Ansvar

Anvsarsområden var uppdelade i semantiska delar. Då var tre i gruppen blev uppdelningen som följer:

Jacob Bergqvist

- inputGetter
- views

Björn Andersson

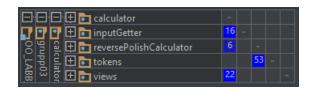
- tokens
- reversePolishCalculator

August Alexandersson

- calculator
- Delar av paketet tokens (som exempelvis allt med klassladdning och sådant att göra: OperatorModule, OperatorFactory, Math)

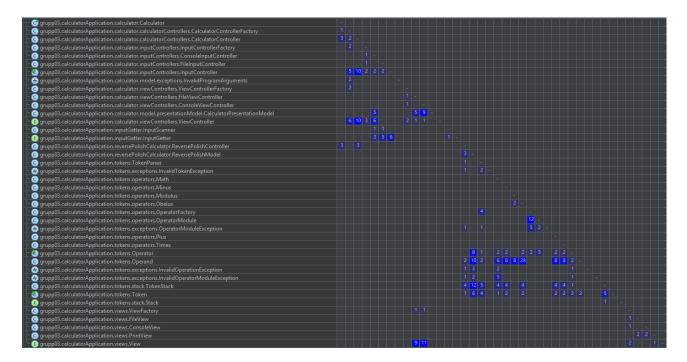
Beroenden, modularitet och kontroller

Projektet är består överst av paketet "calculatorApplication" som huserar alla nödvändiga delar för applikationen. Delarna är separerade för bättre modularitet och underhållbarhet (samt efter semantik). Detta *DSM* (Dependency Scope Matrix) visar på förhallandet mellan delarna eller "modulerna" (då de tekniskt sett inte är Javamoduler):



calculator erhar beroenden från alla paket utom tokens där endast paketet för algoritmen som direkt manipulerar klasser från tokens är inkluderat (i detta fallet är den enda algoritmen för att beräkna uttryck i Reverse Polish Notation och finns i reversePolishCalculator). De andra paket som har ett beroende från calculator är naturliga MVC element som kontroll (controller) för input och vyer (views). Inga cykliska beroenden finns utifrån denna nivå från detta DSM.

En större överblick avslöjar däremot inga cykliska beroende bland klasser från alla paket.



Calculator skapar en CalculatorController genom ett fabriksmönster som baserat på applikationens parametrar kommer generar en kontroll som arbetar mot en fil eller en mot en konsoll. Fabriken kommer alltså att beroende på argumenten initiera en ny CalculatorController med en instans från en vykontrollfabrik och en från en indatakontrollfabrik.

Vykontrollerna är ett lager av abstraktion som hanterar speciell logik som gäller för en speciell vy. Varje vykontroll ansvarar själv för att skapa en ny vy. Exempelvis så kommer vykontrollen för konsollen skapa en ny *ConsoleView* och skriva ut speciella meddelanen när användaren använder och stänger av programmet. Detta till skillnad från filvykontrollen som endast skriver ut resultat utan dessa meddelanden. På detta sättet kan exempelvis olika mönster för inmatning med olika meddelanden skapas som en ny kontroll och appliceras på en vy (som är ett grännsnitt alla vyer implementerar).

Vidare finns också en kontroll för indata som initieras med en referens till en vykontroll. Detta är för att denna indatakontroll förser vykontrollen med data som gäller innan viss data ska läsas in. Exempelvis guide text för hur indata ska formateras av användaren i fallet att indata kommer därifrån. Den abstrakta klassen *InputController*, som indatakontrollerna ärver av, kräver inte en vykontroll för en ny instans och detta implementeras endast av de klasser som behöver.