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: 30. 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 faglærere Yngve Dahl og Hallvard Trætteberg og kvalitetssikret av Dag Svanæs.

Målform/språk: Bokmål

Antall sider: 4

Antall sider vedlegg: 0

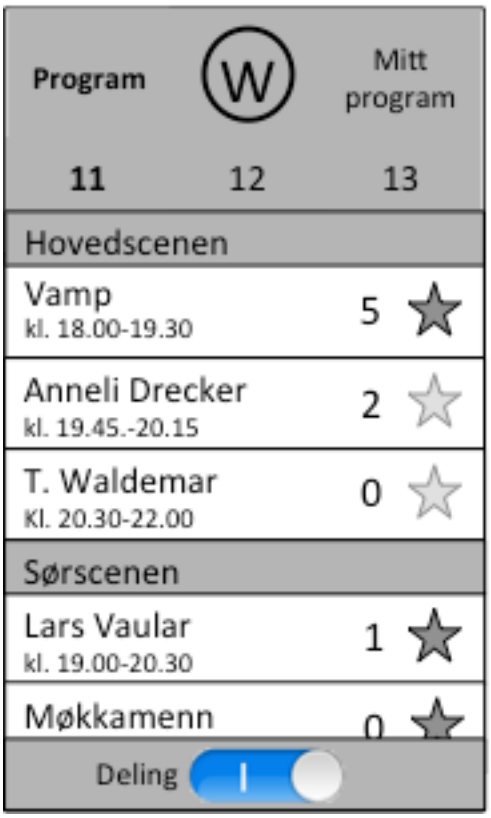
Kontrollert av:

_____

Dato Sign

Tema for eksamen er en mobil applikasjon (app) for en årlig norsk pop- og rockefestival kalt WATT. Årets festival har en varighet på tre dager og vil arrangeres i tidsrommet 11.-13. juli. Det blir arrangert 100 konserter i løpet av festivalen. Konsertene holdes parallelt på fire ulike scener som er satt opp på festivalområdet. Det forventes at ca. 30 000 personer vil delta på festivalen hvorav de fleste er studenter.

Del 1 – Grensesnittdesign (45%)



Figur 1

Figur 2

Figur 1 og Figur 2 viser to skisserte skjermbilder for en app kalt WATT'sApp, som er ment å hjelpe brukere med å (1) planlegge hvilke konserter de vil delta på, (2) varsle brukere før oppstart av en utvalgt konsert (påminnerfunksjonalitet) og (3) koordinere sine planer for deltakelse med andre brukeres planer.

WATT'sApp virker som følger:

- Lista viser en oversikt over konserter med navn på artist/band, tidsrom og hvilken scene de arrangeres på. Lista kan scrolles ved å dra den opp eller ned.
- Brukeren kan merke av hvilke konserter han/hun er interessert i ved å trykke på stjerne-feltet ved siden av en av konsertene. Stjerne-feltet vil da endres fra lys grå (ikke interessert) til mørk grå (interessert).
- Lista viser kun konserter for én dato, og datoen velges ved å trykke på ett av tallene *11* (11. juli), *12* (12. juli) eller *13* (13. juli) over lista. Valgt dato er indikert med fet type (bold). I figurene 1 og 2 er 11. juli valgt.
- Lista kan enten vise alle konsertene på programmet eller kun de som brukeren har stjerne-markert (markert som interessante). En velger modus ved å trykke enten *Program* eller *Mitt*

program. Valgt modus er indikert med fet type (bold). Figur 1 og 2 viser skjermbilde med hhv. hele programmet og stjerne-markerte konserter.

- Hvis brukeren trykker på en mørk grå stjerne (slik at den blir lys grå, for *ikke* interessert) og *Mitt program* er aktivert, så vil lista oppdateres og konserten vil bli fjernet fra lista.
- En bruker vil varsles av en melding med lyd 15 minutter før en konsert som han/hun har merket av som interessant.
- ”W”-symbolet plassert øverst og i senter av skjermbildet er festivalens logo. Ved å trykke på symbolet vil WATT sin hjemmeside åpnes i mobilens nettleser.
- Under lista ligger en tilstandsbryter (toggle button) som styrer hvorvidt *deling* av konsertplan er på. Deling innebærer at andre brukere markert som dine ”venner” får tilgang til din plan og omvendt. Styring av vennelista håndteres av et annet skjermbilde som ikke er vist her.
- Når deling er på (som vist figur 1 og 2), så vil det for hver konsert vises *antall venner* som skal på den konserten. Hvis en trykker på tallet så kommer det opp et panel med oversikt over hvem det er.

a) Forklar begrepene *affordance* og *feedback* slik disse forstås i Don Normans designprinsipper, og beskriv kort hvordan de to begrepene henger sammen. Diskuter brukskvaliteten til det foreslåtte designet med utgangspunkt i de to begrepene.

“Affordance”, slik det forstås i Don Norman’s design prinsipper, betegner hva et produkt signaliserer om sin bruk gjennom sin utforming. For eksempel signaliserer utformingen til en brus-flaske at den kan gripes og drikkes av. Vi skiller gjerne mellom tre typer affordance: *Fysisk* (gitt av kroppens anatomi i kombinasjon med et objekts utforming), *kulturell* (konvensjoner som læres) og *kontekstuell* (gitt av konteksten/omgivelsene).

I design av brukergrensesnitt handler affordance om grensesnittets evne til å formidle hva en bruker kan gjøre, og hvilke konsekvenser en bruker kan forvente.

Normans prinsipp om ”feedback” handler om å sende informasjon til brukeren om hvilken aksjon som har blitt foretatt, og hva som har blitt oppnådd. Feedback er med på å bekrefte overfor brukeren at systemet har respondert på brukerens interaksjoner. Feedback kan kommuniseres gjerne visuelt, auditivt, taktilt eller gjennom em kombinasjon av disse.

Affordance og feedback spiller en sentral rolle i å formidle virkemåten til et produkt. Mens affordance signaliserer hva et produkt kan brukes til og hva som *blir* konsekvensen, gir feedback informasjon om hvilken aksjon som er foretatt og hva som *er* konsekvensen). På denne måten er feedback med på å komplementere affordance.

Med utgangspunkt i affordance-begrepet vil følgende aspekter kunne påvirke brukskvaliteten til designet:

- Elementene som brukes for å velge modus og dato signaliseres ikke at de er interaktive elementer (at de kan trykkes). Det samme gjelder for festivalsymbolet.
- Hvorvidt bruk av stjerne-symbol er egnet for å markere hvilke konserter man er interessert i (favoritter) er i høy grad kulturelt betinget (kulturell affordance)
- Det kan stilles spørsmål ved hvorvidt oversikten over konserter signaliserer at dette er en liste som kan scrolles (scrollbar mangler). I figur 1 antydes det at oversikten er en liste gjennom at man bare kan se deler av elementene i den nederste konserten som vises i lista.
- Det er ingenting som signaliserer at elementene som angir antall venner som planlegger å se samme konsert er et interaktivt element som kan klikkes på.

- Bryteren for å dele informasjon med ”vennelista” antyder at elementet brukes for å sette ulike tilstander på samme måte som en lysbryter (hva ”deling” faktisk innebærer er imidlertid uklart).

Med utgangspunkt i feedback-begrepet vil følgende aspekter kunne påvirke brukskvaliteten til designet:
- Det å bruke fet skrift for å indikert hvilket modus som er satt og hvilken dato som er valgt vil muligens være en form for feedback som ikke er tydelig nok for brukere. Dette bør testes.
- Appen gir umiddelbar feedback når det trykkes på et stjerne symbol for antyde hvilke konserter som er av interesse. Fargene stjerne-symbolene har når de er valgt skiller muligens ikke nok, og kan føre til misforståelser.
- Grensesnittet i utgangspunktet ingen indikasjon på at det å melde interesse for en konsert fører til at konserten legges til i ”Mitt program”. Dette kan for eksempel tydeliggjøres gjennom en enkel animasjon.
- Brukeren for ingen tilbakemelding om at det å markere en konsert som interessant medfører at han eller hun får et varsel 15 minutter før konsertstart. En slik tilbakemelding vil kunne være hensiktsmessig, særlig ved førstegangs bruk.

b) Diskuter brukskvaliteten til det foreslåtte designet med utgangspunkt i *gestalt*-prinsippene om nærhet og likhet i form/farge.

- Fargen og plasseringen på elementene som brukes for å velge modus og dato angir ikke tydelig hvilke av dem som logisk hører sammen. Bruk av for eksempel ulik farge og/eller mer luft mellom elementer for valg av modus og elementer for valg av dato kan bidra til å tydeliggjøre dette.
- Mindre luft mellom dato-elementene kan også tydeliggjøre at elementene logisk hører sammen
- Plasseringen av festival-symbolet i forhold til elementene for valg av modus bør vurderes siden funksjonen til førstnevnte ikke logisk er knyttet til sistnevnte.
- Mer ”luft” mellom ”venne-telleren” og stjerne-symbolet i lista kan bidra til å indikere at de to elementene ikke logisk hører sammen.
- Scener og konserter i lista skilles relativt tydelig ved hjelp av ulik fargebruk. Dette er positivt.
- Det at ”scener” i lista har samme fargebakgrunn som modus- og dato-elementene kan være forvirrende for en bruker (”Scener” gir ikke respons når de trykkes på).

c) Basert på diskusjonen i oppgave 1a og 1b, foreslå et alternativt brukergrensesnitt som du mener vil være bedre. Skisser løsningen og begrunn forslaget.

Besvarelsen bør dekke alle aspekter som er diskutert i oppgave 1a og 1b, og som kan ha negativ innvirkning på brukskvaliteten til appen. Besvarelsen bør illustrere tydelig hvilke grep som er gjort i forhold til det opprinnelige designet.

d) Beskriv kort rollen metaforer har i design av brukergrensesnitt. Relater beskrivelsen til begrepene *konseptuell modell* og *mental modell*.

En metafor i design av brukergrensesnitt er et sentralt virkemiddel for å formidle den tiltenkte virkemåten (den konseptuelle modellen) til et produkt. Metaforer har generelt til hensikt å skape en assosiasjon mellom brukergrensesnittet og produkter, gjenstander eller konsepter en bruker har kjennskap til fra før. Gjennom bruk av hensiktsmessige metaforer vil brukere kunne benytte kunnskap de har om virkemåten til noe (f.eks. et produkt) de kjenner fra før til å danne seg en

forståelse (mental modell) av systemets virkemåte. Metaforen er m.a.o. med på å skape en forventning til brukeren om hvordan han/hun kan interagere med brukergrensesnittet for å gjennomføre en oppgave.

En hensiktsmessig metafor vil bidra til å kommunisere den konseptuelle modellen til et system slik at brukeren raskt vil kunne danne seg en egnet mental modell av systemet.

e) Det skisserte brukergrensesnittet i Figur 1 og Figur 2 benytter en liste-basert *metafor* hvor brukeren kan velge hvilke konserter han er interessert i fra en sortert festivalprogram-oversikt. Denne metaforen er gjerne uegnet for komplekse data, hvor en kan ønske å se de samme dataene fra ulike perspektiver f.eks. tid og sted langs hver sin akse.

Beskriv en metafor du mener ville være mer egnet for disse formålene. Lag en skisse og en kort forklaring av et skjermbilde som illustrerer designet.

Et eksempel på en metafor som kan bidra til å løse problemet med manglende oversikt vil være en *tidslinje* som viser tidsrommene for de ulike konsertene (bortover) og hvilken scene som brukes (nedover). En slik fremstilling vil for eksempel kunne gi bedre oversikt over konserter en har merket av som interessante og som vil avholdes innenfor samme tidsrom.

Del 2 – Designprosess (30%)

WATT'sApp fra Del 1 ble utarbeidet uten involvering av brukere i designprosessen. Tilbakemeldinger fra festival-deltakerne tydet på at manglende brukervennlighet og få muligheter for å koordinere egne festival-planer med andres festival-planer var en av hovedårsakene til at mange lot være å bruke appen. Festival-arrangøren har et ønske om at det utvikles en ny designløsning for neste års festival.

Du er ansatt som konsulent i et IT-firma som har fått oppdraget å utarbeide den nye designløsningen. Det er vedtatt at den nye løsningen skal utarbeides gjennom en brukersentrert prosess, og du er satt til å lede prosessen.

a) Beskriv hvordan du ville bruke brukersentrerte metoder for å studere brukssammenhengen (context of use) for ny versjon av WATT'sApp. Beskrivelsen bør inneholde kort hva de ulike metodene/teknikkene du velger består i, hvordan du ville bruk dem i praksis, og hvorfor du anser dem som relevant.

Følgende metoder og teknikker vil være hensiktsmessig for å få innsikt i brukskonteksten for løsningen som skal designes:

- **Feltobservasjoner**: Å følge en gruppe festival-deltakere en dag kan gi verdifull informasjon om f.eks. om behov for festivalplanlegging og koordinering med venner, hvordan dette håndteres, og opplevde utfordringer. Metoden kan også gi innsikter med hensyn til utfordringer den fysiske brukskonteksten kan by på med hensyn til en designløsning (f.eks. støy, regn, trengsel). Avhengig av tidsrammene kan det imidlertid være en utfordring å finne en relevant festival for å kunne gjennomføre feltobservasjoner.
- **Intervjuer**: Intervjuer med deltakere (individuelt eller i grupper) på en festival kan bidra til å gi dybde innsikt i forhold til punktene over. Intervjuer kan også gjøres med personer som har tidligere erfaring med festival-deltakelse. Imidlertid kan det være en fordel å samle inn data fra personer som befinner seg i den reelle brukskonteksten, siden det da vil være lettere for potensielle brukere og reflektere over egne behov.

- **Spørreskjema**: Spørreskjema kan bidra til å bygge en overordnet forståelse av behov og ønsker festivaldeltakere har i forhold til en festival-planlegger, men er bidrar generelt ikke til en kvalitativ forståelse av brukerbehov.
- **Personas**: Basert på innsamlede data fra feltobservasjoner og intervjuer kan det være hensiktsmessig å lage et sett med personas som reflekterer ulike segmenter av festival-deltakerne. Personas vil kunne være nyttig for å diskutere og formidle aspekter ved brukssammenheng med andre designere og utviklere og evt. med kunden (i dette tilfellet festival-arrangøren).

b) Diskuter metodiske styrker og svakheter knyttet til brukbarhetsevalueringer gjennomført i laboratorium kontra brukbarhetsevalueringer gjennomført i felt (reelle brukssituasjoner). Gi eksempler på fordeler og ulemper du ser i forhold til å bruke de to evalueringsformene i tilknytning til WATT'sApp.

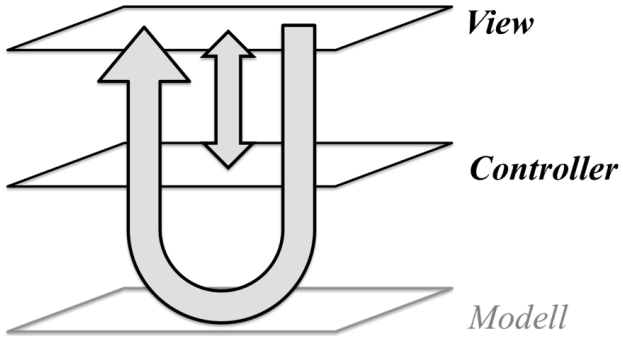
Metodiske styrker tilknyttet brukbarhetsevalueringer gjennomført i laboratorium sammenlignet med felt ligger først og fremt kontrollen laboratorie-evalueringer gir over test-situasjonen, mens svakheten ligger hovedsakelig i manglende realisme. Motsatt: Evaluering i felt gir en mer realistisk brukssituasjon, men en mister en deler av kontrollen en har i et laboratorium. De to evalueringsformene vil ha ulike styrker og svakheter i tilknytning til WATT'sApp:

Lab.:
+ Raskt tilgang til relevante test situasjoner, samme scenario kan testes flere ganger,
+ Man kan nyttiggjøre et rikt utvalg av datainnsamlingsverktøy (videokamera, speiling av grensesnitt, lydopptak, mm.)
+ Kan være hensiktsmessig med tanke på å teste aspekter som ikke påvirkes mye av fysisk og sosial kontekst (f.eks om begrepsbruken i grensesnittet gir mening for brukere).
÷ Å gjenskape en realistisk brukskontekst lar seg ikke gjøre. I praksis betyr det at det er en rekke aspekter som kan påvirke brukskvaliteten som en ikke vil fange opp. Dette vil kunne ha konsekvenser for validiteten til dataene som samles inn.

Felt:
+ Realistisk testsituasjon. For eksempel vil flere mobile aspekter ved brukskonteksten gjenskapes – høy validitet.
÷ Kan være tidkrevende. Det kan gå lang tid mellom relevante brukssituasjoner.
÷ Kan vanskeliggjøre bruk av opptaksutstyr for datainnsamling.
÷ Økt risiko for at uforutsette hendelser kan forhindre at testen kan gjennomføres.

Del 3 – Teknikk (25%)

Ved realisering av brukergrensesnitt med MVC-arkitekturen så deler en gjerne systemet i tre lag, kalt modell, controller og view. Forholdet mellom disse er illustrert i Figur 3.



Figur 3

a) Beskriv kort hva som er oppgaven til de tre lagene og forklar hvordan data flyter mellom dem, med referanse til Figur 3.

Modell-laget holder **applikasjonsdataene** (ofte de som lagres mellom sesjoner) og er i utgangspunktet uavhengig av brukergrensesnittet. For å støtte MVC-arkitekturen så må den være **observerbar**, dvs. en må 1) kunne lese tilstand og 2) kunne registrere lyttere som får beskjed når modellen endres.
View-laget er den **synlige delen av brukergrensesnittet**, både rent **visuelle og interaktive** elementer. De interaktive elementene har en lyttermekanisme tilsvarende modellen, for å rapportere om brukerens handlinger.
Controller-laget er limet mellom modell og view og sørger for at de holdes oppdatert, altså at **viewet oppdateres når modellen endres** og omvendt at **brukerens aksjoner forplanter seg til modellen**. Controller-laget håndterer også (de)aktivering av elementer inkl. **navigering**, **validering** av verdier i innfyllingsfelt, **avhengigheter** innen viewet osv.
Figuren illustrerer ulik type dynamikk (det er ikke viktig å navngi dem):
- Den **venstre delen av U-en** består i å oppdatere viewet med data fra modellen, både initielt og når modellen senere endres
- Den **høyre delen** forplanter endringer i viewet til modellen og annen interaksjon som trigger endringer i modellen. Disse vil igjen trigge oppdatering av viewet.
- **I-en i midten** er view-intern dynamikk som ikke involverer modellen, f.eks. navigering, validering, seleksjon, osv.

b) Beskriv hvordan man *generelt* realiserer *View*-laget med JavaFX og FXML og hvordan dette knyttes til *Controller*-laget.

JavaFX bruker en såkalt *scene graph* for å representere viewet, dvs. en **hierarkisk struktur** av alle visuelle og interaktive elementer. Objektene i strukturen har ulike **egenskaper (properties** – observerbare felt med metoder iht. JavaBeans-konvensjonen) som styrer både funksjonell og visuell oppførsel. Å rigge opp viewet består essensielt av å bygge opp hierarkiet og sette egenskapene.
FXML er et XML-basert format for å beskrive dette hierarkiet og disse egenskapene. XML-elementene (tags) har typisk **navn etter klasser** og et element tilsvarer en instans av denne klassen. Attributtene har **navn etter egenskapene** og én attributt tilsvarer én verdien til én egenskap. Egenskaper av komplekse typer representeres som XML-elementer hvor verdien er strukturen av elementene inni. FXML har også støtte for å oppgi **kontroller-klassen**. Selve viewet og evt. kontrolleren bygges av (en instans av) **FXMLLoader**-klassen, som **"oversetter" elementer og attributter til instanser og egenskapsverdier** ved å utnytte JavaBean-konvensjonene.
FXML(Loader-en) har to mekanismer for å knytte viewet og kontrolleren sammen:
1. Et element kan inneholde et **fx:id-attributt** (fx er et spesielt FXML-navnerom) og et felt med samme navn i kontrolleren (som er public eller annotert med @FXML) vil automatisk blir satt til instansen som elementet er "oversatt" til.
2. En kan **navngi metoder i kontrolleren** (må ha riktig signatur) som skal kalles når **hendelser eller egenskapsendringer** inntreffer.

I tillegg finnes det en inkluderingsmekanisme så en dele opp FXML-koden i flere filer, med hver sin kontroller.

b, for de som har hatt Swing) Beskriv hvordan man *generelt* i Swing kobler modell og view til controller-logikken (logikken som håndterer dynamikken).

De er to teknikker som er i bruk:
1. Bruk av standard lyttergrensesnitt som **ChangeListener**, **ItemListener**, **ActionListener** osv. for viewet og **PropertyChangeListener** eller egendefinerte lyttergrensesenitt for modellen.
2. Bruk av modell-adaptere mellom modellen og view-modeller som **SpinnerModel**, **ListModel**, **TreeModel** osv.

I begge tilfeller rigges det opp ved aktivering av viewet.

c) Lag et klassediagram over dataene som ligger i *Modell*-laget for WATT'sApp-versjonen beskrevet i Del 1. Skisser strukturen av *View*- og *Controller*-elementer som finnes ved kjøring og forklar hva som skjer (med referanse til diagram og skisse) når:

- brukeren velger visning av *Mitt Program* (når *Program* var valgt fra før)
- brukeren trykker på en *mørk grå* stjerne når *Mitt Program* er valgt (deling er *av*)
- brukeren slår *på* deling
- deling er *på* og en venn trykker på en *lys grå* stjerne

Klassediagrammet bør inneholde klasser for **arrangement/konsert**, **band/artist**, **scene**, tidsrom, **person** med nødvendige **assosiasjoner** (arrangement->band, arrangement<->scene, person<->arrangement (interesse), person<->person (venner). Slik modellering er ikke fokuset i dette faget, så vi er ikke så nøye på detaljene. Men det som er viktig er at det i tillegg må det være klasser/grensesnitt som støtter ***observerbarhet***, ved at en definerer lytterrensesnitt, bruker PropertyChangeListener og PropertyChangeSupport eller JavaFX sine Property-klasser.

Strukturen av view- og kontroller-elementer kan skisseres med et ***objektdiagram*** og bør inneholde elementer tilsvarende (deler av) figur 1 eller 2. Vi er ikke så nøye på notasjonen, så lenge det er rimelig godt korrespondanse med figuren(e). Controller-delen bør inneholde tilstand for valg av program og dato, siden det ikke er naturlig at det ligger i modellen eller viewet.

Forklaringen kan være tekst og/eller sekvensdiagram, og det viktigste er at en får med seg fasene beskrevet i deloppgave a) og refererer til klasse- og objektdiagrammene (skissen):
- brukeren velger visning av Mitt Program (når Program var valgt fra før): Dette er en view-intern endring, hvor **kontrolleren** endrer hvilke data som **lista** viser. En må også oppdatere hvilken av de to **program-indikatorene i viewet** som er uthevet.
- brukeren trykker på en mørk grå stjerne når Mitt Program er valgt (deling er av): Dette endrer statusen og dermed **modellen** via kontrolleren. Dette trigger **oppdatering av viewet**, og siden den tilsvarende konserten ikke lenger er interessant, så fjernes den fra lista.
- brukeren slår på deling: Det er ikke viktig hvordan data utveksles mellom enhetene, men at alle elementene i **lista må oppdateres**, slik at **tallene vises**.
- deling er på og en venn trykker på en lys grå stjerne: Når nye data tas imot så **endres modellen**, og dette må trigge **oppdatering av viewet**, dvs. tallet som vises på en linje i lista.