Disainimustrid (GRASP)

Muster on probleemi / lahenduse paar, mida võib rakendada erinevates kontekstides. Mustril on nimi, mis seda identifitseerib.

GRASP (**G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns) disainimustrid pärinevad C. Larmani raamatust "Applying UML and Patterns" ja kirjeldavad objektide ülesannete / vastutuste jaotamise fundamentaalseid printsiipe.

Nimi: Ekspert (Expert)
Probleem: Milline on kõige üldisem ülesannete jaotamise printsiip objektorienteeritud disainis?
Lahendus: Ülesanne tuleb anda informatsiooni omavale eksperdile st. klassile, mis omab selle ülesande läbiviimiseks vajalikku informatsiooni.

Nimi: Looja (Creator)
Probleem: Millise klassi ülesandeks peaks olema mingi klassi eksemplari tekitamine?
Lahendus: Klassi B ülesandeks võiks klassi A tekitamise anda juhul, kui kehtib üks alljärgnevatest tingimustest:

- B *agregeerib* klassi A objekte
- B *sisaldab* klassi A objekte
- B-s *salvestatakse* klassi A objekte
- B *kasutab tihedalt* klassi A objekte
- B *omab* klassi A *initsialiseerimiseks vajalikke andmeid* (B on klassi A loomiseks **ekspert**)

Nimi: Mõõdukas seostatus (Low Coupling)
Probleem: Kuidas toetada mõõdukat sõltuvust ja korduvkasutatavust?
Lahendus: Ülesandeid tuleks jaotada nõnda, et klasside omavaheline seostatus jääks mõõdukaks.

Nimi: Kõrge kokkukuuluvus (High Cohesion)
Probleem: Kuidas hoida keerukus hallatavana?
Lahendus: Ülesandeid tuleks jaotada nõnda, et sama klassi ülesanded oleksid kokkukuuluvad.

Disainimustrid (GRASP) jätkub

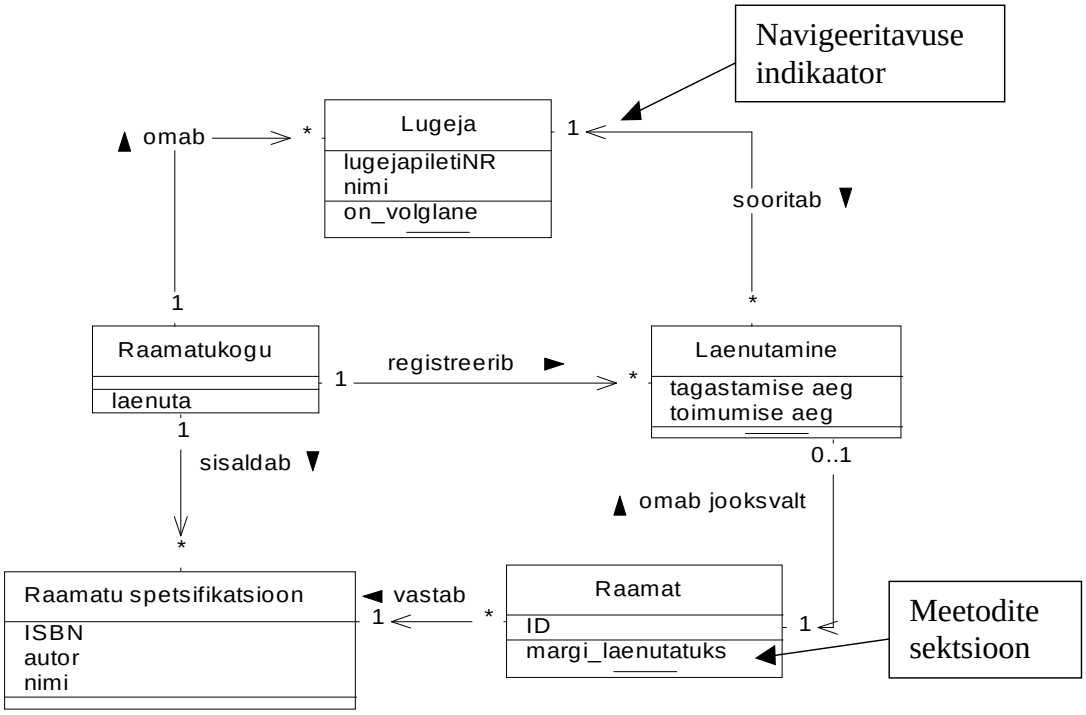
Nimi: Kontroller (Controller)
Probleem: Kes peaks vastutama süsteemivälisele sündmusele reageerimise eest?
Lahendus: Süsteemivälisele sündmusele reageerimise ülesannet peaks täitma klass, mis kuulub ühte järgnevatest kategooriatest:

- esindab terviksüsteemi (fassaadikontroller), näiteks “Süsteem”
- esindab tervikorganisatsiooni (fassaadikontroller), näiteks “Raamatukogu”
- esindab käitumist omavat reaalse maailma objekti (näiteks isiku rolli / töökohta), kes võiks seda ülesannet täita (rollikontroller), näiteks “Broneerija”
- esindab kunstliku ühe use case-i sündmuste haldajat (*handler*), tavaliselt nimetatud “<use case’i nimi>Haldaja / Handler” (use case’i kontroller), näiteks “Raamatute laenutamise haldaja”

Ülesannete jaotamist tuleks alustada ülesande täpse sõnastamisega.

Disaini klassidiagramm

Disaini klassidiagramm kirjeldab tarkvaraklasside ja liideste spetsifikatsioone ja erineb seega kontseptuaalmudelist, mis kirjeldab rakendusvaldkonna mõisteid.



Disaini klassidiagrammi näide

Disaini klassidiagrammi loomisel tuleks mõelda diagrammi kasutusalale.

- Kui diagrammi kasutatakse CASE vahenditega koodi genereerimiseks, siis on vajalik täpne ja kõikehõlmav detailsus.
- Kui diagramm on mõeldud tarkvaraarendajatele tutvumiseks võib liigne detailsus vähendada signaali / müra suhet.

Olulisemad sammud disaini klassidiagrammi koostamisel:

1. Leia interaktsioonidiagrammide abil kõik loogilise lahenduse seisukohalt olulised klassid ja joonista need klassidiagrammile.
2. Lisa atribuudid kontseptuaalmudeli vastavatest kontseptsioonidest.
3. Lisa meetodid analüüsides interaktsioonidiagramme.
4. Lisa assotsiatsioonid klasside vahel, mis kujutavad atribuudi-nähtavust (*attribute visibility*).
5. Lisa navigeerimisnooled, mis kujutavad atribuudi-nähtavuse suunda.