Gustav Eriksson Söderlund
SUT24

# Reflektion avancerad labb 3

**Single-responsibility principle:**

Jag provade att införa repositories och DTO:s för att försöka hålla mig till SRP. Jag la märke till att Controller-klasserna också utförde konvertering mellan DTOs och Objekt, så jag införde en egen klass för att sköta det. Den borde däremot ha abstraherats och delats upp i fler klasser, eftersom den sköter konvertering av flera olika saker.

**Open-Closed principle:**

De flesta av klasserna "behöver" tyvärr fortsatt vara öppna för modifikation. En eventuell lösning hade varit att göra bas-interface för t.ex DTOs som sedan hade kunnat utökas vid behov.

**Liskov-Substitution principle:**

Det enda i koden som jag skrivit för programmet som skulle kunna bryta med Liskov är mina IRepository-interface. Men de ändrar för tillfället inget i funktionaliteten, så de skulle kunna fungera som sin "basform" också.

**Interface-Segregation principle:**

Denna bryter jag definitivt mot i applikationen, då jag försökte skapa ett generiskt interface för mina repositories. Det betyder att varje repo i nuläget implementerar flera metoder som de inte behöver använda. IRepository-interfacet skulle alltså kunna delas upp i flera, mer specifika interfaces (till exempel ett för att hämta data från flera tabeller, ett för varje CRUD-del etc.)

**Dependency inversion principle**:

Controller-klasserna är beroende av abstraherade Interface, vilket gör att vi enkelt skulle kunna byta ut dem mot en ny klass, om vi t.ex. skulle behöva byta databassystem. Man skulle också kunna injicera en eventuell abstraherad ResponseConverter i Controller-klasserna, i nuläget behöver man in och ändra vid ändringar i ResponseConvertern.