

# Gruppe 16 - Boligformidling

Eivind Schulstad - s198752 - Dataingeniør

Gretar Ævarsson - s198586 - Informasjonsteknologi

Sigurd Hølleland - s198597 - Informasjonsteknologi

# Produktdokumentasjon

Programmet Boligformidleren er en applikasjon programmet i Java, som er ment for å brukes av et firma som skal formidle boliger som en tredjepart mellom utleiere og boligsøkere. Programmet muliggjør registrering, visning og muligheter for å holde oversikt over et stort antall utleiere, deres utleieobjekter og boligsøkere.

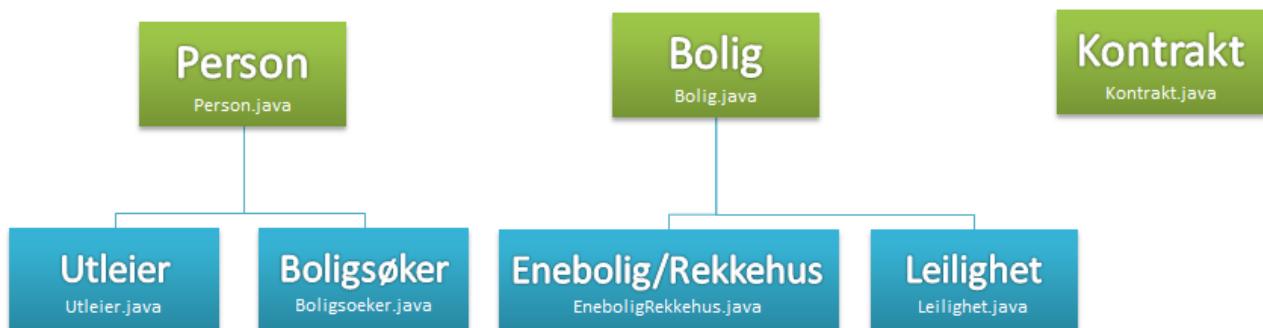
En stor del av programmet er dessuten myntet på å gi brukeren informasjon om hvilke boligsøkere som kan være interessert i en bolig, og hvilke boliger som kan være passende for en boligsøker. Det er også flere funksjoner for å sortere/filtrere boliger utifra diverse søkerkriterier. Vår idé har vært å lage et program der brukeren skal gis en god og informativ oversikt over et stort, forvirrende boligmarked.

Det går dessuten ann å registrere kontrakter i programmet, og det er også gode muligheter for å holde oversikt og føre statistikk over disse. Under følger en inngående beskrivelse av programmets funksjonalitet, ikke minst også hvilke forenklinger som er måttet blitt gjort, og også hvordan disse og annet kan forbedres.

## 1. Oppbygning av programmet

### 1.1. Klassehierarki

Vi har tre klasser som representerer fysiske objekter/entiteter som brukes i programmet, dvs. Person, Bolig og Kontrakt. Person-klassen har to subklasser, Utleier og Boligsøker. Bolig-klassen har også to subklasser, Enebolig/Rekkehus og leilighet.

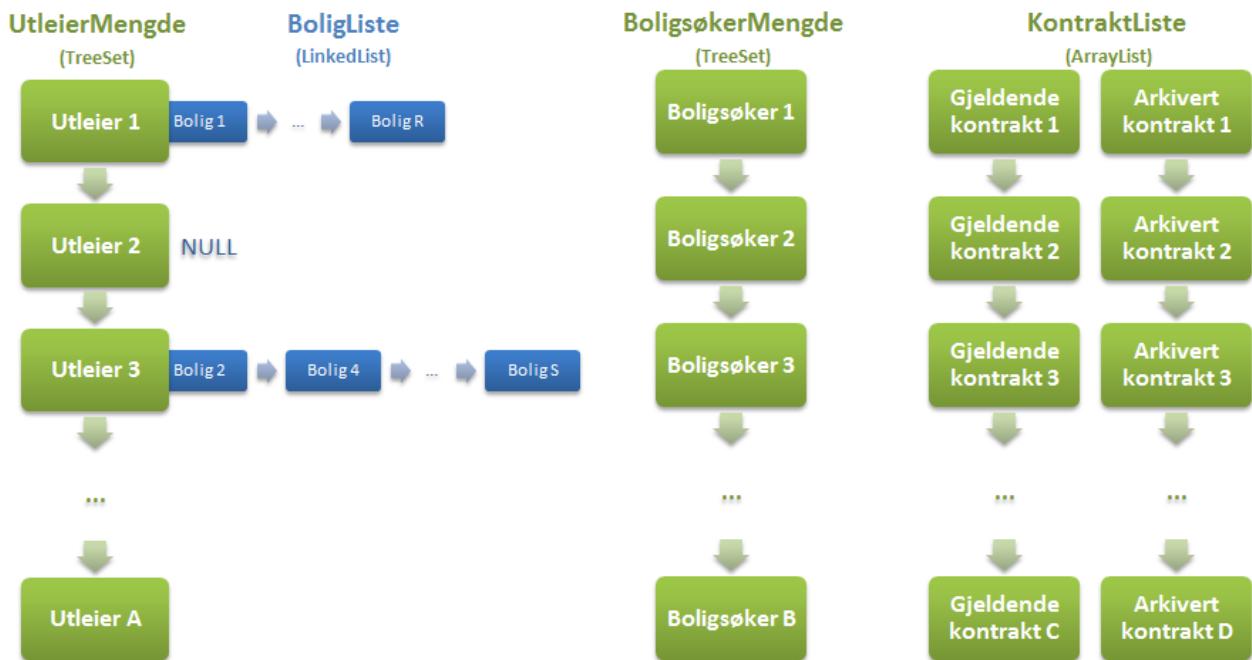


## 1.2. Datastruktur

Nevnte objekter lagres i en datastruktur, dvs. Person-objektene lagres i hver sin TreeSet. Videre innholder hvert Utleierobjekt en LinkedList der alle boliger vedkommende er knyttet til lagres. Programmets datastruktur er altså satt sammen ved hjelp av Java's Collections. Vi valgte nevnt datastruktur for Personer fordi det hindrer dobbeltlagring og er sortert (f.eks. mht. personens navn).

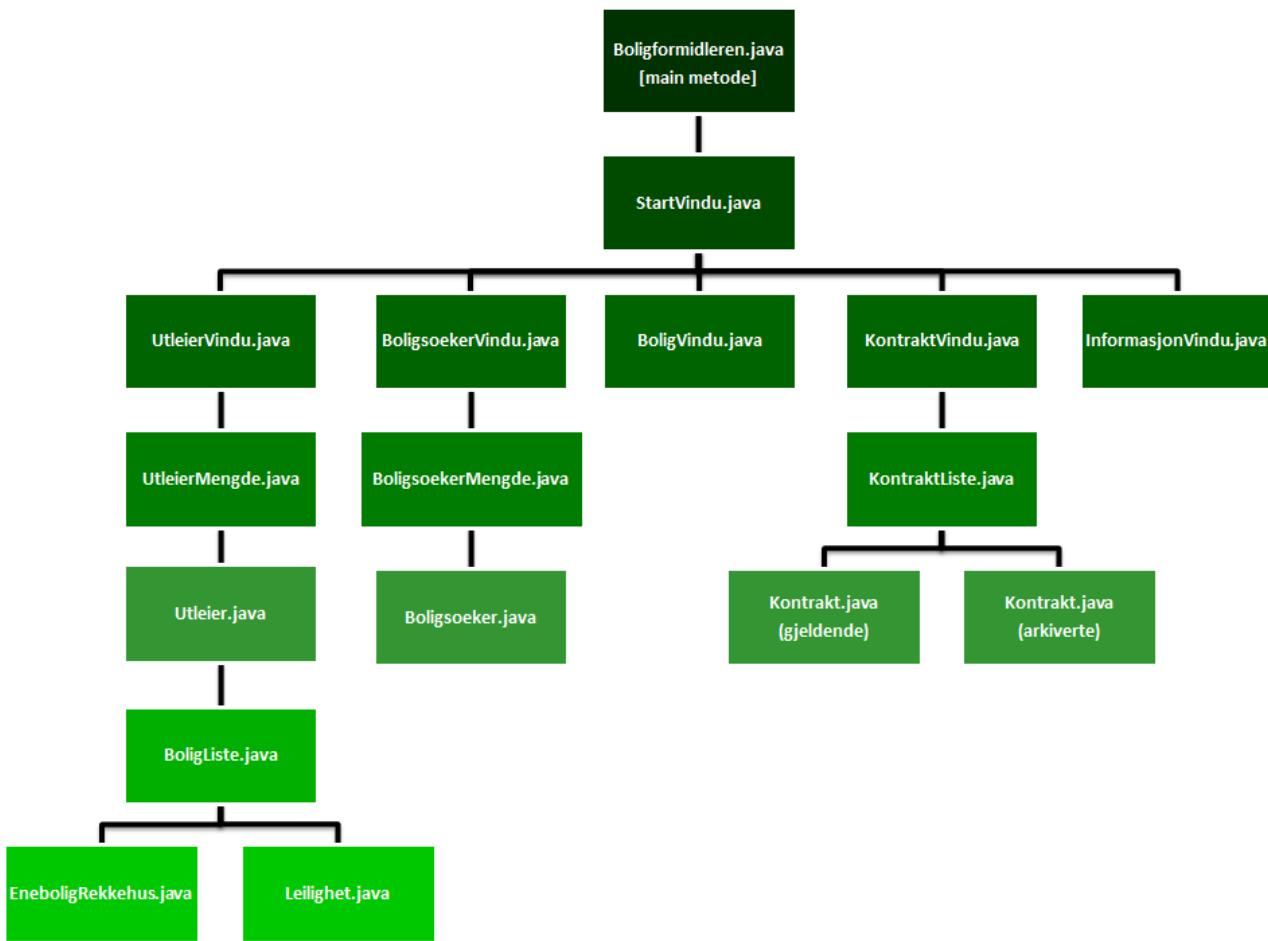
Det er også definert en egen, norskvennlig sorteringsrekkefølge i programmet gjennom klassen Personsammenlikner.java. TreeSet'ene opprettes med denne rekkefølgen. Vi valgte LinkedList datastrukturen for Boligene fordi den er effektiv når man skal sette inn og fjerne objekter, som er en stor del av vårt program.

Kontrakt klassen inneholder to ArrayList'er, en for gjeldende kontrakter og en for arkiverte kontrakter. Vi valgte ArrayList fordi den er effektivt i å søke gjennom og hente ut data (bruke indeks). Samtidig er det ikke så mye fjerning og innsetting til og fra listene.



## 1.3. Programstruktur

Programmet har en tresstruktur, dvs. alle klassene tilhører en øvrig klasse, utenom klassen som inneholder main metoden (se diagram under). Hver klasse er beskrevet EirikDataIng - s198752, Gretar (IT - s198586), Sigurd (IT - s198597).



### 1.3.1. Boligformidler.java

Denne klassen inneholder main-metoden og den oppretter et StartVindu-objekt, som åpnes når programmet kjøres. Lukkeknappen på StartVinduet er programmert slik at den skriver alle mengder og lister til fil før programmet avsluttet.

Boligformidler.java inneholder følgende metoder

- `public static void main(String[] args)`

### 1.3.2. StartVindu.java

Denne klassen inneholder alle andre vindu-klassene (utleievindu, boligsøkervindu, boligvindu, kontraktvindu, informasjonvindu) og kobler dem sammen med get-metoder (static). Dette betyr at alle klassene har tilgang til hverandre igjennom denne klassen. StartVindu-klassen inneholder i tillegg alle konstanter (static) som brukes av mer enn en klasse, f.eks. regulære uttrykk (regex) mønstre. Til slutt inneholder denne klassen generelle metoder som brukes i andre klasser, eks. for regex, skrive data til fil m.fl. StartVindu.java inneholder følgende metoder

- `public StartVindu() - konstruktør`
- `public static UtleierVindu getUtleierVindu()`
- `public static BoligsoekerVindu getBoligsoekerVindu()`
- `public static BoligVindu getBoligVindu()`
- `public static KontraktVindu getKontraktVindu()`
- `public static InformasjonVindu getInformasjonVindu()`
- `public static boolean kontrollerRegEx(String pattern, JTextField[] input)`
- `public static String kontrollerRegExTomFeltOK(String pattern, JTextField[] input)`
- `public static boolean kontrollerRegEx(String pattern, String input)`
- `public static int konverterBlanktFeltTilHeltall(JTextField jtf)`

- public static boolean tekstFeltErTomt(JTextField[] jtf)
- public static Date konverterDato(String datostreng)
- public void generateData() - ikke i bruk (brukt ved utvikling av programmet)
- public void skrivTilFil()
- public static void visFeilmelding(StackTraceElement[] ste)
- public static void visFeilmelding(Object o)
- public static String visJaNeiMelding(String melding, String vindutekst)
- public void actionPerformed(ActionEvent e)

### 1.3.3. UtleierVindu.java

Denne klassen brukes når man skal registrere, slette eller endre en utleier, dvs. den inneholder alle nødvendige felter og knapper og alle metodene som kalles når brukeren trykker på knappene. I tillegg inneholder den et UtleierMengde-objekt (hvor alle utleierobjektene ligger). UtleierVindu.java inneholder følgende metoder

- public UtleierVindu() - konstruktør
- public UtleierMengde getUtleierMengde()
- public void regUtleier()
- public void slettUtleier()
- public void blankFelter()
- public boolean regexOK()
- public void endreFelt(String felt, String ny)
- public void skrivUtleierTilFil()
- public void lesUtleierFraFil()
- public void actionPerformed(ActionEvent e)
- public void focusGained(FocusEvent fe)
- public void focusLost(FocusEvent fe) - brukt for auto-fill

### 1.3.4. UtleierMengde.java

Denne klassen inneholder mengden selv (TreeSet) og alle metodene som brukes når man vil sette inn et nytt utleier-objekt, fjerne eller finner en utleier. Før en utleier kan fjernes, sjekker metoden om den har registrerte boliger eller ikke, dvs. den kan ikke fjernes hvis den ikke har tom Boligliste. I tillegg inneholder den metoder for å registrere bolig-objekter til en utleier, dette fordi man først må søke gjennom alle utleierene for å forvisse seg om at boligen ikke eksisterer fra før. UtleierMengde.java inneholder følgende metoder

- public void focusLost(FocusEvent fe) - brukt for auto-fill
- public boolean fjern(Utleier ul)
- public Utleier finnUtleier(String fornavn, String etternavn)
- public Bolig finnBolig(String gateadresse, int postnr, String poststed)
- public boolean regBolig(Utleier u, Bolig b)
- public Set kopierMengdeUsortert()
- public int antallBoliger()
- public Set getSortertMengde()
- public String toString()

### 1.3.5. Utleier.java

Denne klassen er en konkret subklasse til Person-klassen. Implementerer interface Comparable for å kunne sortere utleierne i en TreeSet etter egendefinert rekkefølge. Legger til to ekstra datafelt, et om hvilket firma utleieren representerer og en boligliste av typen ArrayList, som inneholder alle boligobjekter utleieren har registrert. Har derfor også metoder for å sette inn og fjerne et boligobjekt fra listen. I klassen ligger også en metode "tilRad" som bestemmer hvilke og i hvilken rekkefølge datafeltene hos en utleier skal vises i en tabell ved å legge dem inn i en array. Det er

også redefinert en compareTo-metode for å kunne legge uteierobjektet i en treeSet som ikke tar imot en Comparator i konstruktøren, og en toString-metode som bestemmer hvordan informasjonen om en uteier skal skrives ut i brukergrensesnittet. Utleier.java inneholder følgende metoder

- public Utleier(String fornavn, String etternavn, String gateadresse, int postnr, String poststed, String epost, int tlfnr, String firma) - konstruktør
- public String getFirma()
- public Boligliste getBoligliste()
- public void setFirma(String f)
- public Object[] tilRad()
- public void regBolig(Bolig b)
- public void slettBolig(Bolig b)
- public int compareTo(Utleier ul)
- public String toString()

### 1.3.6. Boligliste.java

Hvert uteierobjekt inneholder et objekt av denne klassen, og dessuten inneholder dette objektet listen selv (LinkedList) og alle metodene som brukes når man vil sette inn et nytt bolig-objekt (enten av type 'EneboligRekkehus' eller 'Leilighet'), fjerne eller finne en bolig. Boligliste.java inneholder følgende metoder

- public List getListe()
- public void settInn(Bolig b)
- public boolean fjern(Bolig b)
- public Bolig finnBolig(String gateadresse, int postnr, String poststed)
- public String toString()

### 1.3.7. EneboligRekkehus.java

Denne klassen er en sub-klasse til Bolig-klassen og den arver alt fra Bolig, og har i tillegg datafelter for antall etasjer, tomtens areal og om huset har kjeller. toString-metoden bestemmer hvordan fulstendig informasjon om en leilighet skal presenteres i brukergrensesnittet. Se avsnitt Bolig.java for mer informasjon om hvordan boliger håndteres. EneboligRekkehus.java inneholder følgende metoder

- public EneboligRekkehus(String gateadresse, int postnr, String poststed, String type, String beskrivelse, Date annonsedato, int inneAreal, int antRom, int byggeaar, int pris, int antEtasjer, int tomtAreal, boolean kjeller) - konstruktør
- public int getAntEtasjer()
- public int getTomtAreal()
- public boolean getKjeller()
- public void setAntEtasjer(int a)
- public void setTomtAreal(int t)
- public void setKjeller(boolean k)
- public String toString()

### 1.3.8. Leilighet.java

Arver fra Bolig, og har i tillegg datafelter for hvilken etasje leiligheten befinner seg i, om det er heis og om det er balkong. toString-metoden brukes på samme måte som i EneboligRekkehus-klassen. Leilighet.java inneholder følgende metoder

- public Leilighet(String gateadresse, int postnr, String poststed, String type, String beskrivelse, Date annonsedato, int inneAreal, int antRom, int byggeaar, int pris, int etasje, boolean heis, boolean balkong) - konstruktør
- public int getEtasje()

- public boolean getHeis()
- public boolean getBalkong()
- public void setEtasje(int e)
- public void setHeis(boolean h)
- public void setBalkong(boolean b)
- public String toString()

### **1.3.9. BoligsoekerVindu.java**

Denne klassen brukes til å registrere, slette eller endre en boligsøker, dvs. den inneholder alle nødvendige felter og knapper og alle metodene som kalles når brukeren trykker på knappene. I tillegg inneholder den BoligsoekerMengde-klassen (hvor alle boligsoeker-objektene ligger). BoligsoekerVindu.java inneholder følgende metoder

- public BoligsoekerVindu() – konstruktør
- public BoligsoekerMengde getBoligsoekerMengde()
- public void regBoligsoeker()
- public void slettBoligsoeker()
- public boolean regexOK()
- public void endreFelt(String felt, String ny)
- public void blankFelter()
- public void skrivBoligsoekerTilFil()
- public void lesBoligsoekerFraFil()
- public void visFeilmelding(StackTraceElement[] ste)
- public void visFeilmelding(Object o)
- public void actionPerformed(ActionEvent e)
- public void focusGained(FocusEvent fe)
- public void focusLost(FocusEvent fe)

### **1.3.10. BoligsoekerMengde.java**

Denne klassen inneholder mengden selv (TreeSet) og alle metoder som brukes når man vil sette inn et nytt boligsoeker-objekt, fjerne eller finne en boligsøker. BoligsoekerMengde.java inneholder følgende metoder

- public void settInn(Boligsoeker b)
- public boolean fjern(Boligsoeker bs)
- public Set getMengde()
- public Boligsoeker finnBoligsoeker(String fornavn, String etternavn)
- public Set kopierMengdeUsortert()
- public String toString()

### **1.3.11. Boligsoeker.java**

Denne klassen er også en konkret subklasse til Person-klassen. Implementerer interface Comparable for å sortere boligsøkerne i en treeSet etter egendefinert (norsk) rekkefølge. Boligsøkeren får en rekke ekstra datafeltene som skal tilsvare datafeltene i en Bolig. Dette er fordi en boligsøker skal kunne registrere seg med egne krav til hvilke boliger han/hun ser på som interessante. For eksempel dersom boligsøkeren har lagret 5 som "maxEtasje", skal ikke boligsøkeren vises som interessert i leiligheter som har "etasje" = 6. Det er også et datafelt for at boligsøkeren skal kunne skrive en kort tekst om seg selv, og et datafelt som indikerer om boligsøkeren for øyeblikket leter etter en bolig. Dette er "true" som default og endres kun når boligsøkeren registrerer en kontrakt med firmaet, og evt. når kontrakten avsluttes. Det kunne vært aktuelt å legge til funksjon så brukeren manuelt kan endre dette datafeltet. I klassen ligger det også en metode som skal sammenlikne nevnte datafeltene med de tilsvarende datafeltene hos en bolig. Dersom ett av boligsøkerens

krav ikke oppfylles av boligen det sammenliknes med, skal denne boligen ikke regnes som interessant for boligsøkeren. Et viktig moment er at boligsøkeren skal kunne fylle ut så mange, eller så få, krav som han/hun ønsker. Dersom det ikke er registrert noe krav skal derfor defaultverdier som 0/"ingen krav"/null, ikke være begrensende for søkeret. I klassen ligger også en metode "tilRad" som bestemmer hvilke og i hvilken rekkefølge datafeltene hos en boligsøker skal vises i en tabell ved å legge dem inn i en array. Det er også redefinert en compareTo-metode for å kunne legge boligsøkerobjektet i en TreeSet som ikke tar imot en Comparator i konstruktøren, og en toStringmetode som bestemmer hvordan informasjonen om en boligsøker skal skrives ut i brukergrensesnittet. Boligsoeker.java inneholder følgende metoder

- public Boligsoeker(String fornavn, String etternavn, String gateadresse, int postnr, String poststed, String epost, int tlfnr, String pInfo, String type, int areal, int soverom, int byggeaar, int pris, Date dato, int maxAntEtasjer, int tomtestorrelse, boolean kjeller, int maxEtasje, boolean heis, boolean balkong) - konstruktør
- public String getPersInfo()
- public int getAreal()
- public int getByggeaar()
- public Date getDate()
- public int getPris()
- public int getSoverom()
- public String getType()
- public int getMaxAntEtasjer()
- public int getTomtestorrelse()
- public boolean getKjeller()
- public int getMaxEtasje()
- public boolean getBalkong()
- public boolean getHeis()
- public boolean getLeterEtterBolig()
- public void setPersInfo(String p)
- public void setAreal(int a)
- public void setByggeaar(int b)
- public void setDate(Date d)
- public void setPris(int p)
- public void setSoverom(int s)
- public void setType(String t)
- public void setMaxAntEtasjer(int m)
- public void setTomtestorrelse(int t)
- public void setKjeller(boolean k)
- public void setMaxEtasje(int m)
- public void setBalkong(boolean b)
- public void setHeis(boolean h)
- public boolean passerTilBolig(Bolig b)
- public void leterEtterBolig()
- public void leterIkkeEtterBolig()
- public Object[] tilRad()
- public int compareTo(Boligsoeker bs)
- public String toString()

### **1.3.12. BoligVindu.java**

Denne klassen brukes når man skal registrere, slette eller endre en bolig, dvs. den inneholder alle nødvendige felter og knapper og alle metodene som kalles når brukeren trykker på knappene. BoligVindu.java inneholder følgende metoder

- public BoligVindu() - konstruktør
- public void regBolig(String boligtype)
- public void slettBolig()
- public boolean regexOK()
- public void endreFelt(String felt, String ny)
- public void boligUtskrift()

- public void blankFelter()
- public void actionPerformed(ActionEvent e)
- public void focusGained(FocusEvent fe)
- public void focusLost(FocusEvent fe)

### 1.3.13. KontraktVindu.java

Denne klassen brukes når man skal registrere eller si opp en kontrakt, dvs. den inneholder alle nødvendige felter og knapper og alle metodene som kalles når brukeren trykker på knappene. Den inneholder et Kontraktliste-objekt hvori alle kontraktobjektene ligger. Hver gang programmet kjøres, opprettes et kontraktVindu-objekt, som i sin tur kaller på en metode som sjekker alle gjeldende kontrakter i GjeldendeKontrakter-listen og flytter alle utløpte kontrakter til ArkiverteKontrakter-listen.

Når en kontrakt skal registreres, sørger denne klassen for å sjekke at boligen er ledig og at boligsøkeren ikke har en gjeldende kontrakt. Når den er registrert, settes boligen til "ikke ledig" og boligsøkeren settes til "leter ikke etter bolig".

Kontraktregistreringen fungerer mer detaljert slik at en boligsøker kun kan ha en gjeldende kontrakt registrert på seg til enhver tid. Det er altså ikke mulig å leie fler enn en bolig av gangen. Det er valgt slik for å gjøre programmet og håndteringen av kontrakter enklere. Dette betyr at så lenge en boligsøker er registrert med en gjeldende kontrakt, er det ikke mulig å registrere en ny kontrakt på samme boligsøker. Det samme gjelder for boligen. Den har et statusfelt "ledig"(true). Når en bolig knyttes til en kontrakt blir statusfeltet satt til "ikke ledig"(false), og det er ikke mulig å registrere en ny kontrakt på denne boligen. Når en kontrakt går ut forandres statusen samtidig til "ledig" igjen.

Lagringen og behandlingen av opprettede kontrakter foregår i klassen Kontraktliste. Her ligger det to lister av typen ArrayList, den ene for kontrakter som fortsatt er gjeldende, den andre for kontrakter som er utløpt. Ideen/hensikten er at alle nylige opprettede kontrakter legges i en separat liste, når utløpsdatoen for kontraktene er passert flyttes disse automatisk over i en annen liste(arkiv). Dette skiller gjeldende og kontrakter fra kontrakter som kun er av historisk betydning på en enkel og oversiktlig måte, forutsetningen er selvsagt at datokontrollen er god nok. En mulig svakhet ved kontrolleringer av data i dette programmet er at den kun gjennomføres for hver oppstart, dvs. programmet sjekker utløpsdatoen i kontrakten opp mot dagens dato, er utløpsdatoen passert legges kontrakten(e) over i arkivet. Brukeren av programmet er derfor avhengig av å avslutte og starte opp igjen programmet minst en gang daglig. Dette fører til en subtil betydning ved registrering av kontrakter, nemlig at alle nyregistrerte kontrakter behandles som "gjeldende" uavhengig av om den satte sluttdatoen er før dagens dato. Det er altså først etter en restart av programmet at kontrakten vil bli sjekket og lagt i arkivet.

KontraktVindu.java inneholder følgende metoder

- public KontraktVindu() - konstruktør
- public Kontraktliste getKontraktliste()
- public void regKontrakt()
- public boolean regexOK()
- public void siOppKontrakt()
- public void sjekkOmKontraktErUtløpt()
- public void utskrift()
- public void blankFelter()
- public void skrivKontraktTilFil()
- public void lesKontraktFraFil()

- public void actionPerformed(ActionEvent e)
- public void focusGained(FocusEvent fe)
- public void focusLost(FocusEvent fe)

### **1.3.14. KontraktListe.java**

Denne klassen inneholder to lister (ArrayList), en for gjeldende kontrakter og en for arkiverte kontrakter. Den innholder også alle metodene som brukes når man vil sette inn, fjerne eller finne et kontrakt-objekt. KontraktListe.java inneholder følgende metoder

- public List getKontraktListeGjeldende()
- public List getKontraktListeArkiv()
- public List getAlleKontrakter()
- public void settInn(Kontrakt k)
- public int antGjeldendeKontrakter()
- public int antArkiverteKontrakter()
- public Kontrakt finnGjeldendeKontrakt(Boligsoeker bs)
- public Kontrakt finnGjeldendeKontrakt(Bolig b)
- public String sjekkUtlopteOgArkiver(Date idag)
- public void fjernGjeldendeKontraktOgArkiver(Kontrakt k)
- public String toString()

### **1.3.15. Kontrakt.java**

Denne klassen representerer en kontrakt der det inngår en bolig, utleier og en boligsøker (leietaker). Kontrakt-klassen binder derfor på en måte sammen alle programmets "fysiske" entiteter. Forandringer i Kontraktklassen(e) må derfor utføres med stor forsiktighet, da selv små forandringer kan få stor betydning for hele programmets funksjonalitet og ikke minst stabilitet. Hver kontrakt har en startdato og sluttdato og en bestemt pris som trenger ikke at være lik prisen som er registrert for den boligen. Kontrakt.java inneholder følgende metoder

- public Kontrakt(Bolig b, Utleier u, Boligsoeker bs, int pris, Date startdato, Date sluttdato) - konstruktør
- public int getPris()
- public Date getStartdato()
- public Date getSluttdato()
- public Boligsoeker getBoligsoeker()
- public Bolig getBolig()
- public Utleier getUtleier()
- public void setSluttdato(Date sluttdato)
- public String toString()
- public Object[] tilRad()

### **1.3.16. InformasjonVindu.java**

Denne klassen er den viktigste klassen i programmets grensesnitt. Den brukes til å vise alle informasjon om utleiere, boligsøkere, boliger og kontrakter. Klassen bruker mange paneler til å vise dataene, begge for utskriftsområder (JTextArea) og for tabeller (JTable). Den har metoder for å finne kontrakter og boliger utifra kriterier(intervaller i pris etc.) som brukeren skriver inn i tekstfelter. Det er på en måte her alt skal synes, prioritet bør derfor holdes på at vinduet skal være oversiktlig og ha informativ visning. Det er brukt mye tabeller av typen JTable for å oppnå dette, og derfor er mange metoder dedikert til dette. Det er dessuten en svært omfattende hendelseshåndtering som skal utføres i denne delen av programmet, og actionPerformed, focusGained, focusLost-metodene har uungåelig blitt noe uoversiktlig. InformasjonVindu.java inneholder følgende metoder

- public InformasjonVindu() - konstruktør
- public void hentInfoPerson()
- public void visBoligInfo()
- public void tegnUtleierTabell()
- public void tegnBoligsøekerTabell()
- public void tegnBoligtabell()
- public void tegnKontraktTabell()
- public void hentBoligsøekerFraTabell()
- public void hentUtleierFraTabell()
- public void hentBoligFraTabell()
- public boolean finnBoliger()
- public void finnKontrakter()
- public void melding(String s)
- public void actionPerformed(ActionEvent e)
- public void focusGained(FocusEvent fe)
- public void focusLost(FocusEvent fe)

### **1.3.17. Andre klasser som ikke vises på diagrammet**

#### **Person.java**

Klassen Person er en abstrakt klasse som bestemmer hovedtrekkene ved programmets definisjon av en person. Klassen inneholder datafeltet for navn, personens hjemadresse, epost og telefonnr.

Klassen bestemmer også hvordan personer regnes som like, evt. deres ID. Dette er definert gjennom klassens equals-metode som sammenlikner to personer. Dersom de har likt fornavn og etternavn regnes de som samme person. Dette er en bevisst forenkling av virekligheten for å minke arbeidsmengden noe, og en eventuell ting å forbedre i fremtiden.

Det kan være for mye jobb at brukeren kan endre navnet på en utleier eller boligsøker, så vi gjør forutsetningen at dette skal ikke være en mulighet.

#### **Bolig.java**

Alle boliger i programmet er definert av den abstrakte klassen bolig. Her ligger alt som er felles for de to konkrente subklassetyperne Leilighet og EneboligRekkehus. Dette er datafeltet for adresse, hvilken type bolig det er, beskrivelse av boligen, inneareal, soverom, byggeår, leiepris, hvilken dato den ble avertert fra og om boligen er ledig eller ikke. Alle boliger er automatisk ledige når de er opprettet. I klassen er det også bestemt hvordan boliger regnes som like. Dette er definert gjennom klassens equals-metode, som tar to boliger og sammenlikner gateadresse, postnr og poststed. Dersom alle feltene er like hverandre er også boligene like. Det er derfor nødvendig å fylle inn ekstra informasjon, som f.eks leilighetsnummer, i adressefeltet dersom man skal registrere flere leiligheter som ligger i samme blokk. Det ligger også en metode, kalt tilRad, som bestemmer hvilke og i hvilken rekkefølge datafeltene i en bolig skal vises i en tabell ved å legge dem inn i en array. Siden Bolig-klassen er abstrakt er det kun objekter av typen Leilighet og EneboligRekkehus som opprettes. Når en av boligtypene er opprettet lagres de direkte i en utleiers tilhørende boligliste.

#### **UtleierTabellmodell.java, BoligsøekerTabellmodell.java, BoligTabellmodell.java, KontraktTabellmodell.java**

Alle tabeller i programmet er av typen JTable, og har en egen definert tabellmodell som bestemmer innhold og utseende til tabellen. Dette ligger i klassene

BoligTabellmodell, BoligsoekerTabellmodell, KontraktTabellmodell og UtleierTabellmodell. En viktig funksjon ved tabellene er at brukeren skal kunne velge en rad, trykke på "Gå til ..." knappen for å få opp detaljert informasjon om personen/boligen i raden. For å få til dette ligger det to kolonner i hhv. utleier og boligsøkertabellen som inneholder fornavn og etternavn som er skjulte for brukeren, og som kun brukes av metodene hentXxxFraTabell i InformasjonVindu.

## 2. Filbehandling

Programmet lagrer data på 3 forskjellige filer, hhv. boligsoekermengde.data, kontraktliste.data og utleirmengde.data. Når for eksempel boligsøkere registreres legges objektene inn i en TreeSet, og det er denne som i sin tur skrives til fila boligsoekermengde.data. Omrent tilsvarende skjer for utleiere og kontrakter.

Hver av disse datastrukturene har altså sine egne skrivTilFil og lesFraFil-metoder, som ligger i deres tilsvarende -Vindu klasser. Disse kalles opp hver gang programmet avsluttes og startes opp.

Vi valgte å definere en egen norsk tilpasset sorteringsrekkefølge for utleiere og boligsøkere. Dette ble gjort ved at TreeSet'en de to datatypene ligger lagret i er opprettet sammen med en Comparator som definerer sorteringen gjennom en RuleBasedCollator. Denne RuleBasedCollator er ikke serialiserbar noe som kompliserer filbehandlingen av utleiere og boligsøkere noe. Måten det er løst på i programmet er at de sorterte mengdene av utleiere og boligsøkere kopieres over i en mengde som ikke benytter seg av et Personsammenlikner(Comparator)-objekt, når skrivTilFil-operasjonen gjennomføres. Dette gjennom metodene kopierMengdeUsortert i UtleierMengde og BoligsoekerMengde. Objektene skrives altså uten egendefinert norsk sortering over til fil. Når objektene deretter leses inn, blir de en for en lagt inn i TreeSet'en hvor de sorteres på riktig måte. Det er også verdt å merke seg at alle boligobjektene blir skrevet til fil sammen med utleierobjektene

For kontrakter er det hele mye enklere. Programmet oppretter et KontraktListe-objekt der listen over gjeldende og utløpte kontrakter ligger. Hele dette objektet skrives til og leses fra fil ved hver oppstart/avslutning.

## 3. Prioritering

I planleggingsfasen snakket vi om å avgrense oppgaven og derfor opprettet vi en prioritiseringsrekkefølge. Vi snakket om hvilke deler av programmet som er viktig (must-have) og hvilke deler som ikke er så viktig (nice-to-have). Her er hvordan vi prioriterte oppgavene:

1. Første prioritet (*grønt = ferdig, rød = ikke ferdig*):
  - Må kunne registrere alle objekter og skrive ut alt slik at man får verifisert at informasjonen er blitt lagret.
  - Kunne matche boligsøkere med boliger.
  - Lage enkelt, fungerende brukergrensesnitt(flowlayout).
  - Fungerende skriving/lesing fra fil.
2. Annen prioritet:
  - Sletting.
  - Utbedre brukergrensesnitt, utseende og funksjon(mer avansert layout).
  - Vise leiekontrakt-historikk.

- Vise antall boliger for utleie og antall kontrakter som har blitt formidlet.
- Legge data inn i programmet (for sensor).
- Behandle brukerinput (regex, feilhåndtering).

### 3. Tredje prioritet:

- Sortering.
- Siste finish layout.
- Søke i historikk for leiekontrakter utfra kriterier.
- Bruke JTable i utskrift-området til at få opplysingene til å se bedre ut.
- Opprette f.eks. "postnummer.java" hvor man har alle postnummere og poststeder i Norge. Når man skal registrere person eller bolig, så bruker man denne filen for "auto-fill", dvs. man trenger ikke å skrive poststed. Det hjelper til ved feilhåndtering.
- Håndtere hvis to personer har samme navn, f.eks. istedenfor at `finnUtleier()` eller `finnBoligsoeker()` metodene returnerer et Utleier/Boligsoeker objekt, kan disse returnere en array av objekter. Så kan man sjekke størrelsen på arrayen når man skal sjekke hvis to eller flere personer har samme navn.

## 4. Generering av data

Vi brukte en spesiell metode (ca. 150 kodelinjer) i StartVinduet til å generere data, som vi brukte for testing av programmet. Vi opprettet en knapp som kallet på metoden, som les data fra fire .txt filer (utleiere.txt, boligsoekere.txt, boliger.txt og kontrakter.txt). Vi genererte .txt filene ved å bruke random-funksjonene i Microsoft Excel (se bilde nedenfor).

Først fann vi lister av norske fornavn og etternavn, samt gateadresser, postnummere og poststeder i Norge. Så lagde vi tilfeldige krav for boligsøkere og tilfeldige attributer for boliger. Boligene måtte inneholde gyldig fornavn og etternavn for utleierene.

Kontraktene var mest vanskelig å opprette, fordi vi måtte være sikker på at boligen som skulle gå i kontrakten, skulle ha akkurat samme utleier som skulle registreres i kontrakten.

Til slutt genererte vi 593 boliger, 201 boligsøkere, 100 utleiere og 100 kontrakter. Disse brukes når man vil hente informasjon fra programmet.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Fornavn	Etternavn	Gataadresse	Postnr	Poststed	Epost	Tekstområde	Beskrivelse	Boligtype	Areal	Soverrom	Biggjør	Pris	Averdert pris	Etasje	Tomtstørrelse	Kjeller	Etasje	Heis	
2	Mathena	Devrik	Brekkeveien 1	1233	Sand	cadam@wpaportal.com	85-07	Ingen krav	Leilighet	57	8	0	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2));IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2)))	0	0	0	0	0	0	0
3	Øyvind	Bjørnen	Engsveien 6	0663	Moisjær	adams@wpaportal.com	25-07	Struts	Leilighet	110	8	0	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	1918	0	0	0	0	0
4	Oda	Leifsdøn	Granåsen 8	0303	Oslo	dpc0126@wpaportal.com	75-07	Skje	Leilighet	0	0	0	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	0	0	0	0	0	0
5	Viggo	Merland	Gjæstgivervei 6	2680	Nord-Torpa	albs56@wpaportal.com	55-07	Garasje	Leilighet	0	0	0	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	0	0	0	0	0	0
6	Aisha	Næss	Bratteli 1	5429	Molde	lee7310@aol.com	8E-07	Foster	Ingen krav	147	0	0	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	0	0	0	0	0	0
7	Brita	Gjæst	Briegata 9	0283	Oslo	r1rlphson@wpaportal.com	8E-07	Fot	Leilighet	168	1	1950	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	0	0	0	0	0	0
8	Kjellaug	Gundersen	Bjerkeløkka 6	3796	Skien	pmac10@wpaportal.com	8E-07	Stråle	Eneboligrekkehus	99	3	1979	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	5	66	true	0	0	0	0
9	Vanessa	Bjerkli	Gjøkkbakken 9	4830	Hønefoss	alfranbensam@charter.net	8E-07	Hanske	Eneboligrekkehus	0	5	2005	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	5	233	true	0	0	0	0
10	Hamza	Gjørre	Gjørnunet 5	4613	Kristiansand S	rbabinho0063@aol.com	4E-07	Duft	Ingen krav	138	6	0	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	0	0	0	0	0	0
11	Fabian	Rostrand	Gaugeparet 8	6397	Bogstad	asku3@verizon.net	8E-07	Gebiss	Eneboligrekkehus	0	4	1950	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	3	132	false	0	0	0
12	Christel	Staare	Bragdveien 4	1237	Audnedal	ale.vaneira@adcom.no	2E-07	Armmalgåm	Eneboligrekkehus	0	3	1954	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	1	1	477	false	0	0	0
13	Ulrik	Olsvik	Almenning 6	5604	Narvik	bewarwinnings@yahoo.com	8E-07	Sjeld	Leilighet	109	0	0	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	0	0	0	0	0	0
14	Jørund	Ricnes	Amprøysen 2	2667	Lærd	kbk0464@wpaportal.com	8E-07	Pipe	Eneboligrekkehus	0	2	0	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	5	263	true	0	0	0	0
15	Susan	Skjærestad	Årupsgate 3	7466	Trondheim	donbarney@aol.com	2E-07	Buse	Leilighet	159	5	0	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	0	0	0	0	0	0
16	Tuva	Jensen	Gamelihagen 5	5558	Sætreid	rogbaron@aol.com	3E-07	Spitt	Ingen krav	228	8	2002	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	0	0	0	0	0	0
17	Peter	Lura	Grenseveien 4	5143	Kjøpmannskjær	gramanina@aol.com	1E-08	Pomp	Ingen krav	79	1	1916	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	0	0	0	0	0	0
18	Adolf	Ramstad	Brattbakkven 2	3562	Hostle	billbasnetto@cox.net	3E-07	Gritte	Eneboligrekkehus	0	6	1944	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	5	222	true	0	0	0	0
19	Dordi	Pauslen	Baggerløret 4	8608	Mo i Rana	jules728@yahoo.com	5E-07	Pynt	Leilighet	173	4	0	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	0	0	0	0	0	5
20	Amalie	Hilde	Graahbakkven 7311	7031	Tonsetberg	rbattiat@comcast.net	1E-07	Marmelade	Eneboligrekkehus	263	6	1970	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	2	158	true	0	0	0
21	Hamid	Evertsen	Arbins gate 4	1201	Ullensaker	jimbo10@casper.com	7E-07	Just	Leilighet	134	0	0	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	0	0	0	0	0	0	0
22	Anja	Ødegård	Glaciagata 2	7034	Fjellheim	mitred@wpaportal.com	2E-07	Just	Eneboligrekkehus	261	8	1961	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	4	458	false	0	0	0	0
23	Vera	Haraldsvold	Gjøvrevei 1	7111	Halden	edusbekker@live@wpaportal.com	9E-07	Brokk	Eneboligrekkehus	193	2	1931	=IF(RAND()<0,25;0;"Ingen krav";ROUND(RANDBETWEEN(2000;40000);-2);IF(I3="Leilighet";ROUND(RANDBETWEEN(2000;22000);-2);ROUND(RANDBETWEEN(6000;40000);-2))))	1	306	true	0	0	0	0

# **Verktøy**

Vi brukte NetBeans for programmering av java koden og GitHub for å holde mest mulig rede på og oversikt over koden.

Systemkrav til maskinen er den samme som vi har i lab-timene og Java versjon 7 brukes for dette prosjektet.

Microsoft Excel var brukt til å generere tilfeldige data for programmet og Microsoft Word for diagramer.

## **5. Videreutvikling**

Vi vil trekke fram enkelte punkter der programmet kan viderutvikles, evt. trenger videreutvikling.

### **5.1. Brukergrensesnitt/layout**

Programmet består av svært mange vinduer det kan være vanskelig å holde oversikt over. En smartere/smidigere løsning kan være aktuell. Selve visningen av hver enkelt bolig og hver enkelt person er ikke spesielt avansert, og begrenser seg til ren tekst i et utskriftsfelt. Å programmere støtte for bildevisning (av boliger, evt. også boligsøkerer og utleiere) hadde f.eks forbedret denne delen av programmet. Andre forbedringer kan også gjennomføres som f.eks egen tabellvisning av utleiers boliger/interessante boliger for en søker, og diverse formattering av tekst for en mer variert layout. Programmets grensesnitt bruker flere komponenter som listeboxer og avkryssningsbokser, og det er programmert autofill-funksjoner i de fleste av registreringsvinduene. Det er likevel fortsatt et program der brukeren selv er avhengig av å skrive/fylle inn svært mye av informasjonen selv. Det er mye funksjonalitet som etter vår mening fortsatt kan/burde legges til, eksempelvis menyer, tilbake/framover-knapper, generelt ting som gjør det lettere for bruker å "manøvrere" i programmet. En mulig forbedring for det over er at når man viser en tabell, over for eksempel boliger, å ha en "Gå til Kontrakt"-knapp som automatisk åpner et vindu der en kontrakt kan registreres for valgte bolig. Dette burde ikke være altfor tidkrevende/vanskelig å implementere i programmet slik det er pdd.

### **5.2. ID'er**

Programmet behandler personer med likt navn som like personer. Siden dobbeltlagring ikke tillates i datastrukturen, er det ikke mulig å registrere flere personer med likt navn. Dette er en ganske stor forenkling av virkeligheten, men går greit for vårt programs begrensede bruk. Ved videreutvikling måtte hver person utstyres med et unikt generert personnummer, og alle "erLik" eller "equals"-metoder må redefineres.

### **5.3. Kontroll av utløpte kontrakter**

Programmet fungerer slik at kontraktene kontrolleres ved hver oppstart av programmet. Det ville øke tryggheten til programmet, og vært mer elegant om kontraktene kunne kontrolleres oftere, samtidig som programmet kjører

## **5.4. Leie flere boliger**

Det er kun mulig for en leietaker å leie en bolig av gangen. Dette er for å forenkle håndteringen av kontrakter, men kunne evt. forbedres i framtiden. Da måtte isåfall hele kontraktsystemet forandres noe.

## **5.5. Geografi**

Systemet tar ikke hensyn til den geografiske plasseringen av boligene vs leietakerne når det kommer med forslag. Programmet vil derfor være mer nyttig lokalt enn det vil være f.eks på landsbasis. Å utvide "matche"-funksjonaliteten og de forskjellige klassene som definerer boliger og boligsøkere for å ta hensyn til dette, ville vært en stor forbedring.

## **5.6. Database**

Det kunne tenkes at programmet en gang i fremtiden skulle brukes på mye større datamengder enn det vi har foreløpig generert og testet programmet i. Det er tvilsomt om datastrukturen er god nok for det og det kunne vært ønskelig å programmere inn større for database. Det er dessverre uvisst for oss hvor omfattende og hvor mye endringer en slik implementasjon ville føre til.