**Grupparbete SQL**

**Backendutveckling och systemintegration.**

Av Zein och Stefan Sundström

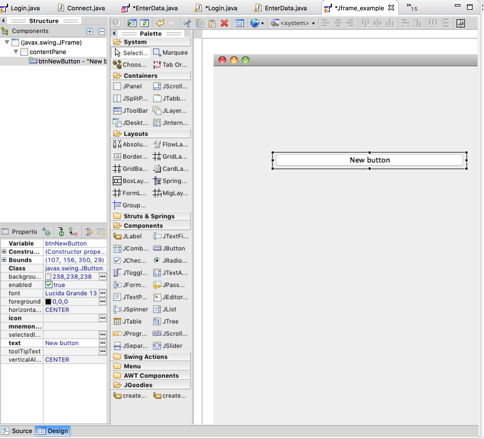
**Beskrivning**

Vi har valt att göra ett program som kan skriva och hämta information om hyresgäster och lägenheter från en databas. Användaren loggar in och kan sedan lista lediga lägenheter och lägga till nya hyresgäster. Vi använder oss av en server för uppkopplingen som måste startas innan programmet körs. ….

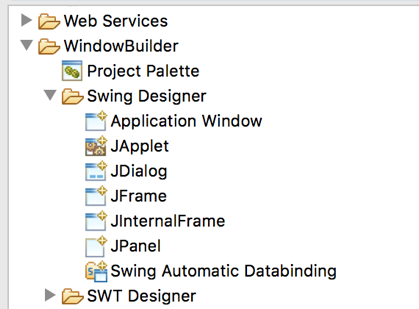
Bla bla bla..lite MVC tänk kanske

**Gränssnitt**

Gränsnittet är gjort i Swing med hjälp av WindowsBuilder - en GU-designer som kan köras i Eclipse. Den gör att man kan gå mellan kod och design-vy, man behöver på så sätt inte ”chansa” när man sätter dimensioner och placering av objekten på skärmen.



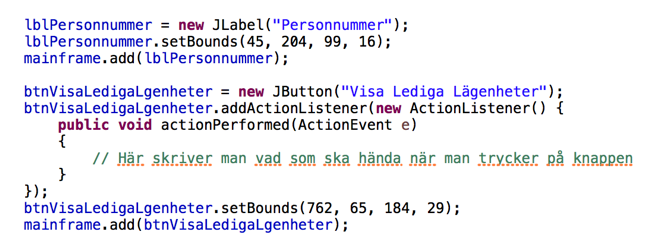
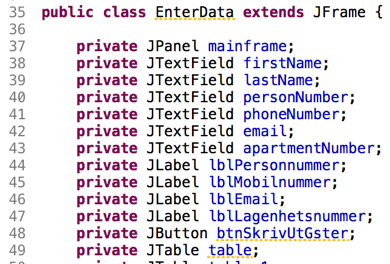
När man installerat WindowsBuilder kan man, istället för att skapa en tom klass, utgå från en preset - JFrame i wizarden skapar en klass som ärver av JFrame. Själva huvudfönstret skapas och sedan kan man lägga till objekt som knappar och textfält.

****

***”new/Other..” i Eclipse***

**Koden**

Om man skriver manuellt så deklarerar man först det objekt man vill använda, sedan skapas en instans från klassen, dimensionerna sätts (x, y, bredd och höjd). Sedan körs metoden add från huvudfönstrets objekt för att det nya objektet ska läggas till. Vill man köra en händelse så lägger man till en ActionListener med Interfacet Actionlistener som inparameter. En så kallad ”anonym inre klass”.

****

**JAR-filer**

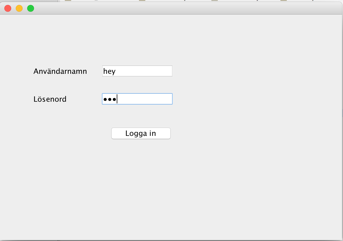
Vi använder oss av två jarfiler som importeras till Bulid Path’en i vår IDE.

RS2XML används för att på ett lätt sätt kunna skriva ut tabeller i Swing och mysql.connector.java är JDBC-drivern för mySQL-databsen vi jobbar mot.

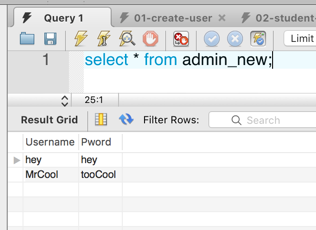
**Funktioner**

**Logga in**

För att logga in måste användaren ange ett användarnamn och ett lösenord.

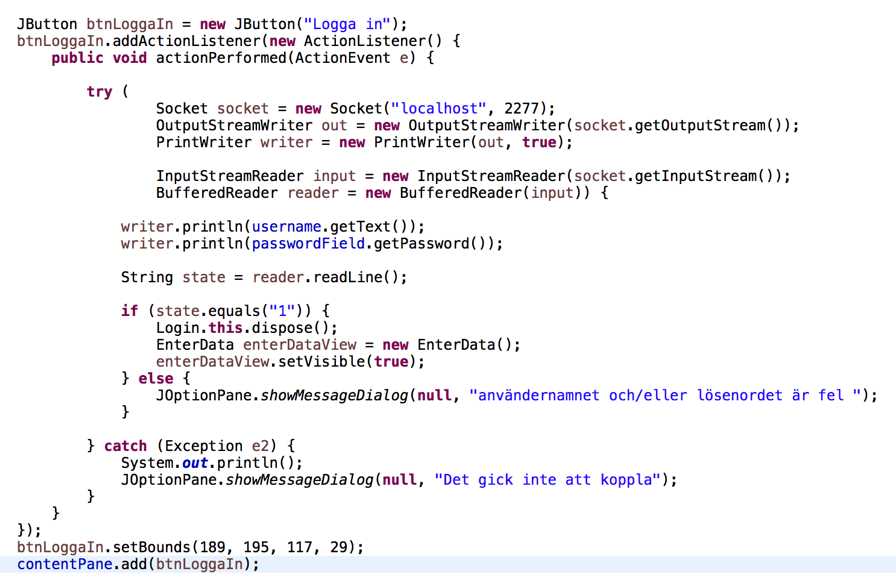
****

Verifieringen görs genom att man jämför de två variabler användaren matar in mot motsvarande hos de administratörer som finns lagrade i databasens tabellen.



I klassen Login är körs en ”händelselyssnare” på knappen ”Logga in”, vilket gör att metoden körs när användaren trycker på knappen. En socket skapas som kopplas till en Server på den egna datorn (local host port 2277).

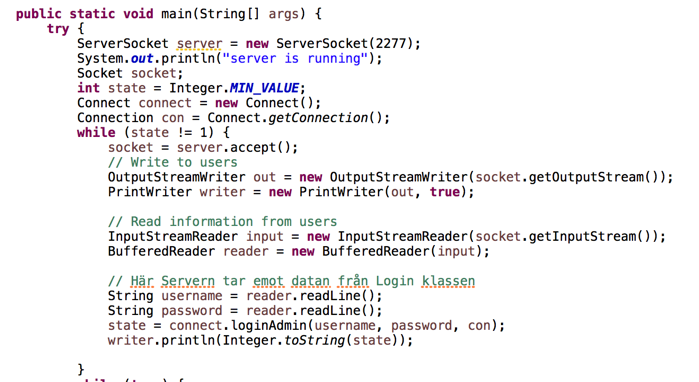
Ett objekt av klassen OutputStreamWriter och PrintWriter används för att skicka iväg användarnamnet och lösenordet till Servern. Lösenordet valideras och svaret läses av med hjälp av BufferedReader-objektet ”reader” och sparas som strängen ”state”. Om state är ”1” , dvs lösenord och användarnamn överensstämmer med en tabellrad i databasen, så öppnas startar programmet (huvudfönstret ”EnterData” öppnas).



**Valideringen i detalj**

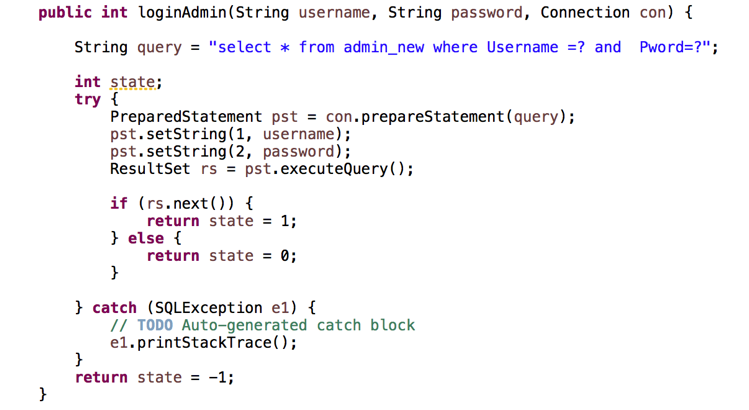
**Server.java**

Användarnamnet och lösenordet samlas in i While loopen där vilkoret säger att loopen kommer att snurra så länge lösenordet inte är 1. loginAdmin-metoden anropas som ligger i klassen Connection, användarnamn, lösenord och uppkoppling skickas in i metoden.



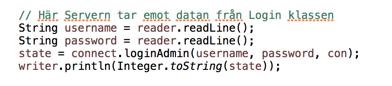
**Connect.java**

I loginAdmin skickas en query som säger ”välj alla rader från tabellen admin som överenstämmer med användarnamn och lösenord”. Ett PreparedStatement-objekt skapas från uppkopplingen med query’n som inparameter, från objektet så talar vi om vad ”?” har för värde på diverse position. Sedan skickas detta till databasen somtalar om för oss hur många rader som överenstämmer med vår fråga och denna sparas som en int i ett ResultSet. Om lösenordet stämmer returneras 1 från metoden.



**Server.java**

Värdet som metoden ovan returnerar lagras sedan som ett heltal (state) i serverklassen, omvandlas till en sträng och skickas ut via port 2277.



**Login.java**

Metoden i knappens händelselyssnar-scoop läser inputen på på 2277 och inkommande data sparas som strängen state. Är lösenordet rätt, dvs 1, så öppnas programmet. Annars så meddelas användaren att lösenordet är fel och får försöka igen.

****