



# DOKUMENTASJON

Prosjektoppgave Programutvikling  
2014

For gruppe 35:  
Kristoffer Gard Osen, s198754, Dataingeniør  
Christer Erik Bang, s198737, Dataingeniør  
Emil Oppegård, s198772 Dataingeniør

# PRODUKTDOKUMENTASJON

Prosjektoppgave Programutvikling  
2014

For gruppe 35:

Kristoffer Gard Osen, s198754, Dataingeniør

Christer Erik Bang, s198737, Dataingeniør

Emil Oppegård, s198772 Dataingeniør

# INTRODUKSJON

Dette er et program utviklet for selskapet "BooleanFormidling". BooleanFormidling driver boligformidling på utleiermarkedet og dette programmet skal gjøre det lettere for firmaet å opprette kontrakt mellom boligsøkere og utleiere. Programmet er utviklet for å bli brukt av en ansatt hos BooleanFormidling. Brukeren skal kunne ta i bruk registeret av boliger, boligsøkere og utleiere og opprette kontrakt mellom de forskjellige aktørene.

## FUNKSJONALITET

### Lagring

Dette programmet kan lagre boligsøkere, utleiere, boliger og kontrakter. Programmets to første faner gir bruker mulighet til å lagre boligsøkere, utleiere og boliger. Kontrakter lagres i den siste fanen. Dette er fordi man selvsagt trenger boligsøkere, utleiere og boliger for å lagre kontrakter. Om boligsøkere og utleiere kan man lagre person-opplysninger. I tillegg, når man har valgt at personen skal være utleier, kan man lagre firma den er knyttet til, hvis det er aktuelt. For boligsøker er det mulig å lagre krav til bolig. Disse kravene er en sentral del av programmet og brukes for å finne boliger som vil passe til boligsøkeren. Kravene består av hvilken boligtype boligsøkeren er interessert i, hvilken by som er aktuell, hvilken prisklasse, samt andre spesifikasjoner knyttet til den aktuelle boligtypen, feks, hvor mange rom skal boligen ha, hvor mange etasjer skal eneboligen være, skal hybelen ha eget kjøkken, hvilken etasje leiligheten ligger i og om den har heis, osv. Disse valgene blir synlig for bruker når boligtype er valgt.

Om boliger lagres informasjon på en liknende måte. Noen spesifikasjoner gjelder for alle boligtyper (feks, by, utleiepris, areal, osv) og noen gjelder kun for den valgte boligtypen. De sist nevnte blir også synlig først når boligtype er valgt.

For en kontrakt lagres det boligsøker, utleier og bolig, samt perioden kontrakten gjelder og når den er tegnet. Når boligsøker velges i brukergrensesnittet, får bruker se en tabell med boliger

som passer til denne søkeren og når bolig er valgt, lagres eieren av boligen som utleier på kontrakten.

### Vis i register

I fanen "Register" har bruker mulighet til å se detaljert informasjon om alle personer og boliger som er lagret. Det vises også bilder av boligene. Bruker har her mulighet til å slette boliger og personer fra registeret.

### Sletting

Sletting av boligsøkere, utleiare og boliger gjøres i fanen forklart over. Bruker markerer en person eller en bolig i tabellen og trykker på slettknappen.

### Match boligsøker med boliger

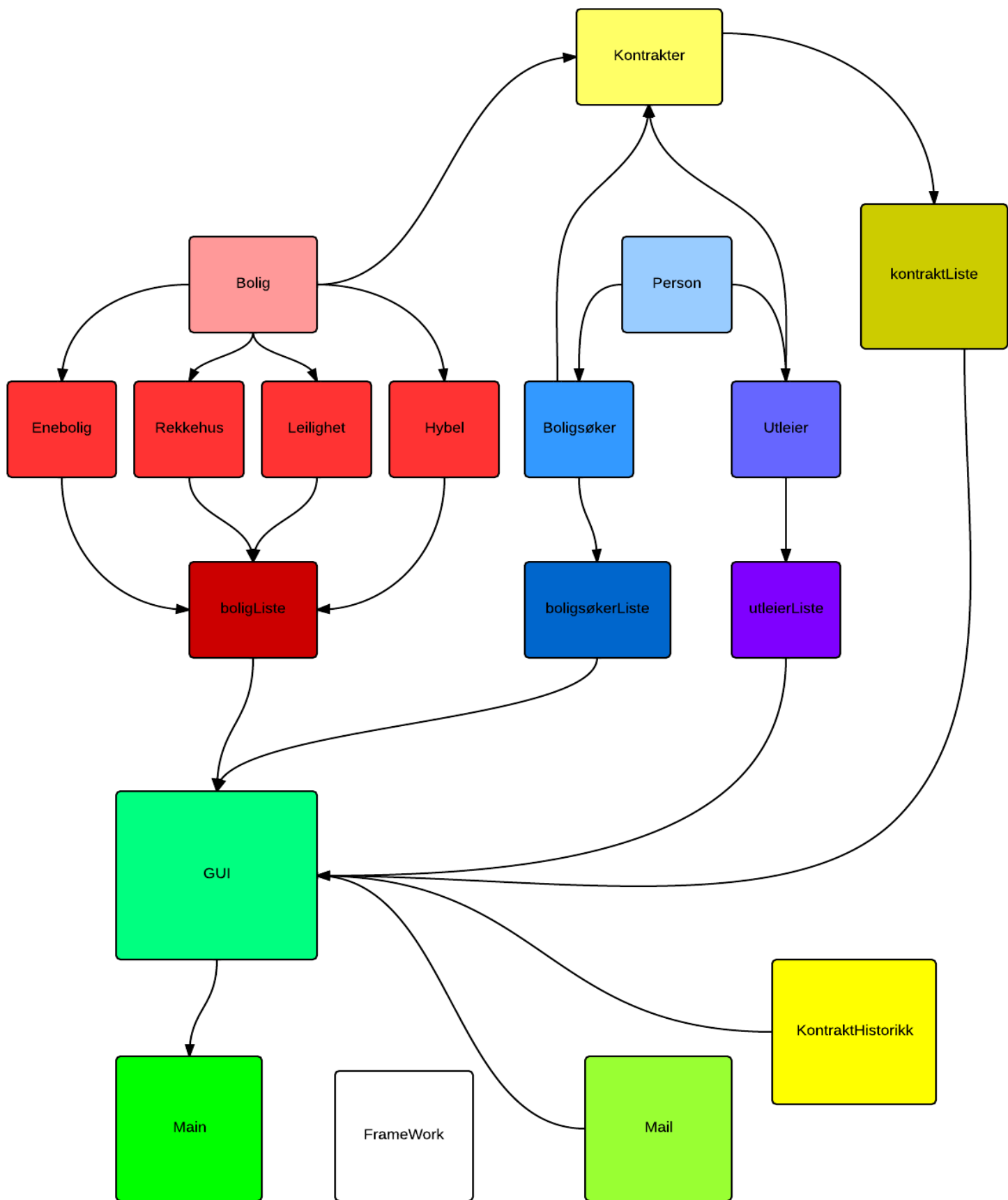
Fanen "MatchMaking" fokuserer på å finne en bolig som passer for en valgt boligsøker og gir bruker av programmet mulighet til å sende informasjon om passende boliger på mail til boligsøkeren. Klassebiblioteket man trenger for å sende mail med java inngår ikke i Javas standard klassebibliotek og denne funksjonaliteten faller da utenfor oppgavens rammebetingelser. **Vi er klar over dette, men har likevel lagt til funksjonen som et easter-egg. Vi ber om at denne funksjonen blir sett bort i fra under sensur.**

For å finne boliger som matcher på en boligsøkers krav, velger bruker en boligsøker fra tabeller og trykker finn match. Programmet har lagret sammensvarende int-arrayer av boligsøkerenes krav og boligens spesifikasjoner (forklart nærmere i avsnitt om datastruktur). Programmet sammenlikner disse arrayene og lager en tabell med de boligene som matchet på nok krav. Disse blir skrevet ut til tabellen med resultater i brukergrensesnittet.

### Oppretting av kontrakter og visning av kontraktregisteret

Kontrakter lagres som forklart i avsnittet om lagring. Med en gang en kontrakt blir lagret blir den vist i tabellen med lagrede kontrakter. For å se mer detaljert informasjon om kontraktene, kan bruker markere en av kontraktene og informasjonen blir skrevet ut i feltet under tabellen.

## PROGRAMSTRUKTUR



Programmets struktur er bygget rundt listene de forskjellige dataen lagres i. GUI-klassen snakker til de forskjellige listeklassene, som igjen snakker til objektene og datafeltene. All innhenting av inndata skjer i GUI-klassen. Objekter bli også opprettet i GUI-klassen og blir sendt herfra for å bli lagret i listeklassene.

## DATASTRUKTUR

Programmet baserer seg på listestruktur. Som vil si at all data som blir behandlet av programmet er lagret i lister. Dette gjør at programmet blir dynamisk og at en kan til en hver tid lagre ny data uten at programmet må utvide lagringskapasiteten. Noe et program som var basert på arrayer måtte ha gjort.

Programmet lagrer og utfører operasjoner på data som er lagret i fire forskjellige lister(egentlig sju, men dette kommer vi tilbake til).

Vi har to forskjellige lister som lagrer data om personer. Disse er boligsøkerListe og utleierListe. BoligsøkerListe er programmert manuelt. Det vil si at vi som programmere har programmert listen ved å gi alle boligsøker objekter en peker kalt "neste". Denne pekeren er en referanse til neste boligsøker som er lagt til i listen. Veldig enkel, og kanskje litt primitivt. En mulighet å endre ved evt. oppdatering av programmet.

For utleierListe bruker vi Collections LinkedList. LinkedList har noen egendefinerte metoder, som gjør slike lister ganske lettvinde å utføre operasjoner på. Blant annet enkle metoder for legge til og fjerne objekter. Den har også en sorter metode som kan bestemmes ved å definere egne rekkefølger. Som vi gjør i utleierSammenlikner. Listen blir sortert alfabetisk.

LinkedList kunne også blitt brukt på boligsøkerListe for å gjøre koden enklere, og listen bedre å utføre operasjoner på.

BoligListe er fire forskjellige TreeSets. En for hver boligtype. Grunnen til TreeSet er fordi slike lister sorteres automatisk når objekter blir lagt til. Bolig objektene blir sortert etter id, fra høyest til lavest.

Grunnen til at vi har fire forskjellige TreeSets er fordi vi ville ha en match-funksjon som baserte seg på hva en boligsøker ønsket seg. Da tenkte vi at det er mye enklere og raskere dele boligtypene opp i forskjellige lister. Siden en boligsøker kun kan ønske seg en bestemt boligtype trenger vi bare å lete i en av listene.

KontraktListe er igjen en manuelt programmert liste, som boligsøkerListe. Grunnen til dette er at vi i denne versjonen av programmet ikke gjør så mange operasjoner mot kontrakt objekter at vi følte det var nødvendig å implementere noen av Collections lister.

For hver gang programmet avsluttets skrives all data til fil. Dette gjøres med ObjectOutputStream og ObjectInputStream, som skriver listen-objektene til fil. Når programmet

startes leser den automatisk all data fra fil. Programmet vil da kunne avsluttes uten at data går tapt.

Vi føler at dette fungerer for slik programmet er i dag. Det er sannsynlig at dette må endres for å øke effektiviteten og oversikten ved videre utvikling, og når mengden data lagret i programmet blir veldig stor.

## DYPERE FORKLARING FOR VIDEREUTVIKLING

Når vi nå skal forklare hvordan en del av koden i programmet fungerer setter vi som en forutsetning at den som skal sette seg inn i koden for å vedlikeholde, videreutvikle eller vurdere den har grunnleggende kunnskaper om java og objektorientert programmering. Det vil si at vi går mer i dybden i noen metoder enn andre for at ikke det ikke skal bli for omfattende.

Alle listene har enkle metoder for å sette inn og fjerne objekter, spesielt de listen som implementerer en av Collections sine lister. Disse er såpass enkle at vi ikke forklarer de.

Dagens versjon av programmet henter og finner mye informasjon basert på et objekts id. Du vil finne flere slike metoder, spesielt i listeklassene. De henter forskjellig informasjon, men er bygget opp på veldig lik måte, ved at den går igjen listen til en finner et objekt som har samme id som den i parameterlista til metoden. Det gjøres på litt forskjellig måte fra hvilke liste du befinner deg i. Eksempel fra boligsøkerListe:

```
public Boligsøker getBoligsøker(String i)
{
    Boligsøker løper = første;

    if(løper == null)
        return null;

    while( løper != null )
    {
        if( løper.getId().equals(i))
            return løper;
        løper = løper.neste;
    }
    return null;
}
```

Her ser vi at et vi oppretter et boligsøker objekt likt det det første i lista. Så "løper" metoden igjennom lista til den finner et objekt som har lik id som den i parameterlista, for så å returnere objektet. Like metoder vil du finne i kontraktlisten, siden begge listene er programmert manuelt.

Neste eksempel er fra boligliste:

```
public Bolig finnBolig(String inn)
{
    int id;
    try{
        id = Integer.parseInt(inn);
    }
    catch (NumberFormatException nfe)
    {
        return null;
    }

    Iterator<Rekkehus> it = rekkehus.iterator();
    Rekkehus re = null;
    Iterator<Enebolig> ite = eneboliger.iterator();
    Enebolig en = null;
    Iterator<Hybel> iter = hybler.iterator();
    Hybel hy = null;
    Iterator<Leilighet> iterat = leiligheter.iterator();
    Leilighet lei = null;

    while(it.hasNext() || ite.hasNext() || iter.hasNext() || iterat.hasNext()){
        if(it.hasNext())
            re = it.next();
        if(ite.hasNext())
            en = ite.next();
        if(iter.hasNext())
            hy = iter.next();
        if(iterat.hasNext())
            lei = iterat.next();

        if (re != null && re.getId() == id )
            return re;
        else if (en != null && en.getId() == id)
            return en;
        else if (hy != null && hy.getId() == id)
            return hy;
        else if (lei != null && lei.getId() == id)
            return lei;
    }
    return null;
}
```

Siden vi her bruker Collections har vi ikke neste pekeren vi kan forholde oss til. Men Collections har metoder for å lete igjennom lister. Vi har her brukt flere Iterator objekter for å kunne lete igjennom alle de fire listene samtidig. Iterator objektene “blar” i listene. Dette gjøres helt til en av bolig objektene har samme id som det i parameterlista til metoden. Den returnerer så dette bolig objektet.

Som sagt er det slik metoder overalt i koden, og det er disse som gjør at programmet fungerer slik det skal. Det er mulig at noen av disse metodene kan endres og forbedres ved videre utvikling. Men mønsteret for gjennomgang er likt for alle metodene.

I GUI brukes det JTables til å vise det meste av informasjon programmet har lagret. Det kreves at informasjonen vi vil vise i JTables kommer i form av en Stringarray. Derfor vil du i alle klassene under GUI finne metoder som disse(bilde på neste side):



```

public String[][] tilTabell()
{
    String[][] ut = new String[liste.size()][6];
    Utleier utleier;
    sorter();
    Iterator<Utleier> iter = liste.iterator();
    int i = 0;

    while(iter.hasNext())
    {
        utleier = iter.next();
        ut[i++] = utleier.tilTabell();
    }
    return ut;
}

//Samme metode som over, bare at den tar med id'n til utleieren
public String[][] tilTabellMedId()
{
    String[][] ut = new String[liste.size()][7];
    Utleier utleier;
    sorter();
    Iterator<Utleier> iter = liste.iterator();
    int i = 0;

    while(iter.hasNext())
    {
        utleier = iter.next();
        ut[i++] = utleier.tilTabellMedId();
    }
    return ut;
}

```

Disse metodene returnerer Stringarrayer som gjør det mulig å vise informasjon om hvert enkelt objekt i listene. Som du ser blir listene gjennomgått på samme måte som eksemplene over. Disse metodene er kun til for å vise riktig informasjon via JTables i GUI. Hvis en ved en videre utvikling av programmet velger å gå bort fra JTables kan disse metodene fjernes.

I GUI viser vi som sagt alt i JTables, og noen/de fleste av gangene vil vi at vi skal kunne hente ut informasjon når vi klikker på en rad i JTable objektet. Dette gjøres ved å lage lyttere til JTable objektet.

```

private class Utvalgslytter implements ListSelectionListener {
    private TableModel tabellmodell;

    public Utvalgslytter(TableModel m) { tabellmodell = m; }

    public void valueChanged(ListSelectionEvent e) {
        if (e.getValueIsAdjusting())
            return;

        ListSelectionModel lsm = (ListSelectionModel) e.getSource();
        if (tabellmodell instanceof resultatTabellModell) {
            if (!lsm.isSelectionEmpty()) {
                System.out.println("Lytter til riktig vindu");
                int valgtRad = lsm.getMinSelectionIndex();
                valgtRad = resultatTabell.convertRowIndexToModel(valgtRad);
                int id;
                try {
                    id = (int) tabellmodell.getValueAt(valgtRad, 9);
                } catch (NullPointerException npe) {
                    return;
                }
                String stringId = Integer.toString(id);

                if (fane.getSelectedIndex() == 4) {
                    String uid = boliger.finnUtleier(boliger.finnBolig(stringId));
                    valgtUtleier.setText(uid);
                    valgtBolig.setText(stringId);
                    valgtUtleier.setVisible(true);
                    utleierLabel.setVisible(true);
                    velgBoligVindu.dispose();
                }
                else if (fane.getSelectedIndex() == 3)
                {
                    valgtBoligId = stringId;
                    sendEmail();
                }
            }
        }
    }
}

```

```

private class resultatTabellModell extends AbstractTableModel {

    int[] kravene = boligsøkere.getKravPåId(valgtId);
    //int type = kravene[0];

    String[] kolonnenavn = getKolonneNavnForBoligtype(kravene);

    Object[][] celler = boliger.matchPåKrav(kravene);

    public int getRowCount() { return celler.length; }

    public int getColumnCount() {
        return celler[0].length;
    }

    public Object getValueAt(int rad, int kolonne) { return celler[rad][kolonne]; }

    public String getColumnName(int kolonne) { return kolonnenavn[kolonne]; }

    public Class getColumnClass(int k) {
        for(int i = 0; i < getRowCount(); i++) {
            Object value = getValueAt(i, k);
            if (value != null)
                return value.getClass();
        }
        return Object.class;
    }
}

private void visMatch() {
    clearResultatPanel();
    resultatTabellModell resultatModell = new resultatTabellModell();
    // resultatTabell = new JTable(resultatModell);
    resultatTabell = new SvartHvitRad(resultatModell);

    TableRowSorter<resultatTabellModell> sorterer = new TableRowSorter<>(resultatModell);

    List<TableRowSorter.SortKey> sortKeys
        = new ArrayList<RowSorter.SortKey>();
    sortKeys.add(new RowSorter.SortKey(0, SortOrder.DESCENDING));

    resultatTabell.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    ListSelectionModel lsm = resultatTabell.getSelectionModel();
    lsm.addListSelectionListener(new Utvalgslytter(resultatModell));
}

```

De øvrige kode eksemplene viser hvordan du oppretter slike tabeller og hvordan du legger til lyttere på JTable objektene slik at vi får en interaktiv tabell.

Men det viktigste og kanskje mest komplekse i koden er hvordan vi finner match mellom boligsøkere og boliger. Dette er den største funksjonen i programmet, og kan være litt vanskelig å sette seg inn i. Eksempel på neste side.

```

public Object[][] matchPåKrav(int[] krav)
{
    DecimalFormat df = new DecimalFormat("0.00");
    Object[][] ut = null;
    int plass = 0;
    int[] specs;
    double matches = 0;
    double matchkoeffisient;
    double urelevante = 0;
    int teller = 0;
    Object[][] dummy = {"Fant", "desverre", "ingen", "passende", "bolig", "for", "valgt", "boligsøker", "!"};
    Object[][] initial = {"Velg", "en", "søker", "for", "å", "finne", "match", "2", "!"};

    if(krav==null)
        return initial; //Når det kjøres for første gang.

    // enebolig
    if(krav[0] == 1) {
        Iterator<Enebolig> iter = eneboliger.iterator();
        Enebolig bolig = iter.next();
        ut = new Object[eneboliger.size()][10];
        while (iter.hasNext()) {
            specs = bolig.getSpecArray();
            if (krav[10] < specs[10] && krav[11] > specs[10] && !bolig.getUtleid() && krav[1] == specs[1]) {
                for (int o = 1; o <= 5; o++) {
                    if (krav[o] == 0)
                        urelevante++;
                    if (specs[o] == krav[o])
                        matches++;
                }
                if (matches >= (3 - urelevante)) {
                    matchkoeffisient = matches / (5 - urelevante);
                    ut[plass] = bolig.tilMatchTabell();
                    ut[plass++][0] = df.format(matchkoeffisient);
                    teller++;
                }
            }
            matchkoeffisient = 0;
            matches = 0;
            urelevante = 0;
            bolig = iter.next();
        }
    }
}

```

h Main

Bildet viser bare en liten del av matchPåKrav metoden i boliglister, men resten er veldig likt som det som står inne i den største if-blokken.

Metoden har en array i parameterlista som blir sendt med fra GUI. Denne er boligsøkeren som vi ønsker å matche sine krav til boliger. Den sjekker først hvilke boligtype vi er på utkikk etter. Krav[1] == 1 = en enebolig, Krav[1] == 2 = et rekkhus, osv.

Ut fra denne if setning vet vi hvilke av TreeSet listene vi skal lete i. Deretter begynner vi å gå gjennom listen. Her bruker vi kjent måte ved hjelp av et Iterator objekt til å "bla" oss igjennom lista. For hvert boligobjekt får vi inn en ny array med datafeltene til en bolig. Hvis prisen, stedet og om boligen ikke er utleid går vi inn i if-blokken. Her skjer magien. Her sjekker metoden om enkelte "relevante" spesifikasjoner til boligen stemmer med boligsøkerens krav. Om den har nok likheter legges den til i arrayen som skal returneres fra metoden.

For hver enkelte boligtype er det litt forskjellige spesifikasjoner, og derfor ser de litt forskjellige ut, men de fungerer på samme måte, bare noen forskjellige verdier.

Ulempen med denne metoden kommer når programmet skal videreutvikles. Hvis en skal feks. legge til flere kriterier hos boligsøker og bolig MÅ også kravarray (i boligsøker) og specarray (i bolig) endres og oppdateres. Samtidig så må også metoden matchPåKrav sjekke de riktige stedene i kravarray og specarray.

Dette kan være en tungvindt måte å “macthe” boligsøkere og boliger. Så ved en videreutvikling kan det hende at dette blir erstattet med en enklere og mer effektiv måte.

Men slik som programmet er i dag, fungerer dette utmerket, og gir programmet et løft.

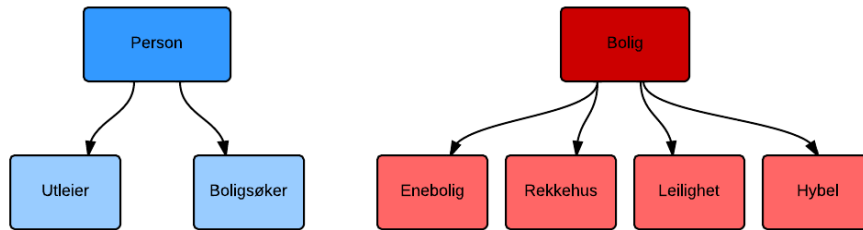
Når det registreres kontrakter settes en boolean verdi i både boligsøker og bolig til true, slik at de ikke skal dukke opp på matching, og når vi skal velge boligsøker/bolig for registrering av ny kontrakt. Eneste måten i dag for å slette en kontrakt er å slette boligsøkere i register vinduet.

Det går heller ikke ann å slette en bolig som er utleid.

Denne funksjonaliteten kan utvides og forbedres ved senere utvikling.

Vi har som sagt ikke konkret gått inn i hver enkelt metode(unntatt macthPåKrav), men heller forklart hvordan en kan enkelt bruke metoder som allerede ligger i klassen, eller skrive egne metoder som egner seg for allerede skrevet kode.

## KLASSEHIERARKI



Person er superklasse for: Utleier og Boligsøker.

Bolig er superklasse for: Enebolig, Rekkehus, Leilighet, Hybel.

Person er en abstrakt klasse som inneholder datafelt og metoder som er felles for utleier og boligsøkere. Mens den abstrakte klassen bolig inneholder datafelt og metoder som er felles for de fire forskjellige boligtype klassene.

## PRIORITERINGER

Det er lagt vekt på den grunnleggende funksjonaliteten beskrevet av oppdragsgiver.

Det er mulig å registrere nye klienter i form av utleiere og boligsøkende. Om begge persontyper lagres ønsket personalia. For boligsøkende lagres også krav til bolig. For utleiere kan det registreres flere boliger i deres navn, lagret med spesifikasjoner, og bilde av boligen. Vi har nedprioritert muligheten til å laste inn bilde av boligsøkende, det kan imidlertid enkelt implementeres i en senere versjon (alle personer vises nå med et standardbilde). Det er enkelt å bla seg gjennom all informasjonen som er lagret i programmet, sortert på henholdsvis personer og boliger. Informasjonen i registeret kan også skrives ut. Om personer vises personalia, og deres krav til bolig. Om bolig vises spesifikasjoner, bilde av bolig, hvem som eier boligen samt hvorvidt den er ledig. Videre er det mulig å velge en boligsøkende person fra en liste, for så å sammenligne/matche kravene til personen, med boliger lagret i registeret. Hvis det finnes en passende bolig for boligsøkende genereres den en tekst i et eget editerbart felt, som enkelt kan sendes med e-post til aktuelle personer.

Det er mulig å opprette en kontrakt mellom utleier og boligsøkende hvor det informeres om kontraktens start og slutt, hvem kontrakten angår og hvilken bolig det er snakk om. Alle kontrakter lagres i et kontraktregister som kan skrives ut.

Vi har valgt å nedprioritere noen av ønskene til oppdragsgiver.

Oppdragsgiver ville ha mulighet til å ha flere bilder knyttet til en bolig. Vi har valgt å begrense det til ett bilde per bolig.

Vi har også valg å nedprioritere søk på kriterier, og heller fokusert på søk i navn og adresse.

Vi bruker JOptionPane til visning av dialogvinduer der det faller seg naturlig. Det var et ønske fra vår side å erstatte dette med egenproduserte vinduer i form av klassen "FrameWork", som vi ikke rakk å ferdigstille. Klassen er tatt med i programmet for å kunne ferdigstille til en senere versjon.

# PROSESSDOKUMENTASJON

Prosjektoppgave Programutvikling  
2014

For gruppe 35:

Kristoffer Gard Osen, s198754, Dataingeniør

Christer Erik Bang, s198737, Dataingeniør

Emil Oppegård, s198772 Dataingeniør

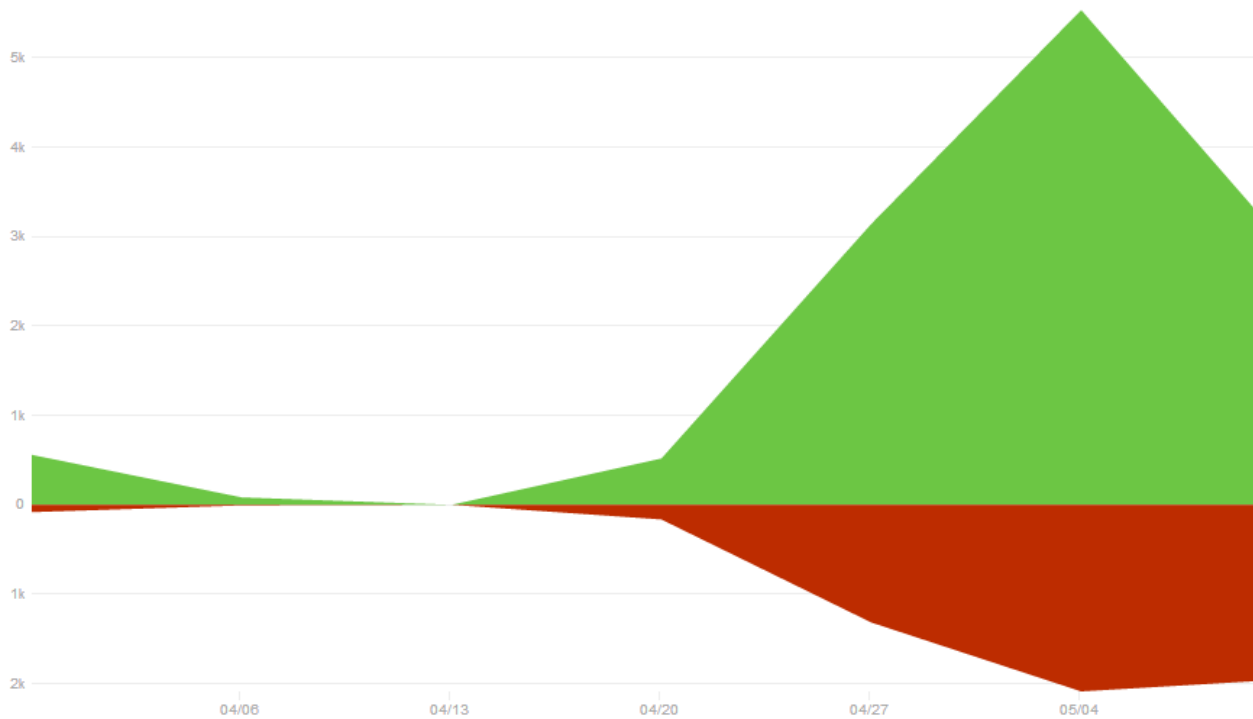


## Hvordan har vi holdt oss til planlagt tidsskjema?

Uke 14	Innlevering av kravspesifikasjon
Uke 15	Oppstart
Uke 16 - (påske)	Ferdig med de grunnleggende klassene
Uke 17	Metoder, GUI
Uke 18	Debugging
Uke 19	Ferdigstille / Sluttrapport
Uke 20	Innlevering - 16.mai

den opprinnelige tidsplanen

Å levere kravspesifikasjonen i uke 14 gikk veldig greit. Vi var tidlig ute, veldig effektive og var ferdig et par dager før fristen. Deretter fulgte en uke preget av store innleveringer i andre fag og det ble gjort lite arbeid på prosjektoppgaven. Påsken kom, og vi hadde fått gjort unna store deler av de grunnleggende klassene, så vi tok oss ferie stort sett hele påsken. Når vi kom tilbake på skolen, begynte vi å føle litt på tidspresset og startet arbeidet for alvor. Den største delen av koden ble skrevet i uke 18-19.



Grafikk fra GitHub

Ettersom at vi kom senere i gang enn planlagt, ble spesielt uke 19 hektisk og vi la ned flere timer hver dag enn det vi kanskje hadde sett for oss. Den opprinnelige planen sier at programmet skulle vært ferdigstilt i uke 19 og at vi skulle bruke uken på å skrive dokumentasjon. Dette var grunnet det korte intervallet mellom denne prosjektoppgaven og neste eksamen. I realiteten ble ikke programmet ferdigstilt før i slutten av uke 19. Et par dager ble brukt på å fjerne noen bugs, mens vi parallelt skrev deler av dokumentasjonen og ryddet i koden. Onsdag 14.05 gjorde vi de siste finpussene og sa oss ferdig med programkoden.

## Hvordan har vi arbeidet?

Når det gjelder koden, ønsker vi ikke å kreditere enkelte klasser eller metoder til en enkeltperson. Selvsagt har vi ikke vært tre om å skrive enkle set- og getmetoder, det er de større metodene hvor alle har bidratt. Det har gjerne fungert slik at en har skrevet et førsteutkast for en metode/funksjon, den er så blitt testet og feil og magler er blitt oppdaget, for så å bli utbedret i plenum. Vi har alle vært åpne til forandringer på kode som vi selv har skrevet, hvis det så ut til å gjøre programmet bedre. I hovedsak har Christer Bang hatt ansvar for Gui-programmeringen, mens Emil Oppegård og Kristoffer Osen har hatt ansvar for datastruktur og annen backend-programmering. Som forutsett ble datastrukturen mye tidligere ferdig enn brukergrensesnittet, så Emil og Kristoffer har jobbet like mye med det som datastrukturen.

## Hvordan har programmet utviklet seg utover den opprinnelige visjonen?

I kravspesifikasjonen skrev vi en liste over klasser vi mente vi trengte å ha med. Den er betydelig kortere enn det antall klasser vi endte opp med. Vi lagde også en modell over programstrukturen og denne er veldig lik det vi endte opp med. Den eneste forskjellen fra den modellen og den vi presenterer i produkt rapporten er et mellomledd mellom Gui-klassen og liste-klassene. Vi forventet at gui-klassen kom til å bli så omfattende at det kunne bli vanskelig å navigere i den. Når vi begynte å utvikle, så vi derimot ikke problemet like godt og valgte å ikke ha noe mellomledd. Vi ser nå, når Gui-klassen er ferdigstilt, at det kan være vanskelig å navigere i koden for en som ikke har satt seg inn i programmet, og at det hadde vært fordelaktig å ha med dette mellomleddet.

Når det gjelder funksjoner vi planla å implementere, er det litt forskjeller. Dette er nok grunnet at vi ikke hadde en helt klar visjon av hvordan vi skulle matche boligsøkers krav på boliger i registeret. Vi skrev at vi ville se og søke i registeret i samme fane som vi ville finne matchende boliger til boligsøker. Lengre ut i utviklingen så vi at det ville være hensiktsmessig å skille disse to funksjonene.

Det ferdigstilte programmets utseende deler samme konsept som det vi presenterte i kravspesifikasjonen, men en del er endret og tilpasset. Vi har brukt faner, som vi hadde planlagt fra starten. De er delt inn etter funksjonalitet og er mer intuitive enn de tidlige utkastene.

# BRUKERVEILEDNING

Prosjektoppgave Programutvikling  
2014

For gruppen:

Kristoffer Gard Osen, s198754, Dataingeniør

Christer Erik Bang, s198737, Dataingeniør

Emil Oppegård, s198772 Dataingeniør

# I n n h o l d

1. Innledning
  2. Starte programmet
  3. Lagre en klient
  4. lagre en bolig
  5. Se register
  6. Slette boliger og klienter
  7. Finn boliger som passer for boligsøker
  8. Send mail
  9. Opprett kontrakt
  10. Se kontrakter
  11. Se all kontrakthistorikk
- 

## 1 . I n n l e d n i n g

Dette er et program for å behandle et register av boliger, utleiere og boligsøkere. Du, bruker av programmet, vil fungere som en tredjepart som formidler boliger mellom utleiere og boligsøkere. I dette programmet har du mulighet til å lagre utleiere, boligsøkere og boliger, finne boliger som passer for boligsøkerene og videreformidle disse til søkeren, dermed opprette kontakt mellom boligsøkere og utleiere. Du kan også lagre kontrakter, se og gjøre endringer i registeret.

## 2. L a g r e   e n   k l i e n t

For å lagre en klient må du først velge fanen "Registrer person". Skriv først inn korrekt data i feltene "Fornavn", "Etternavn", "Adresse", "Mail" og "Telefon".

# Registrer en klient

Fornavn:

Etternavn:

Adresse:

Mail:

Telefon:

☐ Utleier ☒ Boligsøker

Hvis personen du skal registrere er en utleier, velger du det i valget under feltene. Da vil et felt for firma bli synlig. Skriv inn firma. Skriv inn "Privat" hvis utleier ikke er knyttet til et firma. Klikk "Registrer" for å lagre. Hvis personen du registrerer er en boligsøker, velger du boligsøker i valget under feltene. Da vil disse feltene bli synlige:

Boligtype:


By:

Rom:

Min Pris: 0

Max Pris:

☐ Røker ☐ Husdyr

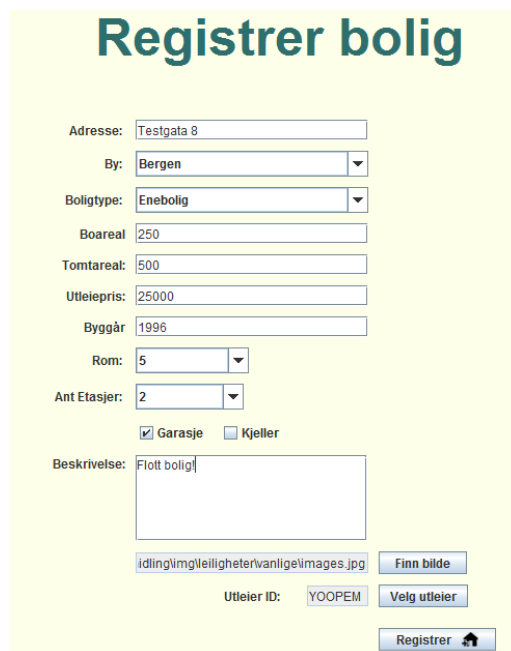


Velg først boligtype. Da vil felter knyttet til boligtypen bli synlig. Fyll inn boligsøkerens preferanser. Hvis en boligsøker ikke har preferanser om et av elementene ved boligen som bestemmes av avkryssingsboksene (feks om eneboligen har kjeller eller ikke), lar du den stå tom. Da blir det registrert at dette elementet ikke skal taes med når du senere skal finne en matchende bolig.

### 3. Lagre en bolig

For å registrere en bolig, velg først fanen “Registrer bolig”. Du vil se flere felter for registrering av data knyttet til boligen. Skriv først inn adresse og velg by. Velg deretter boligtypen. Nå du har valgt boligtypen vil en rekke felter knyttet til den spesifikke boligtypen bli synlig.

1. Fyll inn i alle feltene.
2. Velg bilde til eneboligen ved å trykke på knappen “Velg bilde”. Du vil å opp et vindu hvor du kan velge bilder fra forskjellige mapper. Klikk deg fram til riktig boligtype og finn et passende bilde.
3. Trykk på knappen “Velg Utleier” for å velge eier av boligen. Du vil få opp et vindu med en tabell over alle utleiere. Velg riktig utleier.<sup>7</sup>
4. Trykk på “register” for å lagre bolien.



**Registrer bolig**

Adresse:

By:

Boligtype:

Boareal:

Tomtareal:

Utleiepris:

Byggår:


Rom:

Ant Etasjer:

☒ Garasje ☐ Kjeller

Beskrivelse:

Utleier ID:



Eksempel på registrering av enebolig

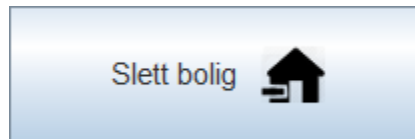
### 4. Se register

Registeret av alle boliger og klienter er tilgjengelig i fanen “Register”. Her kan du velge mellom “Personregister” og “Boligregister”. Personregister viser både utleiere og boligsøkere. Klikk på på enten en person eller en bolig i tabellen for å se mer detaljert informasjon i feltet under tabellen.

## 5. Slette boliger og klienter

Sletting av boliger gjøres i fanen "Register".

For å slette en bolig, velg først "Boligregister", deretter velg en bolig fra tabellen. Trykk på knappen "Slett bolig" for å slette.



For å slette en person, velg først "Personregister", deretter velg en person fra tabellen. Trykk på knappen "Slett person" for å slette.



## 6. Finn boliger som passer for en boligsøker

For å finne boliger som passer til en spesifikk boligsøker velger du fanen "MatchMaking". Her vil du se to tabeller. Den øverste er en tabell med alle registrerte boligsøkere.



# MachMaking

Id	Fornavn	Etternavn	Adresse	Email	Telefon
BØRE	Reidar	Børre	uaingouj	nuijgnqoun	uogn
OSKR	kristoffer	osen	Sinselveien 5b	heipemai@gmail.com	92881543
TRKÅ	Kåre	Traaseth	jaoigho	hwichouq	ouhgo
JOJA	Jan	Johansen	aigho	hohguqo	ohoguqh
DAPO	Pokker	Da	ospigj	iqoiqgo	ioiqgj
OSKE	Ketil	Osen	Grendatunvegen 29d	ketil.osen@me.com	90702827
OSEL	Elizabeth	Osen	økkLØK	økl	ølk

Finn Match!



Velg en boligsøker i tabellen og klikk på knappen “Vis match”. Boliger som samsvarer med boligsøkerens krav vil da blir skrevet ut i den nederste tabellen.

Finn Match!



Matchresultat	By	Areal	Pris/m	Adresse	Antall rom	Parkering
1,00	Bergen	100000 m²	1000 kr/m	Testgata 8	1	<input type="checkbox"/>
1,00	Bergen	100000 m²	1000 kr/m	Liksomveien 3	1	<input type="checkbox"/>
1,00	Bergen	100000 m²	1000 kr/m	Grendatunvegen...	1	<input type="checkbox"/>

## 7. Send mail om match

For å sende en mail til en boligsøker, gjør først som beskrevet i punkt nr.6 for å finne matchende boliger. Velg så en bolig fra tabellen nederst på skjermen. Når du har valgt en bolig, vil det bli skrevet en forhåndsgenerert tekst ut i feltet under overskriften “Mail”. Dette er eposten som skal sendes. Den inneholder allerede info om boligsøker og den aktuelle boligen, men du som bruker står fritt til å redigere den. Trykk på knappen “Send mail” for å sende.

## Mail:

I lei Jan Johansen.  
Vi har funnet en bolig som vi tror passer for  
Boliq: Lurehauken  
Oslo  
15000  
Hvis denne passer for deg, send oss en mail  
Mvh. BoliqFormidling.

Send mail



Mail Sendt!

## 8. Opprett en kontrakt

Opprett kontrakt

Velg en Leietaker OKNR

Kontrakten starter:

Dag(DD)

Måned(MM)

År(AAAA)

Kontrakten slutter

Dag(DD)

Måned(MM)

År(AAAA)

Lagre kontrakt

Lagrede kontrakter

Utteier	Leietaker	Startdato	Sluttdato	ID
Opet	Joke	registrert	noen	kontrakter

Oppdater Register

For å opprette en kontrakt, velg først fanen “Kontrakter”. Du vil få opp dette vinduet:

Kontrakter registreres i den venstre delen av vinduet.

1. Velg først boligsøkeren som skal leie ved å klikke på knappen “Velg leietaker”. Du vil få opp et vindu med en tabell med boligsøkere. Klikk på en boligsøker for å velge.
2. Knappen velg bolig vil nå bli synlig. Klikk på den for å få opp en tabell med boliger som passer til valgt boligsøker. Klikk på en bolig for å velge den.
3. Fyll inn datofeltene og pass på at startdatoen er før sluttdatoen.
4. Klikk på “Lagre kontrakt” for å lagre kontrakten.

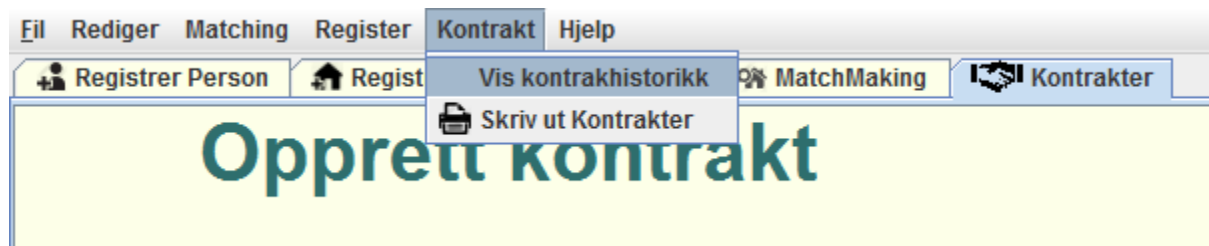
## 9. Se kontrakter

Kontrakter vises automatisk i tabellen i fanen “Kontrakter” når de blir registrert. Hvis du ikke finner kontrakten du leter etter, klikk på knappen “Oppdater register”.

For å vise detaljert informasjon om en kontrakt, velg en kontrakt fra tabellen og informasjonen vil bli skrevet ut i feltet under tabellen.

## 10. Se all kontrakthistorikk

For å se all lagret kontrakthistorikk, velg “Kontrakt” i menylinjen, og deretter “Vis kontrakthistorikk”. Du vil da få opp et vindu med all kontrakthistorikk.



# TESTRAPPORT

Prosjektoppgave Programutvikling  
2014

For gruppen:

Kristoffer Gard Osen, s198754, Dataingeniør

Christer Erik Bang, s198737, Dataingeniør

Emil Oppegård, s198772 Dataingeniør

## Registrering av personer

Handling	Ønsket respons	Resultat
Registrere utleier som allerede finnes i register	Feilmelding, endrer farge på felt til rød. *	Ok
Registrere utleier som allerede finnes i register	Feilmelding, endrer farge på felt til rød. *	Ok
Unnløte å fylle inn alle felter i utleier registrering	Aktuelt felt endrer farge til rød. *	Ok
Tallverdi i Adressefelt	Felt endrer farge til rødt. *	
Registrere boligsøker som allerede finnes i register	Feilmelding.	Ok
Registrere mailadresse med ugyldig format.	Mailfeltet endre farge til rød *	Ok
Registrere telefonnummer med ugyldig format.	Telefonfeltet endrer farge til rød *	Ok
Tallverdi i firmafelt	Endre farge til rødt	Ok
Unnløte å velge by	Felt endrer farge til rød *	Ok
Unnløte å velge rom	Felt endrer farge til rød *	Ok
Unnløte å velge etasje	Felt endrer farge til rød *	Ok
Unnløte å velge plan	Felt endrer farge til rød *	
Skrive inn riktig informasjon der felt har blitt farget rødt (merket med * over)	Felt endrer farge til hvit	Ok

## Registrere bolig

Handling	Ønsket respons	Resultat
----------	----------------	----------

Unnlate å fylle ut påkrevde felt (adresse, boareal, pris, byggår, tomtareal, etasjer)	Felt blir rødt	Ok
Unnlate å velge by	Felt blir rødt	Ok
Unnlate å velge boligtype	Felt blir rødt	Ok
Tallverdi i adressefelt	Felt blir rødt	Ok
Bokstaver eller symboler i boarealfelt	Felt blir rødt	Ok
Negativ tallverdi i boarealfelt	Felt blir rødt	Ok
Bokstaver eller symboler i prisfelt	Felt blir rødt	Ok
Negativ tallverdi i prisfelt	Felt blir rødt	Ok
Bokstaver eller symboler i byggårfelt	Felt blir rødt	Ok
Skrive inn år etter 2016	Felt blir rødt	Ok
Negativ eller lavere enn 1000 i byggårfelt	Felt blir rødt	Ok
Bokstaver eller symboler i tomtarealfelt	Felt blir rødt	Ok
Negativ tallverdi i tomtarealfelt	Felt blir rødt	Ok
Utleier ID med ugyldig format	Felt blir rødt	Ok

## Register

Handling	Ønsket respons	Resultat
Trykker på fanen "Register"	Første gang: Personregister opprettes og lastes inn Hvis radioknapp er valgt: last inn valgt register.	Ok
Endre type register	Valgt register lastes inn på nytt.	Ok
Finner aktuell person ved søk på fornavn/etternavn	Personens plassering blir markert.	Ok
Trykke på slettknapp uten å ha valgt person/bolig	Ingen hendelse	Ok

Velge person fra tabell	Informasjon om peroson vises på skjerm	Ok
Velge bolig fra tabell	Informasjpn om bolig vises på skjerm, hvis bilde er lagt ved vises også dette	Ok

## MatchMaking

Handling	Ønsket respons	Resultat
Valgt boligsøker, trykke "match"	Resultattabellen viser passende boliger, hvis det ikke finnes match gis det beskjed om det	Ok
Valgt bolig fra matchtabell	Generere mailtekst med informasjon om bolig.	Ok
Trykk send mail	Teksten fra mailfelt sendes til valgt mailadresse.	Ok
Ingen match funnet	Tilbakemelding	Ok
Mail mislykkes	Feilmelding	Ok
Mail er sendt	Tilbakemelding	Ok

## Kontrakter

Handling	Ønsket respons	Resultat
Velg en leietaker	vindu med tabell åpnes	Ok
Velg en bolig for leietaker uten match	Feilmelding	Ok
Velg en leietaker fra tomt register	Feilmelding	Ok
Oprette kontrakt med sluttdato før startdato	Feilmelding	Ok
Skrive bokstaver i datofelt	Feilmelding	Ok
Unnlate å fylle inn ett/flere felter	Feilmelding	Ok
Lagre kontrakt med gyldige verdier	Kontrakt opprettes og vises i tabell	Ok
Velge kontrakt fra tabell	Kontraktens innhod vises i display	Ok

## Filmeny

Handling	Ønsket respons	Resultat
Fil-> Lagre	Programmet skriver til fil	Ok
Fil -> Avslutt	Programmet avsluttes med exit kode 0.	Ok
Matching -> Skriv ut matchresultat	Viser utskriftsdialog, skriver ut	Ok
Register -> Skriv ut Boligregister	Viser utskriftsdialog, skriver ut	Ok
Register -> Skriv ut Personregister	Viser utskriftsdialog, skriver ut	Ok
Kontrakt -> vis kontrakthistoprikk	Åpner vindu som inneholder alle kontrakter	Ok
Kontrakt -> Skriv ut kontrakter	Viser utskriftsdialog, skriver ut	Ok
Hjelp-> Om	Åpner vindu med informasjon om programmet	Ok
Status -> Statistikk	Åpner vindu med informasjon om antall boliger og registrerte personer i registeret	Ok.