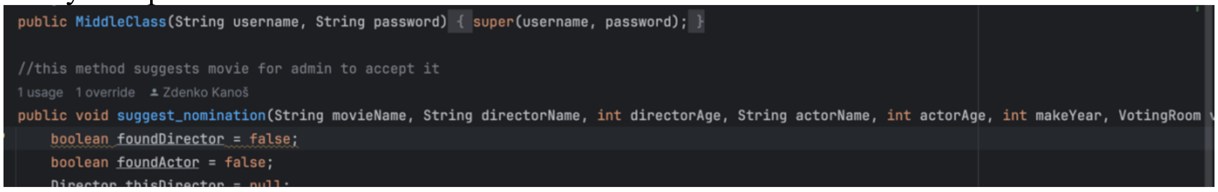
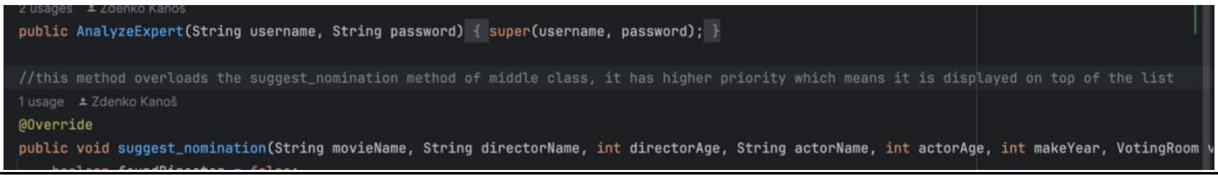
Zdenko Kanoš

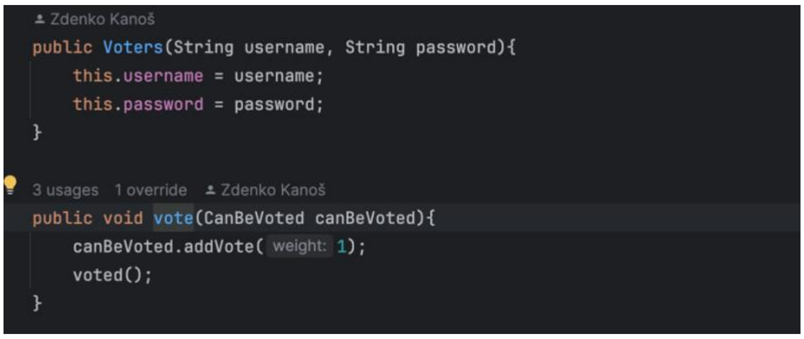
CineVote
HLAVNÉ KRITÉRIÁ:
Prekonávanie metód – vyskytuje sa pri triedach MiddleClass a MovieExpert, MiddleClass a Analyze expert.



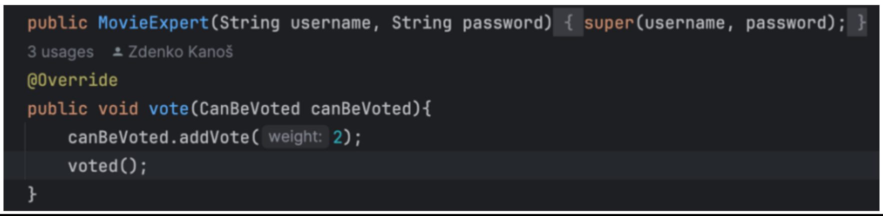
```
public MiddleClass(String username, String password) { super(username, password); }

//this method suggests movie for admin to accept it
1 usage   1 override   Zdenko Kanoš
public void suggest_nomination(String movieName, String directorName, int directorAge, String actorName, int actorAge, int makeYear, VotingRoom v
    boolean foundDirector = false;
    boolean foundActor = false;
    Director thisDirector = null;
```



```
2 usages   Zdenko Kanoš
public AnalyzeExpert(String username, String password) { super(username, password); }

//this method overloads the suggest_nomination method of middle class, it has higher priority which means it is displayed on top of the list
1 usage   Zdenko Kanoš
@Override
public void suggest_nomination(String movieName, String directorName, int directorAge, String actorName, int actorAge, int makeYear, VotingRoom v
    boolean foundDirector = false;
```

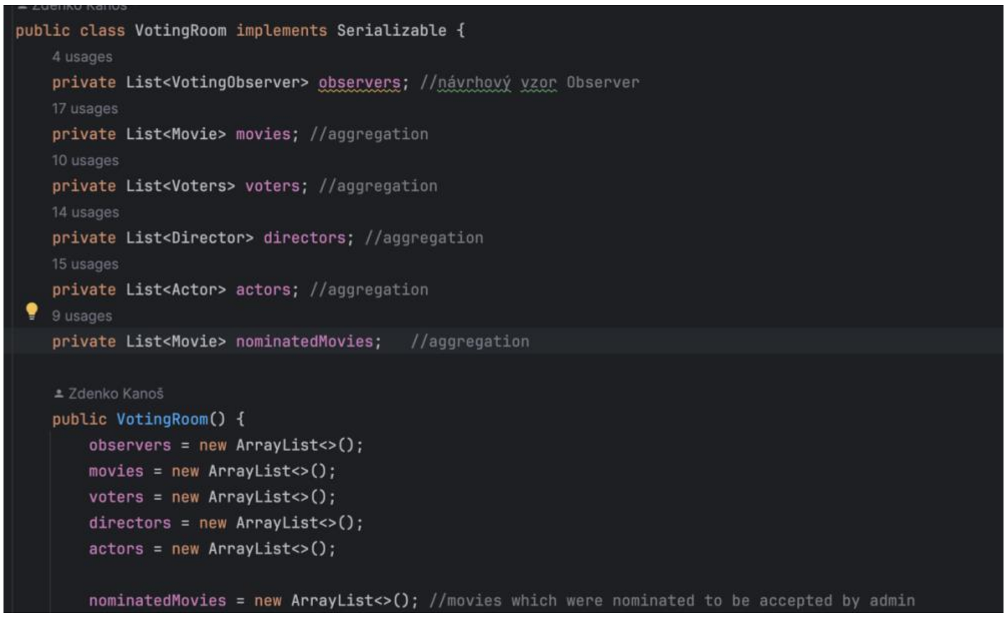


```
Zdenko Kanoš
public Voters(String username, String password){
    this.username = username;
    this.password = password;
}

3 usages   1 override   Zdenko Kanoš
public void vote(CanBeVoted canBeVoted){
    canBeVoted.addVote( weight: 1);
    voted();
}
```



```
public MovieExpert(String username, String password) { super(username, password); }
3 usages   Zdenko Kanoš
@Override
public void vote(CanBeVoted canBeVoted){
    canBeVoted.addVote( weight: 2);
    voted();
}
```

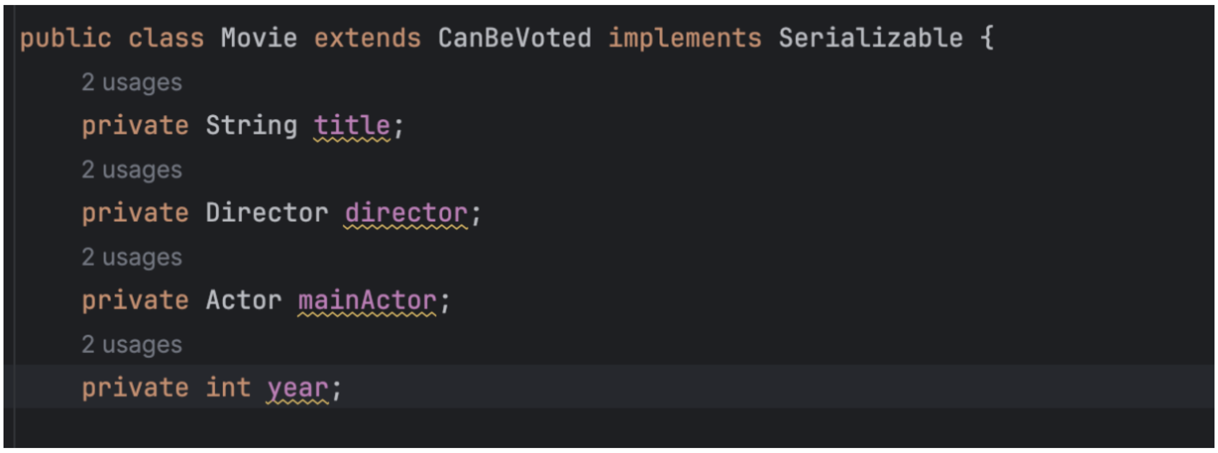
Agregácia - sa vyskytuje v classe VotingRoom



```
Zdenko Kanoš
public class VotingRoom implements Serializable {
    4 usages
    private List<VotingObserver> observers; //návrhový vzor Observer
    17 usages
    private List<Movie> movies; //aggregation
    10 usages
    private List<Voters> voters; //aggregation
    14 usages
    private List<Director> directors; //aggregation
    15 usages
    private List<Actor> actors; //aggregation
    9 usages
    private List<Movie> nominatedMovies;   //aggregation

    Zdenko Kanoš
    public VotingRoom() {
        observers = new ArrayList<>();
        movies = new ArrayList<>();
        voters = new ArrayList<>();
        directors = new ArrayList<>();
        actors = new ArrayList<>();

        nominatedMovies = new ArrayList<>(); //movies which were nominated to be accepted by admin
```

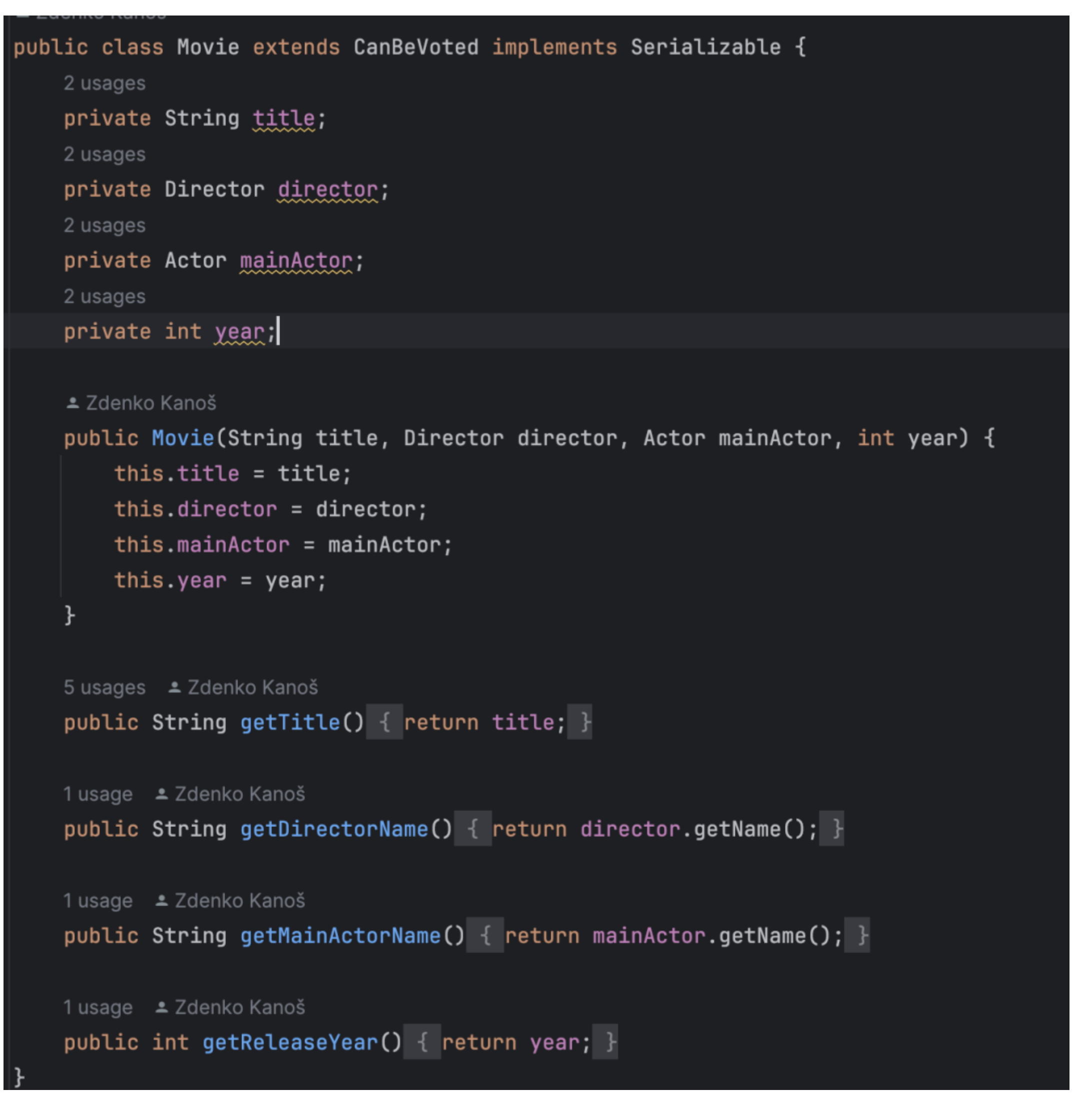
Zdenko Kanoš

Taktiež sa Agregácia vyskytuje v classe Movie kde movie má ako atribúty classy Director a Actor



```
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;
```

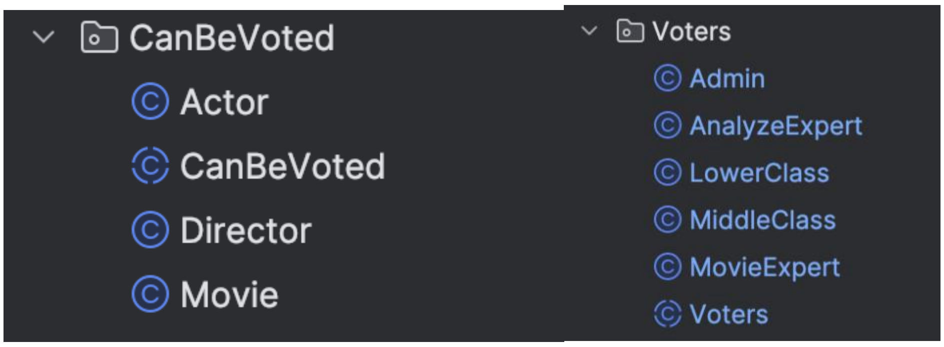
Zapuzdrenie – vyskytuje sa pri každej classe viď. Príklad: (zároveň aj s potrebnými gettermi pre prístup k premenným)



```
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; }
}
```

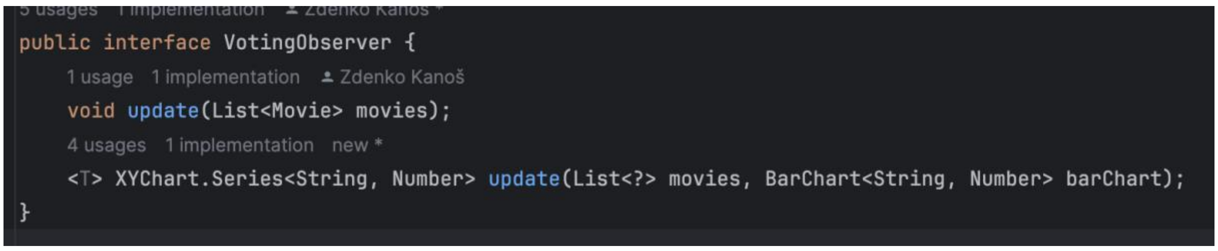
Zdenko Kanoš

V aplikácii sú vybudované dve Hierarchie s abstraktnou classou nad nimi

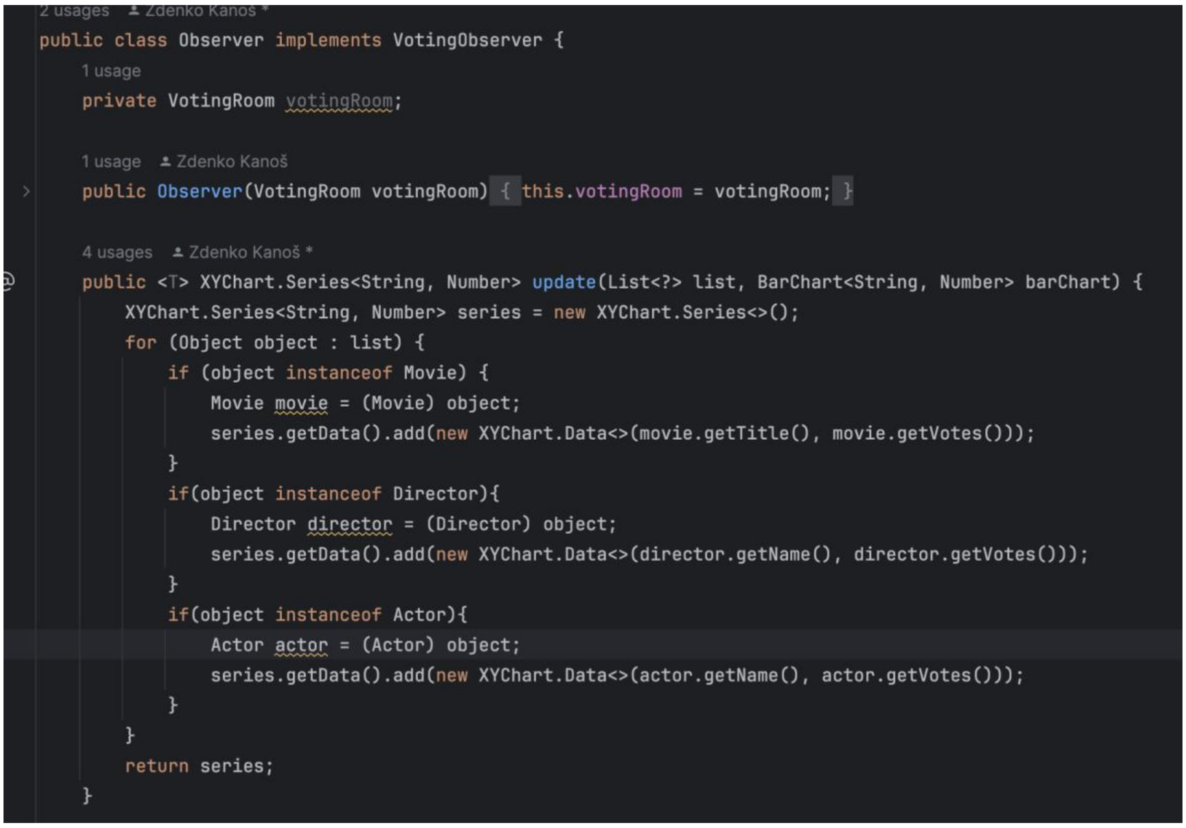


```
CanBeVoted
    Actor
    CanBeVoted
    Director
    Movie
Voters
    Admin
    AnalyzeExpert
    LowerClass
    MiddleClass
    MovieExpert
    Voters
```

ĎALŠIE KRITÉRIÁ:
Návrhový vzor – OBSERVER – využíva sa na aktualizovanie grafov v AdminScene, tieto grafy zobrazujú priebežné výsledky volenia, ktoré môže vidieť len Admin

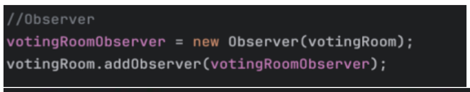


```java
5 usages   1 implementation   Zdenko Kanoš *
public interface VotingObserver {
    1 usage   1 implementation   Zdenko Kanoš
    void update(List<Movie> movies);
    4 usages   1 implementation   new *
    <T> XYChart.Series<String, Number> update(List<?> movies, BarChart<String, Number> barChart);
}
```

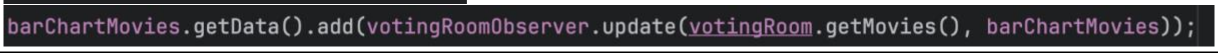


```java
2 usages   Zdenko Kanoš *
public class Observer implements VotingObserver {
    1 usage
    private VotingRoom votingRoom;

    1 usage   Zdenko Kanoš
    public Observer(VotingRoom votingRoom) { this.votingRoom = votingRoom; }

    4 usages   Zdenko Kanoš *
    public <T> XYChart.Series<String, Number> update(List<?> list, BarChart<String, Number> barChart) {
        XYChart.Series<String, Number> series = new XYChart.Series<>();
        for (Object object : list) {
            if (object instanceof Movie) {
                Movie movie = (Movie) object;
                series.getData().add(new XYChart.Data<>(movie.getTitle(), movie.getVotes()));
            }
            if(object instanceof Director){
                Director director = (Director) object;
                series.getData().add(new XYChart.Data<>(director.getName(), director.getVotes()));
            }
            if(object instanceof Actor){
                Actor actor = (Actor) object;
                series.getData().add(new XYChart.Data<>(actor.getName(), actor.getVotes()));
            }
        }
        return series;
    }
```

V AdminScene:



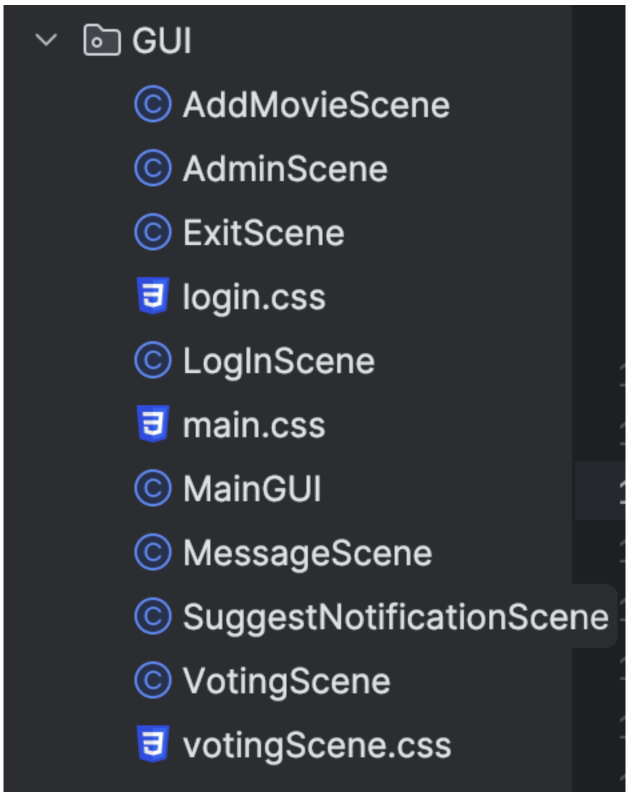
```java
//Observer
votingRoomObserver = new Observer(votingRoom);
votingRoom.addObserver(votingRoomObserver);
```



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

Zdenko Kanoš

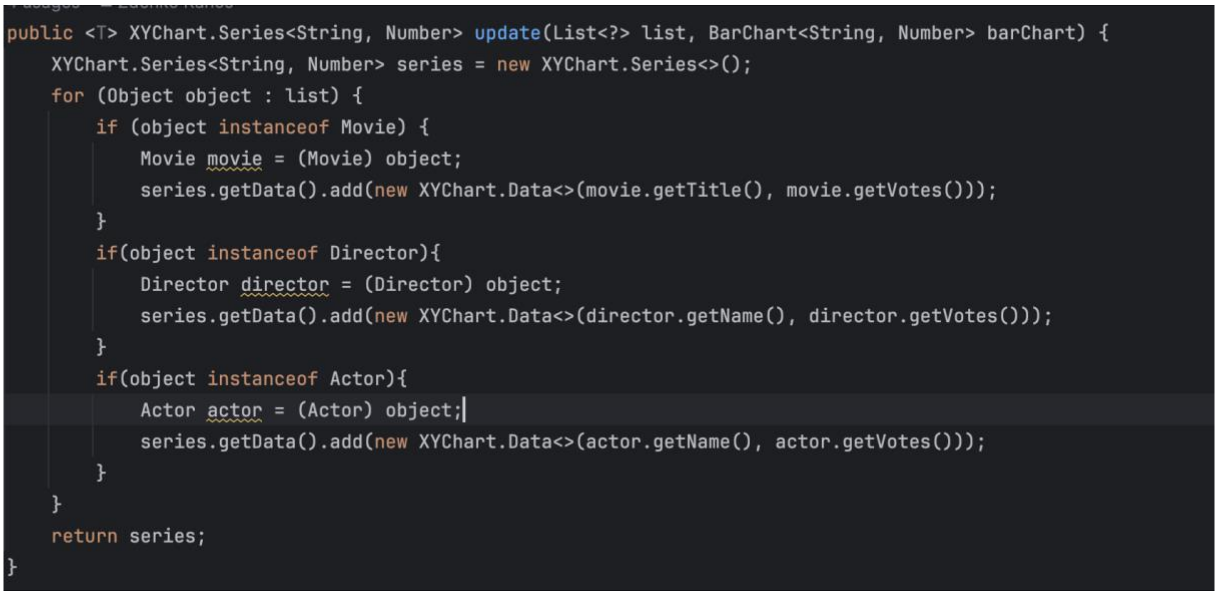
GUI – v aplikácii je zapracované grafické používateľské prostredie, ktoré je celé napísané manuálne



```
GUI
    AddMovieScene
    AdminScene
    ExitScene
    login.css
    LogInScene
    main.css
    MainGUI
    MessageScene
    SuggestNotificationScene
    VotingScene
    votingScene.css
```

---

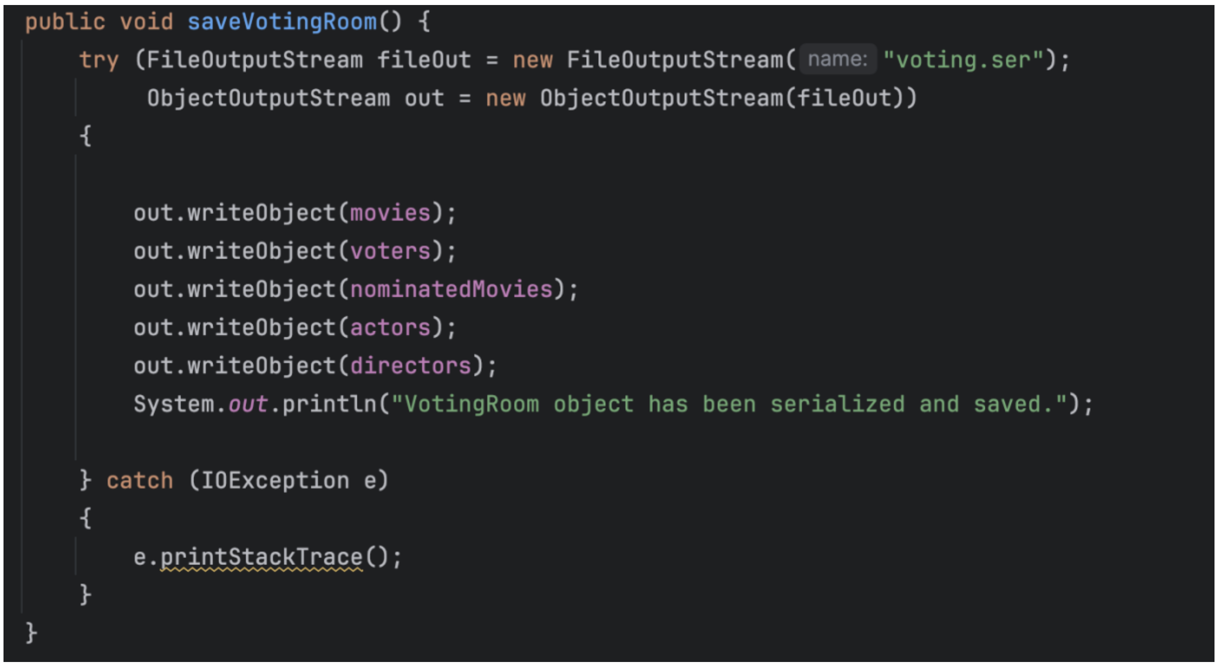
explicitné použitie RTTI – v classe Observer je použité RTTI na aktualizovanie grafov, pretože pri každom type je rôzny getter pre meno a votes a aby sme mohli poslať akýkoľvek list objektov do tejto metódy



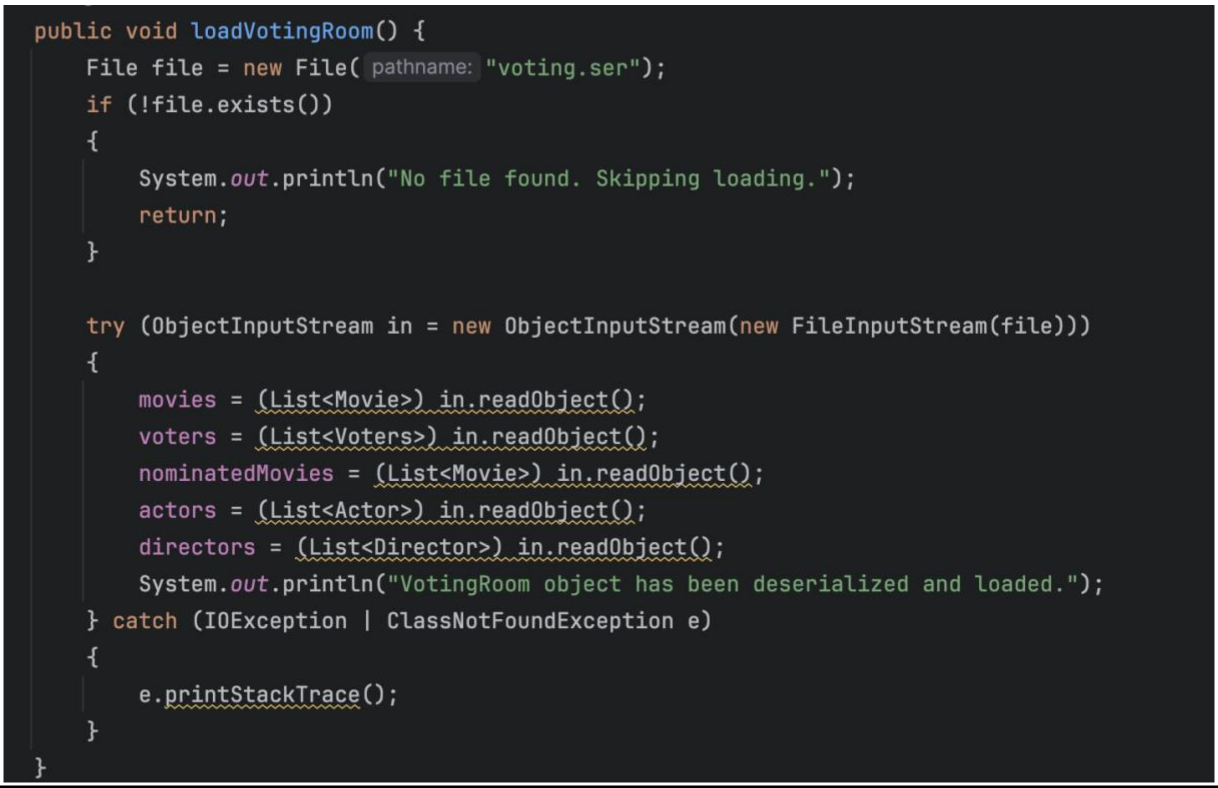
```java
public <T> XYChart.Series<String, Number> update(List<?> list, BarChart<String, Number> barChart) {
    XYChart.Series<String, Number> series = new XYChart.Series<>();
    for (Object object : list) {
        if (object instanceof Movie) {
            Movie movie = (Movie) object;
            series.getData().add(new XYChart.Data<>(movie.getTitle(), movie.getVotes()));
        }
        if(object instanceof Director){
            Director director = (Director) object;
            series.getData().add(new XYChart.Data<>(director.getName(), director.getVotes()));
        }
        if(object instanceof Actor){
            Actor actor = (Actor) object;
            series.getData().add(new XYChart.Data<>(actor.getName(), actor.getVotes()));
        }
    }
    return series;
}
```

Zdenko Kanoš

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)



```java
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);
        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<Voters>) in.readObject();
        nominatedMovies = (List<Movie>) in.readObject();
        actors = (List<Actor>) in.readObject();
        directors = (List<Director>) in.readObject();
        System.out.println("VotingRoom object has been deserialized and loaded.");
    } catch (IOException | ClassNotFoundException e)
    {
        e.printStackTrace();
    }
}
```