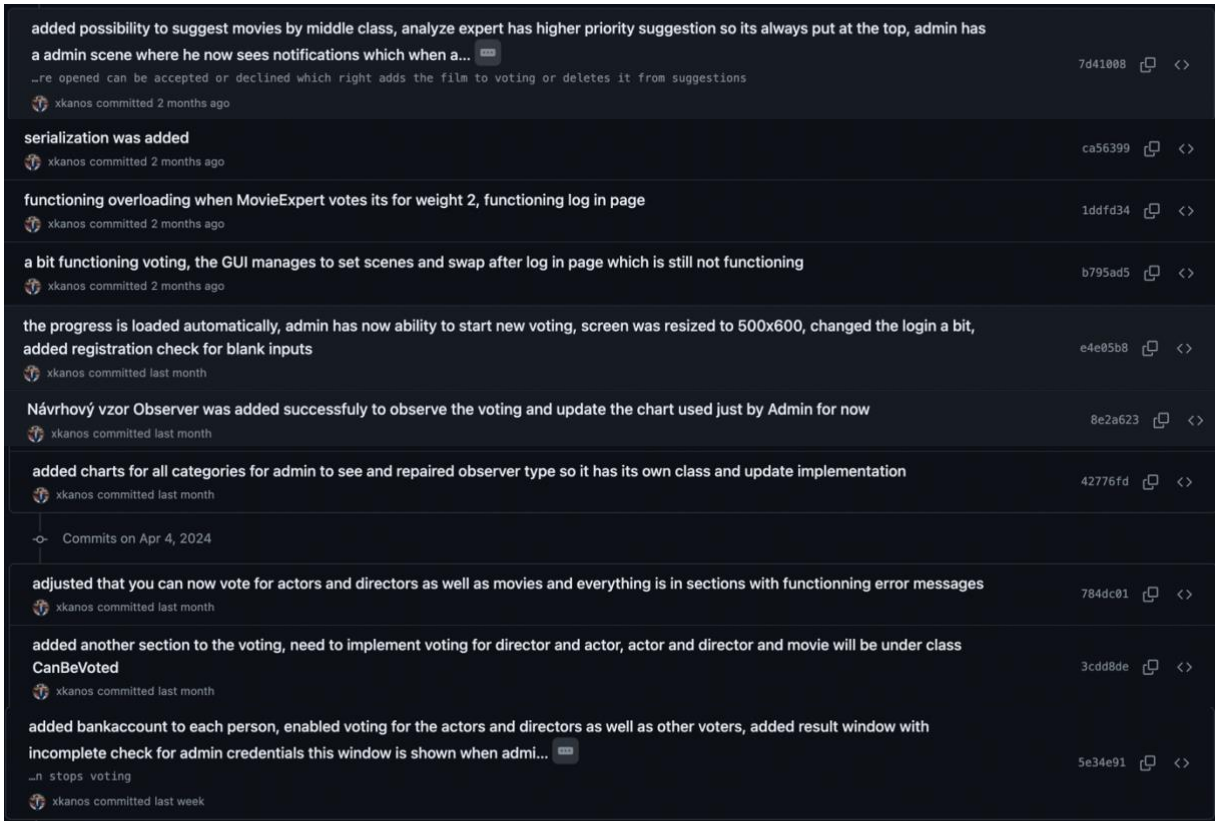
Zdenko Kanoš
Názov projektu: CineVote

OBSAH – Dokumentácia CineVote

Zdenko Kanoš
Názov projektu: CineVote

Zámer projektu:
Zameraním môjho projektu je vytvoriť systém hlasovania o najlepší film, najlepšieho/najlepšiu režiséra/režisérku alebo herca/herečku roka. Účastníci hlasovania budú rozdelení do troch skupín: **Admin**, **Stredná trieda**, **Nižšia trieda**. Admin bude mať možnosť prijímať návrhy nominácií filmov a zároveň nominovať filmy s možnosťou hlasovať práve raz v každej kategórii, zároveň by ako jediný videl priebeh hlasovania a predbežné výsledky. Stredná trieda bude mať možnosť navrhnúť nominácie, ktoré musia byť schválené Adminom a taktiež môžu hlasovať iba raz v každej kategórii, táto skupina bude ešte rozšírená o filmových expertov, ktorý budú mať váhu hlasu 2 a skupinu analytických expertov, ktorých nominácie budú mať vyššiu prioritu pri schvaľovaní. Nižšia trieda má len možnosť hlasovania a nahliadnutie výslednej štatistiky po skončení hlasovania tak ako aj stredná trieda. Pri registrácii bude účastníkovi pridelená vždy rola nižšej triedy (tzn. Vyššie role sú definované pred hlasovaním.) Zároveň, počas hlasovania bude mať každý účastník možnosť vlastniť bankový účet s pridelenou sumou 100€, z ktorého môžu darovať peniaze na účty hercov alebo režisérov.

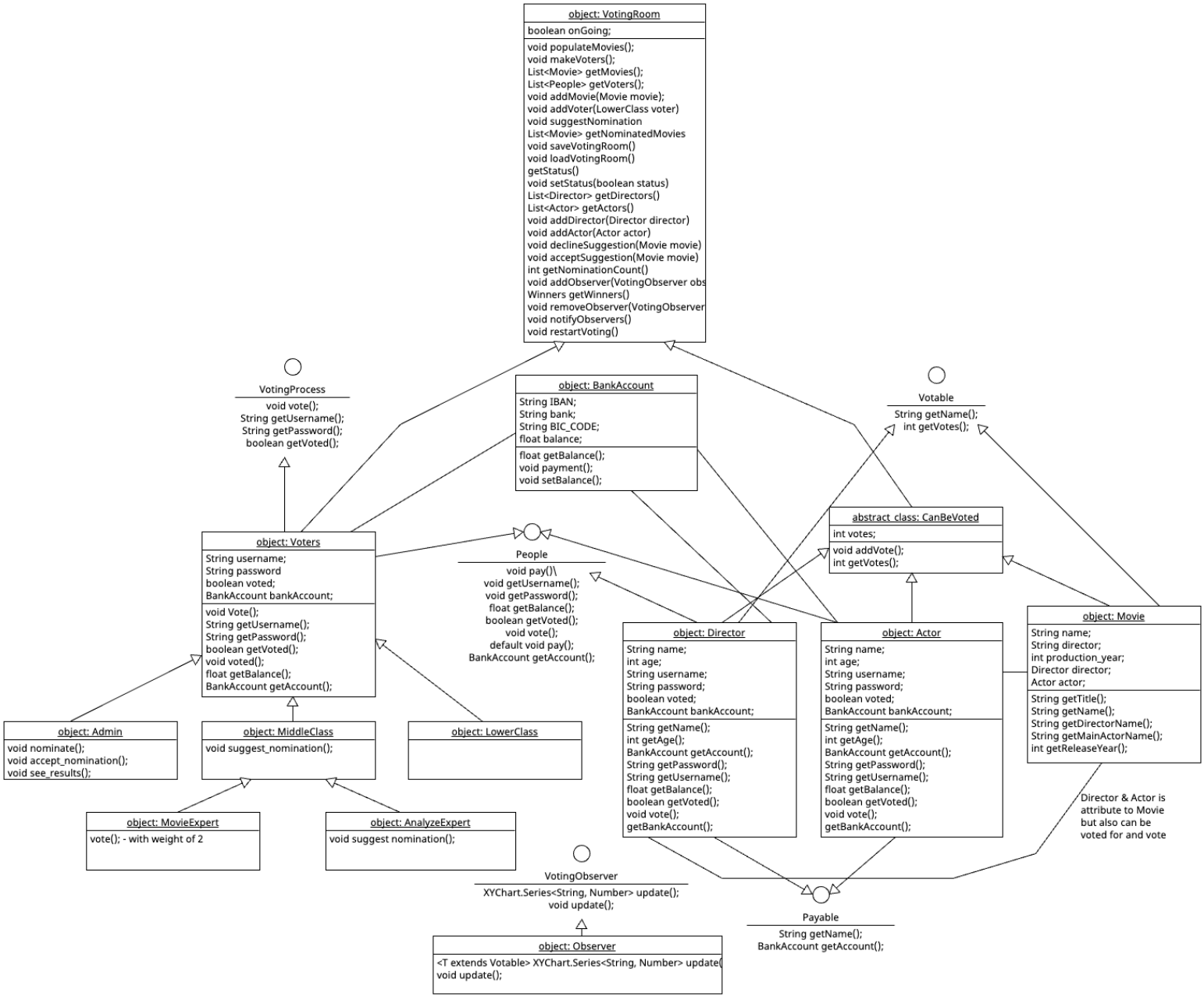
Zoznam hlavných verzií programu:



*popis commitov dostatočne vystihuje zmeny v danom commite

Zdenko Kanoš
Názov projektu: CineVote

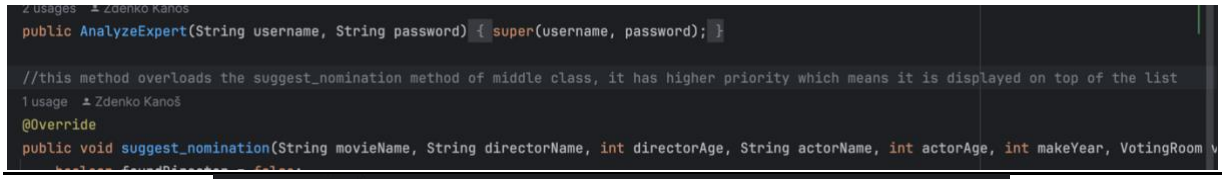
Diagram tried:

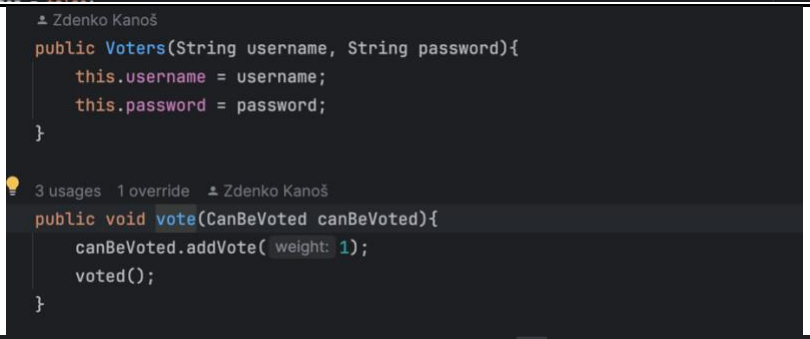


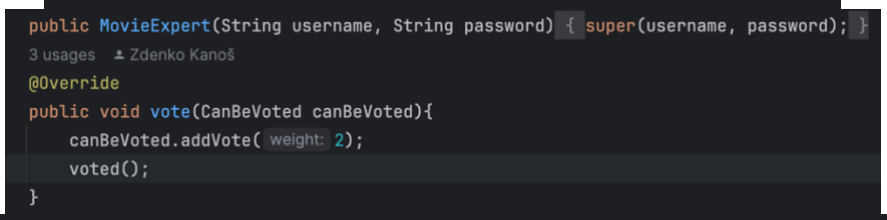
Zdenko Kanoš
Názov projektu: CineVote

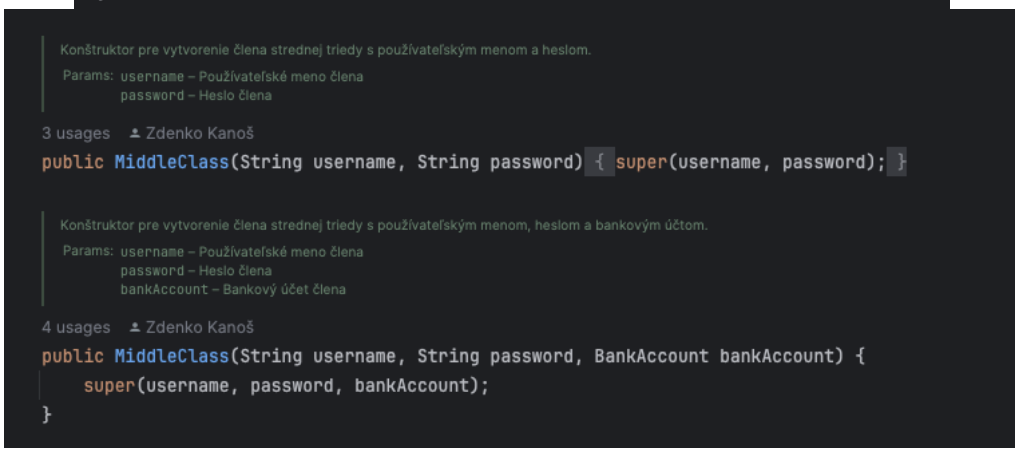
HLAVNÉ KRITÉRIÁ:

Prekonávanie metód a Polymorfizmus – vyskytuje sa pri triedach MiddleClass, MovieExpert a Analyze expert, pri metóde suggest nomination alebo vote. Zároveň aj v triedach patriacich pod Voters sa vyskytuje polymorfizmus pri konštruktoroch, kde sú dva konštruktory, jeden pre voliča s účtom a jeden bez účtu. Tiež aj v druhej hierarchii sú dva konštruktory pre Directora a Actora, (môžu byť s účtom alebo bez).

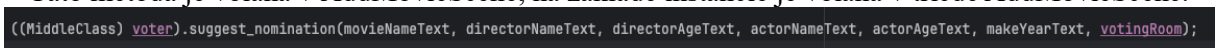




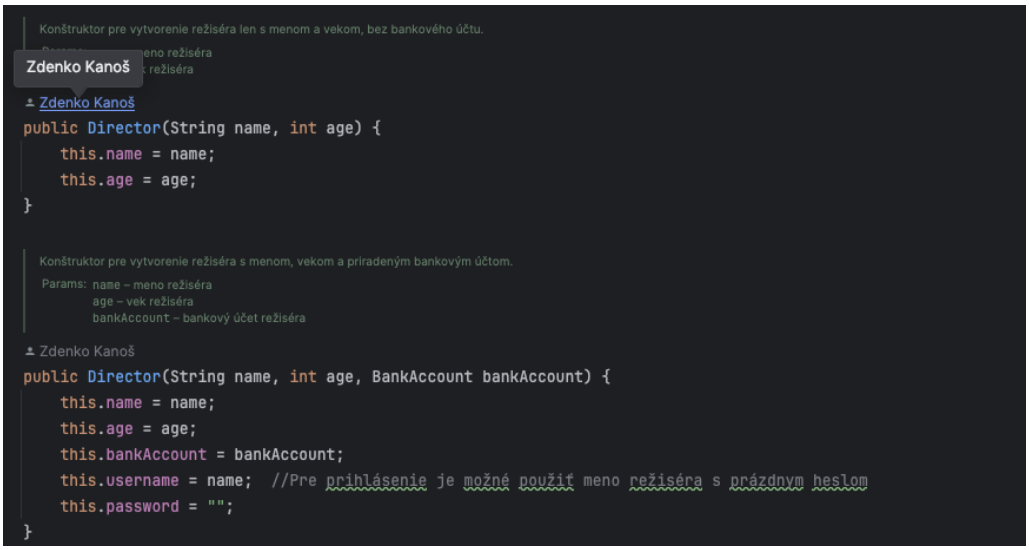




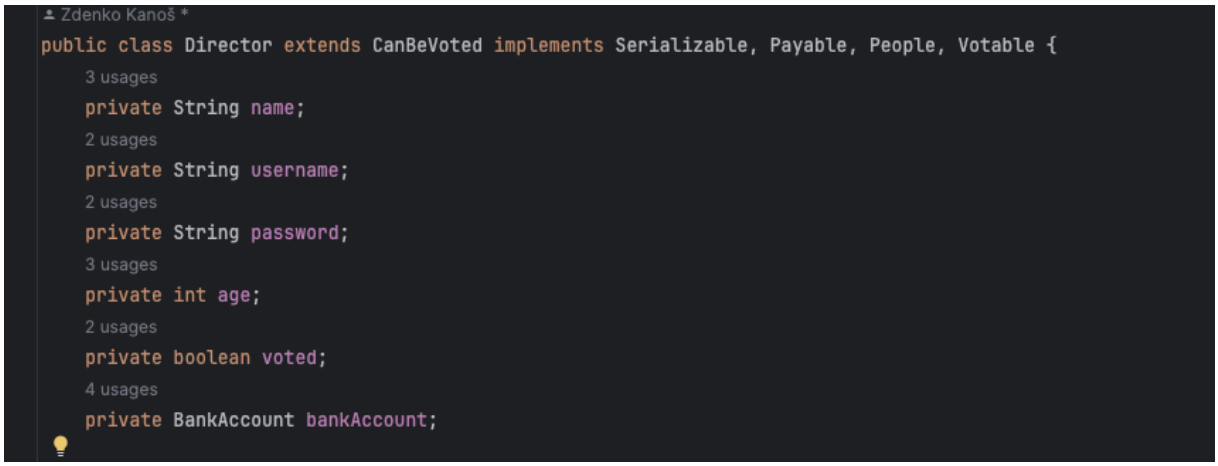
Táto metóda je volaná v AddMovieScene, na základe inštancie je volaná v triede AddMovieScene.



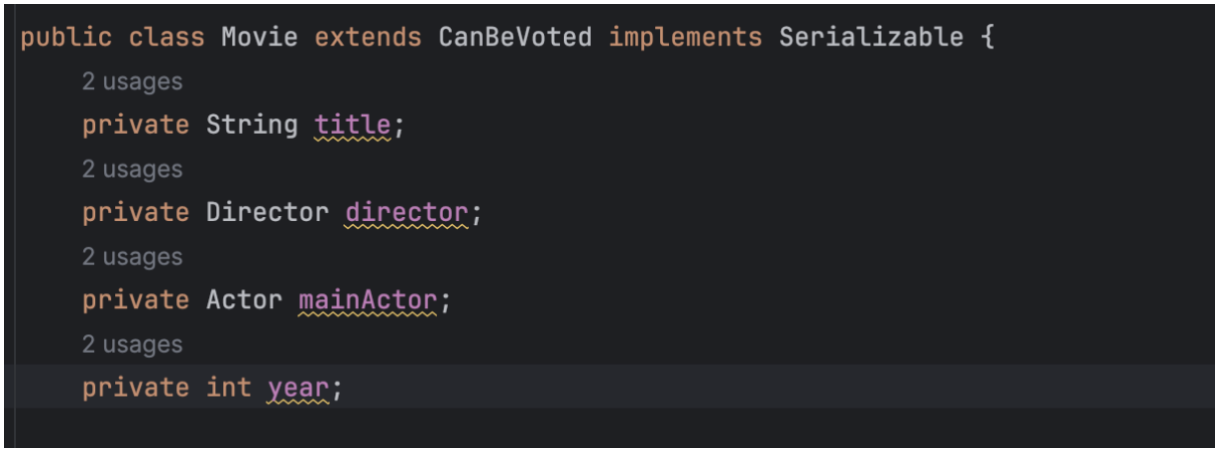
Zdenko Kanoš
Názov projektu: CineVote



Agregácia:
Vyskytuje sa v obidvoch hierarchiách, v triede Director a Actor, ako bankový účet:



Taktiež sa **Agregácia** vyskytuje v druhej hierarchii v triede Movie kde movie má ako atribúty triedy Director a Actor



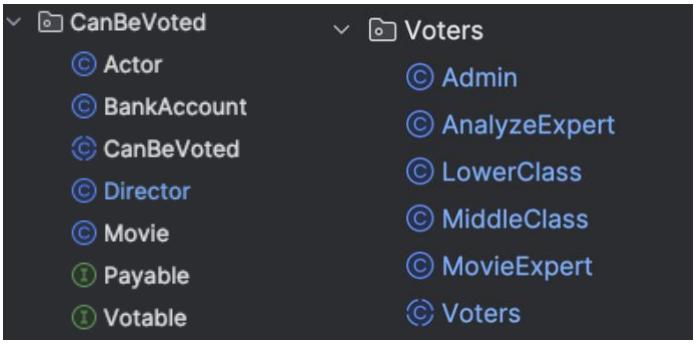
Taktiež aj v triede Account item, ktorá slúži na udržiavanie objektu Payable a zároveň aj reťazec s menom herca alebo režiséra.

```
Trieda AccountItem slúži ako dátový kontajner pre účty, ktoré môžu byť zobrazené a vybrané v rozbaľovacom zozname (ComboBox).
Umožňuje spojenie zobrazovacieho reťazca s objektom implementujúcim rozhranie Payable.

5 usages   Zdenko Kanoš
public class AccountItem {
    3 usages
    private String displayString;
    2 usages
    private Payable payable;
```

Zapuzdrenie – vyskytuje sa pri každej triede v obidvoch hierarchiách viď. príklad: (zároveň aj s potrebnými gettermi pre prístup k premenným)

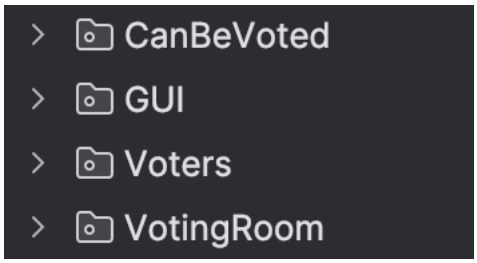
```
public class Movie extends CanBeVoted implements Serializable {
    2 usages
    private String title;
    2 usages
    private Director director;
    2 usages
    private Actor mainActor;
    2 usages
    private int year;

    Zdenko Kanoš
    public Movie(String title, Director director, Actor mainActor, int year) {
        this.title = title;
        this.director = director;
        this.mainActor = mainActor;
        this.year = year;
    }

    5 usages   Zdenko Kanoš
    public String getTitle() { return title; }

    1 usage   Zdenko Kanoš
    public String getDirectorName() { return director.getName(); }

    1 usage   Zdenko Kanoš
    public String getMainActorName() { return mainActor.getName(); }

    1 usage   Zdenko Kanoš
    public int getReleaseYear() { return year; }
}
```

Dve Hierarchie:
V aplikácii sú vybudované dve hierarchie zobrazené na diagrame tried s abstraktnou triedou nad nimi, zároveň Actor, Director a Voters sú prepojení pomocou rozhrania People aby sa mohli všetci ľudia prihlásiť a hlasovať. Payable je rozhranie pre herca a režiséra t.j. pre tých, ktorý môžu dostávať platby.

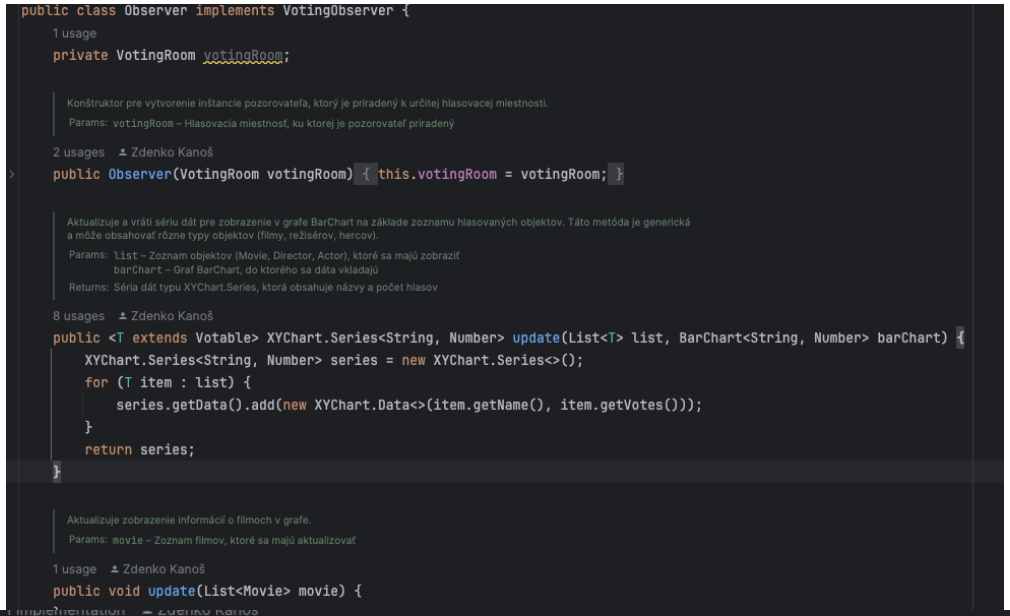


Zdenko Kanoš
Názov projektu: CineVote

Kód vhodne organizovaný do balíkov:



ĎALŠIE KRITÉRIÁ:
Návrhový vzor – OBSERVER – využíva sa na aktualizovanie grafov v AdminScene, tieto grafy zobrazujú priebežné výsledky volenia, ktoré môže vidieť len Admin a aj v results kde zobrazujú tieto grafy výsledky volieb, metóda notifyObservers sa nachádza v triede VotingRoom, kde je aj volaná v prípade pridania nového filmu, herca alebo režiséra.



```
public class Observer implements VotingObserver {
    1 usage
    private VotingRoom votingRoom;

    Konštruktor pre vytvorenie inštancie pozorovateľa, ktorý je priradený k určitej hlasovacej miestnosti.
    Params: votingRoom – Hlasovacia miestnosť, ku ktorej je pozorovateľ priradený
    2 usages  Zdenko Kanoš
    public Observer(VotingRoom votingRoom) { this.votingRoom = votingRoom; }

    Aktualizuje a vráti sériu dát pre zobrazenie v grafe BarChart na základe zoznamu hlasovaných objektov. Táto metóda je generická
    a môže obsahovať rôzne typy objektov (filmy, režisérov, hercov).
    Params: list – Zoznam objektov (Movie, Director, Actor), ktoré sa majú zobraziť
            barChart – Graf BarChart, do ktorého sa dáta vkladajú
    Returns: Séria dát typu XYChart.Series, ktorá obsahuje názvy a počet hlasov
    8 usages  Zdenko Kanoš
    public <T extends Votable> XYChart.Series<String, Number> update(List<T> list, BarChart<String, Number> barChart) {
        XYChart.Series<String, Number> series = new XYChart.Series<>();
        for (T item : list) {
            series.getData().add(new XYChart.Data<>(item.getName(), item.getVotes()));
        }
        return series;
    }

    Aktualizuje zobrazenie informácií o filmoch v grafe.
    Params: movie – Zoznam filmov, ktoré sa majú aktualizovať
    1 usage  Zdenko Kanoš
    public void update(List<Movie> movie) {
```

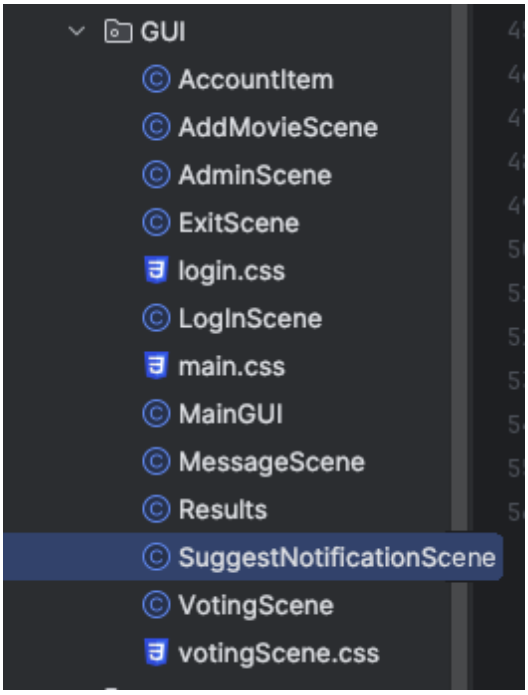
```
public interface VotingObserver {

    Aktualizuje vizuálne zobrazenie na základe poskytnutého zoznamu filmov. Táto metóda je určená na priame aktualizácie bez
    grafickej reprezentácie.
    Params: movies – Zoznam filmov, ktoré sa majú aktualizovať.
    1 usage  1 implementation  Zdenko Kanoš
    void update(List<Movie> movies);

    Generická metóda na aktualizáciu a vytvorenie sérií dát pre zobrazenie v grafickom rozhraní typu BarChart, založené na
    poskytnutom zozname objektov. Táto metóda umožňuje vizualizáciu rôznych typov dát v jednom grafe.
    Params: list – Zoznam objektov, ktoré sa majú zobraziť (môžu to byť filmy, režiséri, herci atď.)
            barChart – Grafický komponent BarChart, kde sa majú dáta zobraziť
    Returns: Séria dát pre BarChart, obsahujúca názvy a príslušné počty hlasov alebo iné numerické hodnoty
    8 usages  1 implementation  new *
    <T extends Votable> XYChart.Series<String, Number> update(List<T> list, BarChart<String, Number> barChart);
}
```

V triede AdminScene:

```
//Observer
votingRoomObserver = new Observer(votingRoom);
votingRoom.addObserver(votingRoomObserver);

barChartMovies.getData().add(votingRoomObserver.update(votingRoom.getMovies(), barChartMovies));
```

Zdenko Kanoš
Názov projektu: CineVote

GUI – v aplikácii je zapracované grafické používateľské prostredie, ktoré je celé napísané manuálne pomocou JavaFX



Explicitné použitie RTTI – je použité pri prihlasovaní v triede LoginScene pomocou instance of na zistenie akého typu je daný užívateľ a podľa toho je prispôsobené hlasovanie. Použité je na viacerých miestach, ešte aj napríklad v AddMovieScene na zistenie či je užívateľ Admin alebo MiddleClass.

```
if (voter instanceof Admin)
{
    ChoiceScene choiceScene = new ChoiceScene(primaryStage, votingRoom, voter);
    primaryStage.setScene(choiceScene);
    break;
} else if (voter instanceof MiddleClass)
{
    ChoiceScene choiceScene = new ChoiceScene(primaryStage, votingRoom, voter);
    primaryStage.setScene(choiceScene);
    break;
} else
{
    if (voter.getBankAccount() != null)
    {
        MessageScene alreadyVotedScene = new MessageScene( message: "You have already voted!", voter, votingRoom);
        primaryStage.setScene(alreadyVotedScene);
    } else
    {
        MessageScene alreadyVotedScene = new MessageScene( message: "You have already voted!", votingRoom);
        primaryStage.setScene(alreadyVotedScene);
    }
    break;
}
```

```
if (voter instanceof Admin)
{
    ((Admin) voter).nominate(movieNameText,
} else if (voter instanceof MiddleClass)
{
    ((MiddleClass) voter).suggest_nominatio
}
```

Ošetrenie mimoriadnych stavov – nachádza sa pri prihlasovaní ak sú obe polia prázdne vypíše sa hláška, že sa musia tieto polia vyplniť

```
public class BlankInputException extends Exception {

    Konštruktor pre BlankInputException, ktorý prijíma správu popisujúcu chybu. Táto správa sa potom môže zobraziť užívateľovi, aby
    vedel, že musí vyplniť všetky požadované polia.
    Params: message – Správa, ktorá sa zobrazí, keď je výnimka zachytená a spracovaná.

    1 usage   Zdenko Kanoš
    public BlankInputException(String message) { super(message); }
}
```

```
try
{
    if (username.isEmpty() || password.isEmpty())
    {
        throw new BlankInputException("Username and password fields cannot be blank.");
    } else
    {
        for (People voter : votingRoom.getVoters())
        {
            if (username.equals(voter.getUsername()))
            {
```

```
} catch (BlankInputException ex)
{
    errorMessageLabel.setText(ex.getMessage());
}
```

Serializácia – v aplikácii sa využíva na uloženie registrovaných užívateľov, možnosť pokračovania v hlasovaní po znovu spustení aplikácie. Ukladajú sa objekty filmov, hercov, režisérov, navrhnutých filmov, používateľov (voters), zároveň aj boolean o tom či práve priebeha hlasovanie alebo nie.

```
public void saveVotingRoom() {
    try (FileOutputStream fileOut = new FileOutputStream( name: "voting.ser");
         ObjectOutputStream out = new ObjectOutputStream(fileOut))
    {

        out.writeObject(movies);
        out.writeObject(voters);
        out.writeObject(nominatedMovies);
        out.writeObject(actors);
        out.writeObject(directors);
        out.writeBoolean(on_going);
        System.out.println("VotingRoom object has been serialized and saved.");

    } catch (IOException e)
    {
        e.printStackTrace();
    }
}
```

```java
public void loadVotingRoom() {
    File file = new File( pathname: "voting.ser");
    if (!file.exists())
    {
        System.out.println("No file found. Skipping loading.");
        return;
    }
    try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(file)))
    {
        movies = (List<Movie>) in.readObject();
        voters = (List<People>) in.readObject();
        nominatedMovies = (List<Movie>) in.readObject();
        actors = (List<Actor>) in.readObject();
        directors = (List<Director>) in.readObject();
        this.on_going = in.readBoolean();
        System.out.println("VotingRoom object has been deserialized and loaded.");
    } catch (IOException | ClassNotFoundException e)
    {
        e.printStackTrace();
    }
}
```

Použitie lambda výrazov:
Využitie vo vnorenej triede Winners na usporiadanie hlasovaných podľa počtu hlasov.

```java
public List<Movie> getWinnersMovies() {
    List<Movie> sortedMovies = new ArrayList<>(movies);
    Collections.sort(sortedMovies, (a, b) -> Integer.compare(b.getVotes(), a.getVotes()));
    return sortedMovies;
}

    Vráti zoznam hercov zoradený podľa počtu hlasov.
    Returns: Zoznam zoradených hercov
2 usages   Zdenko Kanoš
public List<Actor> getWinnersActors() {
    List<Actor> sortedActors = new ArrayList<>(actors);
    Collections.sort(sortedActors, (a, b) -> Integer.compare(b.getVotes(), a.getVotes()));
    return sortedActors;
}

    Vráti zoznam režisérov zoradený podľa počtu hlasov.
    Returns: Zoznam zoradených režisérov
2 usages   Zdenko Kanoš
public List<Director> getWinnersDirectors() {
    List<Director> sortedDirectors = new ArrayList<>(directors);
    Collections.sort(sortedDirectors, (a, b) -> Integer.compare(b.getVotes(), a.getVotes()));
    return sortedDirectors;
}
```

Použitie generickosti vo vlastných triedach – využíva sa na spracovanie objektov pre výpis grafov v triede Observer, do tejto metódy ako parameter list môže vstupovať List filmov, hercov a režisérov a preto bolo potrebné využitie generickosti.

```
public <T extends Votable> XYChart.Series<String, Number> update(List<T> list, BarChart<String, Number> barChart) {
    XYChart.Series<String, Number> series = new XYChart.Series<>();
    for (T item : list) {
        series.getData().add(new XYChart.Data<>(item.getName(), item.getVotes()));
    }
    return series;
}
```

Použitie vhniezdených tried – trieda Winners je vhniezdená v triede VotingRoom pretože úzko spolu súvisia, trieda Winners má na starosti výber víťazov podľa počtu hlasov po skončení hlasovania:

```
public class VotingRoom implements Serializable {
```

```
    Vnútorná trieda Winners zodpovedá za správu a zisťovanie víťazov v rôznych kategóriách.
5 usages  Zdenko Kanoš
public class Winners {
    2 usages
    private List<Movie> movies;
    2 usages
    private List<Actor> actors;
    2 usages
    private List<Director> directors;

    1 usage  Zdenko Kanoš
    public Winners(List<Movie> movies, List<Actor> actors, List<Director> directors) {
        this.movies = movies;
        this.actors = actors;
        this.directors = directors;
    }

    Vráti zoznam filmov zoradený podľa počtu hlasov.
    Returns: Zoznam zoradených filmov
    3 usages  Zdenko Kanoš
    public List<Movie> getWinnersMovies() {
        List<Movie> sortedMovies = new ArrayList<>(movies);
        Collections.sort(sortedMovies, (a, b) -> Integer.compare(b.getVotes(), a.getVotes()));
        return sortedMovies;
    }
```

Použitie implicitnej implementácie metód – je použitá v rozhraní People na spracovanie platieb, ktoré môže vykonávať každá osoba s bankovým účtom (Actor, Director, Voters), táto metóda je implementovaná ako implicitná aby všetky triedy s bankovým účtom nemuseli túto metódu mať napísané osobitne ale mohli ju takýmto spôsobom využívať.

```
    Vykoná platbu z bankového účtu osoby na iný bankový účet.
    Params: receiverBankAccount – bankový účet príjemcu
            value – suma, ktorá sa má previesť
            senderBankaccount – bankový účet odosielateľa
1 usage   Zdenko Kanoš
default void pay(BankAccount receiverBankAccount, float value, BankAccount senderBankaccount) {
    senderBankaccount.setBalance(-value);
    receiverBankAccount.payment(value);
}
```

Táto metóda je volaná v triede MessageScene:

```
voter.pay(selectedAccount.getAccount(), donationAmount, voter.getBankAccount());
```