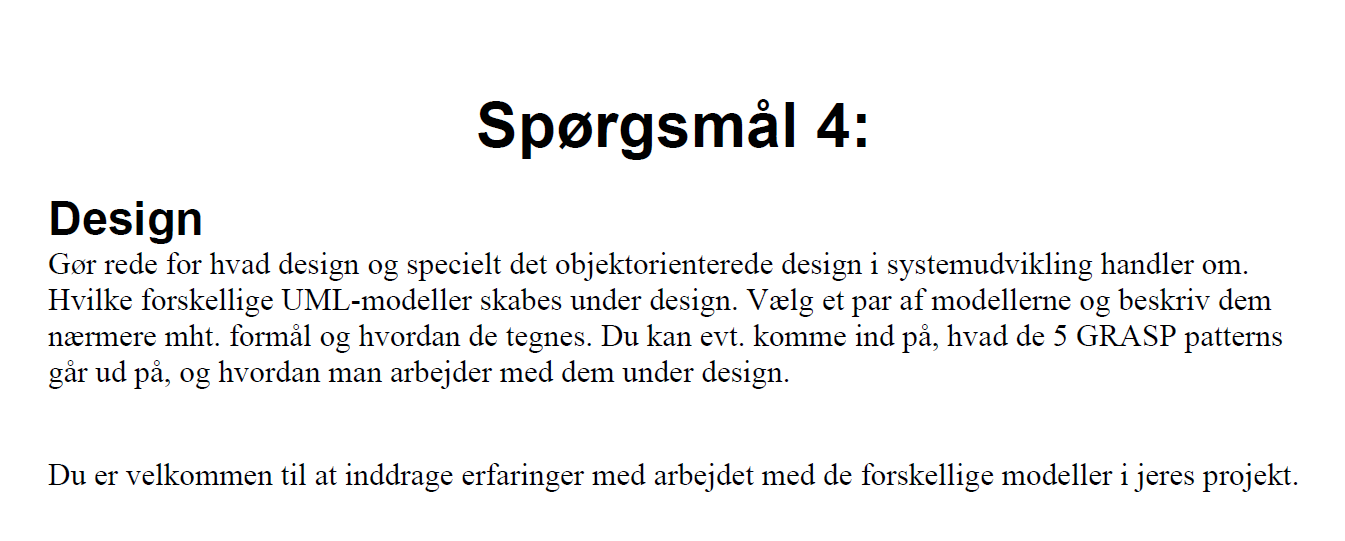
Spørgsmål 4



Inddrag Design klassediagram

Inddrag Design arkitektur.

Inddrag GRASP

Design klassediagram er en videreudvikling af analyse klassediagrammer. Her begynder det at ligne hvordan systemet faktisk er kodet.

De har følgende:

* Klasser
* Attributter - private/public
* Metoder - private/public
* Sammenhænge
* Multipliciteter
* Associationer
* Retninger
* Specielle klasser såsom enum’s, storage og controller.

Der har fjernet følgende fra analysen (associations klasser)

Arkitektur er hvordan du ønsker systemet skal opdeles og arbejde med sig selv.

Den standard arkitektur er 3 lags arkitektur Hvor du GUI i top som kun kan snakke med controller. Controller, samt model i midten hvor controller kun kan snakker med application og storage. Men ikke gui. Og til sidst storage som kun kan snakke med sig selv.

Denne arkitektur er fastlåst i og ved at der kun kan snakkes et tak ned.

GRASP Patterns står for General Responsibility Assignment Software Patterns patterns.

GRASP bruges til generelle ansvarsfordeling, fundamentale principper i design, og et ordforråd ved et højere abstraktionsniveau end kode.

Dets ansvar er sørge for en klasse stiller sig til rådighed overfor andre objekter ved at indkapsle data og skabe objekter/beregninger i andre objekter.

De patterns den bruger er Expert, creator, lav kobling, høj samhørighed og en controller.

1. Expert omhandler at tildele ansvar.
2. Creator er den der står for at oprette objekter af andre klasser.
3. Lav kobling er når en klasse har en attribut af en anden klasse.
4. Høj samhørighed er når en klasse har tilgang til mange andre klasser med færre metoder som er lette at genbruge. Altså en klasse der uddeleger arbejde.
5. Controller en klasse der sørger for at gui og application arbejder sammen og gemmer data til storage.
6. Brugbart: Tilpasningen af systemet til de organisatoriske, arbejdsmæssige og tekniske rammer
7. Sikkert: Sikringen mod uønsket adgang til systemets data og faciliteter
8. Effektivt: Udnyttelsen af faciliteterne i den tekniske platform
9. Korrekt: Opfyldelsen af de opstillede krav
10. Pålideligt: Opfyldelsen af den krævede funktionalitet med den ønskede præcision
11. Vedligeholdbart: Omkostningerne ved lokalisering og retning af fejl i det kørende system
12. Testbart: Omkostningen ved test af systemet i forhold til de opstillede krav
13. Fleksibelt: Omkostningen ved at ændre i det kørende system
14. Forståeligt: Besværet for udvikleren ved at skaffe sig overblik over og forstå systemet
15. Genbrugbart: Anvendeligheden af dele af systemet i andre beslægtede systemer
16. Flytbart: Omkostningen ved at flytte systemet til andre tekniske platforme
17. Integrerbart: Problemerne ved at sammenkoble systemet med andre systemer