

i Forside

IDATx2001 Programmering 2

Ordinær Eksamen vår 2020

Emnekode/Emne: IDATA/IDATT/IDATG 2001 Programmering 2

Institutt: IE/IDI og IE/IIR

Eksamensdato: 28.05.2020

Eksamenstid (fra-til): 09:00 - 14:00 (5 t)

Eksamensform: Hjemmeeksamen

Fagleg kontakt under eksamen for Ålesund: Arne Styve

Tlf: 70 16 12 87

Fagleg kontakt under eksamen for Gjøvik: Kiran Raja

Tlf: 61 13 53 74

Fagleg kontakt under eksamen for Trondheim: Majid Rouhani

Tlf: 73 55 93 55

Teknisk hjelp under eksamen: [NTNU Orakel](#)

Tlf: 73 59 16 00

Hjelpemiddelkode/Tillatte hjelpemiddel:

A / Alle hjelpemiddel tillate (lærebok, egne notat, internett), med UNNTAK av verktøy/løysingar for samhandling (chat, meldingstenester, distribuert/sentralisert versjonskontroll, diskusjonsforum og liknande).

ANNAN INFORMASJON:

Gjer deg opp dine egne meningar og presiser i svara dine kva for føresetnadar du har lagt til grunn i tolking/avgrensing av oppgåva. Fagleg kontaktperson skal berre kontaktast dersom det er direkte feil eller manglar i oppgåvesettet.

Lagring: Svara dine i Inspira Assessment vert lagra automatisk. Jobbar du i andre program – hugs å lagre undervegs.

Juks/plagiat: Eksamen skal vere eit individuelt, sjølvstendig arbeid. Det er tillate å bruke hjelpemiddel. Alle svar vert kontrollert for plagiat. [Du kan lese meir om juks og plagiering på eksamen her.](#)

Varslingar: Dersom det oppstår behov for å gje beskjedar til kandidatane medan eksamen er i gang (f.eks. ved feil i oppgåvesettet), vil dette bli gjort via varslingar i Inspira. Eit varsel vil dukke opp som en dialogboks på skjermen i Inspira. Du kan finne att varselet ved å klikke på bjølla i øvre høgre hjørne på skjermen. Det vil i tillegg bli sendt SMS til alle kandidatar for å sikre at ingen går glipp av viktig informasjon. Ha mobiltelefonen din innan rekkevidde.

Filopplasting: Alle filer må vere lasta opp i Inspira før eksamenstida går ut. Det er lagt til 15 minutt til ordinær eksamenstid for digitalisering av handtegningar/filer. (Tilleggstida inngår i attståande eksamenstid som vises i øvre venstre hjørne på skjermen.)

[Slik digitalisera du handteikningane dine.](#)

[Slik lagrar du dokumentet ditt som PDF.](#)

[Slik fjernar du forfattarinformasjon frå fila\(e\) du skal levera.](#)

[Slik lagar du ZIP-fil av ei mappe på Mac og Windows](#)

Alle oppgåvene skal svarast på digitalt i Inspira, enten ved å svare direkte på spørsmål, eller ved å lasta opp fil som vedlegg til spørsmål.

Programmeringsoppgåver blir gjort i eit IDE (BlueJ, Netbeans, IntelliJ, Eclipse etc). Der du i oppgåvene blir bedt om det, kopierer du kode frå IDE til svarfeltet.

OM LEVERING:

Svara dine vert levert automatisk når eksamenstida er ute og prøven stenger, under føresetnad av at du har svart på minst ei oppgåve. Dette skjer sjølv om du ikkje har klikka «Lever og gå tilbake til Dashboard» på siste side i oppgåvesettet. Du kan opne og redigere svara dine så lenge prøven er open. Dersom du ikkje har svart på nokon av oppgåvene ved prøveslutt, blir ingenting levert.

I siste oppgåve i eksamen skal du **lasta prosjektet ditt opp samla (Innlevering) som ein ZIP-fil. Pakk HEILE PROSJEKTMAPPA DI til ein ZIP-fil og last opp.**

Trekk frå eksamen: Ønskjer du å levere blankt/trekke deg, gå til hamburgermenyen i øvre høgre hjørne og vel «Lever blankt». Dette kan ikkje angrast sjølv om prøven framleis er open.

Tilgang til svara dine: Du finn svara dine i Arkiv etter at sluttida for eksamen er passert.

Målform/språk: Nynorsk

i Innledning

Dersom du ønsker å ha heile oppgåvesettet tilgjengeleg som PDF under eksamen, kan du lasta ned oppgåvesettet her: [IDATx 2001 Programmering 2 V2020.pdf](#)

Denne eksamenen er bygd opp som følgjer:

- Oppgåva inneheld ein **kravspesifikasjon** på eit system du skal laga ei løysing på.
- Deloppgåvene i eksamenssettet i Inspira vil leia deg for å komma fram til løysinga. Det er difor viktig at du **les ALLE oppgåvene** i oppgåvesettet i tillegg til kravspesifikasjonen **før** du byrjar å utvikla løysinga.
- Undervegs i oppgåvene vil du bli beden om å lima inn delar av koden din i svarfeltet. **Les instruksjonane grundig før du gir frå seg svar!**
- I aller siste oppgåve ber vi deg om å **lasta opp HEILE prosjektmappa di** som ein ZIP-fil. Det er viktig at du får med deg ditt lokale GIT-repository i ZIP-fila, så IKKJE bruk ZIP-funksjonen i IntelliJ, men pakk prosjektmappa frå filutforskaren. Ta med **alle** under-mapper i prosjektet. Og **sett av tilstrekkeleg med tid til å sikra riktig opplasting!!**

Alle hjelpemiddel er tillatne brukt under eksamen med unntak av samarbeid og kopiering av andre sine løysingar. Plagiatkontroll vil bli gjennomført.

For alle besvarelser gjeld følgjande vurderingskriterium:

- All kode skal følgja **god kodestandard/kodestil** (gode og fornuftige namn på klassar, metodar, variablar og felt. God struktur med innrykk osv.)
- All kode skal dokumenterast iht prinsippa for dokumentasjon (JavaDoc)
- God design iht prinsippa frå IDATx1001 Programmering 1 (**coupling** og **cohesion**)
- Koden vil bli kontrollert med **CheckStyle** og **SonarLint**. (Checkstyle-fil: [idatx2001_checks.xml.zip](#))

i Kravspesifikasjon/Case beskrivelse

I denne eksamensoppgåva skal de implementera delar av ei løysing basert på følgjande kravspesifikasjon/case:



Bomstasjoner

Det er etablert ei mengde bomringar rundt om i Noreg. Desse finn du bla i rundt Ålesund, Gjøvik og Trondheim.

I samband med dette skal det utviklast eit nytt system for å registrera passeringer av køyretøy i bomstasjonane.

Bomstasjonar (engelsk: Toll Plaza) er ofte organisert i grupper i det som kallast **bomringar** (engelsk: Toll ring), der kvar Bomring kan bestå av ein eller fleire bomstasjonar. Det er bomstasjonane som registrerer **bomstasjon passeringer**.

Ein **bomstasjon-passering** (engelsk: "Toll Passage") inneheld typisk følgjande informasjon:

- Dato for passering
- Klokkeslett (24H) for passering
- En ID i form av eit heiltalfor kva bomstasjon (engelsk: "toll collection point") som vart passert
- Registreringsnummer (engelsk: license plate number) til bilen som passerte - ein tekststreng av typen "UF 32512"
- Beløpet i NOK som blir belasta bilen ved passering - eit positivt flyttal/desimaltal

Takstane skil mellom køyretøytype og om passeringen føregår i eller utanfor rushtida.

Rushtid er definert som følgjande tidsrom:

- 06.30 - 08.59
- 14.30 - 16.29

Følgjande typar **køyretøy** skal kunna registrerast av ein bomstasjon, og har følgjande takstar:

Køyretøytype	Takst i rushtiden	Takst utanom rushtiden
Personbil Diesel (engelsk: Diesel Car)	23 kr	19 kr
Personbil Elbil (engelsk: Electrical Car)	8 kr	4 kr
Personbil Bensin/ladbar hybrid (engelsk: Petrol Car)	21 kr	17 kr
Lett lastebil (over 3500 kg) (engelsk: Truck)	101 kr	86 kr
Motorsykkkel/moped/scooter (engelsk: Motorcycle)	0 kr	0 kr

(Motorsykkkel/moped/scooter har registreringsnummer, og blir derfor registrert og må håndteres av systemet).

Eit køyretøy (uansett klasse) er registrert med følgjande opplysningar:

- Registreringsnummer (engelsk. license plate number) - som ein streng på forma "UF 32512"
- Totalvekt i kg - som eit positivt heiltal

Funksjonalitet

Systemet skal ha følgjande funksjonalitet:

- Det skal vera mogleg å registrera ulike køyretøy (engelsk: vehicle) av dei ulike køyretøya typane i eit køyretøyregister
- Det skal vera mogleg å oppretta ein bomstasjon (engelsk: toll plaza).
- Det skal vera mogleg å registrera passeringer av køyretøy på ein bomstasjon.
- Det skal vera mogleg å gjennomføra ulike søk i registeret av passeringer.
- Det skal vera mogleg å skriva ut alle registrerte passeringer, med kost for kvar passering, og dessutan totalkost for alle passeringer for ein bomstasjon.

Oppgåvene vidare i eksamenssettet vil leia deg igjennom implementasjonen av løysinga, så les kvar enkelt oppgåve grundig!

1(a) Oppgave 1 a)

Opprett eit nytt prosjekt i IDEEN din (IntelliJ, Netbeans, Eclipse, BlueJ). Bruk **kandidatnummeret ditt** som namn på prosjektet og på **mappa** som prosjektet blir oppretta i. Brukar du **Maven** har du moglegheit for nokre ekstra poeng :-). Opprett òg **eit lokalt Git repository** for prosjektet ditt (NB! IKKJE bruk GitHub eller annan nettbasert repository).

Set prosjektet ditt til å kompilera/bygga for **JDK 1.8 (Java versjon 8)**.

Skriv i svarfeltet kva IDE du har valt, og kva byggsystem du har valt å nytta for prosjektet.

Skriv svaret ditt her...

Maks poeng: 5

1(b) **Oppgave 1 b)**

Lag klassar for kvar type køyretøy. Kvar klasse skal ha:

- felt for registreringsnummer iht kravspesifikasjonen
- felt for tillaten totalvekt i kg iht kravspesifikasjonen
- tilgangs-metodar (accessor methods)
- ein metode som returnerer kostnaden for ein passering. Denne metoden skal ha eit **parameter** (av typen boolean) som indikerer om passeringen føregår i eller utanom rushtida.

Bruk arv dersom du meiner det vil vera fornuftig her.
Kopier og lim inn koden for klassane du lagar i svarfeltet under (alle klassar)

Skriv svaret ditt her...

1	
---	--

Maks poeng: 10

2(a) **Oppgave 2 a)**

Opprett eit **register** for å halda på registrerte køyretøy. Dette registeret skal ha følgjande funksjonalitet:

- Det skal vera ein metode for å registrera eit køyretøy
- Det skal vera ein metode for å søka etter eit køyretøy basert på **registreringsnummer**.
- Det skal vera ein metode for å søka etter køyretøy (eitt eller fleire) med totalvekt over ein gitt vekt (gitt som parameter). Tips: bruk av streams og filter gir ekstra poeng.

Lag ein **Unit-test** for å testa registeret. Hugs å gjennomføra både **positive** og **negative** testar. (Du må her òg visa at du kan setja opp Unit-testar riktig i IDEEN din).
Vis òg korleis du vil bruka **exceptions** for å handtera unntakstilfelle i registerklassen din.

Når du er ferdig, kopierer og limer du inn koden for **både registeret og Unit-testen** i svarfeltet under **Skriv svaret ditt her...**

1

Maks poeng: 5

2(b) **Oppgave 2 b)**

Grunngi kvifor du meiner at testane du har laga er høvesvis **positiv(e)** og **negativ(e)** test(ar).

Skriv svaret ditt her...

Format

B


I


U


x_2


x^2


I_x






























ABC



Words: 0

Maks poeng: 5

7/18

2(c) **Oppgave 2 c)**

Køytøyregisteret som den norske staten eig, finst berre i eitt eksemplar. Korleis kan du sikra at det ikkje er mogleg å laga meir enn **ein** instans av køytøyregisteret du oppretta i a)? (Hint: pattern)

Oppdater register-koden din og forklar i Javadoc til klassen kva du har gjort.

Lim inn oppdatert kode i svarfeltet under (ta med heile klassen)

Skriv svaret ditt her...

1

Maks poeng: 5

3

Oppgave 3

Det er ønskeleg å handtera alle typar einingar som skal betala ved bompassering gjennom eit standard **interface**. Dette interfacet skal heita *Payable* og skal ha følgjande metodar:

- `getLicensePlateNumber()`
- `getCost(boolean withinRushHours)`

Gjennomfør nødvendig **refaktorering** (engelsk: *refactoring*) på køyretøyklassane ved å la dei implementera interfacet

Når du er ferdig, kopier de klassene du har gjort endringer i og lim de inn i svarfeltet under.

Skriv svaret ditt her...

1

Maks poeng: 10

4(a) **Oppgave 4 a)**

For å registrera ein bompassering, treng vi ein klasse som representerer **bompasseringen** (engelsk: toll passage). Følgjande informasjon skal lagrast for ein enkelt bompassering:

- ID til bomstasjonen der passeringen er registrert - heiltal
- Registreringsnummer til køyretøyet som blir registrert
- Kostnad/takst belasta køyretøyet (blir henta frå køyretøyet med `getCost()`-metoden)
- Tid for passering (tips: bruk klassen `LocalTime`)
- Dato for passering (tips: bruk `LocalDate`)

Når du er ferdig, kopierer du koden for passeringen frå IDE'ein og limer inn i svarfeltet under.

Skriv svaret ditt her...

1

Maks poeng: 10

4(b) **Oppgave 4 b)**

Opprett **registeret** for **bompassering** (engelsk: toll passage register).
Krav til funksjonalitet:

- Metode for å registrera/legge til ei passering.
- Metode som returnera ei liste av passeringer (returnera **List<TollPassage>** eller liknande).
- Registeret skal implementerast slik at det er mogleg å bruka registeret direkte i ein for-each løkke.
Døme: **for (TollPassage tollPassage : tollPassageRegister){}**.

Rushtid er definert som følgjande tidsrom:

- 06.30 - 08.59
- 14.30 - 16.29

Når du er ferdig, kopierer du koden for registeret frå IDE'ein og limer inn i svarfeltet under.

Skriv svaret ditt her...

1

Maks poeng: 10

5(a) **Oppgave 5 a)**

I denne oppgåva skal du ferdigstillå forretningslogikken ved å implementera konseptet **Bomstasjon** (engelsk: Toll plaza). For å redusera omfanget av oppgåva, vel vi å IKKJE implementera Bomring (engelsk: Toll ring).

Bomstasjonen held på alle bompasseringer (bruk bompasseringsregisteret frå oppgåve 4 b)). Bomstasjonen nyttar køyretøyregisteret for å slå opp nødvendige detaljar om køyretøyet for å kunna registrera ein bompassering.

Ein **Bomstasjon** blir kjenneteikna med følgjande informasjon:

- Eit namn, t.d. "Vikebukt bomstasjon"
- Ei unik ID som eit positivt heiltal

Bomstasjon klassen må minimum innehalda ein metode for å registrera ein bompassering. Metoden kan til dømes ha følgjande signatur:

- boolean registerTollPassage(String licencePlateNumber, LocalDate date, LocalTime time)

Lim inn kjeldekoden for **Bomstasjonen** i feltet under.

Skriv svaret ditt her...

1	
---	--

Maks poeng: 7

5(b) **Oppgave 5 b)**

Lag ein enkel klient (ingen GUI eller meny el.l.) som viser at løysinga di fungerer. Klientkoden din bør oppretta ein bomstasjon, og dessutan eit køyretøyregister med nokre køyretøy.
Ein eller fleire bompasseringer skal registrerast i bomstasjonen.
Til slutt skal du kunna henta ut alle passeringer frå bomstasjonen og skriva dei ut i terminalvindauget/konsollet.

Lim inn kjeldekoden for Klienten i feltet under.

Skriv svaret ditt her...

1	
---	--

Maks poeng: 8

6 Oppgave 6

Database

I denne oppgåva skal du visa at du veit korleis du set opp eit prosjekt for bruk av databasar gjennom Java Persistence API'et (JPA).

Last ned følgende ZIP-fil (innehold filene **persistence.xml**, **jakarta.persistence_2.2.3**, **eclipselink.jar** og **derby.jar** for embedded DB): [database-files.zip](#)

- Plasser filene riktig i prosjektet ditt. Dersom du nyttar Maven: oppdater pom-filen med avhengnad (dependency) til jar-filene.
- Utfør nødvendige endringar på klassen TollPassage slik at denne er forberedt for å kunne lagrast i ein tabell i ein database.

Beskriv med egne ord der du plasserte fila(e), og kva endringar du måtte gjera i TollPassage-klassen for å gjera den "databaseklar".

NB! Du skal ikkje implementera full database-funksjonalitet (søkje, lagra osv).

Skriv svaret ditt her...

Maks poeng: 10

7(a) **Oppgave 7 a)**

Designpatterns

```
2 // From JavaFX-application
3 Button okButton = new Button("Click me");
4 Lable label = new Label("No clicked");
5
6 okButton.setOnAction(e ->
7     label.setText("Clicked!")
8 );
```

Sjå på koden over. Kva designpattern er i bruk her?

Vel eitt alternativ

- ☐ Singleton
- ☐ Decorator
- ☐ Observer
- ☐ Factory

Maks poeng: 2.5

7(b) **Oppgave 7 b)**

Designpatterns

```
public class SomeClass {
    private static final Logger logger = Logger.getLogger(SomeClass.class.getName());
    public static void main(String[] args) {
        logger.info("Logger Name: " + logger.getName());
        logger.warning("Can cause ArrayIndexOutOfBoundsException");
    }
}
```

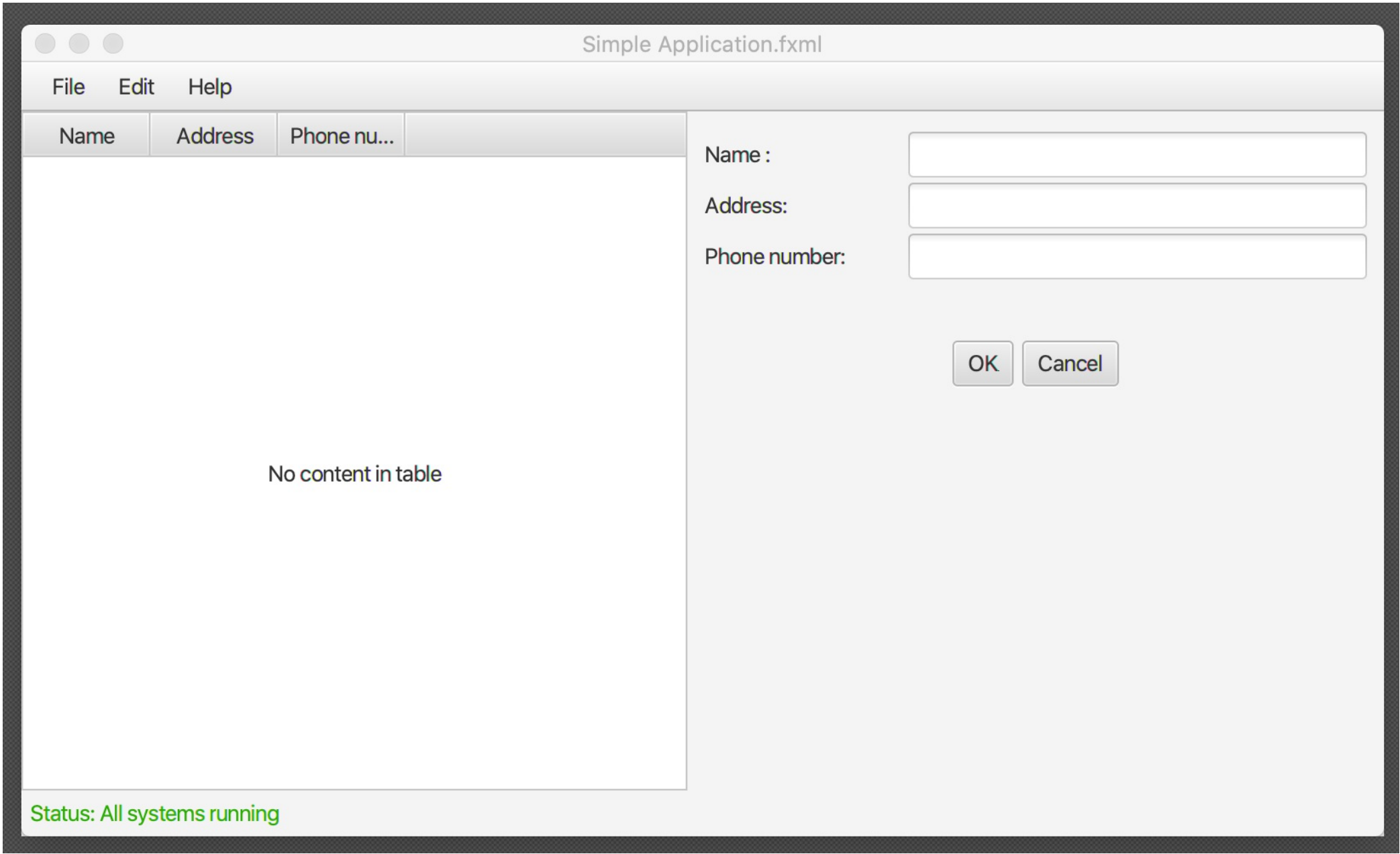
Sjå på koden over. Kva designpattern er i bruk her?

Vel eitt alternativ

- ☐ Singleton
- ☐ Observer
- ☐ Factory
- ☐ Decorator

Maks poeng: 2.5

8(a) **Oppgave 8 a)**



Sjå på forslaget til eit GUI over (NB! Dette er ikkje eit forslag til GUI for bomstasjons-applikasjonen). Beskriv i svarfeltet under korleis du ville ha designa eit slikt GUI ved å skissera eit **hierarki** av ulike **"panes"** (BorderPane, HBox, VBox osv.). Bruk innrykk for å indikera nivå, som til dømes:

- HBox
 - Button OK
 - Button Cancel
- osv.

Menyene inneheld følgjande menyval:

- File: Open, Close, Exit
- Edit: Copy, Paste, Delete
- Help: About

Skriv svaret ditt her...

Format | B | I | U | x₂ | x² | I_x | [Icon] | [Icon] | [Icon] | [Icon] | [Icon] | [Icon] | [Icon] | [Icon] | [Icon] | [Icon] | [Icon] | [Icon] | [Icon]

Words: 0

8(b) Oppgave 8 b)

Grafisk brukergrensesnitt og JavaFX

```
Button okButton;  
Lable statusBarLabel;  
  
// Fill in your code below  
|
```

Skriv kode slik at eit klikk på "OK"-knappen resulterer i følgjande tekst i statusbaren:

"Status: OK button clicked..."

Skriv svaret ditt direkte i svarfeltet under (treng ikkje å skriva koden i IDEEN din først)

Skriv svaret ditt her...

1

9

Filopplasting

Last til slutt opp ein ZIP-fil av **heile prosjekt-mappa di**. Pass på at du får med alle IDE-filer og mapper og den skjulte .git-mappen

[Slik lagar du ZIP-fil av ei mappe på Mac og Windows](#)



Last opp fila her. Maks ei fil.

Det er høve til å laste opp følgjande filtypar: **.zip** Maksimal filstorleik er **35 GB**.



Vel fil for opplasting

Maks poeng: 0