Noter til "SOLID - L"

1. Introduktion til SOLID Principper

- Forelæser: Henrik Bitsch Kirk, Aarhus Universitet
- Institution: Institut for Elektroteknologi og Computerteknik

2. SOLID Akronym

- **S:** Single Responsibility Principle (SRP)
- **O:** Open Closed Principle (OCP)
- **L:** Liskov's Substitution Principle (LSP)
- **I:** Interface Segregation Principle (ISP)
- **D:** Dependency Inversion Principle (DIP)

3. Liskov's Substitution Principle (LSP)

- **Definition:** Hvis S er en subtype af T, kan objekter af type T erstattes med objekter af type S uden at ændre nogen af programmets ønskelige egenskaber.
- **Forklaring:** En underklasse (S) skal kunne bruges i stedet for en overklasse (T) uden at bryde programmets adfærd.
- **Eksempel:** Circle-Ellipsis problem, hvor cirkler ikke nødvendigvis kan erstatte ellipser korrekt på grund af forskellige præ- og postbetingelser.

4. Præ- og Postbetingelser

- **Præbetingelser:** Må kun erstattes med en lige så stærk eller svagere betingelse i underklassen.
- **Postbetingelser:** Må kun erstattes med en lige så stærk eller stærkere betingelse i underklassen.

5. Eksempel med Konto

- **SavingsAccount:** Basiskonto med grundlæggende funktioner.
- **CheckingsAccount:** Udvidet konto med overtræksgodkendelse.
- **Diskussion:** Vigtigheden af at forstå superklassens antagelser og betingelser for ikke at bryde programmet ved subclassing.

6. Alternativ til SOLID

- **CUPID:** Composable, Unix philosophy, Predictable, Idiomatic, Domain-based.
- Kritik af SOLID: Svært at anvende, nogle principper kan være vage eller svære at implementere korrekt.

7. Recap

- **L:** Liskov's Substitution Principle – Klienter af en klasse skal kunne bruge underklasser af den klasse uden problemer.

Noter til "SOLID - ID"

1. Introduktion til SOLID Principper

- Forelæser: Henrik Bitsch Kirk, Aarhus Universitet

- Institution: Institut for Elektroteknologi og Computerteknik

2. SOLID Akronym

- **S:** Single Responsibility Principle (SRP)
- **O:** Open Closed Principle (OCP)
- **L:** Liskov's Substitution Principle (LSP)
- **I:** Interface Segregation Principle (ISP)
- **D:** Dependency Inversion Principle (DIP)

3. Interface Segregation Principle (ISP)

- **Definition:** Interfaces skal være små og sammenhængende, hvilket giver to fordele:
  - Implementører behøver ikke implementere "dummy" metoder.
  - Forbrugere skal ikke tage højde for metoder, de ikke har brug for.
- **Eksempel:** Temperatur sensor interface opdelt i ITemperatureProvider og ITemperatureSensorCalibration for at undgå afhængighed af irrelevante metoder.

4. ISP Overtrædelse Eksempler

- **Consumers:** Overforbrugere tvinges til at håndtere unødvendige metoder.
- **Implementors:** Implementører tvinges til at implementere unødvendige metoder, der ikke giver nogen fordel.

5. ISP i den Virkelige Verden

- **C#'s List<T>:** Implementerer mange interfaces som ICollection<T>, IEnumerable<T>, IList<T>, etc., for at holde interfaces små og specifikke.

6. Dependency Inversion Principle (DIP)

- **Definition:** Højniveau moduler bør ikke afhænge af lavniveau moduler. Begge skal afhænge af abstraktioner.
- **Eksempel:** Environmental Control System (ECS) der regulerer temperatur i et drivhus ved at adskille høj- og lavniveau moduler.

7. DIP Eksempel - ECS

- **Problem:** Højniveau moduler afhænger af lavniveau implementeringer.
- **Løsning:** Anvendelse af abstraktioner til at adskille høj- og lavniveau logik.