Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота №11

із дисципліни «Структури даних та алгоритми»

Тема: «Колекції. Множина TreeSet»

Виконали:
Студенти групи ІА-24
Сіденко Д.Д.
Філімонов Є.А.
Іскандаров Е. Е. огли
Яблонський Д.Б.

Перевірив:
Колеснік Валерій Микола

Київ-2022

Хід роботи:
1. Ознайомитись з javadoc для наступних інтерфейсів та класів:
- Set
- TreeSet
- Comparable
- Comparator
- SortedSet
- NavigableSet

2. Виконати завдання попередньої лабораторної роботи, замінивши списки List (ArrayList та LinkedList) на множини Set (TreeSet). При цьому необхідно щонайменше один раз використати Comparable та щонайменше один раз – Comparator.

3	Класи: База даних про кіно (Список усіх фільмів, список усіх акторів) Фільм (Назва, Список акторів які знялись у фільмі) Актор (Ім'я, Список фільмів в яких зіграв актор) Задача: 1) Визначити, чи є актор, який не зіграв в жодному фільмі 2) Скласти список акторів, з якими коли-небудь в одному фільмі грав заданий актор 3) Знайти фільм з найбільшою кількістю акторів

Лістинг програми:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Results();
        task1();
        task3();
        task2("Robert Downey Jr.");
    }
    static void task1(){
        try{
        System.out.println("The actor without career: "+
Database.getAllActors().first());
    }catch(NullPointerException e){
            System.out.println("Database is empty");
        }
}
    static void task2(String actorFullName){
        Actor tempActor = null;
        TreeSet<Actor> actorsList = new TreeSet<Actor>();
        for (Actor actor : Database.getAllActors()) {
            if (actor.getFullName().equals(actorFullName)) {
                tempActor = actor;
                break;
            }}

        try{
        for (Movie movie : Database.getAllFilms()) {
            TreeSet<Actor> cast = movie.getCast();

            if (cast.contains(tempActor))
```

```
                actorsList.addAll(cast);
                actorsList.remove(tempActor);
        }
        if (actorsList.size()==0){
            System.out.println(actorFullName+" hasn`t colleagues");
        }
        else{
        System.out.print("Colleagues of "+tempActor+" : ");
        for(Actor i:actorsList) {
            System.out.print(i + ", ");

        }}}catch(NullPointerException e){
            System.out.println("Can`t find the actor in database!");
        }

    }
    static void task3(){
        try{
            System.out.println("The film with the biggest cast: " +
Database.getAllFilms().last());
        }catch(NullPointerException e){
            System.out.println("Database is empty");
        }

    }
    static void Results(){
        Movie film1 = new Movie("Avengers");
        Movie film2 = new Movie("Iron Man");
        Movie film3 = new Movie("Black Widow");

        Actor actor1 = new Actor("Robert", "Downey Jr.");
        Actor actor2 = new Actor("Chris", "Evans");
        Actor actor3 = new Actor("Mark", "Ruffalo");
        Actor actor4 = new Actor("Scarlett", "Johansson");
        Actor actor5 = new Actor("Gwyneth", "Paltrow");
        Actor actor6 = new Actor("Jeff", "Bridges");
        Actor actor7 = new Actor("Florence", "Pugh");
        Actor actor8 = new Actor("Rachel", "Weisz");
        Actor actor9 = new Actor("Angelina", "Jolie");

        film1.addAllActors(List.of(actor1,actor2,actor3,actor4));
        film2.addAllActors(List.of(actor1,actor5,actor6));
        film3.addAllActors(List.of(actor4,actor7,actor8));

        Database.addAllFilm(List.of(film2,film1, film3));

        Database.addAllActor(List.of(actor1, actor2, actor3, actor4, actor5,
actor6, actor7, actor8, actor9));

        actor1.addAllFilms(List.of(film2,film1));
        actor2.addAllFilms(List.of(film1));
        actor3.addAllFilms(List.of(film1));
        actor4.addAllFilms(List.of(film3,film1));
        actor5.addAllFilms(List.of(film2));
        actor6.addAllFilms(List.of(film2));
        actor7.addAllFilms(List.of(film3));
        actor8.addAllFilms(List.of(film3));

    }
}
```

```
import java.util.Collection;
import java.util.Comparator;
import java.util.Set;
```

```java
import java.util.TreeSet;

public class Actor implements Comparable<Actor> {
    String name;
    String surname;
    static TreeSet<Movie> career = new TreeSet<Movie>(new
Movie.MovieComparator());

    public Actor(String name, String surname) {
        this.name = name;
        this.surname = surname;

    }

    public static TreeSet<Movie> getCareer() {
        return career;
    }

    public void addAllFilms(Collection<Movie> movies) {
        this.career.addAll(movies);
    }
    public String getFullName() {
        return name+" "+surname;
    }
    @Override
    public String toString() {
        return  name + " " + surname;
    }

    @Override
    public int compareTo(Actor other) {
        int thisCareerSize = this.career.size();
        int otherCareerSize = other.career.size();

        if (thisCareerSize > otherCareerSize) {
            return 1;
        } else if (thisCareerSize < otherCareerSize) {
            return -1;
        } else {
            return this.name.compareTo(other.name);
        }
    }
}

import java.util.Collection;
import java.util.Set;
import java.util.TreeSet;

public class Database {
    static TreeSet<Actor> allActors = new TreeSet<>();
    static TreeSet<Movie> allFilms = new TreeSet<Movie>(new
Movie.MovieComparator());

    public static void addAllFilm(Collection<Movie> film) {
        allFilms.addAll(film);
    }
    public static void addAllActor(Collection<Actor> actor) {
        allActors.addAll(actor);
    }

    public static TreeSet<Actor> getAllActors() {
        return allActors;
    }
```

```
    public static TreeSet<Movie> getAllFilms() {
        return allFilms;
    }

}
import java.util.Collection;

import java.util.Comparator;
import java.util.TreeSet;

public class Movie  {
    String name;
    TreeSet<Actor> cast = new TreeSet<Actor>();

    public Movie(String name) {
        this.name = name;

    }
    public void addAllActors(Collection<Actor> actor) {
        this.cast.addAll(actor);
    }

    public TreeSet<Actor> getCast() {
        return cast;
    }

    @Override
    public String toString() {
        return  name;
    }


    public static class MovieComparator implements Comparator<Movie> {
    @Override
    public int compare(Movie movie1, Movie movie2) {
        int thisCastSize = movie1.getCast().size();
        int otherCastSize = movie2.getCast().size();

        if (thisCastSize > otherCastSize) {
            return 1;
        } else if (thisCastSize < otherCastSize) {
            return -1;
        } else {
            return movie1.toString().compareTo(movie2.toString());
        }
    }
}}
```

Результат виконання:

```
The actor without career: Angelina Jolie
The film with the biggest cast: Avengers
Colleagues of Robert Downey Jr. : Chris Evans, Gwyneth Paltrow, Jeff Bridges, Mark Ruffalo, Scarlett Johansson,
```

3. Відповісти на контрольні питання.

Висновок: В даній лабораторній роботі ми ознайомились з множиною TreeSet та методами використання її при виконанні поставлених задач. Порівняли використання множин зі списками та виявили переваги та недоліки цього способу.