

Oppgavesettet består av 6 sider

## WESTERDALS OSLO ACT INNLEVERING PG3300 Programvarearkitektur

Leveranseinfo:

- Skal gjøres i grupper på 2-3 personer.
  - Leveres senest søndag 20. november 2016, klokka 23:59.
  - Leveres via it's learning. (Lever litt før fristen – det kan ta tid å laste opp filer!)
  - Kode skal skrives i C#, og kompilere og kjøre i Visual Studio Enterprise 2015.
  - UML diagrammer kan være Visual Studio eller bildeforformat (bildeforformat inkluderer screenshot).
  - Tekstdokumentasjon skal være i pdf format.
- 

### Oppgave 1: UML og refactoring av SnakeMess

Se vedlagt solution/kodefil for SnakeMess.

Kodefila 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.

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

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

- Bytt ut variabel-, klasse- og metodenavn med mer beskrivende varianter.
- Splitt kode i flere (mange) klasser og metoder. (Og kanskje bytte ut noen eksisterende elementer med standardelementer fra .NET bibliotekene?)
- Implementere arv(?).
- Tenk GRASP (få med uttrykkene i dokumentasjonen der det er relevant!)
- Se om det kan være fornuftig å benytte design pattern(s).
- Se forøvrig refactoringlista i forelesningsmaterialet.
- Benytt parprogrammering (i alle fall til deler av arbeidet), og reflekter over dette i dokumentasjonen.

Diskuter muligheter innad i gruppen. Skriv i tekstdokumentasjonen fremgangsmåte (rekkefølge på punkter, hva dere prioriterer) og begrunn valgene deres (f.eks. hvorfor dere velger variant A fremfor B, der hvor dere kommer opp med flere muligheter). Dette skal bli en beskrivelse av prosessen som legges ved resten av innleveringen!

**Merk #2:** Refactoring av et dårlig strukturert prosjekt betyr ofte at koden vokser litt. I et prosjekt med så få kodelinjer som SnakeMess, betyr det at løsningen vokser prosentvis MYE. Dette er helt ok! :-)  
(I skolesammenheng – i arb.livet er det ikke like stas å gjøre koden dobbelt så stor for at den skal bli "penere" å jobbe med. Men på en større kodebase vil størrelsen endre seg prosentvis lite.)

#### Oppg. 1 a)

Sett opp et Implementation Class Diagram ("UML Class diagram" i Visual Studio) som viser hvordan dere tenker dere ferdig refactored versjon.

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

#### Oppg. 1 b)

Implementer den nye arkitekturen/koden i Visual Studio.

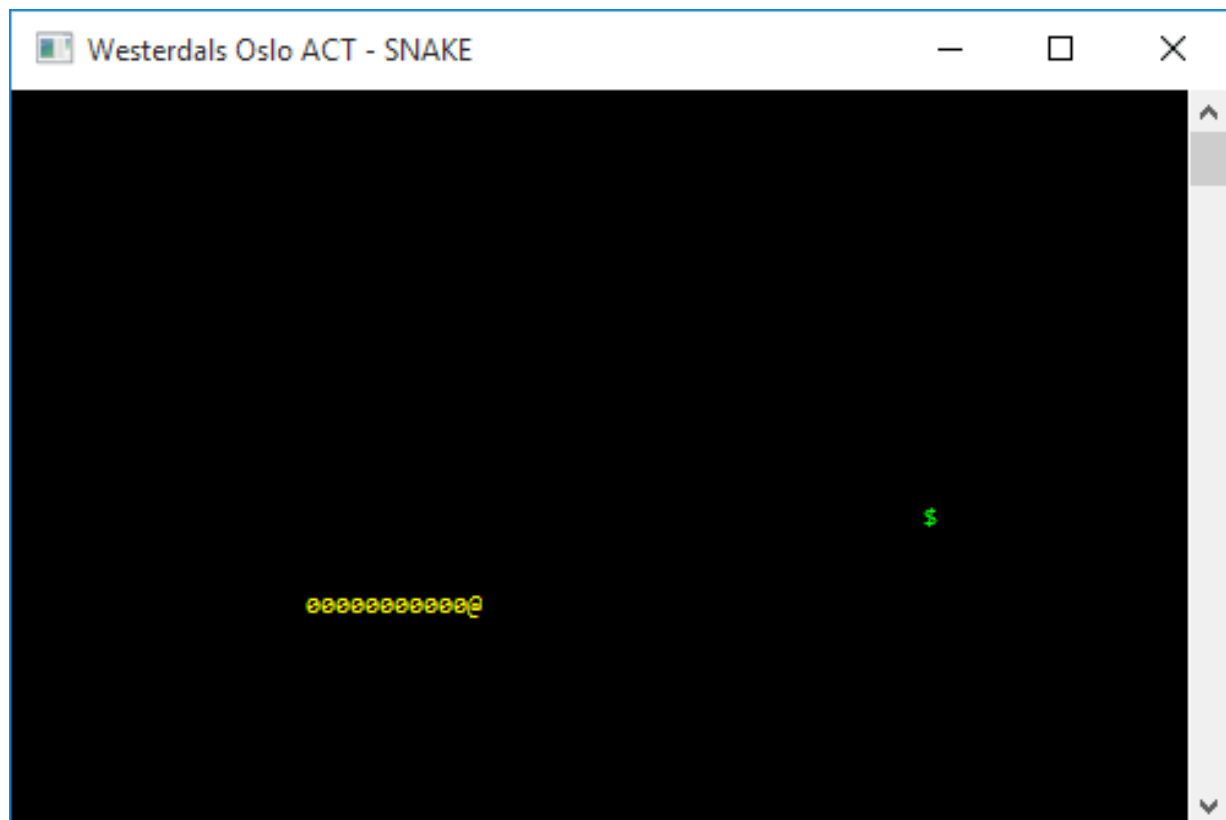
(Eller egentlig: sørg for at dere bygger og leverer en fungerende Visual Studio 2015 C# solution m/kodefiler – om dere vil jobbe i et annet verktøy underveis er det lov.)

#### Oppg 1 c)

Ferdigstill tekstdokumentasjonen (pdf) der dere gjør rede for valgene og prosessen deres. Begrunn hvordan design guidelines og evt. design patterns har påvirket resultatet deres. Husk at dette er en del av sensuren: Det er viktig at dere forklarer løsningen og valgene deres for sensor!

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

Snake - slik det skal se ut når det kjører, både *før* og *etter* refactoring:



## Oppgave 2: Forklaring av Multithreading

Forklar følgende begreper i forbindelse med multithreading: (med utgangspunkt i C#, om det håndteres forskjellig i forskjellige språk)

### Oppg. 1 a) Race conditions

Hva er race conditions?

Hvordan kan disse oppstå?

### Oppg. 1 b) Locks

Hva er locks? Hva slags objekter er egnet som locks?

Hva er forskjellen på static og local locks?

Hvordan benytter vi locks i C#?

### Oppg. 1 c) Deadlocks

Hva er deadlocks?

Hvordan kan disse oppstå?

## Oppgave 3: The Cookie Bakery

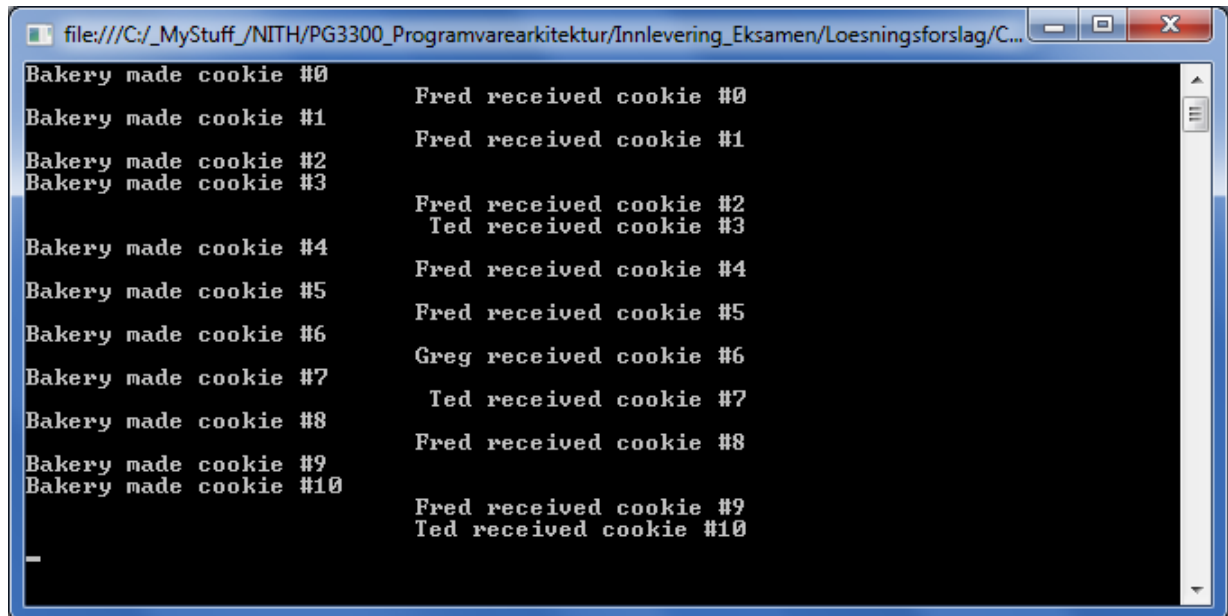
"The Cookie Bakery" er et lite bakeri med tre faste stamkunder: De glupske herrene Fred, Ted og Greg.

The Cookie Bakery lager kun cookies (of course), og selv om produksjonen kan være nokså lav (minimum et dusin cookies om dagen, utover det velger dere selv) hindrer ikke det Fred, Ted og Greg fra å storme mot disken jævnt og trutt for å se om det ligger nye cookies der! (Rekkefølgen ferdige kaker legges ut for salg i trenger ikke være lik rekkefølgen de stekes i.) Men uansett hvor samtidig herrene kaster seg mot disken, er det jo bare én som kan få tak i hver cookie, ikke sant?

Bakeriet putter sine cookies i en kurv på disken etterhvert som de er ferdig bakt (for eksempel hvert 667. millisekund). De holder åpent til de har bakt en dagsvorte og denne er solgt unna.

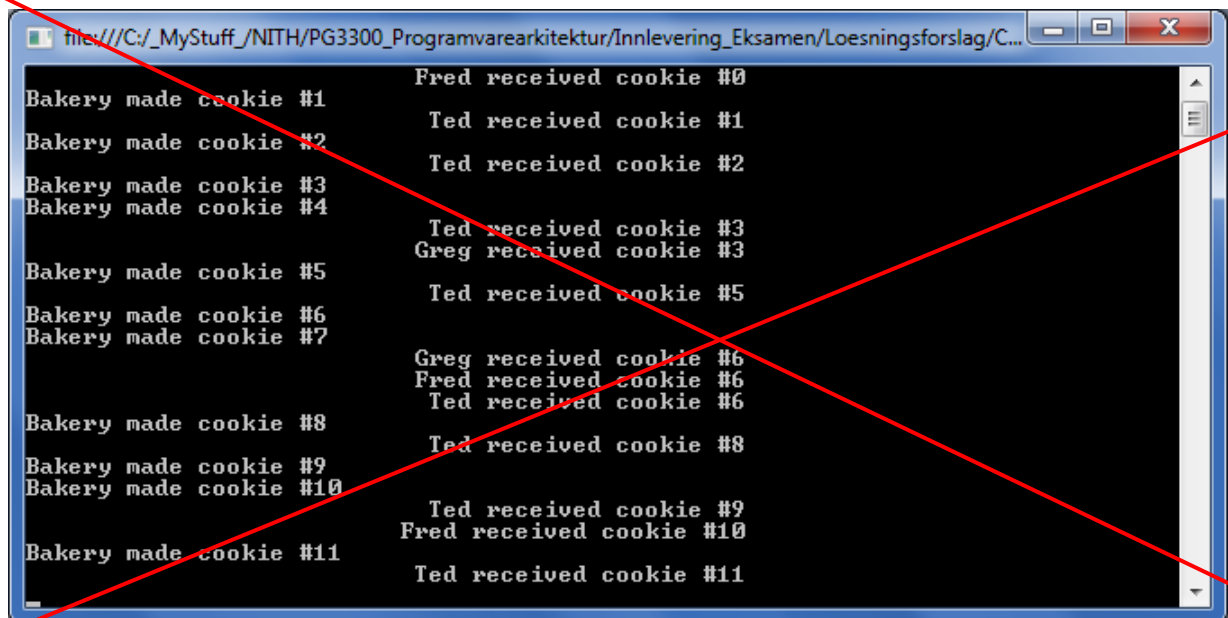
Fred, Ted og Greg prøver alle å få tak i cookies ved javne mellomrom! Implementer funksjonalitet som gjør de i stand til å prøve å plukke opp kaker samtidig (multithreading), men sørg også for at koden er trådsikker slik at bare én kan få fingrene i hver kake som kommer. (Teknisk: Bakeriet selger kakene gjennom en egen metode – `"SellCookieTo(Customer customer)"` eller tilsvarende. Fred, Ted og Greg kaller denne metoden jevnlig gjennom hver sin thread, for eksempel en gang per 1000 millisekunder - velg et tall som er litt høyere enn for bakingen.)

Eksempel på kjøring – i dette tilfellet er Fred helt klart den beste cookie knabberen:



```
file:///C:/_MyStuff/_NITH/PG3300_Programvarearkitektur/Innlevering_Eksamen/Loesningsforslag/C...
Bakery made cookie #0      Fred received cookie #0
Bakery made cookie #1      Fred received cookie #1
Bakery made cookie #2      Fred received cookie #2
Bakery made cookie #3      Ted received cookie #3
Bakery made cookie #4      Fred received cookie #4
Bakery made cookie #5      Fred received cookie #5
Bakery made cookie #6      Greg received cookie #6
Bakery made cookie #7      Ted received cookie #7
Bakery made cookie #8      Fred received cookie #8
Bakery made cookie #9      Fred received cookie #9
Bakery made cookie #10     Ted received cookie #10
_
```

Som i skjermbildet under skal det **ikke** se ut! Flere herrer kan nemlig ikke spise den samme kaka! (Ref. cookie #3 og cookie #6.)



```
file:///C:/_MyStuff/_NITH/PG3300_Programvarearkitektur/Innlevering_Eksamen/Loesningsforslag/C...
Bakery made cookie #1      Fred received cookie #0
Bakery made cookie #2      Ted received cookie #1
Bakery made cookie #3      Ted received cookie #2
Bakery made cookie #4      Ted received cookie #3
Bakery made cookie #5      Greg received cookie #3
Bakery made cookie #6      Ted received cookie #5
Bakery made cookie #7      Greg received cookie #6
Bakery made cookie #8      Fred received cookie #6
Bakery made cookie #9      Ted received cookie #6
Bakery made cookie #10     Ted received cookie #8
Bakery made cookie #11     Ted received cookie #9
Bakery made cookie #11     Fred received cookie #10
Bakery made cookie #11     Ted received cookie #11
_
```

**Ekstra:** I eksemplene på forrige side er alle cookies av samme type. Men det veier positivt på vurderingen om man lager en mer omfattende løsning. Implementer gjerne flere typer cookies, f.eks. cookies med sjokoladebiter, med rosiner, osv. (har vi kanskje lært om ett eller flere patterns som kan benyttes til dette?). Kanskje også større dagsrasjoner og/eller flere utsalgssteder?

Eksempel på kjøring av [mer avansert](#) cookie bakery:

```
file:///C:/_MyStuff/_Westerdals Oslo ACT/PG3300_SoftwareDesi...
Granny's made cookie #1
    Greg received Granny's Cookie #1 with Chocolate chips
Chuckie's Shack made cookie #1
Granny's made cookie #2
Chuckie's Shack made cookie #2
    Ted received Granny's Cookie #2 with Vanilla
    Fred received Chuckie's Shack Cookie #1 with Vanilla
    Ted received Chuckie's Shack Cookie #2 with Chocolate chips
Granny's made cookie #3
Chuckie's Shack made cookie #3
    Ted received Granny's Cookie #3 with Chocolate chips
    Ted received Chuckie's Shack Cookie #3 with Chocolate chips
Granny's made cookie #4
Chuckie's Shack made cookie #4
Granny's made cookie #5
Chuckie's Shack made cookie #5
    Ted received Granny's Cookie #4 with Vanilla
    Ted received Chuckie's Shack Cookie #4 with Vanilla
    Fred received Chuckie's Shack Cookie #5 with Vanilla
    Fred received Granny's Cookie #5 with Vanilla
Granny's made cookie #6
Chuckie's Shack made cookie #6
    Ted received Granny's Cookie #6 with Chocolate chips
    Ted received Chuckie's Shack Cookie #6 with Chocolate chips
Granny's made cookie #7
Chuckie's Shack made cookie #7
Granny's made cookie #8
Chuckie's Shack made cookie #8
    Ted received Granny's Cookie #7 with Vanilla
    Fred received Granny's Cookie #8 with Chocolate chips
    Greg received Chuckie's Shack Cookie #7 with Vanilla
    Ted received Chuckie's Shack Cookie #8 with Chocolate chips
Granny's made cookie #9
Chuckie's Shack made cookie #9
    Ted received Granny's Cookie #9 with Chocolate chips
    Greg received Chuckie's Shack Cookie #9 with Chocolate chips
Granny's made cookie #10
Chuckie's Shack made cookie #10
Granny's made cookie #11
Chuckie's Shack made cookie #11
    Ted received Granny's Cookie #10 with Chocolate chips
    Fred received Granny's Cookie #11 with Vanilla
    Greg received Chuckie's Shack Cookie #10 with Chocolate chips
    Fred received Chuckie's Shack Cookie #11 with Vanilla
Granny's made cookie #12
Chuckie's Shack made cookie #12
Granny's made cookie #13
    Ted received Granny's Cookie #12 with Vanilla
Chuckie's Shack made cookie #13
    Fred received Granny's Cookie #13 with Vanilla
    Greg received Chuckie's Shack Cookie #12 with Vanilla
    Fred received Chuckie's Shack Cookie #13 with Vanilla
Granny's made cookie #14
```

### Oppg. 3 a) Dokumentering

Diskuter muligheter innad i gruppen, og begrunn valgene deres (bl.a. hvorfor dere velger variant A fremfor B, der hvor dere kommer opp med flere muligheter). Dette skal danne grunnlaget for et dokument (pdf) der dere gjør rede for valgene og prosessen deres. Husk at dette er en del av sensuren: Det er viktig at dere forklarer løsningen deres for sensor!

Med andre ord: I dette tekstvedlegget, sørg for å få fram hvorfor akkurat deres løsning er så bra, dvs. vis (med ord og evt. figurer) at dere har gjort gode valg for løsningen deres! (Som for oppgave #1.)

Sørg også for å begrunne eventuelle design patterns dere benytter, samt hva dere har gjort for å følge design guidelines/GRASP.

### Oppg. 3 b) Modellering

Benytt UML og sett opp et use case diagram for bakeriet.

Når du er ferdig med kodingen, generer Implementation class diagram i Visual Studio for koden din.

### Oppg. 3 c) Implementering

Implementer "The Cookie Bakery" som en multithreaded applikasjon der Fred, Ted og Greg kniver om å få tak i cookies!

Benytt parprogrammering til (minst deler av) implementeringen på denne oppgaven også.

Merk – som for oppgave #1: Tenk god struktur også her. Altså GRASP, design patterns, m.m.

- Slutt på oppgavesettet -