IN2090: Obligatorisk oppgave 6 (Java)

I denne oppgaven får du et delvis fullført program for å hente ut informasjon om timelister, og legge til nye timelistelinjer i en PostgreSQL-database. Oppgaven går ut på å fullføre dette programmet slik at programmet henter ut informasjon om timelister fra en database ved hjelp av *JDBC*, og legger inn ny informasjon i databasen.

Koden du får utdelt inneholder en ferdigskrevet main-metode og kode for å koble deg til en database fra Java. Det du trenger å skrive, er metodene i *TimelisteDb.java*.

Start med å laste ned nødvendige filer herfra. Pakk ut filene på et egnet sted. Du skal ha følgende filer:

- **Tilkobling.java**: Det kan være knotete å koble deg til en database, så vi har skrevet all nødvendig kode for å koble deg til PostgreSQL-databasen din på IFI fra Java. Du kan gjerne se på innholdet i filen, men du trenger ikke endre noe her.
- **TimelisteDb.java**: Det er i denne klassen du skal skrive koden i denne obligen. Metodene er deklarert, men du må sørge for at metodene gjør det de skal.
- **Oblig6.java**: Her finnes main-metoden, som kaller på metodene i TimelisteDb.java. Her må du skrive inn ditt eget brukernavn, ellers trenger du ikke endre noe.
- **postgresql.jar**: JDBC er en del av Java, men PostgreSQL-driverne er ikke det, og ligger i denne filen.
- **oppsett.sql**: Denne filen skal du importere i databasen din, på samme måte som du gjorde i oblig 3. Dette er en forenklet versjon av tabellene fra oblig 3, og inneholder tabellene `tliste` og `tlistelinje`.

Oppsett

1: Last ned og pakk ut filene i oblig6-java.zip.

2: Du må være på UiO-nettet for å koble deg til PostgreSQL-databasen på IFI. Om du ikke befinner deg på UiO-nettet, må du sette opp en VPN-klient/program på PC-en din. Instruksjoner for oppsett av VPN-klient/program finner du her (NB: ikke bruk view.uio.no).

3: Bytt ut «`dittBrukernavn`» med brukernavnet ditt i *Oblig6.java*.

4: Prøv å kjøre programmet og se om du får koblet til. Kompilering av Java-programmet gjør du på vanlig måte. For å kjøre programmet, må du bruke følgende kommando:

Windows:

```
java -cp ".;postgresql.jar" Oblig6
```

Unix (Mac/Linux):

```
java -cp .:postgresql.jar Oblig6
```

Om du bruker IntelliJ eller Eclipse må jar-filen legges til i prosjektet. Instruksjoner for IntelliJ. Instruksjoner for Eclipse.

5: Importer *oppsett.sql* til databasen din, på samme måte som du gjorde i oblig 3: Kopier filen til et eget område på IFI, logg inn på PostgreSQL med `psql -h dbpg-ifi-kurs -U brukernavn`, og kjør kommandoen `\i oppsett.sql`. Alternativt kan du kopiere alle spørringene inn i `psql` for å opprette tabellene og sette inn data.

Oppgave

Du skal fylle ut metodene i *TimelisteDb.java*. Alle metodene kaster SQLException videre, så du trenger ikke selv å håndtere SQLException.

`printTimelister()`

Skal printe ut `timelistenr`, `status` og `beskrivelse` for alle timelistene i databasen. Du velger selv hvordan utskriften skal se ut.

`printTimelisteLinjer(int timelisteNr)`

Skal printe ut `linjeNr`, `timeantall`, `beskrivelse` og `kumulativt_timeantall` for alle linjer med `timelisteNr`. Du trenger ikke å skrive ut timelistenummeret. Du velger selv hvordan utskriften skal se ut.

`medianTimeantall(int timelisteNr)`

PostgreSQL har ingen innebygget aggregeringsfunksjon for å regne ut medianverdi. Derfor ønsker vi å bruke Java til å hjelpe oss med utregningen. Metoden skal *returnere* (ikke skrive ut) median-timeantallet.

Medianen er tallet «midt på» listen av tall, dersom listen er sortert. Om det er et partall antall elementer, tar man gjennomsnittet av de to midterste tallene.

Tips: Det kan være lurt å bruke `ORDER BY` i SQL-spørringen for å sortere etter timeantall. Mens du itererer over resultatene, kan du legge inn verdiene i f.eks. en `ArrayList<Integer>` med timeantall. Du kan bruke hjelpemetoden `median(List<Integer>)` som tar imot en sortert liste (kan f.eks. være en `ArrayList<Integer>`), og returnerer mediantallet som en `double`.

`settInnTimelistelinje(int timelisteNr, int antallTimer, String beskrivelse)`

Tar imot de tre parameterne `timelisteNr`, `antallTimer` og `beskrivelse`, og kjører en `INSERT INTO`-spørring for å legge inn en timelistelinje med disse verdiene.

Merk: Du må først finne ut hva som er det høyeste linjenummeret, og i den nye linjen du legger inn må linjeNr være én høyere enn det høyeste. F.eks. om du har linjeNr 1-4, må den nye linjen ha linjeNr 5.

`regnUtKumulativtTimeantall``(int timelisteNr)`

Et kumulativt timeantall er summen av timeantall *så langt i timelisten*. Du skal altså summere linjens timeantall med timeantall til alle linjer du allerede har iterert gjennom. For timeliste 2 vil kumulativt timeantall se slik ut:

```
 linjenr | timeantall | kumulativt_timeantall
---------+------------+-----------------------
       1 |          1 |                     1
       2 |          4 |                     5
       3 |          3 |                     8
       4 |          5 |                    13
```

Dette er noe som kan være knotete å få til i SQL, men lett å gjøre i Java. Derfor skal du i denne oppgaven hente ut alle linjene med det oppgitte timelisteNr. Mens du itererer gjennom resultatene, skal du regne ut den kumulative verdien, og sette den tilbake inn i databasen ved hjelp av `UPDATE`-spørringer. Det er altså lurt å kjøre én `UPDATE` for hver rad i løkken.

Innlevering

Siden du kun skal ha gjort endringer i *TimelisteDb.java*, er det kun denne filen du trenger å levere.