Oppgavesettet består av 4 sider

HØYSKOLEN KRISTIANIA – EKSAMEN
PG3300 Software design

Leveranseinfo:

- Skal gjøres i grupper på **2-3 personer**.
- Dette er en **sammensatt eksamen**, delene (prosjekt og skriftlig) beskrives samlet.
- Delene leveres samlet, senest **mandag 19. november 2018, klokka 09:00**.
  Lever litt før fristen: det kan ta tid å laste opp filene!

Skriftlig eksamen (tekstdokumentasjon)

- Dette er dokumentasjonen (av prosessen og løsningen). Skal være i pdf format.

Prosjekteksamen (kode og kjørbart program)

- Dette er selve modelleringen, koden og programmet. Skal skrives i C#. Dere velger IDE selv, men av sensurhensyn må innleveringen kompilere og kjøre i Visual Studio Enterprise 2017.
- UML diagrammene kan inkluderes i pdf dokumentasjonen over, eller legges ved som bildefiler.
- Legg også ved ferdig bygget og kjørbart program (.exe fil fra build prosessen).

Oppgave: Simulasjon av loppemarked

Ut i fra pensum, hva bør med
Oppgaven går ut på å designe en løsning basert på hva vi har lært om UML, GRASP, design patterns, m.m. Deretter programmere løsningen i C#.

Merk: Tenk god struktur, ikke nødvendigvis hva som hadde vært praktisk for et så lite prosjekt utenfor skolesammenheng. Prøv å få med flest mulig elementer fra pensum. Gjør dere noe som ikke virker logisk (fordi dere vil vise at dere behersker en gitt del av pensum, men finner ikke en perfekt plass å vise den på), forklar hvorfor dere har det med både med kodekommentar og i deres vedlagte pdf dokument.

Noen hovedtemaer dere bør ta stilling til:

1. Benytt flere typer UML diagrammer!
2. Tenk GRASP (og beskriv punktene dere har hatt fokus på i dokumentasjonen).
3. I denne oppgaven bør det være mulig å benytte flere design patterns av de vi har lært om.
4. Multithreading bør med i denne oppgaven. Men pass på at dere unngår kode som ikke er trådsikker!
5. Unit testing passer fint inn, i alle fall på deler av løsningen. (Viser dere at dere mestrer unit testing for noen av kodefilene, og kommenterer dette i dokumentasjonen, får dere full uttelling uten å implementere unit testing for øvrige kodefiler. Velg kodefiler som gjør at dere får vist full beherskelse av Unit testing.)
6. Sørg for at koden er oversiktlig og lett å sette seg inn i. Dvs. gode variabel-, metode- og klassenavn, samt et ryddig oppsett.
7. Dere SKAL benytte parprogrammering på denne oppgaven! I alle fall på deler av arbeidet. Beskriv i tekstdokumentasjonen dere legger ved hvor dere har benyttet parprogrammering, og reflekter over resultatet.
8. Diskuter muligheter innad i gruppen. Skriv i tekstdokumentasjonen fremgangsmåte (rekkefølge på punkter, hva dere prioriterte) og begrunn valgene deres. Dette skal bli en beskrivelse av prosessen og er en del av vurderingsgrunnlaget i innleveringen.

Beskrivelse av domene/business logic
NB: Fokuser på funksjonaliteten beskrevet under, ikke et ekte loppemarked.

"Lottas Lopper" (eng: "Lottas Flea Market") er et (fiktivt) loppemarked. De har en digital løsning der alle varer registreres elektronisk (på selgeren som legger de ut), det samme gjør alle kjøp (hvem som kjøper varen, ikke pris).

Et loppemarked baserer seg på hva folk flest har hatt hjemme, som de ønsker å kvitte seg med. Varene er avarter av samme utgangspunkt. Men så har kanskje enkelte varer noen riper, mangler eller hjemmesnekra tilpasninger.

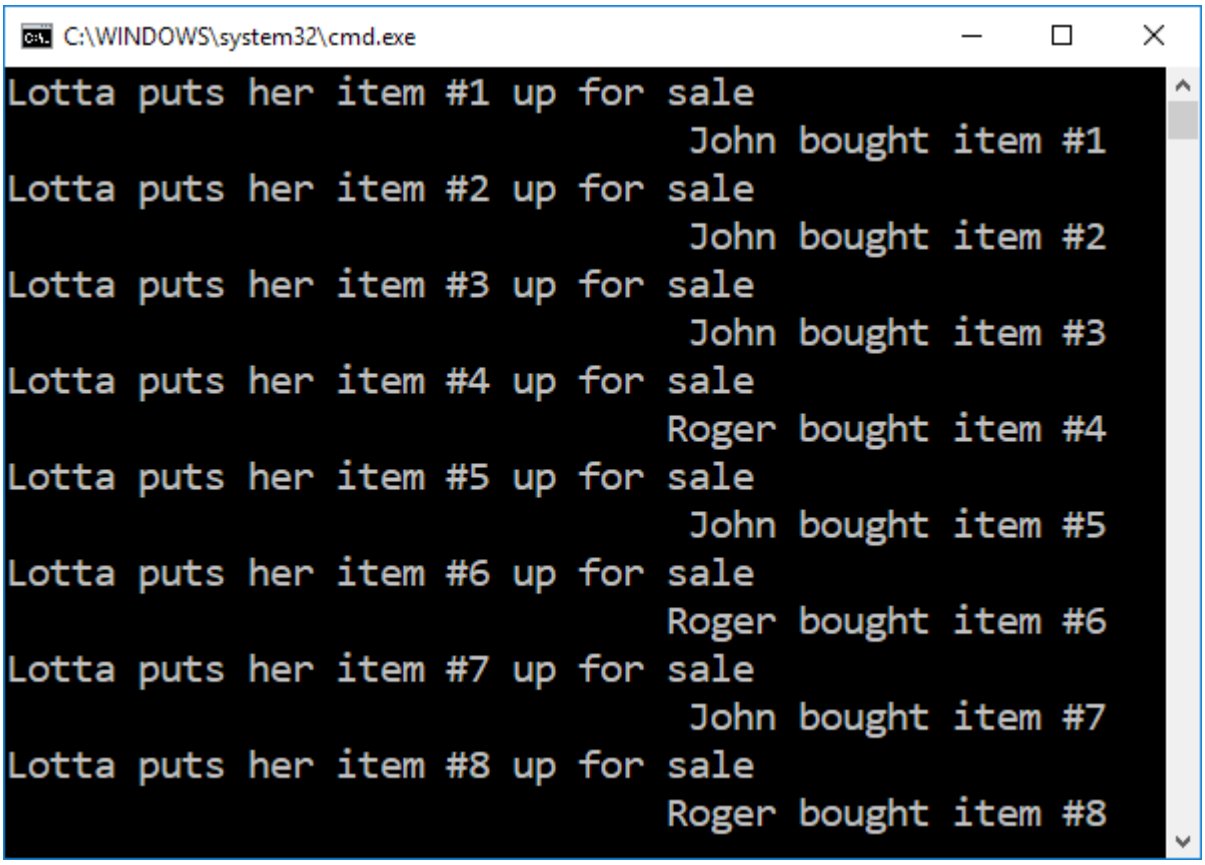
På travle dager er det flere selgere som legger ut gjenstander parallelt. Selgerne står klare med hver sin haug lopper og putter disse ut for salg i høyt tempo. Loppemarkedet holder åpent til alle selgere har lagt ut alle sine varer, og alle disse varene deretter er solgt.

Lottas Lopper er veldig populær! Kundene (i flertall) elsker lopper, de har god råd og de handler svært ukritisk – de nærmest sloss om å kjøpe varene. Tilsynelatende kaster de seg over samme vare, men i en godt implementert løsning vil bare én kunde kunne "vinne" en gitt vare.

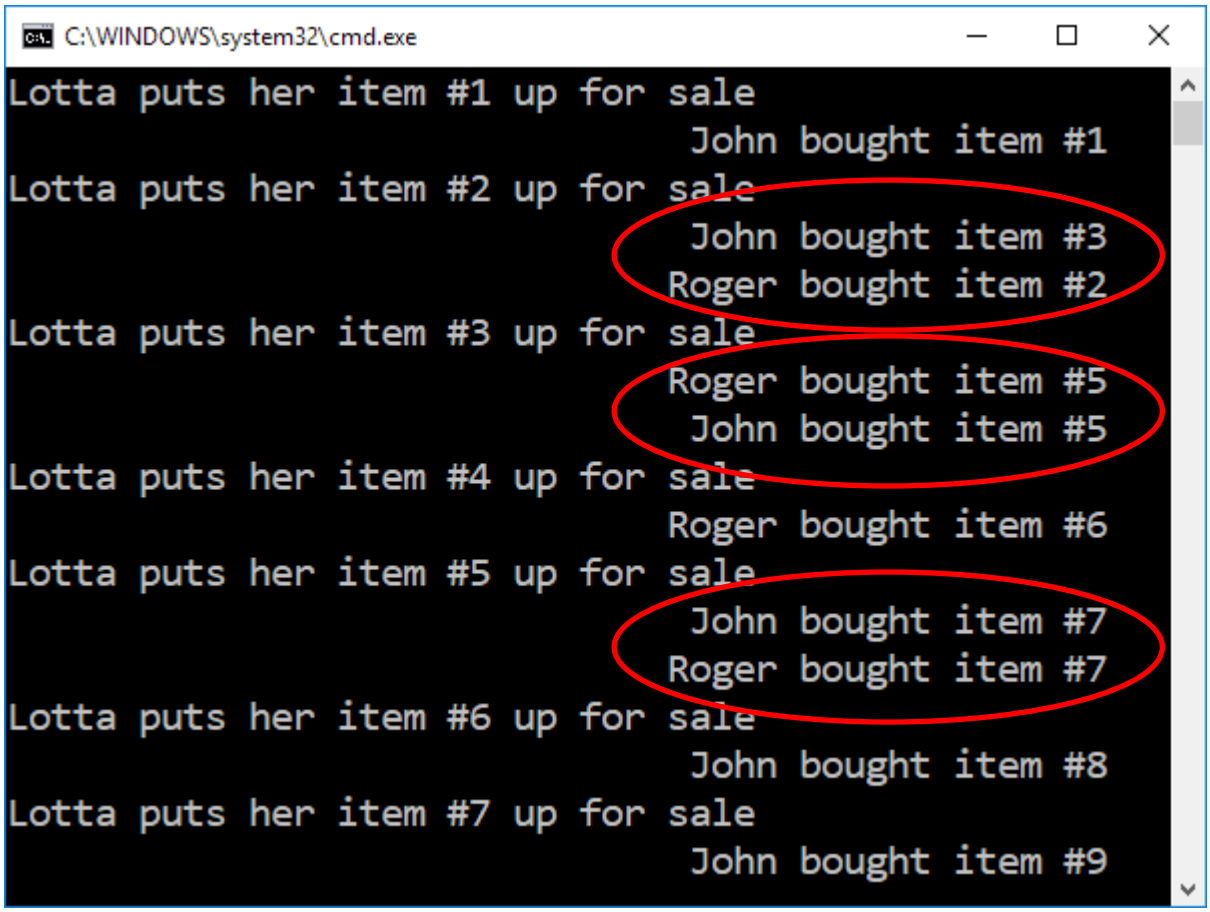
Implementer helst funksjonalitet som gjør kundene i stand til å *prøve* å kjøpe samme vare samtidig, men sørg for at koden er sikret slik at bare én kan få fingrene i hver vare.

Eksempler på utskrift

Eksempel, relativt simpel men fungerende løsning
Én Selger, to kunder, ikke flere varekategorier. Løsningen er trådsikker:

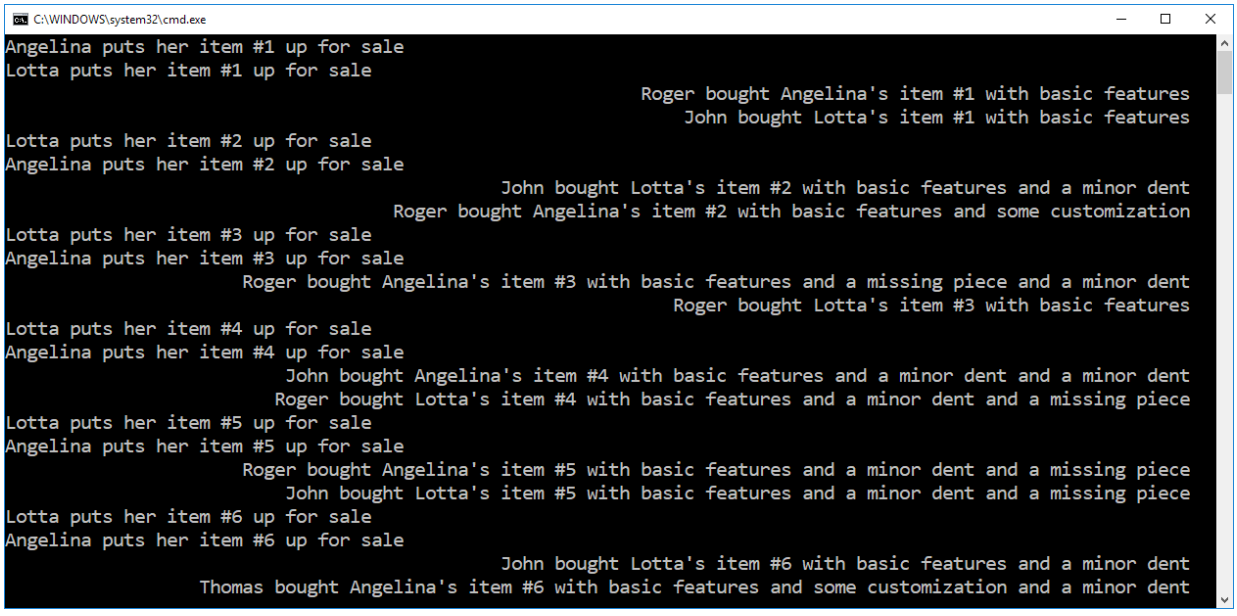


Eksempel, liknende løsning som over, men ikke trådsikker
Som dette skal det ikke se ut! Flere kunder kan nemlig ikke kjøpe samme vare! (Ref. `item #5` og #7.)
De kan heller ikke kjøpe varene *før* de legges ut for salg! (Ref. salg av `item #3` og utover.)



Eksempel, mer avansert loppemarkedløsning:
2 butikker, 3 kunder, varer med et knippe tilpasninger.

(NB – feature, ikke bug: 1) samme "tilpasning" kan forekomme flere ganger på en vare. 2) varene nummereres ut fra selger, ikke totalt sett. så begge selgere har en `item #1`, en `item #2`, osv.)



Forslag til rekkefølge på løsning

a) UML design
Benytt UML og sett opp et use case diagram for deres loppemarked.

Lag deretter UML class diagram.

Disse skal implementeres i pdf dokumentet som er del av besvarelsen.

b) Modellering
Tenk GRASP prinsipper samt mulige design patterns.

Vurder multithreading og fordeler/ulemper med dette.

Dokumenter valgene deres.

Tips: Se over sjekklista fra side 1 en gang til – dere har vel ikke glemt noe?

c) Implementering
Implementer koden til deres design og modell!

Benytt parprogrammering til (minst deler av) implementeringen.

Husk: Tenk god struktur på koden.

Et par små implementeringstips:
* C# sin Random funksjonalitet er i seg selv ikke trådsikker, løsninger på dette kan googles.
* Det er greit å benytte Sleep for å simulere at deler av løsningen tar tid/få tiden til å gå.

d) Unit testing
Unit testing bør gjøres underveis i utviklingen. Så gjør denne i parallell med c, over.

e) Ferdigstilling
Ferdigstill tekstdokumetasjonen (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 og valgene deres for sensor!

- Slutt på oppgavesettet -