INF315 Praktisk databaseadministrasjon
Obligatorisk oppgave i Persistens høsten 2015

Praktisk

Oppgaven kan gjøres individuelt eller i gruppe. Dere leverer ved å zippe hele løsningen og levere i Fronter. Lag én undermappe for applikasjonslaget, en for klienten (den har allerede en egen mappe) og en for db4o. Dere leverer ingenting på papir. Dere behøver ikke lage noen bruksanvisning, prosjektrapport e.l. – det holder med de komplette programmene. Frist for innlevering: fredag 13. november kl. 9.00.

Send meg en e-post med gruppemedlemmer og på hvilket navn i Fronter det er levert innen leveringsfristen.

Oppgave

En eiendomsmegler skal ha et objektorientert program for registrering av eiendommer og bud på disse eiendommene. Klassemodellen er gjengitt på neste side. Dere skal bruke de navn og datatyper som er angitt. Det er lov å legge til ytterligere medlemmer ved behov (jeg tror ikke at dette er nødvendig), men de skal i så fall ikke vises til brukeren. Dere kan regne med at alle objektene får plass i RAM samtidig.

Dere skal lage et program der persistensen sikres med *db4o*. De nødvendige bibliotekene legges i en mappe sammen med programmet.

Som det fremgår, er systemet bygget over trelagsmodellen. All kommunikasjon med brukeren skal skje gjennom grenseklassen (presentasjonslaget) *Menyskjema*. Alle endringer i entitetsklasser skjer gjennom kontrollklassen som også sikrer persistens og konsistens. Feilmeldinger må kastes som feil til grenseklassen og vises til brukeren der, hvis ikke de kan rettes opp på lavere nivå (det tror jeg det er lite av her, da feilene stort sett genereres av brukeren og må rettes av brukeren).

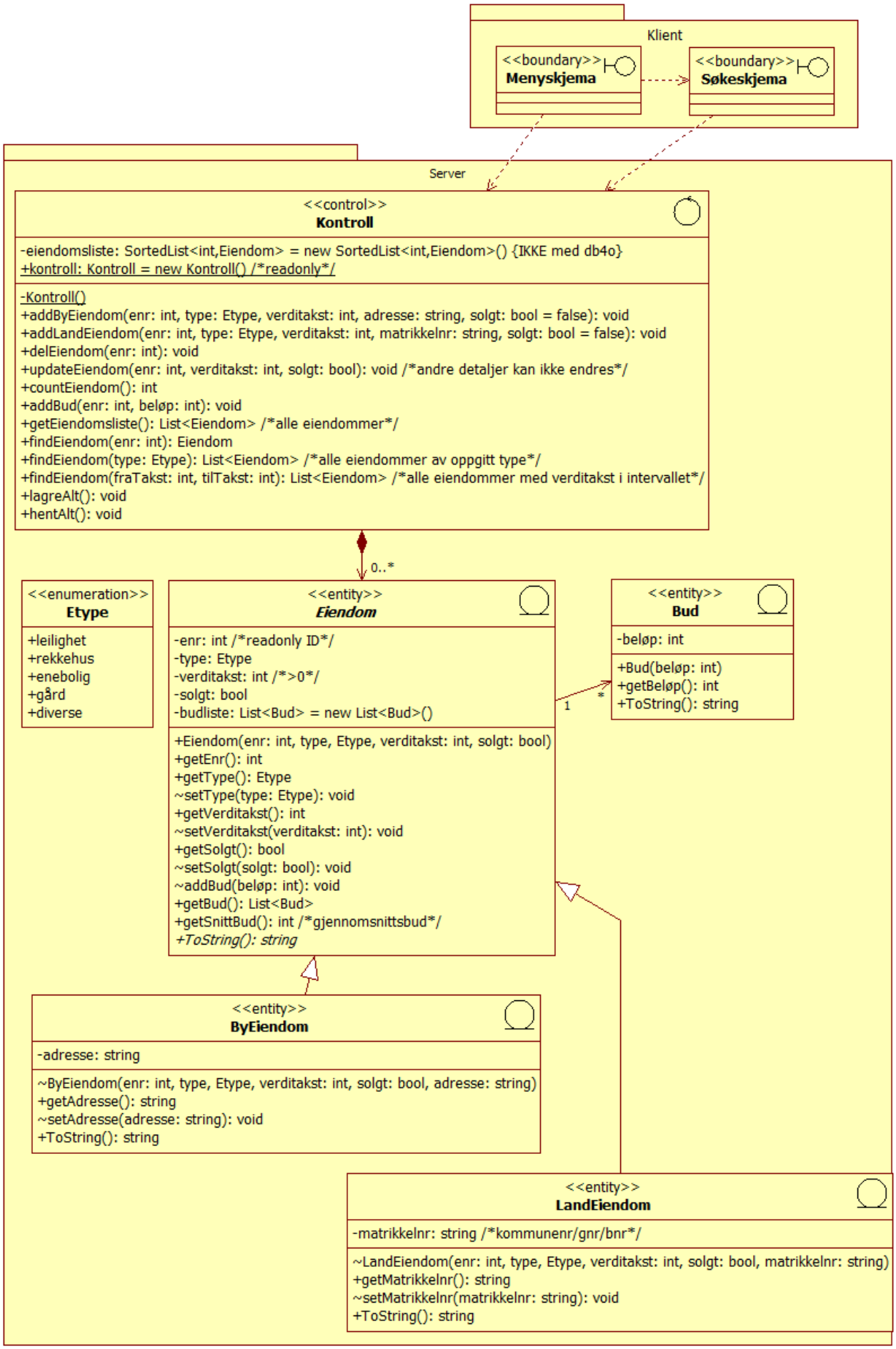
Konkrete entitetsklasser skal være persistente, men ikke kontroll- og grenseklasser. Til testing må dere legge inn data og dem kan dere gjerne la ligge igjen i leveringen.

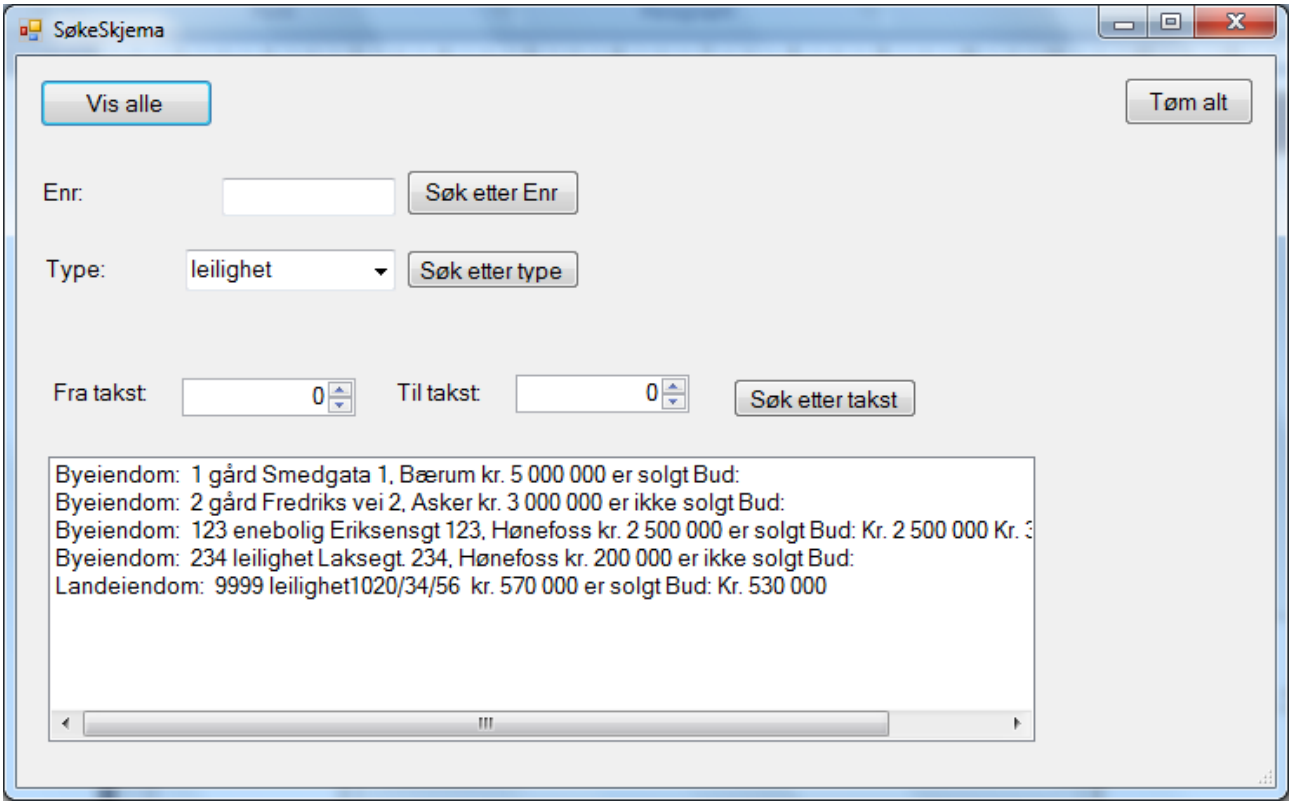
Attributter merket *ID* brukes av brukerne som identifikator og skal være entydig for hvert objekt som er lagret (sikres i programmet – velg selv hvordan).

I klassediagrammet for *Kontroll* finnes samlingen *eiendomsliste*. Når persistensen sikres med db4o, er den unødvendig *og skal ikke være med*. Søking (*find*-metodene) må da skje direkte i objektdatabasen.

Noen tilleggsopplysninger

1. Klienten er laget ferdig og finnes sammen med denne oppgaven som zip-fil. **OBS!** Denne applikasjonen er prøvekjørt bare av meg – den *kan* derfor være feil i den. Hvis du finner slike, håper jeg du vil tipse meg og andre snarest.
2. Alle funksjonene *ToString()* returnerer alle opplysninger om objektet. Formatet fremgår av skjemaet avbildet nedenfor.
3. Det er ikke nødvendig å kontrollere *adresse* og *matrikkelnummer*.
4. I henhold til UML-syntaks er klasser og medlemmer med *kursiv* navn abstrakte. Medlemmer med strek under navnet, er klassemedlemmer (*static*).
5. Medlemmene er merket med synlighet. De som er merket med ~ skal være *internal*.


Klient
<<boundary>>
Menyskjema
<<boundary>>
Søkeskjema
Server
<<control>>
Kontroll
-eiendomsliste: SortedList<int,Eiendom> = new SortedList<int,Eiendom>() {IKKE med db4o}
+kontroll: Kontroll = new Kontroll() /*readonly*/
-Kontroll()
+addByEiendom(enr: int, type: Etype, verditakst: int, adresse: string, solgt: bool = false): void
+addLandEiendom(enr: int, type: Etype, verditakst: int, matrikkelnr: string, solgt: bool = false): void
+delEiendom(enr: int): void
+updateEiendom(enr: int, verditakst: int, solgt: bool): void /*andre detaljer kan ikke endres*/
+countEiendom(): int
+addBud(enr: int, beløp: int): void
+getEiendomsliste(): List<Eiendom> /*alle eiendommer*/
+findEiendom(enr: int): Eiendom
+findEiendom(type: Etype): List<Eiendom> /*alle eiendommer av oppgitt type*/
+findEiendom(fraTakst: int, tilTakst: int): List<Eiendom> /*alle eiendommer med verditakst i intervallet*/
+lagreAlt(): void
+hentAlt(): void
0..*
<<enumeration>>
Etype
+leilighet
+rekkehus
+enebolig
+gård
+diverse
<<entity>>
Eiendom
-enr: int /*readonly ID*/
-type: Etype
-verditakst: int /*>0*/
-solgt: bool
-budliste: List<Bud> = new List<Bud>()
+Eiendom(enr: int, type, Etype, verditakst: int, solgt: bool)
+getEnr(): int
+getType(): Etype
~setType(type: Etype): void
+getVerditakst(): int
~setVerditakst(verditakst: int): void
+getSolgt(): bool
~setSolgt(solgt: bool): void
~addBud(beløp: int): void
+getBud(): List<Bud>
+getSnittBud(): int /*gjennomsnittsbud*/
+ToString(): string
1
*
<<entity>>
Bud
-beløp: int
+Bud(beløp: int)
+getBeløp(): int
+ToString(): string
<<entity>>
ByEiendom
-adresse: string
~ByEiendom(enr: int, type, Etype, verditakst: int, solgt: bool, adresse: string)
+getAdresse(): string
~setAdresse(adresse: string): void
+ToString(): string
<<entity>>
LandEiendom
-matrikkelnr: string /*kommunenr/gnr/bnr*/
~LandEiendom(enr: int, type, Etype, verditakst: int, solgt: bool, matrikkelnr: string)
+getMatrikkelnr(): string
~setMatrikkelnr(matrikkelnr: string): void
+ToString(): string



Output av *ToString()* fremgår av søkeskjemaet ovenfor.