



NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET
INSTITUTT FOR DATATEKNIKK OG INFORMASJONSVITENSKAP

Faglig kontakt under eksamen:
Hallvard Trætteberg,
Tlf: 7359 3443

EKSAMEN I FAG SIF8040 - MMI OG GRAFIKK

Onsdag 14. mai 2003
Tid: kl. 0900-1400

Bokmål

Sensuren faller 11. juni

Hjelpemiddelkode: **D** Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt enkel kalkulator tillatt.

Oppgave 1 - Brukersentrert systemutvikling (25%)

- a. Man skiller ofte mellom horisontale og vertikale prototyper.
- Hva er forskjellen på en horisontal og en vertikal prototyp?
 - I hvilke sammenhenger er det hensiktsmessig å lage en horisontal h.h.v. en vertikal prototyp?

Forslag til løsning 1a:

- *En horisontal prototyp viser "bredden" av et system, med spesiell fokus på det brukeren vil se. En vertikal prototyp går i "dybden" på ett eller flere steder i forhold til funksjonalitet, uten nødvendigvis å vise helheten. Hvis vi bruker en nettbankløsning som eksempel, vil en horisontal prototyp vise alle skjermbildene uten spesielt mye funksjonalitet. En vertikal prototyp vil her gå i dybden på ett eller flere steder, for eksempel i forhold til innlogging og autorisasjon. I praksis vil prototyper ofte være kombinasjoner av vertikale og horisontale, med litt bredde og litt dybde.*
- *Horisontale prototyper brukes for å kunne få tilbakemelding på helheten av en løsning, mens vertikale prototyper brukes for å få tilbakemeldinger på spesifikk funksjonalitet. Ofte vil man begynne med en mer eller mindre horisontal prototyp, for så å gå i dybden på visse deler etter hvert som prosjektet skrider fram.*

HORISONTAL PROTOTYP

V E R T I K A L P	
---	--

Snu arket!

- b. Graden av kontroll regnes som en viktig parameter i karakteristikken av empiriske metoder.
- Sorter de tre empiriske metodene ”brukbarhetstest”, ”feltstudie” og ”felt-test” i forhold til graden av kontroll. Begrunn svaret.

Forslag til løsning 1b:

Med kontroll menes her i hvor stor grad vi som utviklere påvirker det vi observerer. Grad av kontroll er en problemstilling som er viktig i samfunnsvitenskapene, og følgelig også når man i systemutvikling benytter metoder hentet fra samfunnsvitenskapene.

1. Av de tre metodene i oppgaven er det feltstudie som har minst grad av kontroll. Metoden går ut på å observere brukere i deres vanlige miljø med så lite påvirkning som mulig. Idealet er å være ”flue på veggen”. Hvis vi for eksempel skal lage et læreprogram for grunnskolen, så ville et feltstudie innebære å gjøre observasjoner av elever på en skole i deres normale skolehverdag. Som observatør vil man alltid i større eller mindre grad påvirke situasjonen, men på et slikt feltstudie vil man søke i så stor grad som mulig å minimalisere den påvirkning man har på de som observeres. Dette krever ofte tid og forståelse av egen rolle.
2. En felttest skiller seg fra et feltstudie ved at man her bringer noe nytt inn i situasjonen, for eksempel et dataprogram. Hvis vi igjen bruker eksempelet med et læreprogram, så ville en felttest innebære å teste ut en prototyp eller et produkt i den faktiske skolehverdagen. Vi kontrollerer her ett element av situasjonen, nemlig det programmet vi har brakt inn, men velger å ikke kontrollere hvordan det skal brukes. Vi har her mer kontroll enn for et feltstudie.
3. I en brukbarhetstest kontrollerer vi både hvilken teknologi som skal brukes, og hvordan brukeren skal anvende den. Vi bringer brukere ut av deres vanlige sammenheng og gir dem oppkonstruerte oppgaver i en kontrollert omgivelse. Målet er her å kontrollere situasjonen slik at vi kan få repeterbare resultater. Hvis vi igjen bruker eksempelet med et læreprogram, kunne vi ta elevene ut av klassesituasjonen og teste deres bruk av programmet i et brukbarhetslaboratorium

Grad av kontroll:

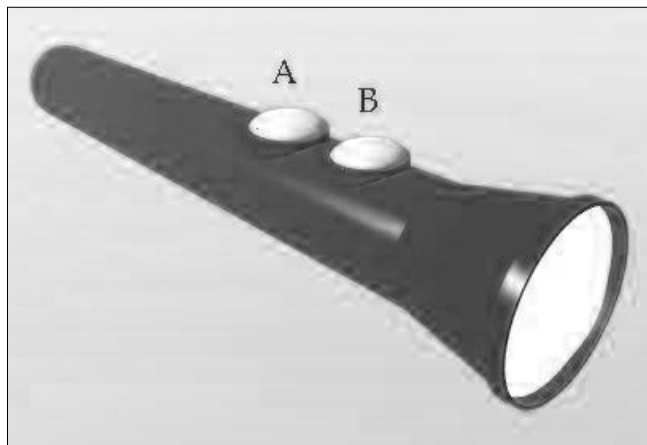
Lite kontroll

Mye kontroll



Oppgave 2 – Design av brukergrensesnitt (40%)

- a. Tilstandsdiagrammer brukes ofte til å spesifisere oppførselen til brukergrensesnitt.



Figuren over viser en modell av en ny lommelykt med to trykknapper A og B. Det er bestemt at lykten skal kunne ha de tre lysstyrkene ”svak”, ”medium” og ”sterk” i tillegg til at den skal kunne være avslått.

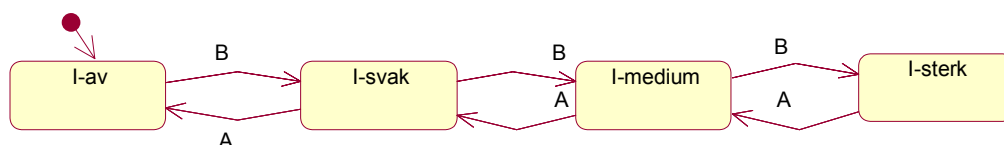
Det er kommet tre forslag til hvordan de to knappene A og B skal kunne brukes til å operere lommelykten:

- I. Et trykk på knapp B øker lysstyrken med ett nivå, mens et trykk på knapp A minsker lysstyrken med ett nivå. ”Av” regnes da som den minste lysstyrken, etterfulgt av ”svak”, ”medium” og ”sterk”.
- II. Knapp A er en av/på knapp for lykten. Knapp B øker lysstyrken med ett nivå fra ”svak” til ”medium” og ”sterk”. Et trykk på B i ”sterk” tar lykten tilbake til ”svak”. Når den slås av og på havner den automatisk i lysstyrke ”svak”.
- III. Som forslag II, men lykten ”husker” sin lysstyrke når den slås av og på.

Lag et detaljert tilstandsdiagram for hver av de tre løsningene. Du kan velge om du vil bruke UML *Statechart* formalismen eller en vanlig notasjon med sirkler og piler.

Forslag til løsning 2a: Tilstandsdiagrammer for I, II og III.

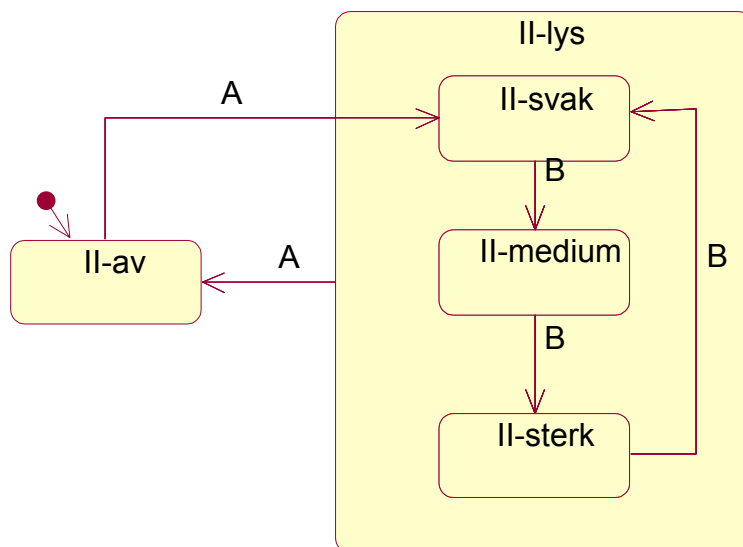
- I. Her blir tilstandsmaskinen en kjede av tilstander, én tilstand pr. lysnivået. Tilstandene er bundet sammen med transisjoner i begge retninger, hvor en transisjon er merket med trykknappen som trigger den. B-transisjonene fører til en tilstand tilsvarende mer lys, A-transisjonene til en tilstand med mindre lys.



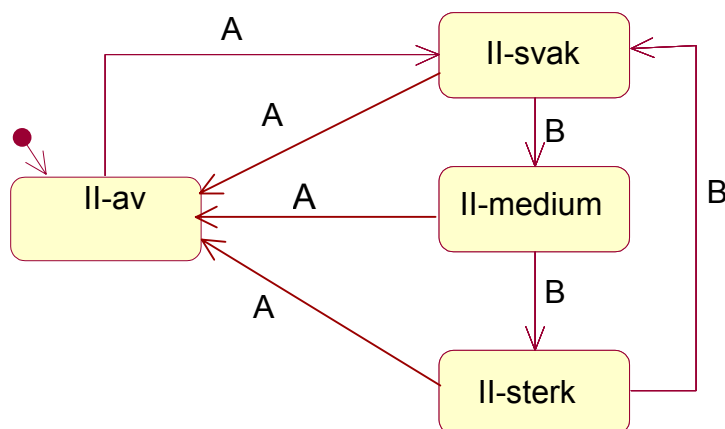
Tilstandsdiagram for nr I.

- II. Her blir tilstandene som tilsvarer lys (dvs. alle utenom av-tilstanden) en ring, med en B-transisjon for hver av dem. I tillegg har disse tre en A-transisjon som leder til av-tilstanden, og av-tilstanden har en A-transisjon som leder til tilstanden tilsvarende ”svak” lysstyrke. Dersom en bruker Statechart er det naturlig å samle de tre tilstandene tilsvarende lys i én overordnet lys-tilstand og de tre A-transisjonene til av-tilstanden erstattes med én fra den overordnede tilstanden (denne blir dermed felles for alle undertilstandene), som vist i figuren. (Alternativt

kan en tenke seg at en flater ut diagrammet ved å fjerne den overordnede tilstanden og legger på en A-transisjon fra hver undertilstand.)

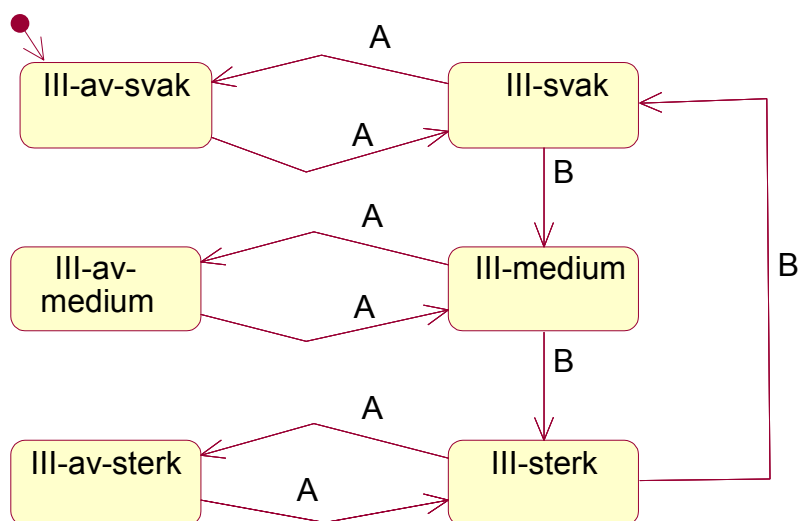


Nr. II med bruk av en overordnet tilstand.



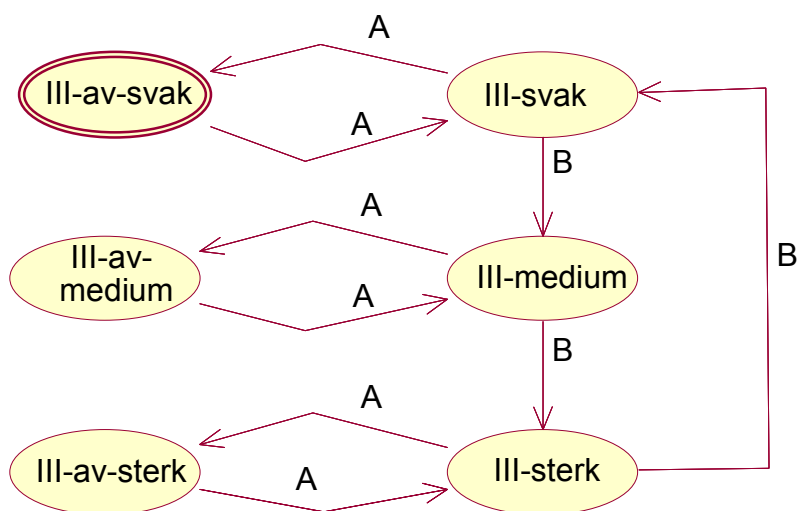
Nr. II, flatt nettverk.

- III. Her har tilstandsmaskinen er ring med tilstander og B-transisjon som i II. Forskjellen er at en må ha en av-tilstand for hver lys-tilstand, for å kunne gå tilbake igjen. Vi får altså tre av-tilstander med en A-transisjon i hver retning mellom tilsvarende lys-tilstand. Merk at her vil en ikke tjene noe på å lage en overordnet lys-tilstand, siden ikke noe er felles for tilstandene inni.



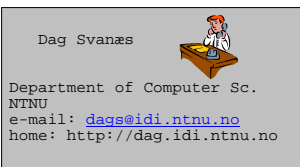
Nr II (UML notasjon).

Om UML-notasjonen: Starttilstand vises med en pil fra en lukket sirkel. Ved bruk av den klassiske piler og sirkel notasjonen brukes dobbelsirkel for å vise starttilstand. Figuren under viser dette for tilstandsmaskinen over:



Nr III i "sirkler og piler" notasjon.

- b. Figuren under viser en skisse til en Java-basert webtjeneste for spesifikasjon og bestilling av visittkort for NTNU-ansatte. Brukerne skal kunne gi inn tekst og spesifisere farge, språk, og evt. bilde. Ønsket antall visittkort skal så kunne bestilles.

Forhåndsvisning:		Antall: 200 <input type="button" value="Bestill"/>
Farge: <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>		
Bilde: <input type="checkbox"/> <input type="button" value="Importer bilde"/>		
Navn: <u>Dag Svanæs</u>		
Enhet: <u>IDI</u>		
e-mail: <u>dags@idi.ntnu.no</u>		
Web: <u>http://dag.idi.ntnu.no</u>		
Språk:		
<input type="radio"/> Norsk		
<input checked="" type="radio"/> Engelsk		

I analyser av brukergrensesnitt skiller man ofte mellom semantiske, syntaktiske og leksikalske modeller/beskrivelser.

For hvert av de fem utsagnene under, bestem om de er på semantisk, syntaktisk eller leksikalsk nivå. Begrunn valgene.

Fem utsagn om brukergrensesnittet over:

- i. For å kunne importere et bilde ved å trykke på "Importer bilde" knappen må "Bilde" boksen først være valgt (avkrysset).
- ii. Et trykk på "Bestill" knappen fører til at det sendes en melding til trykkeriet.
- iii. Det finnes fem små bokser som indikerer forskjellige bakgrunnsfarger.
- iv. Forandringer fører kontinuerlig til at forhåndsvisningen oppdateres.
- v. Knapper vises på skissen som tekster som er omgitt av en avrundet firkant.

Forslag til løsning 2b:

- Utsagn på semantisk nivå handler om betydningen av en handling. Det vil si hva en handling fører til, dens funksjon.
- Utsagn om syntaks handler om hva som er lovlig rekkefølge på enkelthandlinger – hva som er de grammatikalske reglene i grensesnittet.
- Leksikalske utsagn handler om et grensesnitts enkeltdele/byggekløsser. Dette tilsvarer alfabetet i en analyse av et språk.

For hvert av utsagnene:

- i. For å kunne importere et bilde ved å trykke på "Importer bilde" knappen må "Bilde" boksen først være valgt (avkrysset).

Dette omhandler syntaks. Utsagnet sier noe om hva som er lovlig rekkefølge av enkelthandlinger.

- ii. Et trykk på "Bestill" knappen fører til at det sendes en melding til trykkeriet.

Dette er semantikk. Hva skjer når man trykker på en knapp?

iii. *Det finnes fem små bokser som indikerer forskjellige bakgrunnsfarger.*

Dette omhandler grensesnittets elementer og er følgelig på leksikalsk nivå. De fem boksene er symboler i et språk for å beskrive bakgrunnsfarge. Ettersom utsagnet også sier noe om elementenes betydning, kan det argumenteres for at det samtidig er semantisk.

iv. *Forandringer fører kontinuerlig til at forhåndsvisningen oppdateres.*

Dette omhandler semantikken i grensesnittet. Utsagnet sier noe om sammenhengen mellom handlinger og konsekvensen av disse.

v. *Knapper vises på skissen som tekster som er omgitt av en avrundet firkant.*

Dette er et utsagn om grensesnittets byggesteiner. Det er følgelig et leksikalsk utsagn.

Oppgave 3 – Java og Swing (35%)

Java appleten som ble skissert i oppgave 2b skal realiseres v.h.a. en Model-View-Controller (MVC) arkitektur og Swing. Det er et krav at forhåndsvisningen skal oppdateres kontinuerlig når inndata forandres.

NB!:

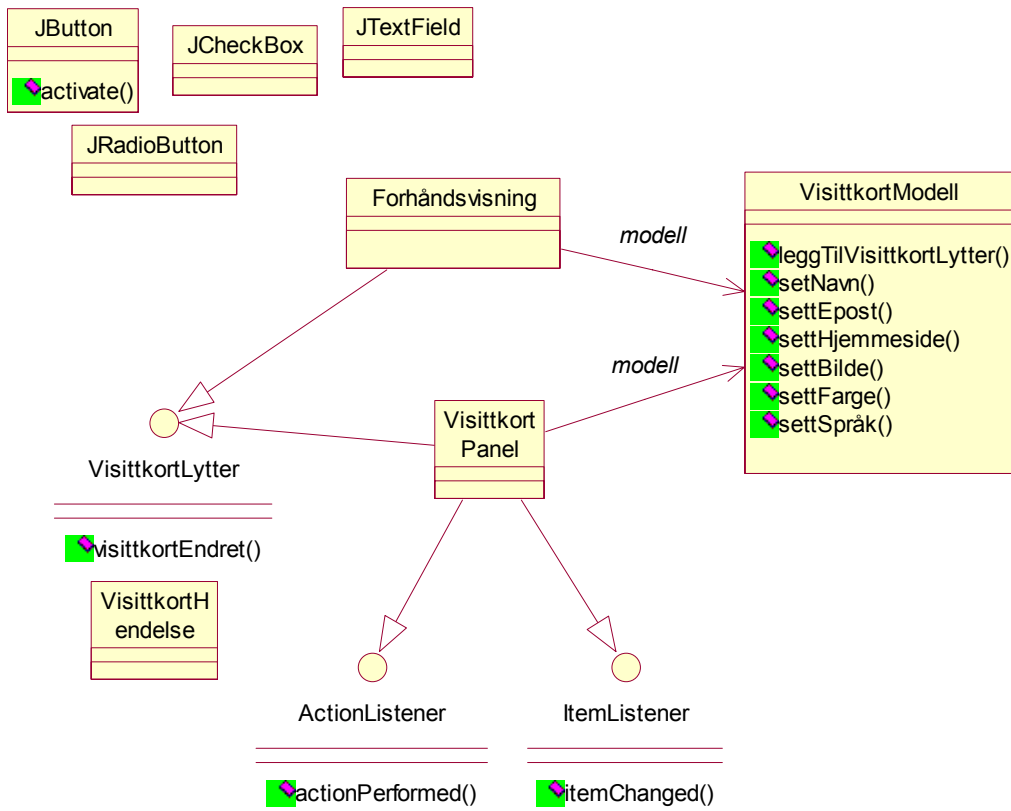
- Du skal i denne oppgaven ikke lage Java kode, kun angi strukturer og sammenhenger.
 - Du skal heller ikke detaljbeskrive grafikkrutinene for tegning av visittkortet.
- a. Skisser en løsning til en overordnet struktur/arkitektur som på en systematisk måte ivaretar kravet til kontinuerlige oppdateringer på skjermen. Hvilke klasser vil du definere, og hvordan skal de samarbeide? Tegn UML Collaboration diagram og forklar.
 - b. Vis et sekvensdiagram som angir meldingsflyten når en bruker skifter fra grå til hvit bakgrunnsfarge. Forklar.
 - c. Knappen "Importer bilde" skal kun være aktiv når avkrysningsboksen "Bilde" er valgt. I motsatt tilfelle er knappen grå for å indikere passiv tilstand, og den reagerer følgelig ikke på knappetrykk.
 - Skisser to forskjellige løsninger for å realisere denne avhengigheten mellom de to GUI elementene. Angi fordeler og ulemper med de to løsningene du har skissert.

Kort liste av noen relevante Swing-klasser:

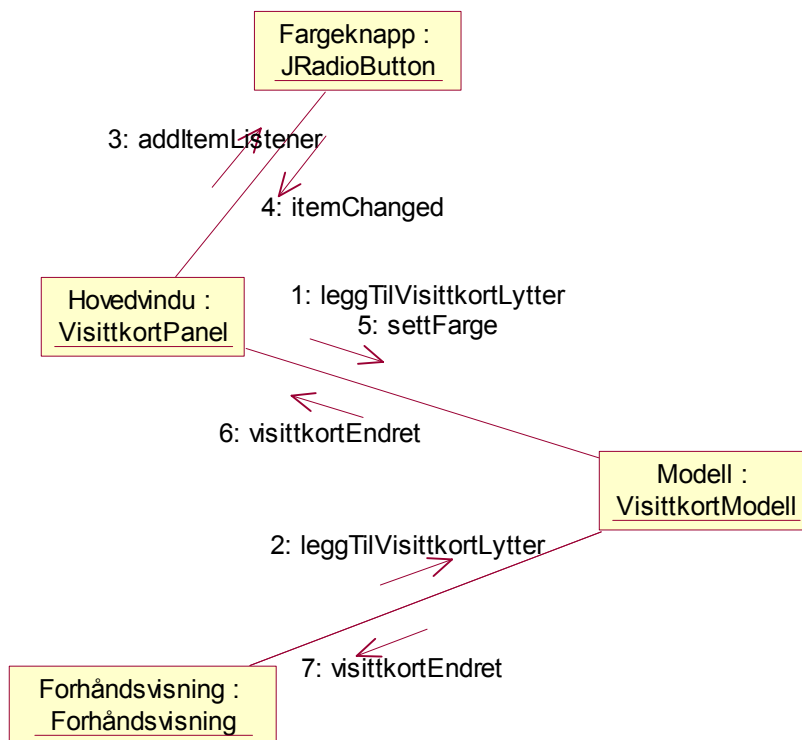
- Knapper: *JButton*
- Avkrysningsboks: *JCheckbox*
- Radioknapper: *JRadioButton* (den kan også bruke egendefinerte ikoner)
- Samling av gjensidig utelukkende knapper: *ButtonGroup*
- Statisk tekst i bakgrunn: *JLabel*
- Tekst input felter: *JTextField*
- Statisk HTML boks: *JEditorPane* (Viser HTML, f.eks. for forhåndsvisning)

Forslag til løsning til a, b og c.

- a) Her er poenget å la skjemaet og forhåndsvisningsvinduet dele en felles modell, som skjemaet endrer og forhåndsvisningsvinduet lytter til endringer i. I tillegg til klasser/objekter for alle synlige vinduer og dialogelementer, må en derfor innføre en modellklasse, et lytter-grensesnitt og en hendelsesklasse. Modellklassen må inneholde en metode for å registrere lyttere, mens lyttergrensesnittet må inneholde en metode som kalles når en endring skjer, og som tar et hendelsesobjekt som parameter (det eneste). Klassediagrammet blir omtrent som følger (diagrammet er ikke nødvendig i besvarelsen, men er nyttig for å vise metodene som angis i Collaboration-diagrammet):



Collaboration-diagrammet blir omtrent som følger:



Vi er ikke så nøye på notasjonen, det blir lett at en blander klasse og objekter i samme diagram, bare de skjønner rollen til hver klasse/hvert objekt. Det viktigste er at det er med metoder for å endre modellen, registrere lyttere og reagere på endringshendelser, og at det kommer frem at skjemaet må registrere seg hos dialogelementene (og modellen) og at forhåndsvisningsvinduet registrerer seg hos kun modellen. Det er ikke viktig å si hvilke Swing-klasser som brukes som dialogelementer, men dersom en tar det med bør det være omtrent riktig. F.eks. bør fargeknappene være JRadioButton-objekter i en ButtonGroup. Det spiller ingen rolle om vinduet som helhet er en JFrame/Frame, JApplet/Applet eller JPanel/Panel. Det er forsåvidt greit om skjema-klassen arver fra JPanel, men ikke nødvendig (at den arver noe i det hele tatt). Den må uansett vite om og registrere seg hos hvert enkelt dialogelement.

- b. *Sekvensen i kall blir som følger (kan leses fra Collaboration-diagrammet over):*
- *1 og 2. Vinduene for skjema og forhåndsvisning legger seg på som lyttere til modellen (strengt tatt trenger ikke skjemaet lytte til endringer, da den ikke kan endres av andre.)*
 - *3. Skjemaet registrerer seg som lytter til innfyllingsfeltene (her er bare fargeknappen vist).*
 - *4. Et innfyllingsfelt (her fargeknappen) sier fra at noe er endret. I tilfellet med fargeknapper, kan det tenkes at en får en hendelse også når en knapp slås av, men er ikke så viktig.*
 - *5. Modellen endres tilsvarende.*
 - *6 og 7. Lytterne får beskjed om endringen (og oppdateres deretter).*
- c. *Koblingen mellom avkrysningsboksen for bildet og knappen "Importer bilde" kan ordnes på flere måter. Vi er ute etter varianter med og uten en eksplisitt modell av tilstanden til avkrysningsboksen, som deles av både avkrysningsboksen og importer-knappen.*
- *Den enkleste måten er å la importer-knappen subklasse JButton, la den lytte på avkrysningsknappen (JCheckBox) dvs. implementere ItemListener-grensesnittet og så aktivere/deaktivere knappen (this) deretter. En tilsvarende og litt mer komplisert måte, er å lage en egen klasse (gjerne anonym) som lytter på avkrysningsboksen og aktiverer/deaktiverer importer-knappen deretter. Fordelen med disse to teknikkene er at de er enkle og kun krever "lokal" samhandling mellom objekter. Dersom en f.eks. gjør om avkrysningsboksen til et tekstfelt (filnavn/URL) og lar importer-knappen alltid være aktiv, så påvirkes ikke andre deler.*
 - *Den andre teknikken vi ønsker de skal se er å legge støtte for bildet inn i modellen, dvs. modellen forandres når avkrysningsknappen brukes. Skjemaet lytter til modellen og aktiverer/deaktiverer importer-knappen deretter (egentlig skal avkrysningsboksen også oppdateres, i tilfelle modellen er endret utenifra). Denne teknikken er altså mer indirekte, ved at en endrer på modellen og så reagerer på en endringshendelsen. Fordelen her er at endringer i modellen utenifra vil oppdatere grensesnittet riktig, det er mao. enklere å sikre konsistens i grensesnittet.*