## **SPAC ugeopgave 2 Software udvikler - uge 5**

I de næste 2 uger skal i arbejde med lagerstyringssystemer. Du skal i samarbejde med dit team lave en præsentation af jeres kodeprodukt som skal fremlægges fredag d. 22.

**Formatet for fremlæggelsen:**

Præsentationen skal vare omkring 15 minutter, præsentationen skal lægge vægt på udvalgte nøgle-funktionaliteter og hvordan i har løst det at udvikle dem. Herefter bliver der sat 5 minutter af til feedback og spørgsmål. Det er valgfrit om man vil præsentere i PowerPoint eller om i vil vise jeres kildekode i jeres runtime environment/konsollen.

Husk at sætte tid af i løbet af ugen til at forberede præsentationen, samt at anvende de værktøjer i fik præsenteret i oplægget om feedback og præsentationsregler.

Præsentationerne skal ikke uploades!

Inden i går i gang med selve kodeopgaverne er det vigtigt at i fastlægger samarbejdet. Få en snak om hvordan i griber det an, opsæt jeres kanban-board, diskuter hvordan i arbejder bedst, og få lavet et udkast til en kravspecifikation sammen. Det er også vigtigt at få brudt opgaver op i passende størrelser, så i kan følge med i hvad hinanden udvikler og løbende koordinere udviklingsarbejdet.

**Opgaven:**

I skal i fællesskab skal udvikle et simpelt lagerstyringssystem ved hjælp af OOP-principper og anvende mindst to designmønstre: Singleton for databaseforbindelse og Factory til oprettelse af forskellige typer af varer.

* **Singleton Mønster for Databaseforbindelse:** Sikr, at der kun kan være én instans af databaseforbindelsesklassen i hele din applikation. Dette sikrer, at alle dele af din kodebase bruger den samme databaseforbindelse.
* **Factory Mønster til Oprettelse af Varer:** Implementer en Factory klasse, der skaber forskellige typer af varer baseret på input. For eksempel, hvis din lagerstyring håndterer tøj, kan Factory klassen skabe "T-Shirt", "Hoodie", eller "Hat" objekter baseret på det ønskede produkt.

**Foreslået Løsningsmetode:**

1. **Definer Klasser:** Start med at definere en baseklasse for varer med fælles egenskaber og metoder. Herefter oprettes specifikke underklasser for forskellige varer.
2. **Implementer Singleton:** Opret en klasse for din databaseforbindelse. Brug en privat konstruktor og en statisk metode til at håndtere instansoprettelse, sikring af kun én instans.
3. **Opret Factory Klasse:** Denne klasse skal have en metode, der accepterer en parameter, som angiver typen af vare, der skal skabes, og returnerer en instans af den ønskede vareklasse.
4. **Test Systemet:** Opret en driverklasse til at simulere systemet i aktion, opret forskellige varer gennem lageret, og anvend din Singleton for at sikre, at alle disse interaktioner bruger den samme databaseforbindelse.

**Yderligere Tips til Opgaveløsning:**

* **For Singleton:** Overvej at bruge en thread sikker implementering, hvis dit system forventes at håndtere flere tråde.
* **For Factory Mønster:** Tænk på at implementere en simpel id-generator for hver varetype, så hver ny vare får et unikt id, hvilket kan hjælpe med at spore varerne i systemet.
* **Integrationstest:** Sørg for at skrive nogle integrationstests, der tester interaktionen mellem din databaseforbindelse (Singleton) og vareoprettelse (Factory) med de specifikke varer.

Målet med denne opgave et at give dig en praktisk anvendelse af både designmønstre og OOP-principper, hvilket vil forbedre forståelsen af disse koncepter i en real-world kontekst. Ved at implementere disse mønstre styrker du softwarens design og gør det lettere at vedligeholde og udvide i fremtiden.