**OPPGAVE h)**

I denne oppgaven har vi forsøkt å kode etter 3 – lags arkitekturens prinsipper. De 3 lagene våre består av en SQlite database, et brukergrensesnitt og en Controller som fungerer som vårt bindeledd mellom databasen og brukergrensesnittet. For å forsikre persistens i databasen har vi brukt design mønsteret, Data Access Object Pattern, til å separere «low-level» data fra forretningslogikken. Vi har tatt i bruk «mock»-entitetsbønner – i form av klasser hentet fra UML-en og laget et tilhørende DAO, som implementerer et DAO-interface, til hver entitet. «Mock»-bønnene lagrer dataen midlertidig før de legges til i databasen og etter at en query henter ut data. På denne måten setter vi databasen utenfor applikasjonens rekkevidde, og med det gjort det mulig å endre database eller bytte til en annen form for lagringsteknologi.

Istedenfor å bruke Dao-Factory initialiserte vi DAO-objektene som final felt i Controller-klassen. Final sikrer at DAO-objektene ikke kan endres på. Som nevnt, skal Controller-klassen bygge opp forretningslogikken mellom brukergrensesnittet og databasen. For å kunne nå databasen og bruke metoder til å hente ut, legge inn, oppdatere eller lignende vil man være nødt til å gå gjennom DAO-objektene for å nå databasen. Noe av det vi fant mest nyttig ved å ta i bruk DAO og interfacet var at vi fikk definert et sett med kjørbare SQL statements som satt rammene for programmets funksjonalitet.

Hvorvidt det er andre utformingsmønstre som kunne gjort utviklingen enklere er vanskelig å si. Man kunne muligens tatt i bruk DAO-factory i større grad eller prøvd å følge MVC-prinsippene i stedet for 3 - lags strukturen. DAO-factory følte vi ikke var vits å bruke i større grad da det bare var 5 klasser. Vi er relativt fornøyd med implementeringen, men skulle gjerne ha klart å strukturere Controller-klassen bedre. GUI pakken kunne vært delt opp i flere underklasser.