Høgskolen i Telemark

UTSATT EKSAMEN

5609 OBJEKTORIENTERT PROGRAMMERING

30.5.2011

Tid:	*9.00 – 13.00*
Målform:	*Bokmål*
Sidetall:	*6 (inkludert denne forsiden og ett vedlegg)*
Hjelpemidler:	*Alle trykte og skrevne (IKKE elektroniske)* *Java API dokumentasjon er elektronisk tilgjengelig lokalt på PC'en.*
Merknader:	*Besvarelsen kan skrives med tekstprogram som ett dokument.* ***Lever besvarelsen i oppgaverekkefølge og skriv oppgavenummer foran hver delbesvarelse.*** ***Besvarelsen skal leveres som papirutskrift påført kandidatnummer på hver ark.***
Vedlegg:	Utdrag av Java API dokumentasjon for class Arrays

Eksamensresultata blir offentliggjort på Studentweb.

Besvarelsen kan skrives med et tekstbehandlingsprogram (Word, TextPad eller Notisblokk), uten kompileringsmulighet for Java. Fargekoding av Java syntaks er tillatt. Hele besvarelsen skal skrives som ett dokument (én fil) for å forenkle utskrift.

Besvarelsen skal leveres på papir, dvs. som utskrift fra skriver. Eksamensvaktene vil hjelpe til med utskrift. Du må selv se gjennom og kontrollere utskriften. Du er ansvarlig for at det som leveres inn er komplett. Figurer og tegningen som du har tegnet for hånd legges ved besvarelsen ved innlevering. Tid til utskrift og kontroll regnes ikke inn i eksamenstiden.

Oppgavesettet består av tre oppgaver som kan løses uavhengig av hverandre. Selv om det er én oppgave du ikke har løst, kan du løse neste som om den forrige var løst. Du kan utvide klassene fra tidligere oppgaver med nye metoder og variabler etter hvert som du får behov for det, men vis tydelig hva som er svar på hver hovedoppgave.

Disponer tiden godt, slik at du får gjort mest mulig på alle tre oppgavene. Det er viktigere å få fram programlogikken, enn at programmet er helt fritt for syntaksfeil og kjørbart. Om en oppgave er uklar så lag dine egne forutsetninger, og forklar disse.

Felles problemstilling for de tre oppgavene
Den nye operaen i Bjørvika trenger et program med grafisk brukegrensesnitt til å holde orden på billettbestillinger. Du skal lage (deler av) en første versjon av dette programmet, som kun skal kunne håndtere billettene til én bestemt forestilling på hovedscenen.

Oppgave 1 Modellklasser (30 %)

I denne oppgaven skal du bestemme og lage modellklassene for programmet basert på følgene opplysninger:

I salen til operaens hovedscene skal det være det 22 rader med 15 seter i hver rad. Hvert sete er unikt identifisert med rad- og setenummer.

Det finnes tre forskjellige typer "kunder" i operaen:

Tilfeldige: Dette er kunder som ringer og reserverer ett eller flere seter per telefon. For hver tilfeldig kunde skal det lagres navn og telefonnummer, samt hvilke seter som er bestilt. Når kunden møter fram og henter de reserverte billettene skal dette registreres. (Du kan anta at kunden henter alle eller ingen av de reserverte billettene.)

Privatabonnent: Dette er en person som abonnerer på ett fast sete i salen til alle forestillinger. For hver Privatabonnent skal det lagres navn, adresse og telefonnummer, samt hvilket sete det abonneres på.

Bedriftsabonnent: Dette er en bedrift som abonnerer på ett eller flere faste seter i salen til alle forestillinger. For hver bedriftsabonnent skal det lagres navn på firmaet, adresse, kontaktperson og telefonnummer, samt hvilke seter bedriften abonnerer på.

Oppgave 1a

De ulike "kundetypene" skal representeres som Java-klasser i et klassetre. **Programmer alle klassene i klassetreet med konstruktører og fornuftige get- og set-metoder.**

Oppgave 1b

Programmer en ny klasse **Sal** som kan representere alle setene i salen slik at disse lar seg reservere. Du kan forutsett at alle rader i én bestemt sal har samme antall seter, men ulike saler kan ha forskjellig anrall rader (og seter pr. rad.) Reservasjoner skal representeres som referanser til objekter av klassene i 1a. Klassen må ha disse public metodene:

- **reserverSete** – skal kalles med hensiktsmessige parametere for å få lagt inn en ny reservasjon/abonnement på et sete i salen.
- **erReservert** – skal sjekke om et bestemt sete er reservert eller abonnert på. I så fall skal metoden returnere aktuelt reservasjons/abbonnementsobjekt
- **slettReservasjon** - skal kalles for å slette en reservasjon/abonnement på et sete i salen.

Oppgave 1c

Skriv et Java testprogram med main-metode som:

1) Oppretter et objekt som representerer salen.
2) Oppretter ett objekt for hver av de tre "kundetypene" og "plasserer" disse på i salen. (Du bestemmer selv hvor i salen.)

Oppgave 2 Swing GUI (50 %)

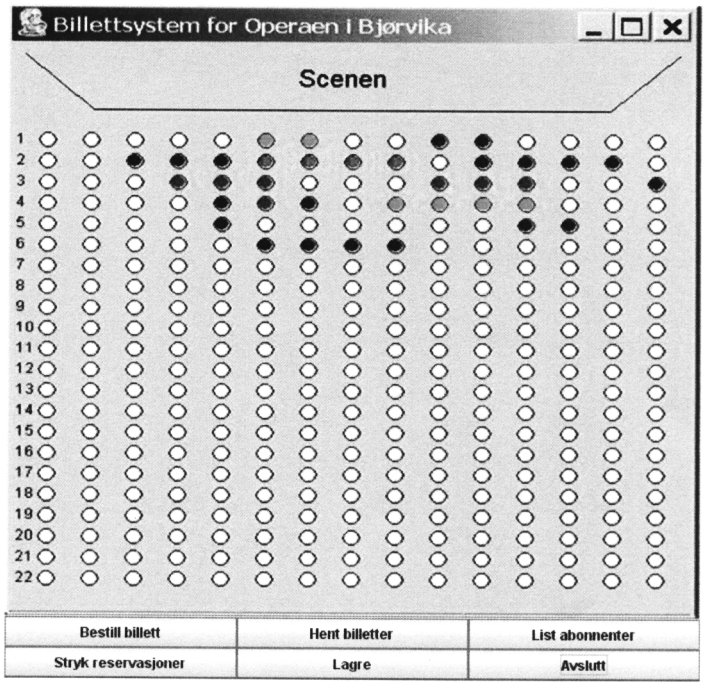
I denne oppgaven skal du programmere et Swing GUI som benytter modellklassene fra oppgave 1 og som kan utføre de oppgavene som er beskrevet nedenfor.

Programmet skal ha grafisk brukergrensesnitt, og skal hele tiden vise frem et kart over salen der det fremgår hvilke seter som til enhver tid er ledige.

Hvert enkelt sete skal vises frem som en sirkel, der sirkelens farge betyr følgende:

- Hvitt = setet er ledig
- Svart = setet er forbeholdt en abonnent (privat eller bedrift)
- Rødt = setet er reservert av en tilfeldig kunde men ikke hentet enda.
- Blått = billett til dette setet er solgt, dvs. en tilfeldig reservasjon som er hentet.
- Grønt = setet er klikket på (markert), men enda ikke tildelt en kunde

Et eksempel på hvordan grensesnittet kan se ut er vist til høyre:



GUI programmet er spesifisert i deloppgavene nedenfor. Dette er bare for gjøre problem-stillingen mer oversiktlig. Programkoden behøver ikke nødvendigvis å ha denne oppdelingen.

Oppgave 2a

Lag et vindu som ser ut som ovenfor, men foreløpig uten farge på setene eller "kode bak" knappene.

Tips:

- Setene bør tegnes som sirkler med fast avstand i et eget JPanel.
- I en JPanel kan du finne bredde og høyde på ved kall på: **`getWidth()`** og **`getHight()`**
- Radnr kan "tegnes" i et JPanel med metoden **drawString()**.
- Du kan beregne et setes x og y posisjon i piksler ut fra rad-/setenr med følgende formler:

```
margForRadnr = 20
setebredde = (panelbredde - margForRadnr) / antall seter
radhøyde = panelhøyde / antall rader
x-pos =  margForRadnr  + ((setenr - 1) * setebredde)
y-pos =  (radnr - 1) * radhøyde
```

Oppgave 2b

Fortsett med det du laget under 2a og legg på lyttere på musa og knappene slik at det er mulig å markere hvert av setene.

Tips:

- Bruk interfacet **MouseListener** for å fange musklikk. Se vedlegg bakerst i oppgavesettet.
- Posisjonen til et **`MouseEvent e`** får du med kall på: **`e.getX()` og `e.getY()`**. Se dokumentasjon av metodene i vedlegget.
- Det kan være lurt å lage en hjelpemetode for å beregne rad/setenr fra x,y verdier i JPanelet.

Oppgave 2c

Knappen **Bestill billett:** Først skal brukeren markere (klikke på) de ledige setene som ønskes bestilt. Disse skal da bli grønne. Deretter skal hun trykke på **Bestill billett** knappen, og programmet skal vise et nytt lite vindu der bruker kan skrive inn navn og telefonnummer til den tilfeldige kunden som reserverer billetten. (Det er ikke nødvendig å kode løsning for å "fjerne merking" fra seter.)

Når bruker trykker OK-knapp i det lille vinduet skal det opprettes en bestilling for en reservasjon av typen "tilfeldig" i systemet, og GUIet skal oppdateres. For brukervennlighetens skyld bør det også finnes en Avbryt-knapp i dette vinduet.

Oppgave 2d

Knappen **Hent billetter**: Her skal programmet be om å få oppgitt telefonnummer på en tilfeldig kunde. (Bruk JOptionpane). Programmet skal finne den tilfeldige kunden og deretter markere de reserverte setene på dette telefonnummeret som solgt og oppdatere GUIet. (Obs! Faste abonnenter skal ikke kunne hente billetter).

Oppgave 2e

Knappen **Stryk reservasjoner:** Denne knappen skal brukes manuelt ca. en halv time før forestillingen begynner. Den skal fjerne all informasjon om bestillinger gjort av tilfeldige kunder som ikke har hentet billettene sine.

Oppgave 3 Lagring i database (20%)

Programmer ferdig klassemetoden lagreData() i klassen DBKontroller nedenfor. Metoden skal lagre opplysninger om alle privat- og bedriftsabonnenter (ikke tilfeldige kunder) for en sal, i en Oracle database. Metoden er tenkt kjørt når brukeren trykker på knappen **Lagre data**, men du skal **ikke** å programmere GUI-mekanismene for å kalle metoden.

```
public class DBKontroller {
  static private Connection forbindelse;

  static public void lagreData(Sal s) {...}

  public static boolean åpneDB() {
      try {
        Class.forName("oracle.jdbc.OracleDriver");
        forbindelse = DriverManager.getConnection(
           "jdbc:oracle:thin:@oraserver:1521:OPERA", "user", "passwd");
        return (forbindelse != null);
      } catch (Exception e) { return false; }
  }

  public static boolean lukkDB()throws Exception {
    try {
       forbindelse.close();
        return true;
      } catch (Exception e) { return false; }
  }
}
```

Du kan anta at det er bygget en ferdig database i Oracle med følgende SQL kommandoer:

```
CREATE TABLE ABONNENTER(
  TYPE           CHAR          NOT NULL,
  TLFNR          VARCHAR2(10)  NOT NULL,
  NAVN           VARCHAR2(40)  NOT NULL,
  ADRESSE        VARCHAR2(40)      NULL,
  KONTAKTPERSON  VARCHAR2(40)      NULL,
  CONSTRAINT ABONNENT_PN PRIMARY KEY (TLFNR)
);

CREATE TABLE SETER(
  SETENR         INTEGER       NOT NULL,
  RADNR          INTEGER       NOT NULL,
  TLFNR          VARCHAR2(10)  NOT NULL,
  CONSTRAINT SETER_PN PRIMARY KEY (SETENR, RADNR),
  CONSTRAINT SETER_ABONNENTER_FN FOREIGN KEY (TLFNR) REFERENCES ABONNENTER
);
```

- **Hvis strukturen i tabellen ABONNENTER passer dårlig med modellklassene dine, kan du godt forutsette din egen tabellstruktur.**
- Oracles datatype VARCHAR2 tilsvarer String i Java.
- Feltet TYPE skal inneholde tegnet T for tilfeldige abonnenter, P for bedriftsabonnenter og B for bedriftsabonnenter.
- Kolonnen TLFNR er primærnøkkel i tabellen ABONNENTER og fremmednøkkel i tabellen SETER.

Vedlegg: Utdrag av Java 6 API dokumentasjon

java.awt.event

Interface MouseListener

```
public interface MouseListener
extends EventListener
```

Method Summary	
void	**mouseClicked**(MouseEvent e) Invoked when the mouse button has been clicked (pressed and released) on a component.
Void	**mouseEntered**(MouseEvent e) Invoked when the mouse enters a component.
Void	**mouseExited**(MouseEvent e) Invoked when the mouse exits a component.
void	**mousePressed**(MouseEvent e) Invoked when a mouse button has been pressed on a component.
void	**mouseReleased**(MouseEvent e) Invoked when a mouse button has been released on a component.

java.awt.event

Class MouseEvent

```
public class MouseEvent
extends InputEvent
```

Method Summary (utdrag av metodene)	
int	**getX**() Returns the horizontal x position of the event relative to the source component.
int	**getY**() Returns the vertical y position of the event relative to the source component.

Slutt på oppgavesettet