SWD Spm 1 – SOLID 1

SRP, ISP og DIP

# SRP - Single responsibilty principle

En klasse bør kun have een grund til at ændre sig.

Eksempel med Rectangel

Klassen bør ikke indeholde funkt. til både draw og calc.

Dette gør at at fx GUI includen skal bygges samtidig med calc.

Hvis ændringer i enten GUI eller computational klassen ændre i rectangle skal det hele bygges, retestes osv igen

* Def. af responsibility

- Noget der kan give anledning til en ændring i klassen

Modem eksemplet: skal vi dele den op? Det kommer an på hvordan applikationen ændrer sig. Hvis connectiondelen ændres skal resten af klassen også rekompilere.

Hvis applikationen ikke ændre sig på disse to responsibility områder, bør vi ikke dele op - needless complexity.

"An axis of change is an axis of change, only if changes occur"

“Dont apply SRP if there is no symptom”

# ISP – Interface segregation principle

På samme måde som SRP handler ISP om at begrænse interfaces, spledes at de klasser som implementerer dem ikke skal tvinges til at implementere metoder den ikke bruger

# Redegør for hvordan du mener at principperne fremmer godt SW design

Alle tre design principper har deres fordele og nogen, deres ulemper.

## SRP

At separere klassers ansvar kan være en god ide, idet ethvert ansvar har dets egen ”Axis of change”. Flere ansvar i klasser medfører kobling blandt dem. Hvis en eneklt af disse ansvar skal ændres kan det medføre at resten af klassen bliver negativt påvirket.

Man skal også passe på ikke at distribuere funktionalitet ud så meget at man får ”needless complexity”. Hvis en klasse ikke har ”brug” for at ændre sig, er der ingen grund til at bruge SRP.

Et personligt eksempel er i faget ISU, hvor distribuerede et parkeringssystems funktionalitet ud på enormt mange små klasser, med enkelte metoder/attributter. Dette medførte en meget uoverskuelig og kompleks kode, og det positive ved SRP opvejede ikke længere unædig kompleksitet.

## ISP