Prosjektoppgave

i PROG1003 - Objekt-orientert programmering våren 2026

Frist: ***Tirsdag 7.april 2026 kl.11.00***
Arbeidsform: *1-3 personer*
Arbeidsinnsats: ***Mye***

Innledning

Dere skal i denne prosjektoppgaven lage et litt større program som holder orden på utleien av ulike gjenstander (tralle, sykkel eller elsparkesykkel) i en *stor* fornøyelsespark.
Parken har flere steder/kontorer der man kan leie/hente ut og levere inn slike gjenstander.

I hovedsak skal programmet håndtere følgende operasjoner:

- **Legge inn / skrive / slette kunde / sted / gjenstand**
- **Hente ut / levere inn en/flere gjenstander**
- **Overføre/flytte gjenstander mellom steder**
- **Finne igjen en gitt gjenstand (på et sted eller hos en kunde)**
- **Skrive steder som har igjen av en gitt gjenstand**
- **Skrive inntjeningen på hver sted**
- **Hele datastrukturen leses fra/skrives til filer**

Globale variable, klasser og datastrukturen

Programmet *skal kun* inneholde to globale objekter av klassene `Kunder` og `Utleiesteder`. ***Se* filene: datastruktur.pdf** og **`mainV26.cpp`**

Programmet *skal* (i hvert fall) inneholde de åtte klassene med (minst) datamedlemmene:

1. **`Kunder`**:
   - `int sisteNr`: fortløpende økende automatisk nummerering av alle nye kunder
   - `list <Kunde*>`: *sortert* liste med de *unikt* nummererte kundene

2. **`Kunde`**:
   - `int`: kundens unike nummer (sortert på dette), mobilnummer
   - `string`: navn
   - `vector <Gjenstand*>`: *usortert* `vector` med nåværende leide gjenstander

3. **`Utleiesteder`**:
   - `int sisteNr`: fortløpende økende automatisk nummerering av alle nye gjenstander
   - `map <string, Sted*>`: `map` med *alle unikt navnede* `Sted`'er (navn er forkortelser/koder, f.eks. «Sted1», «B7», «AKV»)

4. **`Sted`**:
   - `string`: evt. stedets *unike* ID/navn, kort beskrivelse/forklaring av stedet
   - `int`: telefonnummer (for å kontakte stedet/kontoret), inntjente kroner
   - `vector <Tralle*>`: alle *trallene* som stedet for tiden har til utlån
   - `vector <Sykkel*>`: alle *syklene* som stedet for tiden har til utlån
   - `vector <Elsparkesykkel*>`: alle *elsparkesyklene* som stedet har til utlån

5. **`Gjenstand`**:
   - `int`: et *unikt* gjenstandsnummer
   - `enum`-verdi(?) for å representere hvilken subklasse vi *egentlig* har med å gjøre (Brukes bl.a. ved K L når gjenstander skal legges over i rett `vector` på stedet.)

6. **`Tralle`** (subklasse av `Gjenstand`):
   - `bool`: om er utstyrt med drasele/-stropp eller ei (i tillegg til drastaget i metall) (Tralle er en liten kassevogn med fire hjul. Oppi denne kan små barn sitte og/eller bagasjen (sekker, bagger, poser o.l.) ligge.)

7. **`Sykkel`** (subklasse av `Gjenstand`):
   - `bool`: har med tilhenger bak eller ei (til bagasje/sitteplass for mindre barn)

8. **`Elsparkesykkel`** (subklasse av `Gjenstand`):
   - `int`: antall watt

(Blir nok en del bruk for `virtual` ifm. subklassenes kode…..)

Menyvalg / funksjoner

Det skal lages et fullverdig program som har følgende muligheter/menyvalg/funksjonalitet:
(**NB:** *Hvilken kode som må ligge inni i ulike klassene er det opp til dere å designe/bestemme*)

1. **K N** Kunde Ny
   Det opprettes en ny kunde, og vedkommende tildeles automatisk et nummer som er *en* høyere enn siste brukte kundenummer. Kundens navn og mobilnummer leses inn. Lånte gjenstander leses *ikke* inn (det gjøres vha. kommandoen K H).
   NB: Husk å legge kunden *sortert* inn i datastrukturen.

2. **K A** Kunde Alle skrives
   Skriver ut *alle* kundene med deres nummer, navn og antall lånte gjenstander.
   En linje pr.kunde. Stans utskriften og vent på et tastetrykk for hver tjuende kunde.

3. **K 1 <knr>** Kunde skriv <kundenr> ('1' er et ett-tall.)
   Alle kundens data (inkludert *alt* om *alle* lånte gjenstander) skrives ut.

4. **K H <knr>** Kunde Hente (flere) gjenstand(er) til <kundenr>
   Kunden henter/låner (om mulig) et ønsket antall med traller, sykler og/eller elsparkesykler på et eksisterende/lovlig sted. *Den avledede klassens datamedlem(mer) leses og settes her, slik at dets utleiepris kan beregnes eksakt.* Disse objektene flyttes fra vedkommende sted sin datastruktur og over til kundens `vector` (legges bare fortløpende *usortert* bakerst). Gjenstandene kan gjerne komme *i tillegg til* allerede lånte gjenstander. Kunden betaler for lånet til *utleiestedet* (og *ikke* innleveringsstedet).

5. **K L <knr>** Kunde Lever alle gjenstand(er) fra <kundenr>
   Alle kundens evt. lånte gjenstander innleveres på et eksisterende/lovlig sted.
   Dvs. de flyttes fra kundens `vector` til den *aktuelle* `vector`'en hos stedet.
   Husk å «nullstille» datamedlemmer (ingen sele/tilhenger og standard watt er 100).

6. **K S <knr>** Kunde Slett <kundenr>
   Brukeren må bekrefte at *virkelig* ønsker å foreta denne handlingen.
   Om så er tilfelle innleveres evt. lånte gjenstander på et eksisterende/lovlig sted,
   og kunden slettes/fjernes *totalt* fra datastrukturen.

---

7. **S N** Sted Nytt
   Det opprettes (om mulig) et nytt *unikt* sted. Dets beskrivelse og telefonnummer leses.
   Inntjente kroner og de tre `vector`'ene nullstilles (de får *ikke* innhold her, men vha.
   kommandoene S O eller G N).

8. **S A** Sted Alle skrives
   Skriver *alle* stedenes ID/navn, telefonnummer og antallet de har til utleie av de tre
   ulike typene gjenstander.

9. **S 1 <ID>** Sted skriv <stedsID> ('1' er et ett-tall.)
   Alle stedets data (inkluderte antallet av hver gjenstand og *alle* gjenstandenes numre)
   skrives ut.

10. **S T** Sted Tjent
    Skriver *alle* stedenes ID/navn og hva de har tjent inn (*en* linje pr.sted).

11. **S I** Sted hvem har Igjen noe av en gitt gjenstand ('I' er en stor 'i')
    For alle stedene som har igjen/på lager noe av en gitt gjenstand skrives dets ID/navn,
    telefonnummer og det aktuelle antallet (*en* linje pr.sted).

12. **S O** Sted Overfør gjenstand mellom steder
    Overfører/flytter et gitt antall av *en* ønsket gjenstand fra ett sted til et annet.

13. **S S <ID>** Sted Slett <stedsID>
    Brukeren må bekrefte at *virkelig* ønsker å foreta denne handlingen.
    Om så er tilfelle flyttes/overføres alle stedets gjenstander til ett eller flere av de andre
    stedene. Deretter slettes/fjernes stedet *totalt* fra datastrukturen.

---

14. **G N** Gjenstand Ny
    Det opprettes en eller flere nye gjenstander (på *ett gitt* sted) av *en gitt* type.
    Den/disse tildeles automatisk nummer som er en (og en) høyere enn siste brukte
    gjenstandsnummer. Gjenstandene legges bakerst i *aktuell* `vector` på stedet.

15. **G F <gjnr>** Gjenstand Finn <gjenstandnr>
    Finner og skriver hvor en gitt gjenstand befinner seg (på et sted eller hos en kunde).

16. **G S <gjnr>** Gjenstand Slett <gjenstandnr>
    Brukeren må bekrefte at *virkelig* ønsker å foreta denne handlingen. Om så er tilfelle
    letes det etter gjenstanden på *kun* stedene, og den slettes om den blir funnet.
    (Er det en kunde som har gjenstanden, kan den *ikke* slettes før enn at den er innlevert.)

Rimelige verdier for `const`'er (f.eks. priser for leie av gjenstandene og deres tillegg (drasele, tilhenger, ulike watt)) `int`'er (som leses inn fra brukeren) og `enum`'er, og hvilke funksjoner/tjenester de ulike objektene må tilby hverandre (interface), må dere selv finne og bestemme. De aller fleste feilsituasjoner, f.eks. ulovlige kommandoer, ikke-eksisterende stedsIDer, kunde-/gjenstandsnumre, tomme containere, …m.m, og dertil egnede meldinger, er stort sett ikke bemerket ovenfor. Dette må også selvsagt gjøres/kodes.

Forslag til **`mainV26.cpp`** kan hentes ned fra prosjektets hjemmeside. Det må i tillegg *minst* lages en funksjon med utskrift av lovlige kommandoer.

Tips: ***Kommandoene K S, S S, G F og G S bør være noe av det siste dere lager/koder, når alt annet virker og er ferdig laget/testet.***

Data til/fra filer

I programmet er det totalt involvert to ulike filer: **KUNDER.DTA** og **STEDER.DTA**
Hele datastrukturen inni de to globale objektene ligger her lagret på hver sin fil.
Formatene bestemmer dere helt selv.

Prosjekt / multifil-program

Dere *skal* utvikle hele dette programmet som et prosjekt, der programmet er splittet opp i mange ulike filer. Følgende (minst 12) .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)
- LesData3.h (ligger ferdig på EKSEMPLER-katalogen)

Følgende (minst 11) .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)
- LesData3.cpp (ligger ferdig på EKSEMPLER-katalogen)

Hjelp: Se og lær av filene EKS27*.* på EKSEMPLER-katalogen.

Annet (klargjørende?)

- Alle gjenstandene er pålimt/-malt sitt *unike* nummer, uavhengig av hvilken type det er av.
- Ifm. utleie av traller og sykler, så forutsetter vi at *alle* steder *alltid* har henholdsvis nok ekstra seler/stropper og tilhenger tilgjengelig. Alle elsparkesyklene sin toppfart/ytelse kan justeres opp/ned mellom to verdier (100/200 watt). Disse to har ulike priser.
- **NB:** Programmets kun to globale objekter er definert på filen der `main` ligger.
  Når disse trengs brukt i andre filer, så refereres de til vha. `extern` ..... (jfr. EKS_27*.*).
- 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 pkt.2e under «Innlevering» på neste side).

NB NB NB

1. ***Behold norske navn på klassene (og deres da respektive h- og cpp-filer).***
   (Det er da så mye enklere for læringsassistentene (LA) å rette mange prosjekter hver, når filer og klasser har samme navn som angitt i dette dokumentet.)
2. ***La absolutt alt av h-, cpp- og andre filer ligge i en og samme (topp)katalog på GitHub.***
   (Dette gjør også alt rettearbeidet for LAene mye enklere.)
3. ***Lever tidlig en lenke til prosjektet i Blackboard*** (se pkt.1 under «Innlevering» nedenfor).
4. ***Testkjør prosjektet i god tid før fristen ved å clone det hele ned til en helt ny katalog, og kjør det derfra.*** (Blir en simulering av hvordan LAene vil kjøre/oppleve programmet.)
5. ***Meld fra (via mail) til emnelærer om grupper består av bare Mac- eller Linux-brukere.***

Innlevering

Innen tirsdag 7.april 2026 kl.11:00 *skal* dere ha:

1. **levert en *lenke/adresse* til prosjektet på GitHub via Blackboard.**
   Dette gjøres ved å kopiere/klippe ut SSH-adressen for cloning i GitHub.
   Den limes inn i Blackboard via «Skriveinnsending»
   (og *ikke* som «Legg ved filer» -som hittil ved oblig-innleveringer).

2. **lastet opp (committed) på GitHub:**
   a. **deres fungerende, endelige og siste versjon av koden i prosjektet**
   b. **lagt inn *minst* de obligatoriske testdataene i alle filene (jfr. pkt.7 på websiden om prosjektet)**
   c. **de tre stk. ferdig utfylte testskjemaene (pdf) for KN, SO og GF**
   d. ***en* beskrivelse (pdf) av DTA-filenes *format* og eksempler på utseende**
   e. **evt. *en* egen fil (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 bl.a. best ved å sette opp klare og konkrete grupperegler.
- *Jobb mye, effektivt og målrettet allerede fra første stund* (dvs. start «langspurten» straks).
- *Gjør et selvstendig gruppearbeide. Dvs. ikke samarbeid med/kopier fra andre grupper. Det kalles plagiat, og vil medføre stryk på prosjektet for de involverte gruppene.*
- Dere velger selv antallet (1-3 stk) i gruppen. Men, uavhengig av gruppeantallet, så forventes det at dere kommer i mål med prosjektet (*alt* kodes/gjøres og virker).
- Et gruppearbeid er et gruppearbeid. Enten får *alle* godkjent, eller så får *alle* det ikke.
- Det er *ingen reinnlevering på prosjektet* («second chance»). Enten så holder det man har kodet/laget, eller så gjør det ikke det.

Løkke tæll!

FrodeH