Объектно-ориентированное программирование

Задание 1.

Продолжаем работать с проектом с семейным деревом. Реализовать интерфейс Iterable для дерева. Создать методы сортировки списка людей перед выводом, например по имени или по дате рождения (не менее 2). Создать пакетную структуру для проекта

Подсказка № 1

Измените `FamilyTree`, чтобы он принимал тип параметра. Это позволит использовать одно и то же дерево для разных типов объектов.

Подсказка № 2

Интерфейс `FileOperations` также должен быть параметризован, чтобы работать с различными типами данных.

Подсказка № 3

Подумаете о создании класса, который будет управлять взаимодействием с пользователем, предоставляя команды для работы с генеалогическим древом, такими как добавление нового объекта, вывод информации, сохранение и загрузка из файла.

Эталонное решение:

Пакет `model:`

1. **Person.java:**

```
package model;



import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;



public class Person implements Serializable {

    private static final long serialVersionUID = 1L;

    private String name;
```

```
    private int birthYear;
    private Person mother;
    private Person father;
    private List<Person> children;

    public Person(String name, int birthYear) {
        this.name = name;
        this.birthYear = birthYear;
        this.children = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

    public int getBirthYear() {
        return birthYear;
    }

    public void setMother(Person mother) {
        this.mother = mother;
    }

    public void setFather(Person father) {
        this.father = father;
    }
```

```
    public void addChild(Person child) {
        this.children.add(child);
    }

    public List<Person> getChildren() {
        return children;
    }

    public Person getMother() {
        return mother;
    }

    public Person getFather() {
        return father;
    }
}
```

2. **FamilyTree.java:**

```
package model;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

public class FamilyTree<T> implements Serializable, Iterable<T> {
```

```
    private static final long serialVersionUID = 1L;

    private List<T> members;


    public FamilyTree() {

        this.members = new ArrayList<>();

    }


    public void addMember(T member) {

        this.members.add(member);

    }


    public List<T> getMembers() {

        return members;

    }


    @Override

    public Iterator<T> iterator() {

        return members.iterator();

    }


    public void sortByName() {

        Collections.sort(members, (p1, p2) ->
p1.toString().compareTo(p2.toString()));

    }


    public void sortByBirthYear() {

        if (members.get(0) instanceof Person) {
```

```
            Collections.sort(members, (p1, p2) ->
Integer.compare(((Person) p1).getBirthYear(), ((Person)
p2).getBirthYear()));

        }

    }

}
```

Пакет `service`:

1. **FileOperations.java:**

```
package service;



import model.FamilyTree;



import java.io.IOException;



public interface FileOperations<T> {

    void saveToFile(FamilyTree<T> familyTree, String fileName)
throws IOException;

    FamilyTree<T> loadFromFile(String fileName) throws IOException,
ClassNotFoundException;

}
```

2. **FileOperationsImpl.java:**

```
package service;



import model.FamilyTree;



import java.io.*;
```

```
public class FileOperationsImpl<T> implements FileOperations<T> {



    @Override

    public void saveToFile(FamilyTree<T> familyTree, String
fileName) throws IOException {

        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(fileName))) {

            oos.writeObject(familyTree);

        }

    }



    @Override

    public FamilyTree<T> loadFromFile(String fileName) throws
IOException, ClassNotFoundException {

        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(fileName))) {

            return (FamilyTree<T>) ois.readObject();

        }

    }

}
```

Пакет `main`:

1. **Main.java:**

```
package main;



import model.FamilyTree;

import model.Person;

import service.FileOperations;

import service.FileOperationsImpl;



```

```
import java.io.IOException;


public class Main {
    public static void main(String[] args) {
        FamilyTree<Person> familyTree = new FamilyTree<>();


        // Создаем людей
        Person john = new Person("John", 1950);
        Person mary = new Person("Mary", 1955);
        Person susan = new Person("Susan", 1980);


        // Устанавливаем родительские связи
        susan.setMother(mary);
        susan.setFather(john);
        john.addChild(susan);
        mary.addChild(susan);


        // Добавляем людей в древо
        familyTree.addMember(john);
        familyTree.addMember(mary);
        familyTree.addMember(susan);


        // Сортируем по имени
        System.out.println("Сортировка по имени:");
        familyTree.sortByName();
        for (Person person : familyTree) {
```

```
            System.out.println(person.getName() + " - " +
person.getBirthYear());

        }



        // Сортируем по дате рождения

        System.out.println("\nСортировка по дате рождения:");

        familyTree.sortByBirthYear();

        for (Person person : familyTree) {

            System.out.println(person.getName() + " - " +
person.getBirthYear());

        }



        // Сохраняем генеалогическое древо в файл

        FileOperations<Person> fileOps = new FileOperationsImpl<>();

        try {

            fileOps.saveToFile(familyTree, "familyTree.dat");

            System.out.println("\nFamily tree saved to file.");

        } catch (IOException e) {

            e.printStackTrace();

        }



        // Загружаем генеалогическое древо из файла

        FamilyTree<Person> loadedFamilyTree = null;

        try {

            loadedFamilyTree =
fileOps.loadFromFile("familyTree.dat");

            System.out.println("Family tree loaded from file.");

        } catch (IOException | ClassNotFoundException e) {
```

```
            e.printStackTrace();
        }

        // Проверяем, что древо загрузилось правильно
        if (loadedFamilyTree != null) {
            System.out.println("\nLoaded persons:");
            for (Person person : loadedFamilyTree) {
                System.out.println(person.getName() + ", born in " +
person.getBirthYear());
            }
        }
    }
}
```

Пример `CommandManager`:

```
package main;

import model.FamilyTree;
import model.Person;

import java.util.Scanner;

public class CommandManager {
    private FamilyTree<Person> familyTree;
    private Scanner scanner;

    public CommandManager(FamilyTree<Person> familyTree) {
```

```
        this.familyTree = familyTree;
        this.scanner = new Scanner(System.in);
    }

    public void start() {
        while (true) {
            System.out.println("Введите команду (add, list,
sortByName, sortByBirthYear, save, load, exit):");
            String command = scanner.nextLine();

            switch (command) {
                case "add":
                    addPerson();
                    break;
                case "list":
                    listPeople();
                    break;
                case "sortByName":
                    familyTree.sortByName();
                    listPeople();
                    break;
                case "sortByBirthYear":
                    familyTree.sortByBirthYear();
                    listPeople();
                    break;
                case "save":
                    // implement save logic
```

```
                    break;
                case "load":
                    // implement load logic
                    break;
                case "exit":
                    return;
                default:
                    System.out.println("Неизвестная команда");
            }
        }
    }

    private void addPerson() {
        System.out.println("Введите имя:");
        String name = scanner.nextLine();
        System.out.println("Введите год рождения:");
        int birthYear = Integer.parseInt(scanner.nextLine());

        Person person = new Person(name, birthYear);
        familyTree.addMember(person);
        System.out.println("Человек добавлен в дерево.");
    }

    private void listPeople() {
        for (Person person : familyTree) {
            System.out.println(person.getName() + ", родился в " +
person.getBirthYear());
```

```
            }

        }

}
```