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 №12

із дисципліни «*Основи програмування*»

Тема: «Колекції. Множина HashSet. Асоціативні масиви Map»

Виконав:
Студент групи ІА-24
Красношапка Р.О.
Бакалець А.І.
Орловська А.В.

Перевірив:
Колеснік В.М.

Київ-2023

Хід роботи:

1. Ознайомитись з javadoc для наступних інтерфейсів,класів та методів:
   - Set
   - HashSet
   - Object.equals(), Object.hashCode()
   - Map
   - HashMap

2. Виконати завдання лабораторної роботи №10, замінивши списки List (ArrayList та LinkedList) на множини Set (HashSet). Проаналізувати предметну область та на власний розсуд додати функціональність, для реалізації якої використати Map (TreeMap або HashMap).
3. Відповісти на контрольні питання.

Код програми:

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        printResult();
    }

    public static void printResult() {
        Film film1 = new Film("Deadpool");
        Film film2 = new Film("Focus");
        Film film3 = new Film("The Pursuit of Happyness");
        Actor actor1 = new Actor("Ryan Reynolds");
        Actor actor2 = new Actor("Will Smith");
        Actor actor3 = new Actor("Syre Smith");
        Actor actor4 = new Actor("Brad Pitt");

        Database.addFilm(film1);
        Database.addFilm(film2);
        Database.addFilm(film3);
        Database.addActor(actor1);
        Database.addActor(actor2);
        Database.addActor(actor3);
        Database.addActor(actor4);

        film1.addActor(actor1);
        film2.addActor(actor2);
        film3.addActor(actor2);
        film3.addActor(actor3);
        actor1.addFilm(film1);
        actor2.addFilm(film2);
        actor2.addFilm(film3);
        actor3.addFilm(film3);

        System.out.println(getCoActors(actor2));
        System.out.println("The largest number of actors in the film - " + mostActors());
        notFilmed();
    }

    private static String mostActors() {
        int countActor = 0;
        StringBuilder mostActored = new StringBuilder();
        Set<Film> allfilms = Database.getFilms();
        for (Film film : allfilms) {
            if (countActor < film.getActors().size()) {
                countActor = film.getActors().size();
                mostActored = new StringBuilder(film.getName());
            } else if (countActor == film.getActors().size() && film.getActors().size()
!= 0) {
                mostActored.insert(0, film.getName() + " and ");
```

```
            }
        }
        return mostActored.toString();
    }

    private static void notFilmed() {
        Set<Actor> allActors = Database.getActors();
        for (Actor actor : allActors) {
            if (actor.getFilms().isEmpty()) {
                System.out.println(actor.getName() + " - didn't act in any film");
            }
        }
    }
    public static String getCoActors(Actor actor) {
        HashSet<Actor> coActors = new HashSet<>();
        HashSet<Film> films = Database.getFilms();

        Iterator<Film> filmIter = films.iterator();
        while (filmIter.hasNext()) {
            Film film = filmIter.next();
            if (film.getActors().contains(actor)) {
                Set<Actor> actors = film.getActors();
                Iterator<Actor> actorIter = actors.iterator();
                while (actorIter.hasNext()) {
                    Actor coActor = actorIter.next();
                    if (!coActor.equals(actor) && !coActors.contains(coActor)) {
                        coActors.add(coActor);
                    }
                }
            }
        }
        StringBuilder sb = new StringBuilder();
        sb.append("Co-actors of ").append(actor.getName()).append(": ");

        Iterator<Actor> coActorIter = coActors.iterator();
        while (coActorIter.hasNext()) {
            Actor coActor = coActorIter.next();
            sb.append(coActor.getName()).append(", ");
        }
        if (coActors.size() > 0) {
            sb.delete(sb.length() - 2, sb.length());
        }
        return sb.toString();
    }

}

import java.util.HashSet;
public class Database {
    private static final HashSet<Film> films = new HashSet<>();
    private static final HashSet<Actor> actors = new HashSet<>();

    public static void addFilm(Film film) {
        films.add(film);
    }

    public static void addActor(Actor actor) {
        actors.add(actor);
    }

    public static HashSet<Film> getFilms() {
        return films;
    }

    public static HashSet<Actor> getActors() {
        return actors;
    }
```

```
}



import java.util.HashSet;
import java.util.Objects;

class Actor {
    private final String name;
    private final HashSet<Film> films;

    public Actor(String name) {
        this.name = name;
        this.films = new HashSet<>();
    }

    public String getName() {
        return name;
    }

    public HashSet<Film> getFilms() {
        return films;
    }

    public void addFilm(Film film) {
        films.add(film);
    }


    public boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }
        if (obj == null || getClass() != obj.getClass()) {
            return false;
        }
        Actor actor = (Actor) obj;
        return name.equals(actor.name);
    }


    @Override
    public int hashCode(){
        return Objects.hash(name);
    }

}
import java.util.HashSet;
import java.util.Objects;

class Film  {
    private final String name;
    private final HashSet<Actor> actors;

    public Film(String name) {
        this.name = name;
        this.actors = new HashSet<>();
    }

    public String getName() {
        return name;
    }

    public HashSet<Actor> getActors() {
        return actors;
    }
    public boolean equals(Object obj) {
```

```
        if (obj == this) {
            return true;
        }
        if (obj == null || getClass() != obj.getClass()) {
            return false;
        }
        Film Actor = (Film)obj;
        return name.equals(Actor.name);
    }
    public void addActor(Actor actor) {
        actors.add(actor);
    }

    @Override
    public int hashCode(){
        return Objects.hash(name);
    }
}
```

Висновок: під час виконання лабораторної роботи, ознайомилися з javadoc для інтерфейсів та класів: Set, TreeSet, Object.equals(), Object.hashCode(). Виконали завдання попередньої лабораторної роботи, замінивши списки List (ArrayList та LinkedList) на множини Set (HashSet.Також відповіли на контрольні питання.