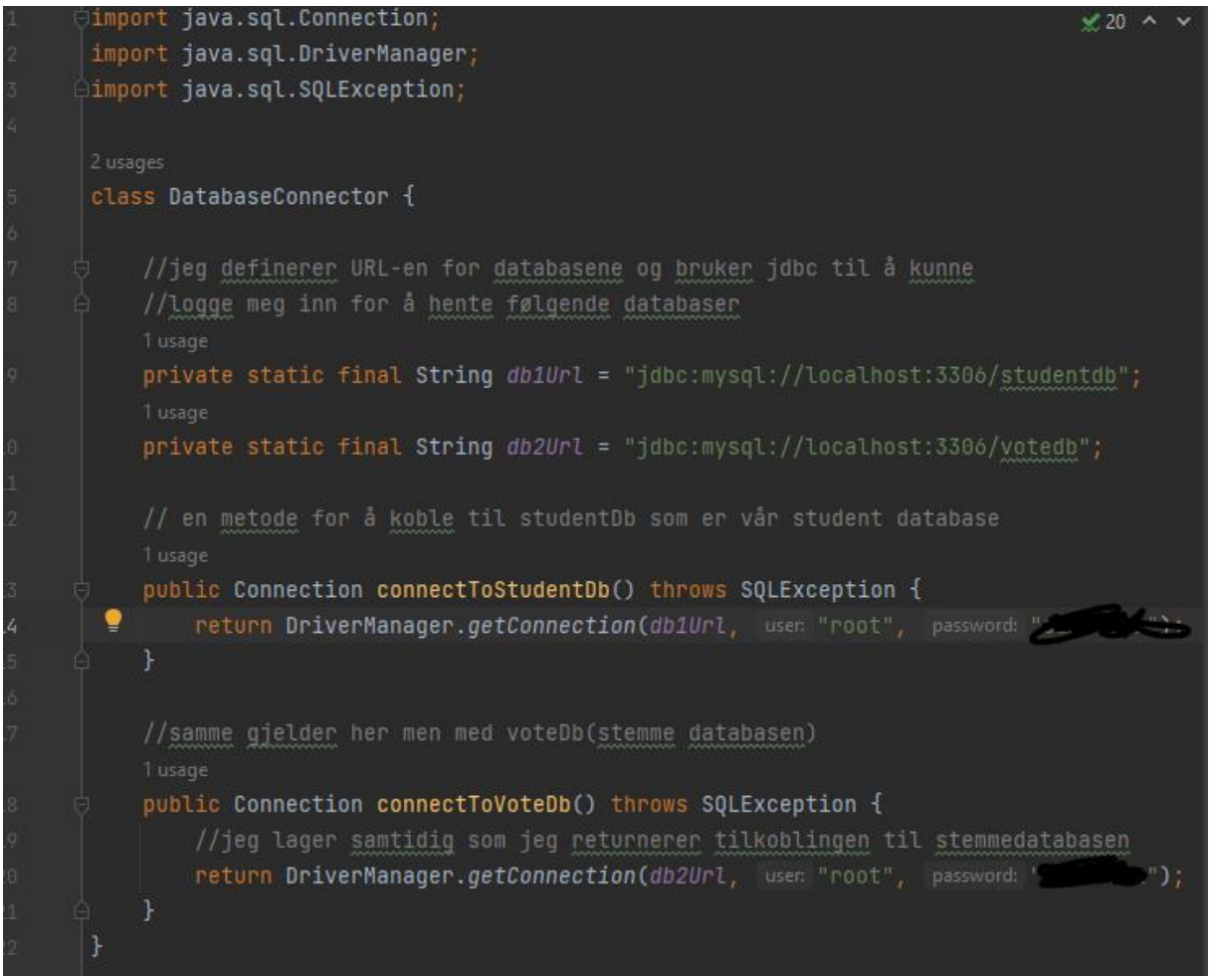
# Rapport for Objektoriented Programming

## DBConnector

- Jeg bruker JDBC til å logge meg inn på databasene jeg har lagd deretter lager jeg metoder for å koble til databasene som studentdb og votedb.



```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

2 usages
class DatabaseConnector {

    //jeg definerer URL-en for databasene og bruker jdbc til å kunne
    //logge meg inn for å hente følgende databaser
    1 usage
    private static final String db1Url = "jdbc:mysql://localhost:3306/studentdb";
    1 usage
    private static final String db2Url = "jdbc:mysql://localhost:3306/votedb";

    // en metode for å koble til studentDb som er vår student database
    1 usage
    public Connection connectToStudentDb() throws SQLException {
        return DriverManager.getConnection(db1Url, user: "root", password: "[redacted]");
    }

    //samme gjelder her men med voteDb(stemme databasen)
    1 usage
    public Connection connectToVoteDb() throws SQLException {
        //jeg lager samtidig som jeg returnerer tilkoblingen til stemmedatabasen
        return DriverManager.getConnection(db2Url, user: "root", password: "[redacted]");
    }
}
```

Main.java

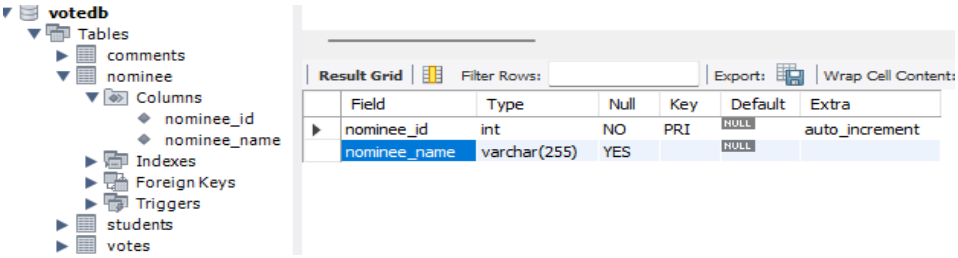
- Denne koden bygger et stemmesystem i java som gjør det enkelt for studenter og kandidater å stemme på å legge til kommentarer. DatabaseConnector brukes til å koble til datbaser «VotingSystem» brukes til å administrere stemmeprossesen og nominerte. Programmet gir valg av registrering, se toppkandidater, legge til kommentarer og avslutte. Dette gjør jeg hjelp av OOP med elementer og metoder som organiserer alt.

- Resultat:

```java
import java.util.Scanner;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;


no usages
public class Main {
    no usages
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {

            // Jeg legger til DatabaseConnector for å koble til følgende databaser

            DatabaseConnector dbConnector = new DatabaseConnector();
            Connection studentDbConnection = dbConnector.connectToStudentDb();
            Connection voteDbConnection = dbConnector.connectToVoteDb();

            // Jeg oppretter et VotingSystem-objekt for å administrere stemmesystemet

            VotingSystem votingSystem = new VotingSystem(studentDbConnection, voteDbConnection);

            // Jeg legger til en Student og legger til i stemmesystemet

            Student student = new Student( studentId: 1,  name: "Karius",  major: "Informatikk");
            votingSystem.addStudent(student);

            // jeg legger til nominerte og legger dem også til stemmesystemet

            Nominee nominee1 = new Nominee( name: "Per");
            Nominee nominee2 = new Nominee( name: "Emely");
            Nominee nominee3 = new Nominee( name: "karl");
            votingSystem.nominate(nominee1);
```

```
"C:\Program Files\Zulu\zulu-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
Velkommen til vår avslutningsseremoni!
Hei, jeg heter Nils, jeg skal hjelpe deg her:
1. Registrer deg, vær så snill.
2. Sjekk ut våre toppnominerte for øyeblikket.
3. Avslutt - takk for at du deltok!
-> 1
Student-ID: 1
Velkommen, Karius! Oi, du har ennå ikke stemt, sjekk ut menyen vår.
1. Se alle nominerte.
2. Forslå en ny kandidat.
3. Velg og legg til en kommentar for en kandidat under:
4. Gå tilbake til hovedmenyen.
-> 2
Skriv inn navnet på den nye kandidaten: jonas
Kandidat 'jonas' har blitt foreslått.
1. Se alle nominerte.
2. Forslå en ny kandidat.
3. Velg og legg til en kommentar for en kandidat under:
4. Gå tilbake til hovedmenyen.
-> 3
Nåværende nominerte:
1. karl
2. jonas
3. Emely
4. Per
Skriv inn nummeret på kandidaten du vil stemme på: 2
Skriv inn kommentaren din for jonas: hei
Stemmen og kommentaren for jonas er blitt registrert.
java.sql.SQLSyntaxErrorException Create breakpoint : Unknown column 'nominee_name' in 'field list'
    at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:121)
    at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping
```

Jeg får ikke til å legge til kommentar fordi den ikke finner kolonne: ‘nominee_name’ som jeg synes er veldig rart for når jeg ser på databasen min så har jeg riktig kolonne som vises følgende bilde nedefor




- Hadde jeg klart å få opp kjøringen for nominee_name hadde programmet kjørt fint og oppdatert seg inni databasen. Jeg har riktig koder med metoder som bekrefter det.

Student.java

- Kodene er grunnlagg for stemmesystemet, jeg har implementert grensesnittet «Votable» som definerer handlinger for stemmeobjektene. Somf.eks studenter med navn og status som er beskrev i koden. votingSystem klassen styrer stemmeprossesen ved å lagre studenter og nominerte kandidater. Gjennom database koblinger brukes kodene til å vise stemmere, nominerende og vinnere. Jeg har brukt OOP som interface, grensesnitt og ovveride for å organisere og vedlikeholde strukturen slik at både koden for meg og lesere blir enklere.

```
// Jeg definerer grensesnitt som brukes til å representere
// egenskapene til noe som kan bli stemt på
1 usage  1 implementation
interface Votable {
    1 usage  1 implementation
    String getName();
    3 usages  1 implementation
    boolean hasVoted();
    1 usage  1 implementation
    void markVoted();
}

// klasse som definerer en student og implemtenterer votable-grensesnittet
// for å definere regler metoden må ha for å håndtere denne kjøringen
13 usages
class Student implements Votable {
    2 usages
    private int studentId;
    2 usages
    private String name;
    2 usages
    private String major;
    3 usages
    private boolean voted;

    1 usage
```

```
}
// jeg bruker ovveride til å holde styr på mine underklasser
//det gjør det enklere med vedlikehold og struktur.
1 usage
@Override
public String getName() {
    return name;
}

no usages
public String getMajor() {
    return major;
}

3 usages
@Override
```

VotingSystem.java

- Koden går ut på hovedklassen VotingSystem som innholder en arrayListe med studenter og mappe over nominerte kandidater. Det klassen gjør er å administrere student og kandidat opplysninger og bruker databaseforbindelser for lagring. Metoden som er implementert gjør at programmet kan kjøre ved å gjøre at studenter kan stemme på kandidater og finne vinnere med flest stemmer. getStudentById søker etter studenter ved gitt ID, addCommentToNominee gjør at du kan legge kommentar inni i databasen. SaveVote og SaveComment gjør at resultatet når disse metodene kjører gjør at det lagres inn i databasen.

```
public void addCommentToNominee(String nomineeName, String comment) {
    if (nominees.containsKey(nomineeName)) {
        Nominee nominee = nominees.get(nomineeName);
        nominee.addComment(comment);
        saveCommentToDatabase(nominee, comment); // Save comment to the database
    }
}

no usages
public List<Student> getStudents() {
    return students;
}

1 usage
private void saveVoteToDatabase(Student student, Nominee nominee) {
    //metode for å lagre stemmer og kommentarer
    // i data basen
    try {
        String query = "INSERT INTO votes (student_id, nominee_name) VALUES (?, ?)";
        PreparedStatement preparedStatement = voteDbConnection.prepareStatement(query);
        preparedStatement.setInt( parameterIndex: 1, student.getStudentId());
        preparedStatement.setString( parameterIndex: 2, nominee.getName());
        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

1 usage
private void saveCommentToDatabase(Nominee nominee, String comment) {
    try {
        String query = "INSERT INTO comments (nominee_name, comment) VALUES (?, ?)";
        PreparedStatement preparedStatement = voteDbConnection.prepareStatement(query);
        preparedStatement.setString( parameterIndex: 1, nominee.getName());
        preparedStatement.setString( parameterIndex: 2, comment);
        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
```

Nominee.java

- Klassen Nominee representerer kandidat/er med navn og funksjoner til å kunne motta stemmer og kommentarer. Konstruktøren har ansvar for å opprette ny kandidat med gitt navn, gi antall stemmer og legge rom til kommentar. Alle gettere i denne klassen henter ut informasjon fra programmet som fungerer som en spørring for å hente liste med navn, votes og kommentarer.

```java
import java.util.ArrayList;
import java.util.List;

// nominee-klassen representerer en kandidat som kan nomineres og motta stemmer og kommentarer
18 usages
class Nominee {
    2 usages
    private final String name;          // kandidatens navn er uforanderlig (final).
    3 usages
    private int votes;
    3 usages
    private final List<String> comments; // en liste over kommentarer til kandidaten.


    //legger til konstruktør for å opprette en ny kandidat med gitt navn.
    4 usages
    public Nominee(String name) {
        this.name = name;
        this.votes = 0;
        this.comments = new ArrayList<>(); // her ppretter jeg en tom liste for kommentarer.
    }

    // metode for å hente kandidatens navn ved hjelp av gettere
    4 usages
    public String getName() {
        return name;
```