Programmeringslaboration 1, HI1030 Klientapplikation till SQL-databas

Inledning

I denna laboration ska ni skapa en databas i ett SQL-DBMS samt skriva en klientapplikation i Java som kommunicerar med detta DBMS via JDBC.

Redovisning

Uppgiften löses i grupp om 2 personer; båda måste ha förkunskaper i Java-programmering motsvarande kursen HI1027, objektorienterad programmering.

Vid redovisningen *ska* följande finnas tillgängligt:

- Två scheman för databasen, enligt ER- respektive enligt relationsmodellen
- Dokumentation om SQL-satserna som skapar databas, tabeller samt ev. vyer och procedurer (lämpligen script-filer).
- Ett klassdiagram som beskriver klientapplikationen (dock inga detaljer om vy-klasser).
- Väldokumenterad källkod för modelldelen av klientapplikationen.

Redovisningsdag anges i kurs-PM och schema. Redovisningstid bokas via Bilda/Verktyg/Inbjudningar. Notera: Modelldelen i applikationen (d.v.s. alla java-klasser utom de som är rena vy-klasser) ska lämnas in via länk på kurswebben senast dagen före redovisningen. Inlämnad kod är en förutsättning för att få redovisa och det är den inlämnade koden som bedöms.

Uppgiften

Databasen ska kunna lagra information om filmer och musikalbum, artister/regissörer, genre, betyg och recensioner för dessa. Databasen ska också innehålla användare (som t.ex. har skapat en recension).
Notera att album respektive artist/regissör är separata entiteter med egna attribut som t.ex. namn och nationalitet respektive titel och utgivningsdatum.

Funktionalitet i klientapplikationen

Allmänna krav, *ska* vara uppfyllda oavsett betygsnivå

1. På server-sidan ska det finnas en specifik användare för klient-applikationen definierad. Denna användare ska endast ha access till vissa tabeller och får endast exekvera SQL-satser på dessa tabeller. Använd GRANT för detta (eventuellt tillsammans med specifika vyer och/eller ”stored procedures”).
2. Datat som returneras vid en fråga ska i applikationen representeras av objekt av typer som CD, Artist o.s.v. Ni måste alltså översätta databasens relationer till lämpliga klasser, dvs.en objektmodell.

3. Koden som kommunicerar med databasen ska ha metoder som motsvarar olika SQL-frågor; *dessa metoder ska returnera objekt eller listor med objekt*, av typer som t.ex. CD (se punkt 2). Dessa metoder ska deklareras i ett java-interface; *vy och kontroller ska endast känna till interfacet, inte implementationen av interfacet*.[1]
4. Varje fråga, insättning i eller uppdatering av databasen ska exekveras i en separat tråd[2].
5. Klientapplikationen ska ha ett grafiskt användargränssnitt.
6. Applikationen ska följa designmönstret MVC. Speciellt ska koden som kommunicerar med databasen och skapar objekten som representerar data separeras från koden som representerar användargränssnittet (två olika paket).
7. Transaktioner ska implementeras där så är lämpligt. Om fel uppstår vid en transaktion ska det resultera i en rollback.
8. När applikationen avslutas ska kopplingen till databasen garanterat stängas.
   Tips: använd stage.setOnCloseRequest för att hantera exit via [X].

Betyg G

Det ska, via klientapplikationen, vara möjligt att lägga till skivor, med artist och genre angivet, samt att betygsätta skivor. Detta görs lämpligen via en modal Stage alt. JDialog.

Det ska också vara möjligt att söka efter skivor via titel samt att söka efter alla skivor för en viss artist.
Notera att det råder ett ”många till många”-samband mellan CD och Artist, det ska alltså finnas en sambandstabell.
Man ska även kunna söka via genre och betyg.

Betyg VG

Förutom kraven för betyg G så ska man även kunna hantera filmer. Filmerna ska kunna också kunna kategoriseras med genre.
Det ska även vara möjligt att för filmer söka via aspekter som regissör, genre och betyg.

Användare ska kunna logga in i programmet för att ange betyg, recensera samt lägga till artister, filmer och album.
En användare ska endast kunna ange ett betyg/recension per film eller album.
Det ska framgå vilken användare som lagt till en artist, film eller album.

Betygssättning

Observera att det även finns kvalitetsmått inom betygen, d.v.s. beroende på hur väl ni genomfört uppgiften så varierar betygen. Har er applikation funktionalitet för betyg G kan ni få 1-2 poäng beroende på hur väl lösningen är skriven och dokumenterad. Har applikationen funktionalitet för betyg VG kan ni få 1-4 poäng beroende på kvaliteten på lösningen.

Några tips

- JDBC, http://download.oracle.com/javase/tutorial/jdbc/index.html
- Bra information om grafiska komponenter i JavaFX och Swing:
  http://docs.oracle.com/javase/8/javase-clienttechnologies.htm

---

[1] Notera att om ni utför detta korrekt kommer ni att kunna återanvända alla klasser i användargränssnittet, samt några i modellen, i nästa laboration (då data i stället ska lagras i en NoSQL-databas).
[2] Notera att du inte får uppdatera UI från någon annan tråd än huvudtråden, se föreläsningsanteckningarna.

- Om användaren bara ska kunna ange vissa specifika alternativ (t.ex. fördefinierade genrer), använd ComboBox istället för TextField.
  Resultat av en sökning kan presenteras snyggt i en TableView (liknar ett kalkylark).