Prosjektoppgave

i «IMT1082 - Objekt-orientert programmering» våren 2019

Frister: ***Mandag 8.april 2019 kl.09:00***
NB: *Fredag 15.mars kl.09:00 (1.delinnlevering)*
Onsdag 27.mars kl.09:00 (2.delinnlevering)
Arbeidsform: *Gruppe (tre (evt. to) personer – flere er ikke relevant)*
Arbeidsinnsats: ***Mye***

Innledning

Dere skal i denne prosjektoppgaven lage **et noe større program som holder orden på ulike kunders kjøp av billetter til ulike arrangementer på ulike spille-/arrangementsteder. Ett og samme spillested kan ha flere (stol)oppsett, hvert bestående av en eller flere soner.**

I hovedsak skal programmet håndtere følgende operasjoner:

- **Legge inn / endre / skrive** / (slette) **en kunde**
- **Legge inn / skrive** / (slette) **et spille-/arrangementsted**
- **Legge inn / endre / skrive** / (slette) **(stol)oppsett på et spille-/arrangementsted**
- **Legge inn / skrive / slette** / (endre) **kjøpe billett(er)** / (returnere billett(er)) **på et arrangement**
- **Lese fra / skrive til** **filer hele/deler av datastrukturen**

Globale variable, klasser (og *litt* datastrukturen)

Programmet skal *kun* inneholde tre globale objekter av klassene `Kunder`, `Steder` og `Arrangementer`. Inni disse objektene ligger til sammen *hele* datastrukturen.

Programmet skal (i hvert fall) inneholde de ni klassene med (minst) sine datamedlemmer:

1. **`Kunder`** – inneholder datastrukturen med `Kunde`'er, samt siste kundenummer brukt hittil.

2. **`Kunde`** – inneholder et unikt kundenummer (sortert på dette), postnummer og telefon (`int`'er), samt navn, gateadresse, poststed og mail (alt `char`-pekere).

3. **`Steder`** – inneholder datastrukturen med `Sted`'ene.

4. **`Sted`** – inneholder et unikt stedsnavn (sortert på dette), antall ulike (stol)oppsett på stedet (max 5 stk), samt datastrukturene for disse oppsettene.

5. **`Sone`** – inneholder et unikt sonenavn (sortert på dette), totalt antall billetter til salgs, antall billetter solgt hittil og prisen pr.billett i sonen.

6. **`Vrimle`** – subklasse av `Sone`. Inneholder (i tillegg til det den arver) kun *alle* kundenumrene til de som har kjøpt billetter i denne unummererte (vrimle) sonen.

7. **`Stoler`** – subklasse av `Sone`. Inneholder (i tillegg til det den arver) antall rader og seter (pr.rad) i sonen, samt *alle* kundenumrene til de som har kjøpt billetter/setene i denne nummererte sonen.

8. **`Arrangementer`** – inneholder datastrukturen med `Arrangement`'er, samt siste unike arrangementsnummer brukt hittil.

9. **`Arrangement`** – inneholder dets navn (sortert på dette, men *ikke* unikt), et unikt arrangementsnummer (men altså *ikke* sortert på dette nummeret), spillested og artist (`char`-pekere), dato og tid (time/minutt) (tre `int`'er). Klassen inneholder også en `enum`-type som forteller om arrangementstype: Musikk, Sport, Teater, Show, Kino, Familie eller Festival. I tillegg inneholder klassen *hele* datastrukturen for sonene (*kopi* av *aktuelt* (stol)oppsett på arrangementstedet) og status for billettsalget i disse sonene til ulike kunder.

NB: Tegn opp, og bli ordentlig sikker på hvordan datastrukturen må være (ser ut), og hvordan den fungerer ifm. de ulike funksjonene dere skal lage (angitt nedenfor). Se også avsnittet "Delinnlevering nr.1" (mot slutten av dette dokumentet).

Menyvalg / funksjoner

Programmet skal (i hvert fall) håndtere tolv menyvalg. De kan oppsummeres i gruppene: (Kommandoer i parentes (fem stk) er det frivillig å lage/implementere.)

1.	**K D** Kunde Display	Brukeren kan velge å få skrevet ut data om *alle* kunder, *en* gitt kunde (ut fra sitt unike nummer), *alle* kunder med et *gitt navn*.
	K N Kunde Ny	Ny kunde legges inn i datastrukturen. Alle dets data leses inn. Tildeles automatisk fortløpende unikt kundenummer.
	K E Kunde Endre	Kan endre alle(?) data/personalia for et gitt nummer.
	(K S Kunde Slett	Kunde med gitt nummer slettes helt fra datastrukturen. (Hva om for tiden har ubrukte billett(er)?))
2.	**S D** Sted Display	Skriver *alle* spille-/arrangementstedenes navn og antall (stol)oppsett pr.sted, men *ikke* detaljer om oppsettene (dette gjøres ved kommandoen i O D).
	S N Sted Nytt	Nytt spille-/arrangementsted legges inn i datastrukturen. *Kun* dets navn leses inn. (Stol)oppsett foretas senere via kommandoen O N.
	(S S Sted Slett	Spille-/arrangementsted med gitt navn slettes. (Hva om det er arrangement på stedet som ikke er avviklet?))
3.	**O D** Oppsett Display	Ut fra et spillestedsnavn, tilbys brukeren hvilket (stol)oppsett (av de opptil fem) hun/han vil se. *Alle* sonene for det aktuelle (stol)oppsettet vises så på skjermen.

O N Oppsett Nytt	Ut fra et spillestedsnavn, opprettes (om plass) et nytt (stol)oppsett . Det nye oppsettet kan lages helt fra «scratch», eller at det blir kopiert fra et allerede eksisterende oppsett på stedet. Husk at et oppsett består av ulike soner (enten stoler eller vrimle). Brukeren kan legge inn nye/endre på kopierte soner til hun/han er ferdig med å definere stedets nye oppsett.
O E Oppsett Endre	Ut fra et spillestednavn og oppsett(nummer) kan brukeren endre på (legge til/fjerne soner, endre på soner) oppsettet (akkurat som ved kopiering under O N).
(O S Oppsett Slett	Ut fra et spillestednavn og oppsett(nummer) slettes(?) bare vedkommende oppsett.)

4. **A D** Arrangement Display	Brukeren får vist/skrevet ut data om: 1) *alle* arrangement, 2) de som inneholder *hele eller deler* av en tekst, 3) spilles på et *visst* sted, 4) en *viss* dato, 5) av en *viss* type (f.eks. Sport), 6) en *viss* artist eller 7) *alle* data (inkl. billettsalget i *alle* soner) om et visst arrangementsnr.
A N Arrangement Nytt	Det spørres om spillestedsnavn, nytt arrangementsnavn og (stol)oppsettnummer. Oppsettet *kopieres* over til det nye arrangementet. Alle andre relevante data for arrangementet blir også lest inn. Det «skjulte» unike nummeret blir også automatisk telt opp og tilordnet. Oppsettet skrives til filen ARR_<nr>.DTA, og slettes fra hukommelsen (og det nye `Arrangement`-objektet).
(A E Arrangement Endre	Mulig å endre på et arrangements data. Men, hvilke?)
A S Arrangement Slett	Mulig å slette et helt arrangement *og(!)* dets datafil (ARR_<nr>.DTA). Dramatisk, men kritisk operasjon?
A K Arrangement Kjøp	Brukeren kjøper billett(er) på et gitt arrangement. Først velges et arrangement (husk at flere kan hete det samme), deretter en sone, så antall billetter og evt. plass i sonen (sete(ne)). Kundenummeret må også lagres på/for disse plassene. Kjøpers hoveddata og billetter skrives (på en *klar* måte) *bakerst* på filen BILLETTER.DTA.
(A R Arrangement Returner	Brukeren gis tilgang til å levere tilbake/returnere allerede kjøpt(e) billett(er).)

Ifm. mange av disse A-kommandoene (D7, E?, K, R) må *hele* datastrukturen for oppsettet og billettsalget på *ett* konket arrangement hentes fra filen ARR_<nr>.DTA, skrives tilbake (om er oppdatert), og deretter bli slettet fra hukommelsen.

Alle slags feilsituasjoner (f.eks. ulovlige kommandoer, ikke-eksisterende navn og numre), og dertil egnede meldinger, er det ikke bemerket ovenfor. Dette må også selvsagt gjøres/kodes.

I tillegg må selvsagt `main` lages (som «styrer hele butikken»), samt funksjoner for å lese brukerens valg/kommando og en lengre utskrift med liste over lovlige valg/kommandoer.
En del annet også: F.eks. at tall virkelig *er numeriske* verdier, og at de er i rimelig intervall.

Data til/fra filer

I programmet er det involvert fem ulike (typer) filer (*alle* filformat bestemmer dere selv):

De tre globale objektenes data lagres på henholdsvis filene:
KUNDER.DTA, **STEDER.DTA** og **ARRANGEMENT.DTA**.

Billetter blir (ved kommandoen «A K») fortløpende skrevet bakerst på filen **BILLETTER.DTA**. Utskriften (formatet) bestemmer dere altså selv, men la det være forståelig og klart, slik dere selv ville forventet at innholdet på en billett ville ha vært.

Dessuten lagres *alle* dataene om *hvert* arrangements (stol)oppsett og status for billettsalget (hvilke kundenumre som har kjøpt hvilke billetter/seter) på filen **ARR_<nr>.DTA**, der <nr> er arrangementet «skjulte» unike nummer.

Prosjekt / multifil-program

Dere *skal* utvikle hele dette programmet som et prosjekt, der programmet er splittet opp i flere ulike filer. Følgende (minst 13) .h-filer må lages:

- en med *alle* const'er
- en med *alle* enum'er
- en med deklarasjon av *alle* 'globale' funksjonsheadinger
- en *pr.klasse* med deklarasjon av dets innhold (datamedlemmer og funksjonsheadinger)
- ListTool2B.h (ligger allerede ferdig på PROSJEKT-katalogen)

Følgende (minst 12) .cpp-filer må lages:

- en som inneholder main og definisjon av de globale variablene
- *minst* en fil som inneholder definisjon (innmaten) av *alle* de 'globale' funksjonene
- en *pr.klasse* med definisjon av klassens funksjoner (deres innmat)
- ListTool2B.cpp (ligger allerede ferdig på PROSJEKT-katalogen)

Hjelp: Se og lær av filene E19*.* på EKSEMPEL-katalog.

Annet (klargjørende?)

- ListTool *skal* brukes ifm. løsningen av denne prosjektoppgaven.
  Legg merke til og bruk 'ListTool2B.H' og 'ListTool2B.CPP' (se rett ovenfor).
- Definer de globale variablene på samme fil som `main`. Om dere trenger å bruke disse på/i andre filer, så refererer dere til dem vha. `extern` …… i disse filene.
- Noen aktuelle `const`'er *kan* være: `NVNLEN, STRLEN, MAXOPPSETT, MAXPRIS, MAXRADER` og `MAXSETER`.
- Det *skal* sikres at alle tall som leses inn virkelig er *numeriske*, samt at de ligger i *fornuftige* intervaller.
- **I hukommelsen er det *aldri* noen `Sone`-objekter, bare `Vrimle` og `Stoler`.**
  Et `Sted`-objekt innholder opptil fem ulike datastrukturer for (stol)oppsett på dette spillestedet, basert på slike objekter. Disse er en *mal* for oppsettet, og brukes *aldri* til å kjøpe billetter fra. Billettkjøp foregår ved at et aktuelt oppsett på spillestedet er *kopiert* over til et `Arrangement`, og fra *dette* objektet forekommer det enkelte og konkrete billettkjøp.

- **I praksis vil det i programmet være desidert mest `Kunde`-objekter, så svært mange `Arrangement`, og minst av `Sted`. Vi lar derfor `Kunde` og `Sted` (selv om de hver kan ha opptil fem oppsett, men allikevel er det «få» spillesteder vi i praksis prater om) ligge i hukommelsen til enhver tid. *Men*, billett-salget (på oppsettet) for *alle* `Arrangement`'ene lar vi *alltid* ligge på filene ARR_<nr>.DTA, og henter inn (og skriver ut igjen) *kun ett og ett når vi virkelig trenger ett av dem*.**
- Husk at flere `Arrangement` kan ha samme navn, men de kan skilles fra hverandre via dato og/eller tid (+ det «usynlige» og unike <nr>).
- Programmet har ingen enkel funksjon for å få skrevet ut *alle* (stol)oppsettene på *ett* eller *alle* spillestedene. Dette må vi (litt søkt) forutsette at (erfarne) billettselgere enten har i hodet, som figurer/tegninger i egne hefter, eller som større plakater på veggen/pulten.
- Programmet skal ikke håndtere slikt som en betalingstjeneste (dvs. kjøperens kortbruk, status (dvs. hva (ikke) betalt), totalt kjøpt for opp gjennom tidene, osv.
- Det er lurt å tidlig planlegge hvilken rekkefølge dere bør implementere dette i, og hva som kan gå parallelt, eller forutsetter at annet er laget/fungerer allerede (se også nedenfor).
- Denne oppgaveteksten er nok ikke helt entydig og utfyllende på alle punkter/måter, derfor er det mulig at dere må gjøre deres egne klargjøringer/presiseringer/forutsetninger. (Se aller siste punkt under «Sluttinnlevering».)

Forslag til rekkefølge på implementasjonen

1. `Main` m/`lesKommando`, `skrivMeny` og omrisset av klassene m/datamedlemmer.
2. Bestem formatet for KUNDER.DTA, STEDER.DTA og ARRANGEMENT.DTA. Legg noen testdata inn på disse, og les dette inn i datastrukturen.
3. Implementer kommandoene 'K D', 'S D' og 'A D'.
4. Implementer kommandoene 'K N', 'S N', og deretter 'K E'.
5. Implementer kommandoene 'O D', 'O N' og kopiering av et steds oppsett.
6. Implementer kommandoen 'A N'.
7. Implementer kommandoen 'A K' (inkl. å skrive til filen BILLETTER.DTA).
8. Implementer kommandoene 'O E' og 'A S'.

Når formatet for ARR_xx.DTA må bestemmes, og når det skal leses fra/skrives til disse, må dere selv bestemme.

Gjøremål 1.arbeidsuka (11.-15.mars)

1. Gjøre pkt 1.1 og punktene 2.1-2.5 på websiden om prosjektet.
2. Sette seg inn i/lese nøye oppgaveteksten (jfr. pkt.5 på websiden om prosjektet). Analyse av problemstillingen og datastrukturen.
3. Delta på de tre forelesningene i uke 11 (mandag, tirsdag *og fredag*).
4. Lage grupperegler (jfr. pkt.1.2 på websiden om prosjektet).
   NB: Dokumentet som ligger ute er *innspill, ingen ferdig* kontrakt.
5. Gjøre pkt.3 og 4 på websiden om prosjektet.
6. Bestemme datoene for når de ulike punktene under «Forslag til rekkefølge på implementasjonen» (rett ovenfor) skal være ferdig.
7. Utføre og avkrysse alt på sjekklisten (jfr. pkt.6 på websiden om prosjektet).
8. Overholde fristen for og innholdet i «Delinnlevering nr.1» (se rett under).

Delinnlevering nr.1

Innen fredag 15.mars 2019 kl.09:00 *skal* dere ha gjort følgende:

Levert *(på papir)* til emnelærer*:*

1. navnet på gruppedeltagerne, inkl. kontaktinfo (mail og mobil) for alle i gruppen.
2. *ett* A4- eller A3-ark med *detaljert* tegning av datastrukturen.
3. gruppereglene, signert av alle gruppens medlemmer.
4. individuelt signert bekreftelse fra *hver* av gruppedeltagerne (jfr. pkt.1.3 på websiden om prosjektet).
5. utført, avkrysset og signert sjekkliste (jfr. pkt.6 på websiden om prosjektet).

Dataene i pkt.1 legges også inn i README-filen for prosjektet på Bitbucket (fjern all den andre teksten Bitbucket foreslår skal være med der).
Dokumentene i pkt.2-3 skal også (innscannet) legges inn i GruppeXX-mappen deres på Google Docs.

Delinnlevering nr.2

Innen onsdag 27.mars 2019 kl.09:00 *skal* dere ha lastet opp (committed) deres siste versjon av koden i prosjektet. Dere skal *minst* ha gjort ferdig (kodet og fungerer korrekt) t.o.m. pkt.4 under «Forslag til rekkefølge på implementasjon» (ellers vil dere ligge dårlig an …… ☹).
I tillegg skal alle aktuelle testfiler i GruppeXX-mappen på Google Docs være oppdatert (jfr. pkt.2.6 på websiden om prosjektet). *Disse lages/fylles ut etter hvert som dere skriver pseudokode for hver kommando (altså **før** dere begynner å skrive selve koden).*

Sluttinnlevering

Innen mandag 8.april 2019 kl.09:00 *skal* dere ha:

- lastet opp (committed) deres fungerende, endelige og siste versjon av koden i prosjektet.
- lagt inn *minst* de obligatoriske testdataene i alle filene (jfr. pkt.7 på websiden om prosjektet).
- *Testkjørt prosjektet i god tid før fristen ved å clone det hele ned til en helt ny katalog, og kjørt det derfra.* (Dette blir en simulering av hvordan læringsassistentene vil teste/kjøre og oppleve programmet.)
- *Grupper med bare Mac-brukere må melde fra om dette i egen mail til emnelærer.*
- i GruppeXX-mappen (på Google Docs):
  - oppdatert og ferdigstilt (fylt ut komplett) alle testfilene.
  - lagt *en fil* med beskrivelse av *alle* filformatene og eksempler på filenes utseende.
  - evt. ha lagt *en* egen fil («readme.doc/pdf») med egne presiseringer/forutsetninger.

Gruppe(sam)arbeid

Sørg for at alle ytre rammer er lagt til rette for et godt og konstruktivt samarbeide. Dette gjøres best ved å sette opp klare og konkrete grupperegler (se pkt.1.2 på websiden om prosjektet, som er *innspill* til dette). *Jobb mye, effektivt og målrettet allerede fra første stund* (dvs. start «langspurten» tidlig, se «Gjøremål 1.arbeidsuka»). »). Og: sørg for å være «i rute» ved delinnlevering nr.2 – ellers får dere en *knallhard* avslutning.

Generelle krav til obligatoriske arbeider

Se: http://folk.ntnu.no/frh/grprog/obliger#Gen_reg (spesielt det femte punktet)

Lykke til!
FrodeH