NTNU
Kunnskap for en bedre verden

Institutt for datateknikk og informasjonsvitenskap

Eksamensoppgave i TDT4180
Menneske-maskin-interaksjon

Faglig kontakt under eksamen: Hallvard Trætteberg

Tlf.: 91897263

Eksamensdato: 31. mai

Eksamenstid (fra-til): 9.00-13.00

Hjelpemiddelkode/Tillatte hjelpemidler: D (Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt enkel kalkulator tillatt.)

Annen informasjon:

Oppgaven er utarbeidet av faglærer Hallvard Trætteberg og kvalitetssikret av Dag Svanæs.

Målform/språk: Bokmål

Antall sider: 4

Antall sider vedlegg:

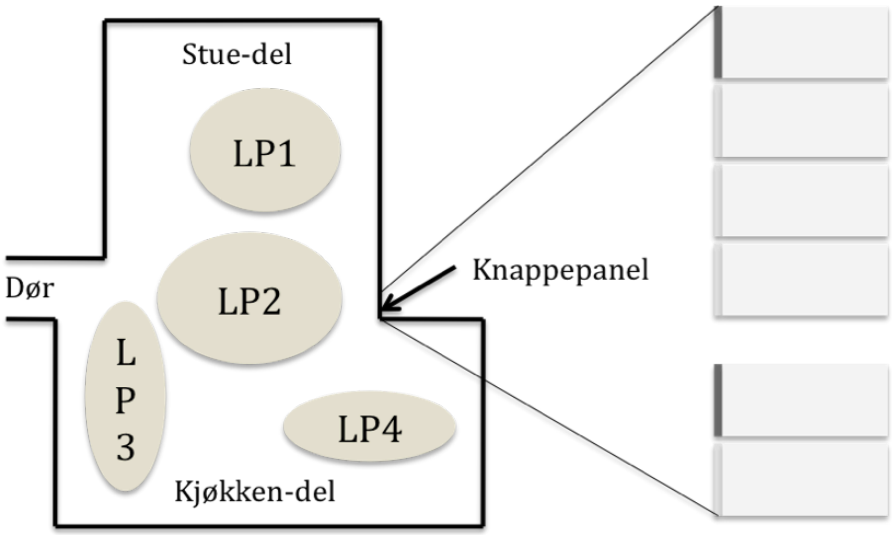
Kontrollert av:

_____

Dato Sign

Del 1 – Grensesnittdesign (30%)

Du ankommer en moderne hytte som du har leid, og ønsker å slå på lysene i det kombinerte stue-kjøkken-rommet. Du finner et panel med knapper, men det er ingen tekst eller annen merking som forteller noe om funksjonen til panelet. Rommet er utformet som vist i figuren under, med knappepanelet forstørret til høyre. De fire lyspunktene er tegnet som ovaler tilsvarende området de lyser opp.



Stue-kjøkken-rommet, med lyspunktene LP1-LP4.

De seks knappene er gruppert i to grupper, fire i den øverste og to i den nederste. Knappene lar seg trykke litt inn, men spretter helt ut når en slipper. Hver knapp har en smal stripe til venstre som lyser opp i blått (vist som mørkegrått i figuren) når du trykker på knappen. Denne stripa forblir lysende når du slipper knappen, inntil en annen knapp i samme gruppe trykkes. I illustrasjonen over til høyre er det altså de to øverste knappene i hver gruppe som er sist trykket på.

Etter litt prøving og feiling finner du følgende sammenheng mellom knappene og de fire lyspunktene i kjøkken-stue-rommet: Hver av de fire knappene i den øverste gruppa tilsvarer ett lyspunkt. Når du trykker lenger enn ett sekund på en av disse knappen, så slår tilsvarende lyspunkt seg på/av. De to nederste knappene dimmer opp og ned så lenge knappen holdes inne. Lyspunktet som dimmes er det som tilsvarer knappen i øverste gruppe som lyser blått. Det nest nederste lyspunktet (LP3) er imidlertid et lysrør som ikke kan dimmes, så i det tilfellet har de to nederste knappene ingen effekt.

a) Kritiser (både positivt og negativt) designet av knappepanelet, med referanse til generelle prinsipper for god design.

Det er mange prinsipper for god design, som er relevante her. En bør få med de fleste av disse, men viktigere enn antallet er at de beskrives og anvendes riktig.
Gestalt: Grupperingen i 4 + 2 knapper er god (+), men det er ingen forskjell på utseendet til knapper som gjør forskjellige ting (-). Dimmeknappene kunne f.eks. hatt annen fasong.
Affordance: Knappene inviterer til trykking (+). Noe av funksjonen krever at en holder en knapp inne og dette indikeres ikke på noen måte (-).
Feedback: Knappene gir taktil og visuell feedback (+). Feedbacken som gis av lyset, kan være misvisende, siden den ikke forteller om effekten av trykket (at lyset skifter tilstand), men at trykket ble registrert (sist trykte knapp i gruppa) (-).

Synlig tilstand: Tilstanden til systemet er om lysene er på/av og dimmenivået. Dette er ikke synlig i panelet, kun hvilken som ble trykt sist (som knapt kan kalles systemtilstand) (-).
Konsistens: Ett lyspunkt er ulikt de andre, siden det ikke kan dimmes, men behandles likt (-).
Mapping: Det er ikke opplagt hvilket lyspunkt som styres av hver knapp (-).
Kunnskap i hodet (recall), ikke i verden (recognition): En må huske mappingen og at noen knapper slår lys av/på, mens andre brukes til dimming (-).

b) Knappepanelet på veggen skal *erstattes* av en høyoppløselig og trykkfølsom fargeskjerm festet på samme sted, med størrelse og format som en smart-telefon, f.eks. iPhone. Skisser og forklar virkemåten til *to ulike* design for et slikt touch-panel, hvor hvert design begrunnes basert på kritikken (både positiv og negativ) i a). Vurder også de to designene opp mot hverandre.

Her er litt av poenget å ha design som er en *forbedring* og som har *vesentlige forskjeller*.
- Forbedringer:
God mapping, f.eks. ved å plassere knappene etter lyspunktenes plassering i rommet.
Synlig tilstand: Indikere tilstanden til lysene (av/på og dimmenivå).
Affordance: Bruke relevante kontroller, f.eks. knapper for av/på og slidere for dimmenivå.
Konsistens: Tydelig vise at lyspunkt 3 ikke kan dimmes, enten ved å utelate dimme-kontroller eller de-aktivere dem.
- Vesentlige forskjeller:
Tid/rom-dimensjon: Egne kontroller pr. lyspunkt eller selektor og ett sett lyskontroller.
Layout: Med eller uten kart/rom-basert layout.
Kontroller: Valg av dimmenivå med slider eller nedtrekksmeny, eksplisitt knapp for av/på vs. implisitt basert på dimmenivå, interaksjonsstil, f.eks. skjema vs. touch-basert interaksjon.
- Vurderingen – en bør gjenta hvordan prinsippene ivaretas, i tillegg til en vurdering av de faktiske forskjellene.

Del 2 – Designprosess (30%)

a) Hva er forskjellen (formål og innhold) på *formativ* og *summativ* evaluering, og når brukes hver av dem i designprosessen?

Formativ evaluering har som formål å (hjelpe til å) *forme/forbedre* designet og bruker derfor kvalitative teknikker som gir innsikt i hva som skjer underveis ved bruk, ikke bare hvor godt det (kvalitativt) gikk. Formativ evaluering gjøres nødvendigvis underveis i designprosessen. **Summativ** evaluering har som formål å *summere opp* hvor godt et design er blitt, f.eks. hvor mange oppgaver som ble gjennomført av og hvor fort. SUS-skjemaer er også relevant.

Du jobber i et IT-firma og har laget to forslag til design for styring av lys i hytta basert på smart-telefon-formatet (tilsvarende oppgave 1b). Nå lurer hyttefirmaet på hvorvidt disse vil fungere som en app som kan kjøre på gjestenes egne smart-telefoner. Tanken er at lysanlegget skal styres via det trådløse nettverket i hytta og at det skal være mulig å styre det med både det eksisterende panelet og app-en kjørende på én eller flere telefoner.

b) Beskriv hvordan du vil bruke brukersentrerte metoder for å studere brukssammenhengen (context of use) for en slik lysstyrings-app.

Dette dreier som om den innledende fasen i et prosjekt, hvor en ønsker å bli kjent med brukere, brukssammenhengen og omgivelsene for bruk. Typiske metoder er **feltstudier** (observasjon av reell bruk i ekte omgivelser), **undersøkelser/spørreskjema** (survey) og **intervju**. En bør si mer enn bare nevne metoden, f.eks. om hva som ligger i teknikken, hvordan en finner informanter osv.

c) Anta at en det er besluttet at en slik lysstyrings-app kan ha noe for seg. Du får i oppgave å gjennomføre én iterasjon av en brukersentrert prosess, for å videreutvikle begge design-forslagene, for så å velge hvilken som skal tas videre. Forklar (og begrunn) hvordan du vil gjennomføre iterasjonen og vurderingen av forslagene.

Her var det et poeng at en skulle skjønne at en allerede hadde kommet litt ut i prosessen og at én iterasjon omfattet (i denne rekkefølge) **prototyping** og **evaluering** av designet, **sammenligning** og **valg** av alternativ og **redesign** (av valgt alternativ) basert på resultatene. Vi reduserte kravet om at en kom inn i iterasjonen på riktig sted, men det er viktig å ha med alle trinnene i en iterasjon. For hvert trinn bør en bruke relevante metoder og begrunne og beskrive dem, ikke bare liste dem opp (og dynge på med så mange en husker). F.eks. er det ikke nok å si at en skal lage en prototype, en bør også si hva slags (papir/digital, horisontal/vertikal osv.). En bør spesielt kommentere hva en gjør for å få resultater som gjør det mulig sammenligne de to designalternativene.

Del 3 – Konstruksjon (40%)

De to design-forslagene fra 1b) vurderes som så likeverdige at det besluttes å lage en app (i Java og Swing) som *kombinerer* dem, ved at en kan bytte mellom dem når som helst uten å restarte app-en.

Anta at lysstyringsanlegget kan styres vha. (en klasse som implementerer) følgende Java-grensesnitt:

```
// controls a set of lights indicated by the String identifiers (lightId) “LP1”-“LP4”
public interface LightSystem {
	// returns the number of light levels/states the light with the given id supports
	public int getNumberOfLightLevels(String lightId);
	// returns the current level/state of the light with the given id (0 means off)
	public int getLightLevel(String lightId);
	// sets the current level/state of the light with the given id (0 means off)
	public void setLightLevel(String lightId, int level);
	// adds a listener that will be notified when one of the lights changes state
	public void addLightListener(LightListener lightListener);
	// removes a previously registered listener
	public void removeLightListener(LightListener lightListener);
}
```

a) Hva slags rolle vil en klasse som implementerer `LightSystem`-grensesnittet ha i en konstruksjon basert på MVC? Spesifier ferdig `LightListener`-grensesnittet og gjør rede for evt. antagelser om virkemåte for både `LightSystem` og `LightListener`.

LightSystem representerer **tilstanden** til systemet og dermed **modellen** i en MVC-basert konstruksjon. LightListener vil typisk inneholde én metode for å varsle om en endring i modellen, f.eks. **void lightChanged(String lightId)**. Mange valgte å la LightListener utvide et eksisterende Swing-grensesnitt, f.eks. ActionListener og ChangeListener (fra vedlegget), men det er unaturlig. **LightSystem** må varsle **LightListener**-ene når det skjer en endring i tilstanden, uansett om endringen skjer fra panelet eller appen.

b) Forklar med diagrammer, tekst og kode hvordan du vil konstruere (GUI-delen av) app-en (som kombinerer begge design-forslagene fra 1b) i Java Swing, basert på MVC-arkitekturen, og som bruker `LightSystem` og `LightListener`. Se vedlegget for en oversikt over en del relevante klasser/grensesnitt. Dersom du trenger andre klasser, men ikke husker nødvendige detaljene, så forklar hvilke antagelser du gjør.

Her bør en ha med et ordentlig klassediagram, med riktig bruk av arv og assosiasjoner, hvor også implementasjon av grensesnitt er med, både LightListener (fra modell) og Swing-grensesnitt (fra GUI). Diagrammet bør dekke begge delene/alternativene. Kode kan kun delvis erstatte klassediagrammet. Et objektdiagram vil kunne gi samme informasjon, men er ikke påkrevd.

c) Forklar med tekst og sekvensdiagram hva som skjer når 1) app-en startes opp og 2) når en slår på et lyspunkt og dimmer det ned.

Her er det viktig at det er tydelig hvilke objekter som har rollen som modell, view og kontroller og å få med de viktige stegene:
1) view/kontroller registrerer lytter på modell, kontroller registrerer lytter på view og view oppdateres med initialtilstand (dette "glemte" de fleste).
2) kontroller (lytter) får beskjed om endring, endring utføres på modell, view/kontroller for *begge* delene får beskjed om (modell)endring og oppdaterer viewet med relevant kall.

d) App-en er ment å kunne styre de samme lyspunktene fra flere smart-telefoner samtidig. Forklar hvordan dette håndteres av din konstruksjon (du kan se bort fra evt. tidsforsinkelser i nettet osv.).

Dette håndteres automatisk av MVC-arkitekturen, dersom modellen legges på en server/i panelet og alle apper kobler seg på denne. Det eneste problemet er helt samtidig bruk, hvor siste endring vinner, men med riktig distribusjon og håndtering av hendelser, så skal systemet forbli konsistent.

Appendix

JLabel

JLabel() - Creates a JLabel instance with no image and with an empty string for the title.
setText(String text) - Defines the single line of text this component will display.
setIcon(Icon icon) - Defines the icon this component will display

JButton

setText(String text) - Defines the single line of text this component will display.
setIcon(Icon icon) - Defines the icon this component will display
setEnabled(boolean b) - Enables (or disables) the button.
addActionListener(ActionListener l) - Adds an ActionListener to the button.

ActionListener

void **actionPerformed**(ActionEvent e) - Invoked when an action occurs.

JSlider

JSlider(int min, int max, int value) - Creates a horizontal slider using the specified min, max and value.
int **getValue**() - Returns the slider's current value.
void **setValue**(int n) - Sets the slider's current value to n.
void **addChangeListener**(ChangeListener l) - Adds a ChangeListener to the slider.

ChangeListener

void **stateChanged**(ChangeEvent e) - Invoked when the target of the listener has changed its state.

JSpinner

JSpinner(SpinnerModel model) - Constructs a spinner for the given model.
getValue() - Returns the current value of the model, typically this value is displayed by the editor.
setValue(Object value) - Changes current value of the model, typically this value is displayed by the editor.
void **addChangeListener**(ChangeListener l) - Adds a ChangeListener to the spinner.

SpinnerNumberModel (implements SpinnerModel)

SpinnerNumberModel(int value, int minimum, int maximum, int stepSize)
Constructs a SpinnerNumberModel with the specified value, minimum/maximum bounds, and stepSize.

JComboBox

JComboBox(Object[] items) - Creates a JComboBox that contains the elements in the specified array.
int **getSelectedIndex**() - Returns the index of the current selected item.
Object **getSelectedItem**() - Returns the current selected item.
addActionListener(ActionListener l) - Adds an ActionListener.