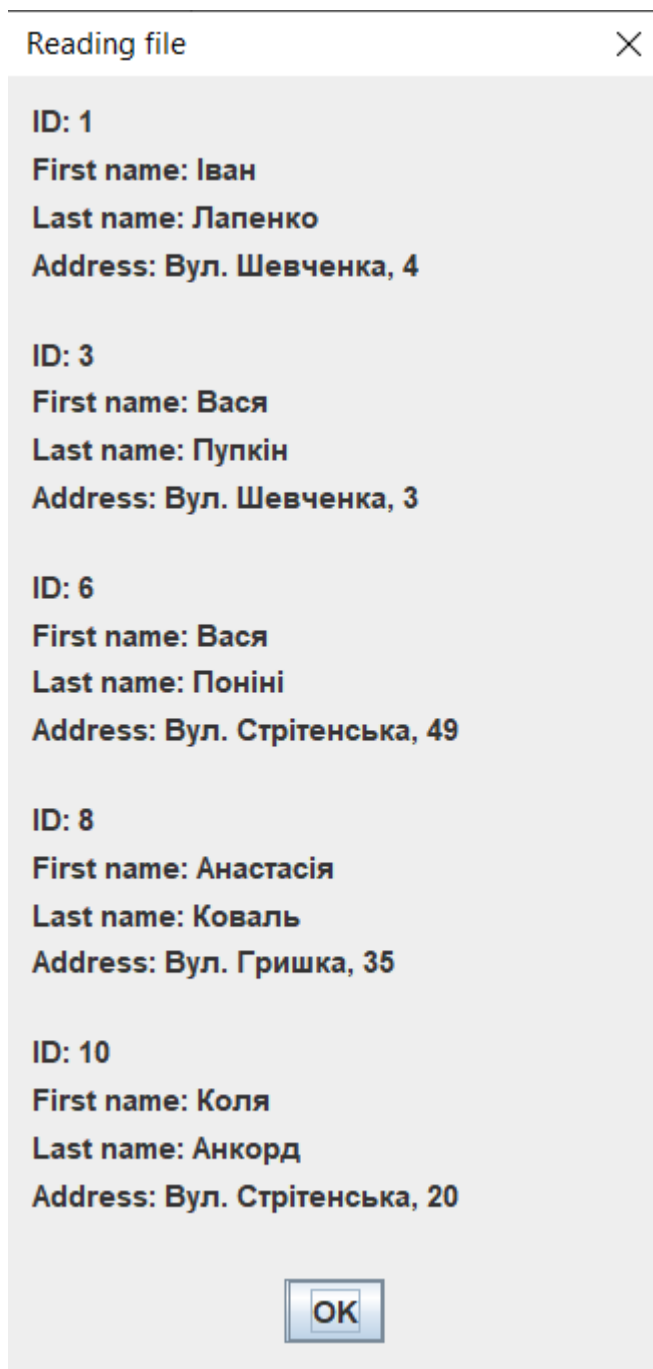


Рисунок 3.2 – Список після сортування за зростанням по ID



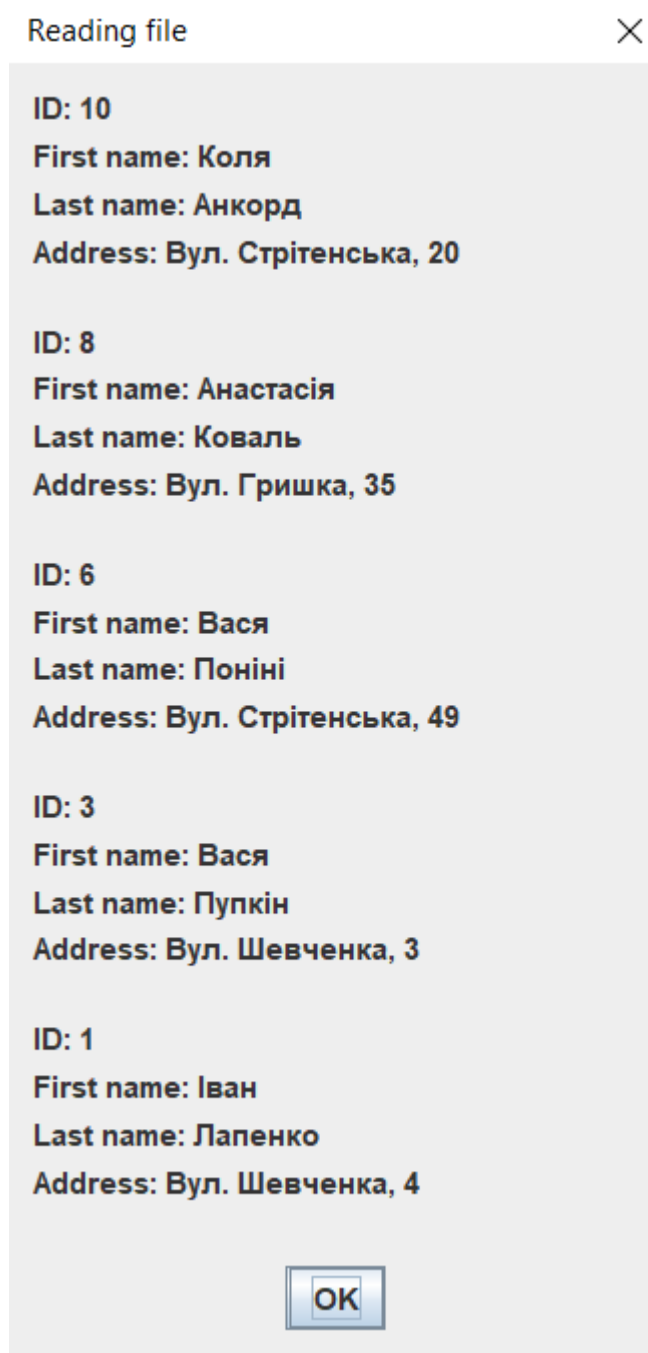


Рисунок 3.3 – Список після сортування за спаданням по ID

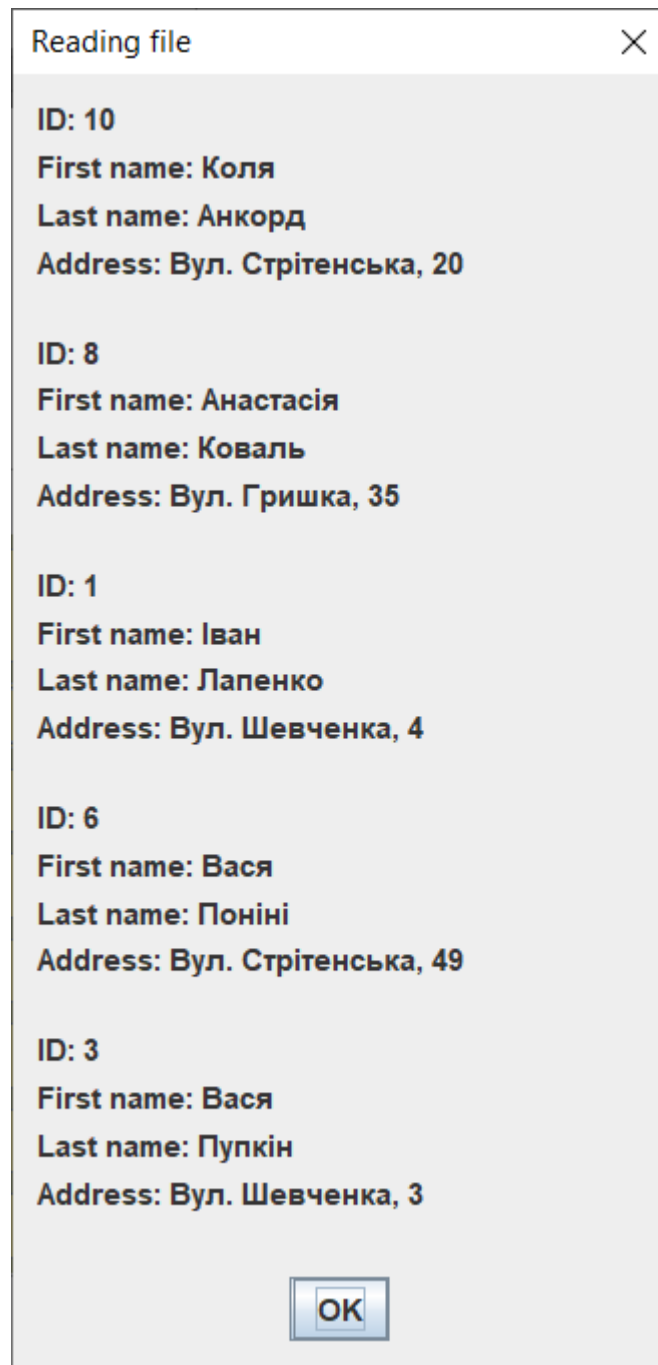


Рисунок 3.4 – Список після сортування за зростанням по Прізвищу

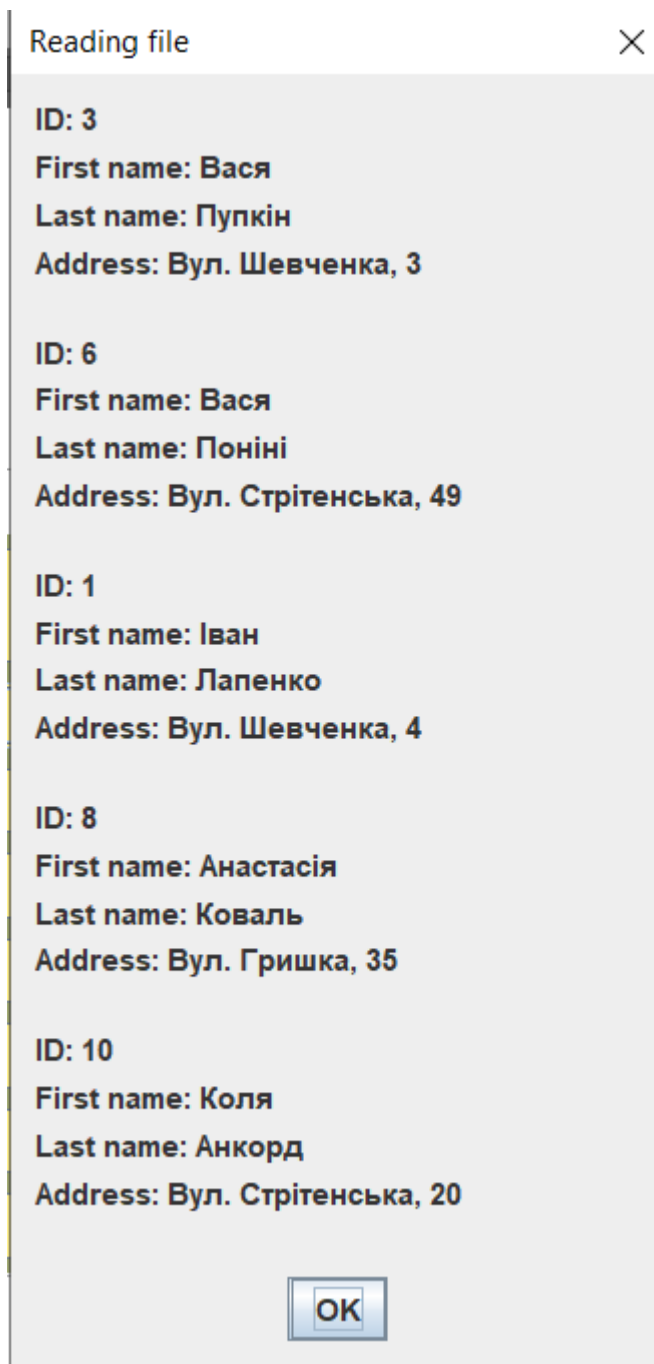


Рисунок 3.5 – Список після сортування за спаданням по Прізвищу

## ВИСНОВКИ

У процесі проходження практики ми закріпили набуті під час навчального процесу теоретичні знання з дисципліни «Об'єктно-орієнтоване програмування» і здобули практичні навички застосування цих знань, а також:

- на етапі написання коду програми вдосконалено навички програмування на мові Java;
- поглибив теоретичні знання і практичні навички, навички пошуку та систематизації інформації, її обробки із застосуванням автоматизованих інформаційних систем та технологій; оволодів методиками теоретичного та експериментального дослідження при розв'язуванні конкретних проблем;
- ознайомилися з відмінностями між об'єктно-орієнтованим програмуванням та функціональним програмуванням;
- ознайомилися з порядком створення класів та їх будову;
- принципи побудови класів та їх методів, основні типи класів та способи їх надбудови та взаємодії;
- ознайомилися з прийомами роботи з одновимірними та багатовимірними масивами, динамічними масивами;
- отримали практичні навички створення об'єктно-орієнтованого програмного коду;
- навчилися обгрунтовувати та проектувати ієрархію класів та об'єктів проекту;
- навчилися організовувати безпечний доступ до інформації завдяки застосуванню засобів інкапсуляції;
- ознайомилися з методами повторного використання коду та інформаційних компонентів через механізми успадкування класів та поліморфізму;

- дізналися про засоби стандартної бібліотеки Java для реалізації інтерфейсів взаємодії з людиною;
- навчилися оперувати даними і методами та створювати програмні засоби керування ними;
- навчилися користуватися середовищем розробки Netbeans;
- навчилися використовувати різні технології програмування на мові Java;
- навчилися самостійно опановувати нові методи та технології розробки програмного забезпечення;
- навчилися створювати та обробляти файли і інформацію, яка зберігається в них;
- навчилися створювати консольне та віконне програмне забезпечення;
- навчилися виконувати тестування та налагодження програми;
- створено програму за зазначеними вимогами;
- підготовлено звіт, в якому задокументовано результати практики;

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. GeeksforGeeks. Java Programming Language. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/java/> (дата звернення 05.03.2021). – Назва з екрану.
2. JavaRush. Помощь по задачам. [Електронний ресурс] – Режим доступу до ресурсу: <https://javarush.ru/help> (дата звернення 05.03.2021). – Назва з екрану.
3. Stack Overflow. [Електронний ресурс] – Режим доступу до ресурсу: <https://stackoverflow.com> (дата звернення 05.03.2021). – Назва з екрану.
4. Освоюємо Java. [Електронний ресурс] – Режим доступу до ресурсу: [https://uk.wikibooks.org/wiki/Освоюємо\\_Java](https://uk.wikibooks.org/wiki/Освоюємо_Java) (дата звернення 03.03.2021). – Назва з екрану.
5. Програмування мовою Java / О.М. Васильєв. – Тернопіль: Навчальна книга – Богдан, 2019. – 696 с.; іл.
6. Руководство по языку программирования Java. [Електронний ресурс] – Режим доступу до ресурсу: <https://metanit.com/java/tutorial/> (дата звернення 03.03.2021). – Назва з екрану.

## ДОДАТОК А. КОД ПРОГРАМИ

### Person.java

```
package domain;

public class Person{
    protected int id;
    protected String firstName;
    protected String lastName;
    public Person() {
        this.id = 0;
        this.firstName = "Name";
        this.lastName = "Last name";
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        if (id >= 0) {
            this.id = id;
        }
        else {
            System.out.println("ID не може бути менше нуля!!!");
        }
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    @Override
    public String toString() {
        return "ID: " + id + "\nFirst name: " + firstName + "\nLast name: " + lastName;
    }
}
```

## Contact.java

```
package domain;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
public class Contact extends Person{
    private ArrayList<Contact> arrayList;
    private String address;
    public Contact(int id, String firstName, String lastName, String address) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.address = address;
        this.arrayList = new ArrayList<>();
    }
    public Contact() {
        this(5, "Petya", "Petrov", "Pushkin St.");
    }
    @Override
    public int getId() {
        return super.getId();
    }
    @Override
    public void setId(int id) {
        super.setId(id);
    }
    @Override
    public String getFirstName() {
        return super.getFirstName();
    }
    @Override
    public void setFirstName(String firstName) {
        super.setFirstName(firstName);
    }
    @Override
    public String getLastName() {
        return super.getLastName();
    }
    @Override
    public void setLastName(String lastName) {
```



```

        super.setLastName(lastName);
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public void fillingArrayList() {
        Contact contact = new Contact(id, firstName, lastName, address);
        arrayList.add(contact);
    }
    public void sortById() {
        Collections.sort(arrayList, new SorterById());
    }
    public void reverseSortById() {
        Collections.sort(arrayList, Collections.reverseOrder(new SorterById()));
    }
    public void sortByLastName() {
        Collections.sort(arrayList, new SorterByLastName());
    }
    public void reverseSortByLastName() {
        Collections.sort(arrayList, Collections.reverseOrder(new SorterByLastName()));
    }
    @Override
    public String toString() {
        return super.toString() + "\nAddress: " + getAddress() + "\n";
    }
    public void writeFile() {
        try(FileWriter fileWriter = new FileWriter("contacts.txt", false)) {
            StringBuilder strBuilder = new StringBuilder();

            arrayList.forEach((person) -> {
                strBuilder.append(person).append("\n");
            });

            fileWriter.write(String.valueOf(strBuilder));
            fileWriter.close();
        }
        catch(IOException e) {
            e.printStackTrace();
        }
    }

```

```

    }
}

```

## Student.java

```

package domain;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
public class Student extends Person{
    private int group;
    private String department;
    private String discipline;
    private byte mark;
    private String nameTeacher;
    private ArrayList<Student> arrayList;
    public Student(int id, String firstName, String lastName, int group, String department, String discipline, byte
mark, String nameTeacher) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.group = group;
        this.department = department;
        this.discipline = discipline;
        this.mark = mark;
        this.nameTeacher = nameTeacher;
        this.arrayList = new ArrayList<>();
    }
    public Student() {
        this(4, "Ivan", "Ivanov", 35, "Some department", "Some discipline", (byte)5, "Petrovna");
    }
    @Override
    public int getId() {
        return super.getId();
    }
    @Override
    public void setId(int id) {
        super.setId(id);
    }
    @Override
    public String getFirstName() {
        return super.getFirstName();
    }

```

```
}  
  
@Override  
public void setFirstName(String firstName) {  
    super.setFirstName(firstName);  
}  
  
@Override  
public String getLastName() {  
    return super.getLastName();  
}  
  
@Override  
public void setLastName(String lastName) {  
    super.setLastName(lastName);  
}  
  
public int getGroup() {  
    return group;  
}  
  
public void setGroup(int group) {  
    this.group = group;  
}  
  
public String getDepartment() {  
    return department;  
}  
  
public void setDepartment(String department) {  
    this.department = department;  
}  
  
public String getDiscipline() {  
    return discipline;  
}  
  
public void setDiscipline(String discipline) {  
    this.discipline = discipline;  
}  
  
public byte getMark() {  
    return mark;  
}  
  
public void setMark(byte mark) {  
    this.mark = mark;  
}  
  
public String getNameTeacher() {  
    return nameTeacher;  
}  
  
public void setNameTeacher(String nameTeacher) {  
    this.nameTeacher = nameTeacher;  
}
```

```

    }
    public void fillingArrayList() {
        Student student = new Student(id, firstName, lastName, group, department, discipline, mark, nameTeacher);
        arrayList.add(student);
    }
    public void sortById() {
        Collections.sort(arrayList, new SorterById());
    }
    public void reverseSortById() {
        Collections.sort(arrayList, Collections.reverseOrder(new SorterById()));
    }
    public void sortByLastName() {
        Collections.sort(arrayList, new SorterByLastName());
    }
    public void sortByMark() {
        Collections.sort(arrayList, Collections.reverseOrder(new SorterByMark()));
    }
    @Override
    public String toString() {
        return super.toString() + "\nGroup: " + group + "\nDepartment: " + department + "\nDiscipline: " + discipline +
            "\nMark: " + mark + "\nNameTeacher: " + nameTeacher + '\n';
    }
    public void writeFile() {
        try(FileWriter fileWriter = new FileWriter("students.txt", false)) {
            StringBuilder strBuilder = new StringBuilder();

            arrayList.forEach((string) -> {
                strBuilder.append(string).append('\n');
            });

            fileWriter.write(String.valueOf(strBuilder));
        }
        catch(IOException e) {
            e.printStackTrace();
        }
    }
}

```

## Order.java

```

package domain;
import java.io.FileWriter;
import java.io.IOException;

```

```

import java.util.ArrayList;
import java.util.Collections;
public class Order extends Person{
    private String name;
    private String dateTime;
    private byte type;
    private ArrayList<Order> arrayList;
    public Order(int id, String name, String courier, String dateTime, byte type) {
        this.id = id;
        this.name = name;
        this.lastName = courier;
        this.dateTime = dateTime;
        this.type = type;
        this.arrayList = new ArrayList<>();
    }
    public Order() {
        this(3, "Nokia 3310", "Ivanov", "12.12.2012 12:12:12", (byte)1);
    }
    @Override
    public int getId() {
        return super.getId();
    }
    @Override
    public void setId(int id) {
        super.setId(id);
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    @Override
    public void setLastName(String lastName) {
        super.setLastName(lastName);
    }
    @Override
    public String getLastName() {
        return super.getLastName();
    }
    public String getDateTime() {
        return dateTime;
    }

```

```

    }
    public void setDateTime(String dateTime) {
        this.dateTime = dateTime;
    }
    public byte getType() {
        return type;
    }
    public void setType(byte type) {
        this.type = type;
    }
    public void fillingArrayList() {
        Order order = new Order(id, name, lastName, dateTime, type);
        arrayList.add(order);
    }
    public void sortById() {
        Collections.sort(arrayList, new SorterById());
    }
    public void reverseSortById() {
        Collections.sort(arrayList, Collections.reverseOrder(new SorterById()));
    }
    public void sortByDate() {
        Collections.sort(arrayList, new SorterByDate());
    }
    @Override
    public String toString() {
        return "ID: " + id + "\nName of product: " + name + "\nLast name of courier: " + lastName + "\nDate Time: " +
dateTime + "\nType: " + type + "\n";
    }
    public void writeFile() {
        try(FileWriter fileWriter = new FileWriter("orders.txt", false)) {
            StringBuilder strBuilder = new StringBuilder();

            arrayList.forEach((string) -> {
                strBuilder.append(string).append("\n");
            });

            fileWriter.write(String.valueOf(strBuilder));
        }
        catch(IOException e) {
            e.printStackTrace();
        }
    }
}

```

```
}
```

### **SorterById.java**

```
package domain;
import java.util.Comparator;
public class SorterById implements Comparator<Person>{
    @Override
    public int compare(Person p1, Person p2) {
        return p1.getId() - p2.getId();
    }
}
```

### **SorterByDate.java**

```
package domain;
import java.util.Comparator;
public class SorterByDate implements Comparator<Order>{
    @Override
    public int compare(Order o1, Order o2) {
        return o1.getDateTime().compareTo(o2.getDateTime());
    }
}
```

### **SorterByLastName.java**

```
package domain;
import java.util.Comparator;
public class SorterByLastName implements Comparator<Person> {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.getLastName().compareTo(p2.getLastName());
    }
}
```

### **SorterByMark.java**

```
package domain;
import java.util.Comparator;
public class SorterByMark implements Comparator<Student> {
    @Override
    public int compare(Student s1, Student s2) {
        return Byte.compare(s1.getMark(), s2.getMark());
    }
}
```

## MenuForm.java

```

package gui;
import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import interfaces.Form;
public class MenuForm extends JFrame{
    private final Form form;
    public MenuForm(final Form form) {
        this.form = form;
        this.setTitle("MAIN MENU");
        this.setBounds(600,300,350,250);
        this.setResizable(false);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel panel = new JPanel(new GridBagLayout());
        panel.setBackground(Color.getHSBColor(0.19f, 0.32f, 0.67f));
        this.setContentPane(panel);
        GridBagConstraints grid = new GridBagConstraints();
        grid.weightx = 0;
        grid.weighty = 1;
        grid.fill = GridBagConstraints.HORIZONTAL;
        grid.ipady = 10;
        grid.ipadx = 10;
        ActionListener contactFormAction = (ActionEvent event) -> {
            form.changeContactForm();
            JOptionPane.showMessageDialog(null,
                "Для того, щоб додати дані про контакт, введіть його дані і натисніть кнопку Add Contact Info",
                "Instruction",
                JOptionPane.INFORMATION_MESSAGE);
        };
        ActionListener studentFormAction = (ActionEvent event) -> {
            form.changeStudentForm();
            JOptionPane.showMessageDialog(null,
                "Для того, щоб додати дані про студента, введіть його дані і натисніть кнопку Add Student Info",
                "Instruction",
                JOptionPane.INFORMATION_MESSAGE);
        };
        ActionListener orderFormAction = (ActionEvent event) -> {
            form.changeOrderForm();
            JOptionPane.showMessageDialog(null,

```



"Для того, щоб додати дані про замовлення, введіть його дані і натисніть кнопку Add Order Info",  
"Instruction",

```

        JOptionPane.INFORMATION_MESSAGE);
    };
    JButton contactFormButton = new JButton("Contact Form");
    grid.gridx = 0;
    grid.gridy = 0;
    contactFormButton.setBackground(Color.getHSBColor(3, 12, 13));
    contactFormButton.addActionListener(contactFormAction);
    panel.add(contactFormButton, grid);
    JButton studentFormButton = new JButton("Student Form");
    grid.gridx = 0;
    grid.gridy = 1;
    studentFormButton.setBackground(Color.getHSBColor(3, 12, 13));
    studentFormButton.addActionListener(studentFormAction);
    panel.add(studentFormButton, grid);
    JButton orderFormButton = new JButton("Order Form");
    grid.gridx = 0;
    grid.gridy = 2;
    orderFormButton.setBackground(Color.getHSBColor(3, 12, 13));
    orderFormButton.addActionListener(orderFormAction);
    panel.add(orderFormButton, grid);
}
}

```

## ContactForm.java

```

package gui;
import domain.Contact;
import interfaces.Form;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.FileReader;
import java.io.IOException;
public class ContactForm extends JFrame{
    private final Contact CONTACT;
    public ContactForm(final Form form) {
        this.CONTACT = new Contact();
        this.setTitle("CONTACT FORM");
        this.setBounds(400,290,780,330);
        this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    }
}

```

```
JPanel panel = new JPanel(new GridBagLayout());
panel.setBackground(Color.getHSBColor(0.19f, 0.32f, 0.67f));
this.setContentPane(panel);
GridBagConstraints grid = new GridBagConstraints();
grid.weightx = 1;
grid.weighty = 1;
grid.fill = GridBagConstraints.HORIZONTAL;
JLabel idLabel = new JLabel("ID: ");
grid.gridx = 2;
grid.gridy = 0;
grid.gridwidth = 1;
idLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(idLabel, grid);
JTextField idField = new JTextField();
grid.gridx = 5;
grid.gridy = 0;
grid.gridwidth = 1;
panel.add(idField, grid);
JLabel firstNameLabel = new JLabel("First Name: ");
grid.gridx = 2;
grid.gridy = 1;
grid.gridwidth = 1;
firstNameLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(firstNameLabel, grid);
JTextField firstNameField = new JTextField();
grid.gridx = 5;
grid.gridy = 1;
grid.gridwidth = 1;
panel.add(firstNameField, grid);
JLabel lastNameLabel = new JLabel("Last Name: ");
grid.gridx = 2;
grid.gridy = 2;
grid.gridwidth = 1;
lastNameLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(lastNameLabel, grid);
JTextField lastNameField = new JTextField();
grid.gridx = 5;
grid.gridy = 2;
grid.gridwidth = 1;
panel.add(lastNameField, grid);
JLabel addressLabel = new JLabel("Address: ");
grid.gridx = 2;
```

```

grid.gridy = 3;
grid.gridwidth = 1;
addressLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(addressLabel, grid);
JTextField addressField = new JTextField();
grid.gridx = 5;
grid.gridy = 3;
grid.gridwidth = 1;
panel.add(addressField, grid);
ActionListener displayInfo = (ActionEvent event) -> {
    try(FileReader fileReader = new FileReader("contacts.txt")) {
        StringBuilder strBuilder = new StringBuilder();
        while(fileReader.ready()) {
            strBuilder.append((char)fileReader.read());
        }
        JOptionPane.showMessageDialog(null, strBuilder.toString(), "Reading file",
JOptionPane.PLAIN_MESSAGE);
    }
    catch(IOException e) {
        e.printStackTrace();
    }
};
ActionListener backToMenu = (ActionEvent event) -> form.changeContactForm();
ActionListener sortById = (ActionEvent event) -> {
    CONTACT.sortById();
    CONTACT.writeFile();
};
ActionListener reverseSortById = (ActionEvent event) -> {
    CONTACT.reverseSortById();
    CONTACT.writeFile();
};
ActionListener sortByLastName = (ActionEvent event) -> {
    CONTACT.sortByLastName();
    CONTACT.writeFile();
};
ActionListener reverseSortByLastName = (ActionEvent event) -> {
    CONTACT.reverseSortByLastName();
    CONTACT.writeFile();
};
ActionListener clearTextFields = (ActionEvent event) -> {
    idField.setText("");
    firstNameField.setText("");

```

```

        lastNameField.setText("");
        addressField.setText("");
    };
    ActionListener addContactInfo = (ActionEvent event) -> {
        CONTACT.setId(Integer.parseInt(idField.getText()));
        CONTACT.setFirstName(firstNameField.getText());
        CONTACT.setLastName(lastNameField.getText());
        CONTACT.setAddress(addressField.getText());
        CONTACT.fillingArrayList();
        CONTACT.writeFile();
    };
    ImageIcon backToMenuIcon = new ImageIcon("D:\\study\\Practic OOP\\Practic-
OOP\\PR_4\\src\\icons\\back_to_menu.png");
    JButton backToMenuButton = new JButton(backToMenuIcon);
    grid.gridx = 0;
    grid.gridy = 0;
    grid.gridwidth = 1;
    backToMenuButton.setBackground(Color.getHSBColor(3, 12, 13));
    backToMenuButton.addActionListener(backToMenu);
    panel.add(backToMenuButton, grid);
    JButton displayInfoButton = new JButton("Display Info");
    grid.gridx = 0;
    grid.gridy = 1;
    grid.gridwidth = 1;
    displayInfoButton.setBackground(Color.getHSBColor(3, 12, 13));
    displayInfoButton.addActionListener(displayInfo);
    panel.add(displayInfoButton, grid);
    JButton sortByIdButton = new JButton("Sorting by ID");
    grid.gridx = 0;
    grid.gridy = 2;
    grid.gridwidth = 1;
    sortByIdButton.setBackground(Color.getHSBColor(3, 12, 13));
    sortByIdButton.addActionListener(sortById);
    panel.add(sortByIdButton, grid);
    JButton reverseSortByIdButton = new JButton("Reverse Sorting by ID");
    grid.gridx = 0;
    grid.gridy = 3;
    grid.gridwidth = 1;
    reverseSortByIdButton.setBackground(Color.getHSBColor(3, 12, 13));
    reverseSortByIdButton.addActionListener(reverseSortById);
    panel.add(reverseSortByIdButton, grid);
    JButton sortByLastNameButton = new JButton("Sorting by Last Name");

```

```

        grid.gridx = 0;
        grid.gridy = 4;
        grid.gridwidth = 1;
        sortByLastNameButton.setBackground(Color.getHSBColor(3, 12, 13));
        sortByLastNameButton.addActionListener(sortByLastName);
        panel.add(sortByLastNameButton, grid);
        JButton reverseSortByLastNameButton = new JButton("Reverse Sorting by Last Name");
        grid.gridx = 0;
        grid.gridy = 5;
        grid.gridwidth = 1;
        reverseSortByLastNameButton.setBackground(Color.getHSBColor(3, 12, 13));
        reverseSortByLastNameButton.addActionListener(reverseSortByLastName);
        panel.add(reverseSortByLastNameButton, grid);
        JButton clearTextFieldsButton = new JButton("Clear text fields");
        grid.gridx = 0;
        grid.gridy = 8;
        grid.gridwidth = 1;
        clearTextFieldsButton.setBackground(Color.getHSBColor(3, 12, 13));
        clearTextFieldsButton.addActionListener(clearTextFields);
        panel.add(clearTextFieldsButton, grid);
        JButton addInfoButton = new JButton("Add Contact Info");
        grid.gridx = 0;
        grid.gridy = 10;
        grid.gridwidth = 0;
        addInfoButton.setBackground(Color.getHSBColor(3, 12, 13));
        addInfoButton.addActionListener(addContactInfo);
        panel.add(addInfoButton, grid);
    }
}

```

## StudentForm.java

```

package gui;
import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import domain.Student;
import interfaces.Form;
import java.io.FileReader;
import java.io.IOException;
public class StudentForm extends JFrame{
    private final Student STUDENT;

```

```

public StudentForm(final Form form) {
    this.STUDENT = new Student();
    this.setTitle("STUDENT FORM");
    this.setBounds(380, 290, 780, 350);
    this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    JPanel panel = new JPanel(new GridBagLayout());
    this.setContentPane(panel);
    panel.setBackground(Color.getHSBColor(0.19f, 0.32f, 0.67f));
    GridBagConstraints grid = new GridBagConstraints();
    grid.weightx = 1;
    grid.weighty = 1;
    grid.fill = GridBagConstraints.HORIZONTAL;
    JLabel idLabel = new JLabel("ID: ");
    grid.gridx = 2;
    grid.gridy = 0;
    grid.gridwidth = 1;
    idLabel.setHorizontalAlignment(JLabel.RIGHT);
    panel.add(idLabel, grid);
    JTextField idField = new JTextField();
    grid.gridx = 5;
    grid.gridy = 0;
    grid.gridwidth = 1;
    panel.add(idField, grid);
    JLabel firstNameLabel = new JLabel("First Name: ");
    grid.gridx = 2;
    grid.gridy = 1;
    grid.gridwidth = 1;
    firstNameLabel.setHorizontalAlignment(JLabel.RIGHT);
    panel.add(firstNameLabel, grid);
    JTextField firstNameField = new JTextField();
    grid.gridx = 5;
    grid.gridy = 1;
    grid.gridwidth = 1;
    panel.add(firstNameField, grid);
    JLabel lastNameLabel = new JLabel("Last Name: ");
    grid.gridx = 2;
    grid.gridy = 2;
    grid.gridwidth = 1;
    lastNameLabel.setHorizontalAlignment(JLabel.RIGHT);
    panel.add(lastNameLabel, grid);
    JTextField lastNameField = new JTextField();
    grid.gridx = 5;

```

```

grid.gridy = 2;
grid.gridwidth = 1;
panel.add(lastNameField, grid);
JLabel groupLabel = new JLabel("Group: ");
grid.gridx = 2;
grid.gridy = 3;
grid.gridwidth = 1;
groupLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(groupLabel, grid);
JTextField groupField = new JTextField();
grid.gridx = 5;
grid.gridy = 3;
grid.gridwidth = 1;
panel.add(groupField, grid);
JLabel departmentLabel = new JLabel("Department: ");
grid.gridx = 2;
grid.gridy = 4;
grid.gridwidth = 1;
departmentLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(departmentLabel, grid);
JTextField departmentField = new JTextField();
grid.gridx = 5;
grid.gridy = 4;
grid.gridwidth = 1;
panel.add(departmentField, grid);
JLabel disciplineLabel = new JLabel("Discipline: ");
grid.gridx = 2;
grid.gridy = 5;
grid.gridwidth = 1;
disciplineLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(disciplineLabel, grid);
JTextField disciplineField = new JTextField();
grid.gridx = 5;
grid.gridy = 5;
grid.gridwidth = 1;
panel.add(disciplineField, grid);
JLabel markLabel = new JLabel("Mark: ");
grid.gridx = 2;
grid.gridy = 6;
grid.gridwidth = 1;
markLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(markLabel, grid);

```

```

JTextField markField = new JTextField();
grid.gridx = 5;
grid.gridy = 6;
grid.gridwidth = 1;
panel.add(markField, grid);
JLabel teacherLabel = new JLabel("Name of teacher: ");
grid.gridx = 2;
grid.gridy = 7;
grid.gridwidth = 1;
teacherLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(teacherLabel, grid);
JTextField teacherField = new JTextField();
grid.gridx = 5;
grid.gridy = 7;
grid.gridwidth = 1;
panel.add(teacherField, grid);
ActionListener displayInfo = (ActionEvent event) -> {
    try(FileReader fileReader = new FileReader("students.txt")) {
        StringBuilder strBuilder = new StringBuilder();
        while(fileReader.ready()) {
            strBuilder.append((char)fileReader.read());
        }
        JOptionPane.showMessageDialog(null, strBuilder.toString(), "Reading file",
JOptionPane.PLAIN_MESSAGE);
    }
    catch(IOException e) {
        e.printStackTrace();
    }
};
ActionListener sortById = (ActionEvent event) -> {
    STUDENT.sortById();
    STUDENT.writeFile();
};
ActionListener reverseSortById = (ActionEvent event) -> {
    STUDENT.reverseSortById();
    STUDENT.writeFile();
};
ActionListener sortByLastName = (ActionEvent event) -> {
    STUDENT.sortByLastName();
    STUDENT.writeFile();
};
ActionListener sortByMark = (ActionEvent event) -> {

```



```

        STUDENT.sortByMark();
        STUDENT.writeFile();
    };
    ActionListener backToMenu = (ActionEvent event) -> form.changeStudentForm();
    ActionListener clearTextFields = (ActionEvent event) -> {
        idField.setText("");
        firstNameField.setText("");
        lastNameField.setText("");
        groupField.setText("");
        departmentField.setText("");
        disciplineField.setText("");
        markField.setText("");
        teacherField.setText("");
    };
    ActionListener addContactInfo = (ActionEvent event) -> {
        STUDENT.setId(Integer.parseInt(idField.getText()));
        STUDENT.setFirstName(firstNameField.getText());
        STUDENT.setLastName(lastNameField.getText());
        STUDENT.setGroup(Integer.parseInt(idField.getText()));
        STUDENT.setDepartment(departmentField.getText());
        STUDENT.setDiscipline(disciplineField.getText());
        STUDENT.setMark(Byte.parseByte(markField.getText()));
        STUDENT.setNameTeacher(teacherField.getText());
        STUDENT.fillingArrayList();
        STUDENT.writeFile();
    };
    ImageIcon backToMenuIcon = new ImageIcon("D:\\study\\Practic OOP\\Practic-
OOP\\PR_4\\src\\icons\\back_to_menu.png");
    JButton backToMenuButton = new JButton(backToMenuIcon);
    backToMenuButton.setBackground(Color.getHSBColor(3, 12, 13));
    grid.gridx = 0;
    grid.gridy = 0;
    grid.gridwidth = 1;
    backToMenuButton.addActionListener(backToMenu);
    panel.add(backToMenuButton, grid);
    JButton displayInfoButton = new JButton("Display Info");
    grid.gridx = 0;
    grid.gridy = 1;
    grid.gridwidth = 1;
    displayInfoButton.addActionListener(displayInfo);
    displayInfoButton.setBackground(Color.getHSBColor(3, 12, 13));
    panel.add(displayInfoButton, grid);

```

```

JButton sortByIdButton = new JButton("Sorting By ID");
grid.gridx = 0;
grid.gridy = 2;
grid.gridwidth = 1;
sortByIdButton.addActionListener(sortById);
sortByIdButton.setBackground(Color.getHSBColor(3, 12, 13));
panel.add(sortByIdButton, grid);
JButton reverseSortByIdButton = new JButton("Reverse Sorting by ID");
grid.gridx = 0;
grid.gridy = 3;
grid.gridwidth = 1;
reverseSortByIdButton.addActionListener(reverseSortById);
reverseSortByIdButton.setBackground(Color.getHSBColor(3, 12, 13));
panel.add(reverseSortByIdButton, grid);
JButton sortByLastNameButton = new JButton("Sorting by Last Name");
grid.gridx = 0;
grid.gridy = 4;
grid.gridwidth = 1;
sortByLastNameButton.addActionListener(sortByLastName);
sortByLastNameButton.setBackground(Color.getHSBColor(3, 12, 13));
panel.add(sortByLastNameButton, grid);
JButton sortByMarkButton = new JButton("Descending Sorting by mark");
grid.gridx = 0;
grid.gridy = 5;
grid.gridwidth = 1;
sortByMarkButton.addActionListener(sortByMark);
sortByMarkButton.setBackground(Color.getHSBColor(3, 12, 13));
panel.add(sortByMarkButton, grid);
JButton clearTextFieldsButton = new JButton("Clear text fields");
grid.gridx = 0;
grid.gridy = 8;
grid.gridwidth = 1;
clearTextFieldsButton.addActionListener(clearTextFields);
clearTextFieldsButton.setBackground(Color.getHSBColor(3, 12, 13));
panel.add(clearTextFieldsButton, grid);
JButton addInfoButton = new JButton("Add Order Info");
grid.gridx = 0;
grid.gridy = 10;
grid.gridwidth = 0;
addInfoButton.addActionListener(addContactInfo);
addInfoButton.setBackground(Color.getHSBColor(3, 12, 13));
panel.add(addInfoButton, grid);

```

```

    }
}

```

## OrderForm.java

```

package gui;
import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import domain.Order;
import interfaces.Form;
import java.io.FileReader;
import java.io.IOException;
public class OrderForm extends JFrame{
    private final Order ORDER;
    public OrderForm(final Form form) {
        this.ORDER = new Order();
        this.setTitle("ORDER FORM");
        this.setBounds(400, 290, 750, 350);
        this.setResizable(false);
        this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        JPanel panel = new JPanel(new GridBagLayout());
        panel.setBackground(Color.getHSBColor(0.19f, 0.32f, 0.67f));
        this.setContentPane(panel);
        GridBagConstraints grid = new GridBagConstraints();
        grid.weightx = 1;
        grid.weighty = 1;
        grid.fill = GridBagConstraints.HORIZONTAL;
        JLabel idLabel = new JLabel("ID: ");
        grid.gridx = 2;
        grid.gridy = 0;
        grid.gridwidth = 1;
        idLabel.setHorizontalAlignment(JLabel.RIGHT);
        panel.add(idLabel, grid);
        JTextField idField = new JTextField();
        grid.gridx = 5;
        grid.gridy = 0;
        grid.gridwidth = 1;
        panel.add(idField, grid);
        JLabel nameLabel = new JLabel("Name of product: ");
        grid.gridx = 2;
        grid.gridy = 1;
    }
}

```

```

grid.gridwidth = 1;
nameLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(nameLabel, grid);
JTextField nameField = new JTextField();
grid.gridx = 5;
grid.gridy = 1;
grid.gridwidth = 1;
panel.add(nameField, grid);
JLabel lastNameLabel = new JLabel("Last Name of courier: ");
grid.gridx = 2;
grid.gridy = 2;
grid.gridwidth = 1;
lastNameLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(lastNameLabel, grid);
JTextField lastNameField = new JTextField();
grid.gridx = 5;
grid.gridy = 2;
grid.gridwidth = 1;
panel.add(lastNameField, grid);
JLabel dateTimeLabel = new JLabel("Date and Time: ");
grid.gridx = 2;
grid.gridy = 3;
grid.gridwidth = 1;
dateTimeLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(dateTimeLabel, grid);
JTextField dateTimeField = new JTextField();
grid.gridx = 5;
grid.gridy = 3;
grid.gridwidth = 1;
panel.add(dateTimeField, grid);
JLabel typeLabel = new JLabel("Type of order: ");
grid.gridx = 2;
grid.gridy = 4;
grid.gridwidth = 1;
typeLabel.setHorizontalAlignment(JLabel.RIGHT);
panel.add(typeLabel, grid);
JTextField typeField = new JTextField();
grid.gridx = 5;
grid.gridy = 4;
grid.gridwidth = 1;
panel.add(typeField, grid);
ActionListener displayInfo = (ActionEvent event) -> {

```

```

try(FileReader fileReader = new FileReader("orders.txt")) {
    StringBuilder strBuilder = new StringBuilder();
    while(fileReader.ready()) {
        strBuilder.append((char)fileReader.read());
    }
    JOptionPane.showMessageDialog(null, strBuilder.toString(), "Reading file",
JOptionPane.PLAIN_MESSAGE);
}
catch(IOException e) {
    e.printStackTrace();
}
};

ActionListener backToMenu = (ActionEvent event) -> form.changeOrderForm();
ActionListener sortById = (ActionEvent event) -> {
    ORDER.sortById();
    ORDER.writeFile();
};

ActionListener reverseSortById = (ActionEvent event) -> {
    ORDER.reverseSortById();
    ORDER.writeFile();
};

ActionListener sortByDate = (ActionEvent event) -> {
    ORDER.sortByDate();
    ORDER.writeFile();
};

ActionListener clearTextFields = (ActionEvent event) -> {
    idField.setText("");
    nameField.setText("");
    lastNameField.setText("");
    dateTimeField.setText("");
    typeField.setText("");
};

ActionListener addContactInfo = (ActionEvent event) -> {
    ORDER.setId(Integer.parseInt(idField.getText()));
    ORDER.setName(nameField.getText());
    ORDER.setLastName(lastNameField.getText());
    ORDER.setDateTime(dateTimeField.getText());
    ORDER.setType(Byte.parseByte(typeField.getText()));
    ORDER.fillingArrayList();
    ORDER.writeFile();
};

```

```

        ImageIcon backToMenuIcon = new ImageIcon("D:\\study\\Practic
        OOP\\PR_4\\src\\icons\\back_to_menu.png");

        JButton backToMenuButton = new JButton(backToMenuIcon);
        grid.gridx = 0;
        grid.gridy = 0;
        grid.gridwidth = 1;
        backToMenuButton.setBackground(Color.getHSBColor(3, 12, 13));
        backToMenuButton.addActionListener(backToMenu);
        panel.add(backToMenuButton, grid);

        JButton displayInfoButton = new JButton("Display Info");
        grid.gridx = 0;
        grid.gridy = 1;
        grid.gridwidth = 1;
        displayInfoButton.addActionListener(displayInfo);
        displayInfoButton.setBackground(Color.getHSBColor(3, 12, 13));
        panel.add(displayInfoButton, grid);

        JButton sortByIdButton = new JButton("Sorting by ID");
        grid.gridx = 0;
        grid.gridy = 2;
        grid.gridwidth = 1;
        sortByIdButton.setBackground(Color.getHSBColor(3, 12, 13));
        sortByIdButton.addActionListener(sortById);
        panel.add(sortByIdButton, grid);

        JButton reverseSortByIdButton = new JButton("Reverse Sorting by ID");
        grid.gridx = 0;
        grid.gridy = 3;
        grid.gridwidth = 1;
        reverseSortByIdButton.addActionListener(reverseSortById);
        reverseSortByIdButton.setBackground(Color.getHSBColor(3, 12, 13));
        panel.add(reverseSortByIdButton, grid);

        JButton sortByDateButton = new JButton("Sorting by Date and Time");
        grid.gridx = 0;
        grid.gridy = 4;
        grid.gridwidth = 1;
        sortByDateButton.addActionListener(sortByDate);
        sortByDateButton.setBackground(Color.getHSBColor(3, 12, 13));
        panel.add(sortByDateButton, grid);

        JButton clearTextFieldsButton = new JButton("Clear text fields");
        grid.gridx = 0;
        grid.gridy = 8;
        grid.gridwidth = 1;
        clearTextFieldsButton.setBackground(Color.getHSBColor(3, 12, 13));

```

```

clearTextFieldsButton.addActionListener(clearTextFields);
panel.add(clearTextFieldsButton, grid);
JButton addInfoButton = new JButton("Add Order Info");
grid.gridx = 0;
grid.gridy = 10;
grid.gridwidth = 0;
addInfoButton.setBackground(Color.getHSBColor(3, 12, 13));
addInfoButton.addActionListener(addContactInfo);
panel.add(addInfoButton, grid);
}
}

```

## Form.java

```

package interfaces;

public interface Form {

    public void changeContactForm();
    public void changeStudentForm();
    public void changeOrderForm();
}

```

## Main.java

```

package test;
import javax.swing.*.*;
import gui.*;
import interfaces.Form;

public class Main implements Form{

    private final MenuForm MENU_FORM;
    private final ContactForm CONTACT_FORM;
    private final StudentForm STUDENT_FORM;
    private final OrderForm ORDER_FORM;

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Main();
            }
        });
    }

    public Main() {
        MENU_FORM = new MenuForm(this);
        CONTACT_FORM = new ContactForm(this);
        STUDENT_FORM = new StudentForm(this);
    }
}

```

```
        ORDER_FORM = new OrderForm(this);
        MENU_FORM.setVisible(true);
    }
    @Override
    public void changeContactForm() {
        MENU_FORM.setVisible(!MENU_FORM.isVisible());
        CONTACT_FORM.setVisible(!CONTACT_FORM.isVisible());
    }
    @Override
    public void changeStudentForm() {
        MENU_FORM.setVisible(!MENU_FORM.isVisible());
        STUDENT_FORM.setVisible(!STUDENT_FORM.isVisible());
    }
    @Override
    public void changeOrderForm() {
        MENU_FORM.setVisible(!MENU_FORM.isVisible());
        ORDER_FORM.setVisible(!ORDER_FORM.isVisible());
    }
}
```