Høgskolen i Telemark

EKSAMEN

6109 OBJEKTORIENTERT PROGRAMMERING

15.12.2014

Tid:	9-13
Målform:	Bokmål/nynorsk
Sidetall:	11 med forside
Hjelpemidler:	Alle trykte og skrevne. Java API dokumentasjon er tilgjengelig lokalt på hver maskin.
Merknader:	Eksamen blir gjennomført på PC uten tilgang på Internett, men satt opp med standard programvare og NetBeans. Besvarelsen skal leveres elektronisk i form av et ferdig NetBeans-prosjekt. Hele mappestrukturen lagres på utdelt minnepinne. Skriv kandidatnummeret ditt i toppen av hver fil.
Vedlegg:	Ingen

Eksamensresultater blir offentliggjort på StudentWeb.

Avdeling for allmennvitenskapelige fag

Bokmål

Oppgavesettet tar for seg ulike deler av et program for å håndtere utleie av containere.

Du skal opprette et NetBeans-prosjekt `Eksamen` med standardpakke `eksamen`. Et komplett prosjekt skal inneholde følgende klasser, der innrykk indikerer subklasser:

- `Container`
- `Oppdrag`
  - `DagOppdrag`
  - `TimeOppdrag`
- `ContainerModell`
- `ContainerGUI`
- `TegneVindu`

Du **skal** bruke **samme klassenavn** for å lette sensur. Du skal også opprette en tom tekstfil med navn `logg.txt` i prosjektmappen.

Merk: Det er ikke nødvendig å markere hvilke oppgaver ulike deler av koden hører til. Noen deloppgaver kan kreve at du legger til kode i flere klasser. Legg til hjelpemetoder der du synes det er hensiktsmessig, og pass på at koden er korrekt innrykket.

Prosentsats ut for oppgaver og deloppgaver antyder vekt ved sensur. Prøv å svare på så mange spørsmål som mulig. Lykke til!

Oppgave 1. Klasser og subklasser (35%)

Du skal i denne oppgaven lage klasser for å ta vare på data om containere og oppdrag. Det er to slags oppdrag:

- *Dag-oppdrag*: Kunden får disponere en container en hel dag, og kan sette minstekrav til størrelsen på containeren målt i antall kubikkmeter. Bedriften står imidlertid fritt til å bruke en større container.
- *Time-oppdrag*: Kunden leier en container for et fritt valgt antall timer, men kan da ikke velge størrelse på containeren. Bedriften kan dermed bruke alle slags containere til slike oppdrag.

I begge tilfeller vil bedriften sørge for å tømme innholdet på nærmeste gjenvinningsstasjon.

1-a (10%)

Programmer klassen `Container`. Den skal ha objektvariabler for å ta vare på løpenummeret til containeren (en unik tallkode), volum i antall kubikkmeter (heltall) og leiepris pr. time (desimaltall).

Klassen skal ha en konstruktør og get-metoder for objektvariablene. Konstruktøren skal kaste et unntak hvis volum eller leiepris er negative tall.

1-b (10%)

Programmer klassen `Oppdrag`. Denne klassen skal ha en objektvariabel for å ta vare på adressen til kunden og en referanse til et `Container`-objekt. Sørg for at begge disse objektvariablene er direkte tilgjengelige i eventuelle subklasser (skal lages i oppgave 1-c).

Konstruktøren skal kun ta adressen som parameter, og ha en set-metode for å knytte en container til oppdraget.

Klassen `Oppdrag` skal være abstrakt og ha to objektmetoder `minsteVolum` og `pris`. Se oppgave 1-c for en nærmere beskrivelse av oppførselen til disse metodene i subklassene.

- Metoden `minsteVolum` tar ingen parametere. Den skal returnere minimumsvolumet (heltall) til en container for dette oppdraget.
- Metoden `pris` tar heller ingen parametere og skal returnere prisen for oppdraget (desimaltall).

1-c (15%)

Programmer klassene `DagOppdrag` og `TimeOppdrag` som subklasser av `Oppdrag`.

`DagOppdrag` skal ha en objektvariabel for å ta vare på minimumsvolum for containere som kan brukes. Prisen for et slikt oppdrag er kr. 500 pr. kubikkmeter.

`TimeOppdrag` skal ha en objektvariabel for å ta vare på leietiden i antall timer. Prisen på et slikt oppdrag er gitt ved leietiden og leieprisen til containeren. Minimumsvolumet for slike oppdrag skal være 0, alle slags containere kan altså brukes.

Begge klasser bør ha en `toString`-metode som viser hva slags oppdrag det er snakk om, adressen til kunden, løpenummeret til containeren som er brukt og prisen på oppdraget.

Oppgave 2. Objektsamlinger og filer (30%)

Klassen `ContainerModell` skal holde styr på en liste med ledige containere og en liste med aktive oppdrag. Du skal bruke to `ArrayList` objektvariabler til dette, se kode under.

2-a (10%)

Programmer konstruktøren til klassen, som forklart med kommentarer i koden under.

2-b (10%)

Programmer metoden `regOppdrag` for å registrere et nytt oppdrag, som forklart med kommentarer i koden under.

2-c (10%)

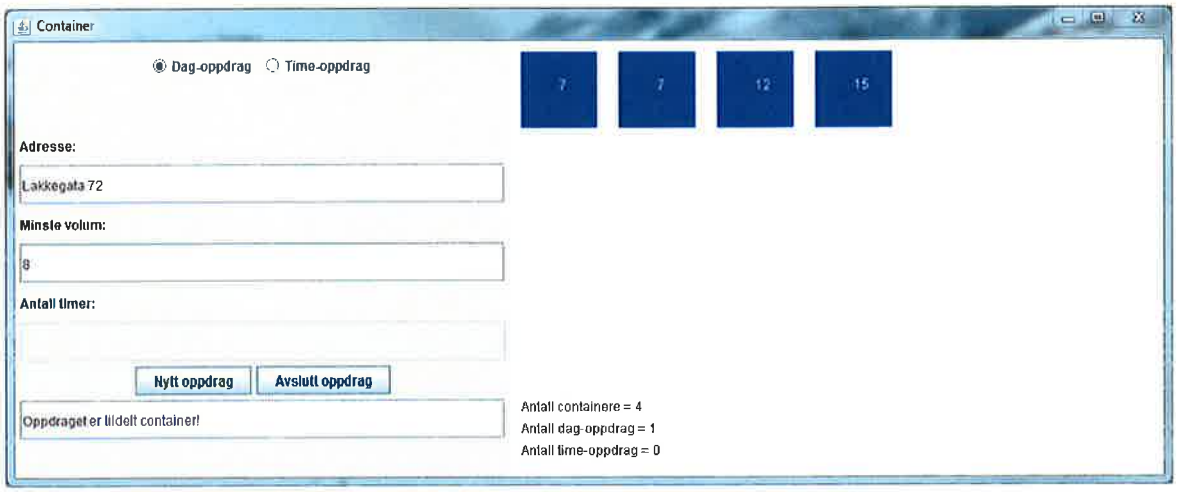
Programmer metoden `avsluttOppdrag` for å avslutte et oppdrag, som forklart med kommentarer i koden under.

```
public class ContainerModell {

  private ArrayList<Container> ledige; // ledige containere
  private ArrayList<Oppdrag> oppdrag;  // aktive oppdrag

  // Legg eventuelt til flere variabler.

  public ContainerModell() {

    int[] volum = {7, 7, 9, 12, 15};
    double[] timepriser = {500, 550, 600, 900, 950};

    // Opprett 5 containere med data fra tabellene over,
    // og legg disse Container-objektene inn i listen ledige.
    // Gi Container-objektene løpenummer fra 1 og oppover.
    // Sett objektvariabelen oppdrag til å være en tom liste.
  }

  public String regOppdrag(Oppdrag oppdrag) {

    // Finn en ledig container, hvis mulig.
    // For dag-oppdrag må containeren være stor nok.
    // Velg den minste mulige containeren.
    // Sett oppdraget inn i listen oppdrag.

    // Returner en passende statusmelding.
  }

  public String avsluttOppdrag(String adresse)
                                      throws IOException {

    // Finn aktivt oppdrag på gitt adresse i listen oppdrag.
    // Det er aldri mer enn 1 oppdrag pr. adresse.
    // Frikoble containeren fra oppdraget.
    // Legg containeren tilbake i listen med ledige containere.
    // Fjern oppdraget fra listen oppdrag.

    // Skriv oppdraget som en ny linje til slutt i loggfilen
    // ved hjelp av toString-metoden i oppdragsklassene.

    // Returner en passende statusmelding.
  }

  // Legg eventuelt til flere metoder.
}
```

Oppgave 3. GUI (35%)

Du skal nå bygge opp deler av brukergrensesnittet til et program for å administrere oppdrag og containere, som vist i figuren under.

Når brukeren registrerer et nytt oppdrag skal programmet finne en passende container.



Brukeren skal kunne registrere nye oppdrag og avslutte oppdrag. Dette skal gjøres ved bruk av metoder i klassen `ContainerModell`. Du kan løse denne oppgaven selv om du ikke har svart på oppgave 2. Lag eventuelt dummy-versjoner av metodene i `ContainerModell` for testing.

Programmet skal også vise listen med ledige containere grafisk til høyre i vinduet, sammen med statusinformasjon om antallet ledige containere og aktive oppdrag.

Følgende viser et kodeskjelett for klassen `ContainerGUI`:

```
public class ContainerGUI
    extends JFrame implements ActionListener {

    private ContainerModell cm;

    private ButtonGroup oppdragstype;
    private JRadioButton btnDag, btnTime;
    private JButton btnReg, btnAvslutt;
    private TegneVindu vindu;
    private JTextField txtAdresse, txtVolum, txtTimer,
                       txtStatus;
    // ...
}
```

3-a (20%)

Du skal programmere deler av klassen `ContainerGUI` i henhold til kodeskjelettet over.

Merk: Det er tilstrekkelig å forklare med kommentarsetninger hvordan man kan bruke paneler og layout-klasser for å oppnå det visuelle resultatet i venstre del av vinduet vist over.

- Brukeren kan velge mellom dag-oppdrag og time-oppdrag med radioknappene øverst.
- Brukeren skal deretter kunne skrive inn en adresse, og enten minimumsvolum (heltall) eller antall timer (heltall) i tekstfeltene.
- Radioknappen for dag-oppdrag skal deaktivere tekstfeltet for antall timer, mens radioknappen for time-oppdrag skal deaktivere tekstfeltet for minimumsvolum.
- Når brukeren klikker knappen «Nytt oppdrag» (`btnReg`) skal det registreres enten et `DagOppdrag` eller et `TimeOppdrag` ved hjelp av metoden `regOppdrag` i klassen `ContainerModell`. En passende melding skal deretter bli vist i statusfeltet.
- Når brukeren klikker knappen «Avslutt oppdrag» (`btnAvslutt`) skal oppdraget avsluttes ved hjelp av metoden `avsluttOppdrag` i klassen `ContainerModell`.

3-b (15%)

Denne oppgaven tar for seg grafikkdelen av brukergrensesnittet (høyre del av vinduet).

Programmer klassen `TegneVindu` som en subklasse av `JPanel`. Klassen bør ha tilgang på et objekt av klassen `ContainerModell`.

Øverst i panelet skal hver ledige container bli vist som et blått rektangel med volumet skrevet i hvitt. Hvis det er mange ledige containere skal de om nødvendig tegnes i flere «rader».

Nederst i panelet skal antall containere, aktive dag-oppdrag og aktive time-oppdrag skrives ut ved hjelp av metoden `drawString`.

Nynorsk

Oppgåvesettet tar for seg ulike delar av eit program for å handtere utleige av containarar.

Du skal opprette eit NetBeans-prosjekt `Eksamen` med standardpakke `eksamen`. Eit komplett prosjekt skal innehalde følgjande klasser, der innrykk indikerer subklasser:

- `Container`
- `Oppdrag`
  - `DagOppdrag`
  - `TimeOppdrag`
- `ContainerModell`
- `ContainerGUI`
- `TeikneVindauge`

Du **skal** nytte **same klassenamn** for å lette sensur. Du skal og opprette ei tom tekstfil med namn `logg.txt` i prosjektmappa.

Merk: Det er ikkje naudsynt å markere kva for oppgåver ulike delar av koden høyrer til. Nokre deloppgåver kan krevje at du legg til kode i fleire klasser. Legg til hjelpemetodar der du meiner det er føremålstenleg, og pass på at koden er korrekt rykka inn.

Prosentsats ut for oppgåver og deloppgåver antyder vekt ved sensur. Freist å svare på så mange spørsmål som mogleg. Lykke til!

Oppgåve 1. Klasser og subklasser (35%)

Du skal i denne oppgåva lage klasser for å ta vare på data om containarar og oppdrag. Det er to slags oppdrag:

- *Dag-oppdrag*: Kunden får disponere ein container ein heil dag, og kan setje minstekrav til storleiken på containeren målt i kubikkmeter. Bedrifta står derimot fritt til å nytte ein større container.
- *Time-oppdrag*: Kunden leiger ein container for eit fritt tal timar, men kan då ikkje vele storleik på containeren. Bedrifta kan dermed nytte alle slags containarar til slike oppdrag.

I begge høve vil bedrifta syte for å tømme innhaldet på næraste gjenvinningsstasjon.

1-a (10%)

Programmer klassa `Container`. Den skal ha objektvariablar for å ta vare på løpenummeret til containeren (ei unik talkode), volum i talet på kubikkmeter (heiltal) og leigepris pr. time (desimaltal).

Klassa skal ha ein konstruktør og get-metodar for objektvariablane. Konstruktøren skal kaste eit unntak viss volum eller leigepris er negative tall.

1-b (10%)

Programmer klassa `Oppdrag`. Denne klassa skal ha ein objektvariabel for å ta vare på adressa til kunden og ei referanse til eit `Container`-objekt. Syt for at begge desse objektvariablane er direkte tilgjengelege i eventuelle subklasser (skal lagast i oppgåve 1-c).

Konstruktøren skal berre ta adressa som parameter, og ha ein set-metode for å knytte ein container til oppdraget.

Klassa `Oppdrag` skal være abstrakt og ha to objektmetodar `minsteVolum` og `pris`. Sjå oppgåve 1-c for nærare forklaring av oppførselen til desse metodane i subklassane.

- Metoden `minsteVolum` tar ingen parametrar. Den skal returnere minimumsvolumet (heiltal) til ein container for dette oppdraget.
- Metoden `pris` tar heller ingen parametrar og skal returnere prisen for oppdraget (desimaltal).

1-c (15%)

Programmer klassene `DagOppdrag` og `TimeOppdrag` som subklasser av `Oppdrag`.

`DagOppdrag` skal ha ein objektvariabel for å ta vare på minimumsvolum for containarar som kan nyttast. Prisen på eit slikt oppdrag er kr. 500 pr. kubikkmeter.

`TimeOppdrag` skal ha ein objektvariabel for å ta vare på leigetida i timar. Prisen på eit slikt oppdrag er gitt ved leigetida og leigeprisen til containeren. Minimumsvolumet for slike oppdrag skal være 0, alle slags containarar kan dermed nyttast.

Begge klasser bør ha ein `toString`-metode som syner kva slags oppdrag det er snakk om, adressa til kunden, løpenummeret til containeren som er nytta og prisen på oppdraget.

Oppgåve 2. Objektsamlinger og filer (30%)

Klassa `ContainerModell` skal halde styr på ei liste med ledige containarar og ei liste med aktive oppdrag. Du skal nytte to `ArrayList` objektvariablar til dette, sjå kode under.

2-a (10%)

Programmer konstruktøren til klassa, som forklart med kommentarar i koden under.

2-b (10%)

Programmer metoden `regOppdrag` for å registrere eit nytt oppdrag, som forklart med kommentarar i koden under.

2-c (10%)

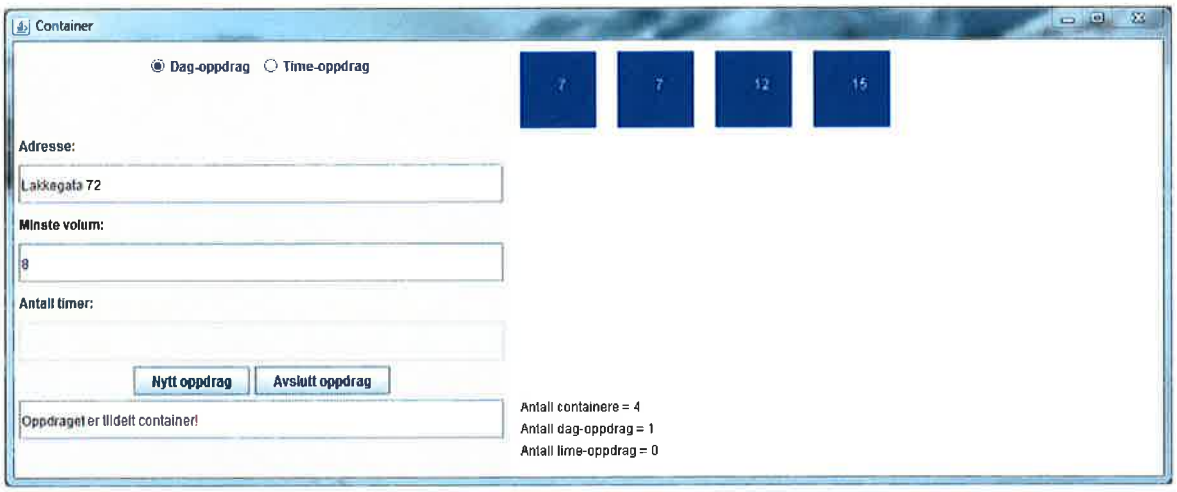
Programmer metoden `avsluttOppdrag` for å avslutte eit oppdrag, som forklart med kommentarar i koden under.

```
public class ContainerModell {

  private ArrayList<Container> ledige; // ledige containarar
  private ArrayList<Oppdrag> oppdrag;  // aktive oppdrag

  // Legg eventuelt til fleire variablar.


  public ContainerModell() {

    int[] volum = {7, 7, 9, 12, 15};
    double[] timeprisar = {500, 550, 600, 900, 950};

    // Opprett 5 containarar med data frå tabellane over,
    // og legg desse Container-objekta inn i lista ledige.
    // Gi Container-objekta løpenummer frå 1 og oppover.
    // Sett objektvariabelen oppdrag til å være ei tom liste.
  }

  public String regOppdrag(Oppdrag oppdrag) {

    // Finn ein ledig container, viss mogleg.
    // For dag-oppdrag må containeren være stor nok.
    // Vel den minste moglege containeren.
    // Sett oppdraget inn i lista oppdrag.

    // Returner ein passande statusmelding.
  }

  public String avsluttOppdrag(String adresse)
                                      throws IOException {

    // Finn aktivt oppdrag på gitt adresse i lista oppdrag.
    // Det er aldri meir enn 1 oppdrag pr. adresse.
    // Frikople containeren frå oppdraget.
    // Legg containeren tilbake i lista med ledige containarar.
    // Fjern oppdraget frå lista oppdrag.

    // Skriv oppdraget som ei ny line til slutt i loggfila
    // ved hjelp av toString-metoden i oppdragsklassene.

    // Returner ein passande statusmelding.
  }

  // Legg eventuelt til fleire metodar.
}
```

Oppgåve 3. GUI (35%)

Du skal no bygge opp delar av brukargrensesnittet til eit program for å administrere oppdrag og containarar, som syna i figuren under.

Når brukaren registrerer eit nytt oppdrag skal programmet finne ein passande container.



Brukaren skal kunne registrere nye oppdrag og avslutte oppdrag. Dette skal gjerast ved hjelp av metodar i klassa `ContainerModell`. Du kan løyse denne oppgåva sjølv om du ikkje har svart på oppgåve 2. Lag eventuelt dummy-versjonar av metodane i `ContainerModell` for testing.

Programmet skal og syne lista med ledige containarar grafisk til høgre i vindauget, saman med statusinformasjon om talet på ledige containarar og aktive oppdrag.

Følgjande syner eit kodeskjelett for klassa `ContainerGUI`:

```
public class ContainerGUI
    extends JFrame implements ActionListener {

    private ContainerModell cm;

    private ButtonGroup oppdragstype;
    private JRadioButton btnDag, btnTime;
    private JButton btnReg, btnAvslutt;
    private TegneVindu vindauge;
    private JTextField txtAdresse, txtVolum, txtTimar,
                       txtStatus;
    // ...
}
```

3-a (20%)

Du skal programmere delar av klassa `ContainerGUI` i henhold til kodeskjelettet over.

Merk: Det er tilstrekkelig å forklare med kommentarsetningar korleis ein kan nytte panelar og layout-klasser for å oppnå det visuelle resultatet i venstre del av vindauget syna over.

- Brukaren kan vele mellom dag-oppdrag og time-oppdrag med radioknappane øvst.
- Brukaren skal kunne skrive inn ei adresse og anten minimumsvolum (heiltal) eller tal på timar (heiltal) i tekstfelta.
- Radioknappen for dag-oppdrag skal deaktivere tekstfeltet for timetal, medan radioknappen for time-oppdrag skal deaktivere tekstfeltet for minimumsvolum.
- Når brukaren klikkar knappen «Nytt oppdrag» (`btnReg`) skal det registrerast anten eit `DagOppdrag` eller eit `TimeOppdrag` ved hjelp av metoden `regOppdrag` i klassa `ContainerModell`. Ein passande melding skal deretter skrivast ut i statusfeltet.
- Når brukaren klikkar knappen «Avslutt oppdrag» (`btnAvslutt`) skal oppdraget avsluttast ved hjelp av metoden `avsluttOppdrag` i klassa `ContainerModell`.

3-b (15%)

Denne oppgåva tar for seg grafikkdelen av brukargrensesnittet (høgre del av vindauget).

Programmer klassa `TeikneVindauge` som ei subklasse av `JPanel`. Klassa bør ha tilgang på eit objekt av klassa `ContainerModell`.

Øvst i panelet skal kvar ledige container bli syna som eit blått rektangel med volumet skreve i kvitt. Viss det er mange ledige containarar skal dei om naudsynt teiknast i fleire «rader».

Lengst ned i panelet skal talet på containarar, aktive dag-oppdrag og aktive time-oppdrag skrivast ut ved hjelp av metoden `drawString`.