KUN PDF FILER ER TILLATT I BESVARELSEN, DERFOR LIGGER SELVE ZIP BESVARELSEN I VEDLEGG

I objektorientert programmering lager vi et dataprogram som baserer seg på objekter. Vi organiserer data i klasser, og denne klassen kan vi opprette instanser av. Klassen vil inneholde felter; som er objektenes verdi (data), og metoder; som er hva objektene kan gjøre. Objektene kan sammenlignes med «objekter» i virkelig liv. I objektorientert programmering har vi 4 grunnleggende konsepter som er arv, polymorfisme, enkapsulering og abstraksjon.

Av disse har jeg benyttet enkapsulering ved å markere felter som private, som gjør at klassen selv kontrollerer egne data, og benytter gettere for å få tilgang til dataene, og evt. settere for å sette data. I forhold til polymorfisme er alle klasser/objekter polymorfiske siden alle klasser i Java arver fra klassen «object», så på denne måten har vi alltid polymerfisme til stede. Objekter er opprettet i flere tilfeller og sees typisk hvor det er bruk av «new».

Eksempler på objekter i oppgaven er som i JDBC klassen lages et connection objekt. Vi har en JDBC klasse, som inneholder metodene med «getConnection xx» som i dette tilfellet peker til hver av sin database. I hver metode i JDCD klassen hentes data ut fra databasen og deretter lages det objekter av de. Disse har hver sin klasse, som det da lages opbjekter/instanser av. I hver jdbc metode opprettes et midlertidig objekt som fylles med verdiene konstruktøren krever for å lage objektet. Eksempelvis; hvert event, hver ceremonibeskrivelse, hver student osv blir et objekt.

Jeg rakk dessverre ikke komme i mål med «Participant» klassen min, men her var tanken at både staff, gjester og studenter som skal på event er alle sammen deltagere (participants) og vil være naturlig å arve denne klasse som blant annet skulle gjøre nytte for seg da listen over *alle* deltagere skulle printes ut for bruker. Her har alle noe til felles, som feks. at alle har et navn. Siden jeg dessverre kom litt for sent i gang med implementasjonen arv her oppstod det en del krøll da student skulle arve, grunnet måten programmet var bygd opp på allerede, dette er hvorfor denne klassen ikke er oppført med arv fra participant. Pr. nå, og er noe jeg med mer tid ville rettet opp i.

## Referanser:

Referansener er benyttet i JDBC klassen.

JDBC konstruktør og getConnection metodene er hentet fra Bogdan git repository, lesson 13.

URL: <https://github.com/bogdanmarculescu/pgr112oopDBsection/tree/main/lesson13>

Øvrige metoder i JDBD klasse er inspirert fra Marcus git reposity:

PGR112-23V/code/lectures/\_22/library/Library.java, samt Bogdan sitt overnevnte.

Url: <https://github.com/kristiania/PGR11223V/blob/master/code/lectures/_22/library/Library.java>

Alle metodene i JDBC er en litt blanding av inspirasjon fra begge, samt egen implementasjon.