SWD spm. 2 - SOLID 2

OCP, LSP og DIP

# OCP – Open closed principle

Open for extension, closed for modification

OCP siger at man bør refkatorere således at yderligere ændringer ikke skaber problemer i resten af programmet.

når OCP er "well applied" betyder det at vi kan tilføje ny kode uden at behøve at ændre i det gamle der i forvejen virker.

De 2 primære attributter

1-Open for extgension - Modulet kan udvides i takt med at krav udvides.

2-Closed for midification - Ved kun at udvide programmet, behøves vi ikka at røre ved exekverbare filer, DLL'er og library filer

**HOW TO OCP**

Abstraktion:

Brug abstrakte base klasser

Eksempel – Client der bruger Server

2 konkrete klasser er lort -> brug et interface (client)

## Perspektiv

Strategy pattern

# LSP – Lisskov Substitution Principle

Hvis S er en subtype af T, så skal objekter af typen T kunne erstattes af subtypen S.

**Eksempel:**

Hvis en Tesla er en specialisering af en bil, så bør jeg kunne bruge bilens ***drive()*** metode på teslaen.

Lad os sige at en base klassen bil, har en ***drive()*** og en ***shiftGearUp()*** metode. Teslaen kan sagtens implementere ***drive()*** metoden men fordi det er en *Tesla* får vi et problem med gearskiftet! (et ***throwExeption)*** vil bryde OCP! Et ***//Do Nothing*** er sikkert fint men ikke særligt sikkert. Altså er en Tesla **ikke** en bil i Barbara Liskovs optik ☺.

LSP handler dermed i bund og grund om ikke at bryde ”er en” kontrakten med klient koden, og altså lave et arvehieraki der opfylder en ægte specialisering.

# DIP – Dependancy Inversion Principle

A: High level modules should not depend on low level modules. Both should depend on abstractions

B: Abstractions should not depend upon details. Details should depend upon abstractions

Det der menes med abstraktioner her, er interfaces. Et eksempel på DIP er vores øvelse med compressionsStocking, hvor vi fx havde nogen knapper (high-level) der kaldte noget low-level funktionalitet (blinkende LED’er osv). I stedet for at gøre knappen afhængig af low-level funktionalitetet, lader vi den afhænge af et interface. Vis klassediagrammet for stockingCompression.

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

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

## OCP

## LSP

## DIP