Tekniskt PM

# Introduktion

Under kursen TDP 028 har vi haft mycket fokus på modularitet, fragments ska kunna gå att återanvända och eftersom projektet jag arbetat med på ovanstående kurs är relativt stort skall det vara enkelt att testa vilket kräver bra struktur och modularitet. Några frågeställningar som jag har velat belysa är:

* Hur skapar vi modulär kod?
* Varför använder vi modulär kod?
* Hur kan modularitet användas i android?

# Hur skapar vi modulär kod

Något en programmerare alltid ska tänka på när hen skapar modulär kod är att moduler endast skall utföra en uppgift.[1] Om en modul heter ”display\_ui” så ska den modulen endast visa UI för användaren och inget annat. Moduler ska fungera självständigt utan att behöva veta hur en annan modul fungerar[2].

# Varför vi använder modulär kod

Kort sagt så blir koden organiserad när vi skapar modulär kod då en modul bara gör en sak[1]. Korrekt skriven modulär kod ska gå att återanvända utan att ändra någonting i modulen då den ska vara skriven på ett abstrakt sätt[1]. En stor fördel med modulär kod är att det gör det enkelt för flera programmerare att arbeta på ett gemensamt projekt då de kan skapa egna moduler för att sedan få modulerna att fungera tillsammans[1]. Modulär kod gör det också enklare att modifiera ett program då en programmerare bara behöver modifiera innehållet i en modul istället för hela programmet[3].

# Moduläritet i android

I en perfekt värld ska moduler fungera som små självständiga enheter[2] och sedan enkelt kunna sättas ihop till att fungera tillsammans, detta är hur fragments i android kan användas. Ett fragment för att visa en lista kan fungera så att den förväntar sig några argument, t.ex vad som ska visas i listan, och sen så gör den inget mer än att visa den listan på UI. Genom detta blir det enkelt att återanvända den till en annan activity där vi vill ha samma funktionalitet. På så sätt så får vi moduläritet i android.

[1] D. Daniel, B. Jane, och B. Mark, ”Untangling Parametric Schemata : Enhancing Collaboration through Modular Programming”, s. 14.

[2] D. T. Ross, J. B. Goodenough, och C. A. Irvine, ”Software Engineering: Process, Principles, and Goals”, *Computer*, vol. 8, nr 5, s. 17–27, maj 1975.

[3] O. Schoett, ”Data abstraction and the correctness of modular programming”, 1986.