Peer Review

Alexander Brunnegård, Julia Böckert, Malte Carlstedt, Steffanie Kristiansson,

Clara Simonsson

TDA367

Grupp ”Laget före jaget”

Chalmers tekniska högskola

Kodstruktur

Kodförståelse

Det är bra med en MVVM för er tilltänkta plattform. Ni följer det bra genom att Model inte dependar på något annat än det som är i model-mappen, men att view-model har models vilket är så detta designmönster fungerar.
Indelningen och namngivningen av packages gör koden lättare att förstå, märker att det är vissa saker som inte är helt klara än men man förstå vad syftet med klasserna är. En liten detalj är att en av packagena (analys) är på svenska medan alla andra är på engelska.

I koden finns tre metoder som heter: getTotalBudgetDB, getTotalBudgetThisMonth, getTotalBudgetPreviousMonth. Även om dessa metoder tillfredsställer dagens behov för applikationen, kan det vara fördelaktigt att implementera en mer mångsidig metod. En metod som med hjälp utav en extra parameter eller användning av typen DATE för att bestämma om användaren vill ha sin budget för denna månad eller föregående.

Abstraktion

Något att påpeka är att "CategorywithMoney" inte extendar Category, trots att det "är" en Category (istället för att "ha" en category). Om Liskov följs borde "CategorywithMoney" vara en äkta subtyp till Category, man kan bara flytta metoden "equals" till en separat klass, om det skulle vara ett hinder för principen att uppfyllas. Det är förvirrande att kategori-klass har ett "has-a"-förhållande till Category.

Namnet IDatabaseHandler förvirrar när det inte implementeras av DBHandler. Om DBHandler också är en DataBaseHandler bör interfacet implementeras, för att göra koden mer förståelig, och det skulle följa Liskovs. Det skulle även Däremot innehåller DBHandler saker som inte är relevant till interfacet, så i det fallet får man döpa om DBHandler till något annat.

Det är bra att det är mycket composition over inheritance, eftersom det blir en svagare dependency. IDataBaseHandler-interfacet bidrar till att både Dependency Inversion och Open-Closed-principle följs.

Design och dependencies

Eftersom koden är uppdelad i packages, är det lättare att se vad som är kopplat till vad. Däremot är det tveksamt om allt som ligger i package "Model" är Model-relaterad. Samma ang resten av packages namn, kanske förtydliga lite vilka som är View etc eftersom det tydligt ska framgå att det är MVVM som används, istället för MVC som man kan dela upp i model, view och controller tillexempel.

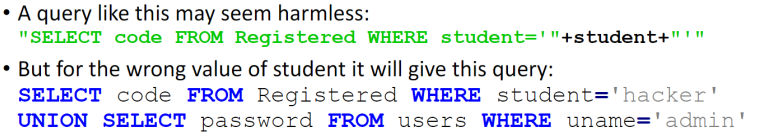
Efter att ha analyserat koden kunde vi inte hitta någon onödig eller dålig dependency, vilket ger koden mindre coupling och lättare att extenda. Detta bidrar till en hållbar kod för framtida utvecklare. Däremot saknas abstraktioner att extenda, förutom ett interface (IDataBaseHandler).

Det skulle underlätta om koden sågs över för att leta efter metoder och variabler att skicka upp i något mer abstrakt. Även fast de inte behövs just nu kan abstrakta klasser underlätta för framtiden och dessutom gör att koden ser mer ”extendable”ut!

Säkerhetsproblem

I databasen finns det metoder där strängar inte ”safety checkas” innan det görs en query (åtminstone inte vad vi kan hitta). På så sätt kan man enkelt avsluta denna query innan den egentligen får avslutas, alternativt skapa problem i databasen. När vi vill selecta genom att föra vidare till exempel ett namn i form av en textsträng, direkt till vår query, kan en användare som satt in ett *”; –”* någonstans i namnet avsluta denna query direkt, och på så sätt kommer ingen del av koden fungera så som den ska. Även kan ett namn på en vara, istället för att vara *”Ice cream”* kategoriseras under *”Ice cream; DROP DATABASE lalu-na.db; –”*, vilket då tar bort hela databasen. Osannolikt att detta skulle hända i detta projektets utsträckning, men det är ett problem om produkten skulle lanseras på detta vis.

Det finns det databasanrop som använder sig av så kallad SQL injection, dvs att en användare kan överlista systemet beroende på när den vill använda slutapostrofen för ett värde(1). Detta verkar vara en väl genomtänkt risk som tas, då de variabler endast kan erhålla vissa värden (till exempel en variabel av typen Date eller liknande), dvs att de inte kan innehålla vissa tecken. Kolla på möjligheten att använda er av PreparedStatement istället.



Figur 1: SQL Injection

Tester

Många tester kvar att göra men det vi la märke till i testerna för DBHandler och SqLiteHandler var att båda variablerna var döpta till db, vilket blev förvirrande när vi började läsa det andra testet och trodde att den hade samma db som den första eftersom det var samma namn. I övrigt ser testerna mycket bra ut!

Namngivning

Metoden ”equals” i klassen Expense bör byta namn, det förvirras lätt med andra metoder som heter equals i Java. Dessutom är det inte ett abstrakt eller beskrivande namn på en metod.

Sammanfattning

Sammanfattningsvis är koden bra dokumenterad och klasserna har tydliga namn. Koden är modulär i och med uppdelningen i packages, vilket bidrar till möjligheten att sätta sig in i koden och vidareutveckla den.

Generellt bra knackad kod kamrater!