Oppgavesettet består av sider

# NORGES INFORMASJONSTEKNOLOGISKE HØGSKOLE INNLEVERING PG3300 Programvarearkitektur

Leveranseinfo:

* Skal gjøres i grupper på 2-3 personer.
* Leveres senest søndag 11. november 2012, klokka 23:55.
* Leveres via it's learning.
* Kode skal kompilere og kjøre i Visual C# 2010 Express.
* Tekstdokumentasjon skal være i docx eller pdf format.   
  (UML kan være bildeformat, inklusive screenshot.)

­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Oppgave 1: UML og refactoring av Snake

Se den vedlagte kodefila: SnakeMess.cs

Denne fila inneholder et (for de fleste) kjent spill. Koden kompilerer og kjører, men er ikke særlig lesbar, utvidbar, eller gjenbrukbar.

Denne oppgaven går ut på å refaktorere SnakeMess.cs til bedre arkitektur og kode! Faktisk så bra som dere er i stand til å skrive den. :-)

Tenk god struktur, ikke nødvendigvis hva som hadde vært praktisk for et så lite prosjekt i arbeidslivet. (M.a.o. ta det dere har lært i dette emnet helt ut, ikke lag forenklede løsninger fordi det er lite kode totalt sett).

"Ta det dere har lært i dette emnet helt ut " inkluderer: (men er ikke begrenset til)

* Bytt ut variabel-, klasse- og metodenavn.
* Splitt kode i flere klasser og metoder.
* Implementer arv(?).
* Tenk GRASP.
* Bruk en eller flere design patterns.
* Se forøvrig refactoring lista i forelesning 8.

Diskuter muligheter innad i gruppen. Noter ned stikkord for fremgangsmåte (rekkefølge på oppgaver, hva dere prioriterte), og begrunn valgene deres (f.eks. hvorfor dere valgte variant A fremfor B, der hvor dere kom opp med flere muligheter).

## Oppg. 1 a)

Sett opp UML skjema(er) for hvordan dere tenker dere den refaktorerte løsningen:

1. Implementation Class Diagram er et minimum.
2. Bruk flere av diagrammene vi har lært om hvis dere synes det virker hensiktsmessig.

Husk at ved refactoring skal programmet oppføre seg likt for brukeren før og etter! M.a.o. funksjonaliteten i programmet skal være den samme før og etter endringene.

## Oppg. 1 b)

Implementer den nye arkitekturen i Visual C#.

(Eller sørg i alle fall for at dere bygger og leverer en fungerende Visual C# solution m/kodefiler mot slutten av prosjektet, om dere jobber i et annet verktøy underveis.)

## Oppg 1 c)

Med bakgrunn i notatene, skriv et tekstvedlegg der dere gjør rede for valgene og prosessen deres. Husk at dette er en del av sensuren, så det er viktig at dere "selger" løsningen deres til sensor.

Med andre ord: I dette tekstvedlegget, sørg for å få fram hvorfor akkurat deres løsning er så bra, og vis (med ord, evt. figurer) at dere har tenkt, planlagt og resonert dere frem til gode valg underveis for løsningen deres.

# Oppgave 2: Multi-threading

Denne oppgaven offentliggjøres ifbm. forelesning 9, (dokumentet oppdateres med den nye oppgaven da), siden multi-threading er tema for hele forelesning 9.

Kort fortalt blir det en ny (kode-)oppgave (bygger *ikke* videre på snake, over) som tar for seg aspekter ved multi-threading.