NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET
INSTITUTT FOR DATATEKNIKK OG INFORMASJONSVITENSKAP

Faglig kontakt under eksamen:
Dag Svanæs, Tlf: 73 59 18 42

LØSNINGSFORSLAG

EKSAMEN I FAG
TDT4180/IT2401 – MMI

Onsdag 23. mai 2007
Tid: kl. 0900-1300

Bokmål

Sensuren faller 13. juni 2007

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

Karakterskalaen ved NTNU

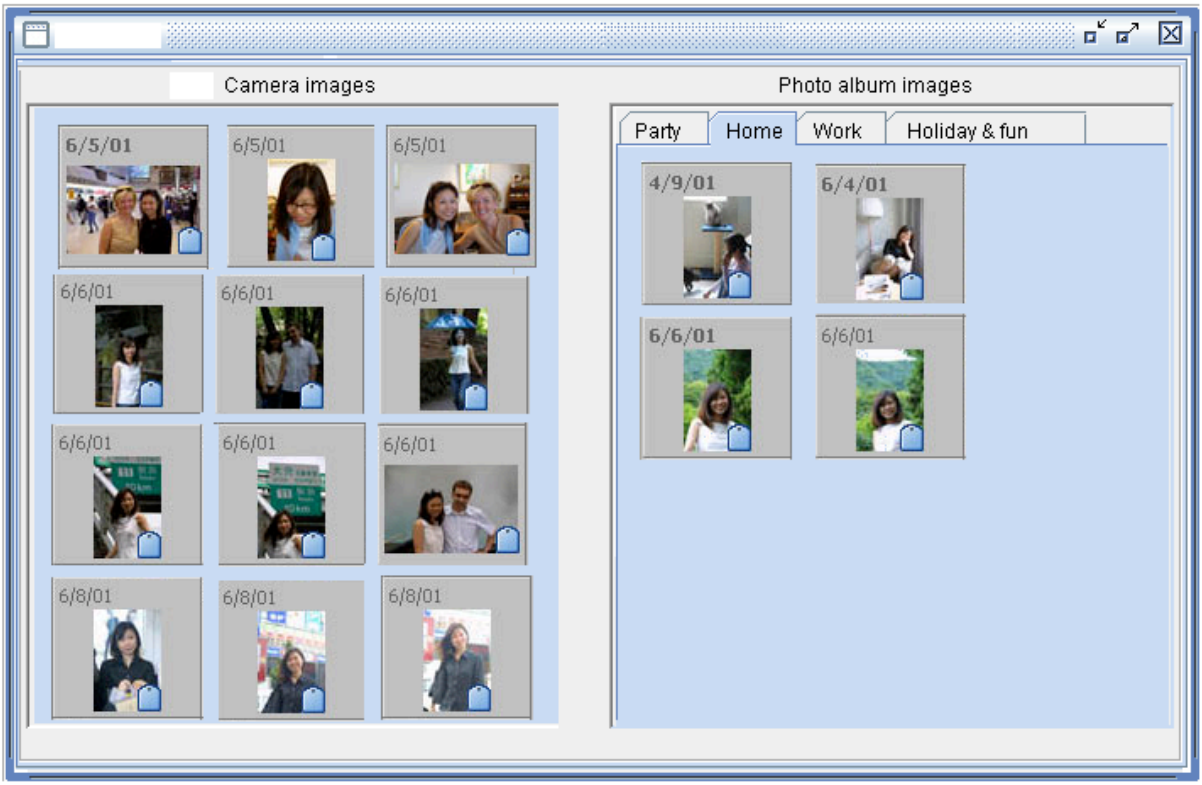
Styret ved NTNU har den 31.08.04 (S-sak 46/04) revidert den generelle beskrivelsen av karaktertrinnene og endret NTNUs studieforskrift § 39. Den reviderte beskrivelsen er anbefalt av Utdannings- og forskningsdepartementet og Universitets- og høgskolerådet.

Symbol	**Betegnelse**	**Generell, ikke fagspesifikk beskrivelse av vurderingskriterier**
A	Fremragende	Fremragende prestasjon som klart utmerker seg. Kandidaten viser svært god vurderingsevne og stor grad av selvstendighet.
B	Meget god	Meget god prestasjon. Kandidaten viser meget god vurderingsevne og selvstendighet.
C	God	Jevnt god prestasjon som er tilfredsstillende på de fleste områder. Kandidaten viser god vurderingsevne og selvstendighet på de viktigste områdene.
D	Nokså god	En akseptabel prestasjon med noen vesentlige mangler. Kandidaten viser en viss grad av vurderingsevne og selvstendighet.
E	Tilstrekkelig	Prestasjonen tilfredsstiller minimumskravene, men heller ikke mer. Kandidaten viser liten vurderingsevne og selvstendighet.
F	Ikke bestått	Prestasjon som ikke tilfredsstiller de faglige minimumskravene. Kandidaten viser både manglende vurderingsevne og selvstendighet.

I tillegg bruker NTNU skalaen bestått/ikke bestått._For obligatoriske aktiviteter som skal registreres, men ikke evalueres - f eks deltakelse på ekskursjoner, innlevering av et bestemt antall oppgaver o l, brukes fullført/ikke fullført.

Oppgave 1 (30%) Grensesnittdesign

På bildet under ser du en første skisse til et program for å håndtere digitale bilder på en PC, et s.k. digitalt fotoalbum. Dette skjermbildet viser hovedstrukturen for hvordan det kan se ut når brukeren skal legge over bilder fra sitt digitale kamera.



Skjermbildet er delt i to. Til venstre er alle foto i kameraet. Til høyre er bildene i albumet. Albumet er delt opp i kategorier som brukeren selv har valgt (”Party”, ”Home” etc.). Det brukes ”tabbed panes” for å velge mellom katagoriene.

a)
Angi tre forskjellige interaksjonsstiler / dialogformer for å flytte bilder fra kamera over til albumet. For hver av løsningene, hvilke grensesnittelementer vil måtte legges til skjermbildet? Hva er fordeler og ulemper med hver av de tre løsningene for dette programmet?

Det som kjennetegner dette programmet:

- Brukergruppe med lite IT-erfaring (eldre). Følgelig viktig at alt er enkelt å forstå.
- Bruk av bilder gjør at man ikke bør sløse med skjermplass. Bilder tar mye plass.
- Vi kan ha svaksynte, noe som gjør at liten tekst kan være vanskelig å lese.

Tre hovedmetoder er:

- Direktemanipulasjon (DM)
- Ikonbasert – knapper.
- Nedtrekksmenyer

Skjermelementer:

- DM krever ingen ekstra elementer. Alt som kreves er at det er mulig å velge ett eller flere elementer og dra de over fra venste til høyre.
- Ikonbasert krever en knapp pr. operasjon. F.eks. ”Slett valgte bilde(r)”, ”Flytt bilder” etc.
- Nedtrekksmenyer krever at det er menyer på toppen, slik det som regel er i appliasjoner. Det er da naturlig at disse ligger under ”edit”, og bruke ”cut”, ”clear”, ”paste” og ”copy”.

Fordeler/Ulemper for dette programmet:

DM:

- Fordeler:
  - Krever ikke ekstra elementer på skjermen, og sparer følgelig skjermplass. Enkelt og effektiv å bruke når man har forstått det.
- Ulemper:
  - Det mangler ”affordance” som forteller at det er mulig å dra ikoner. Dette må forklares på et eller annet vis.

Ikoner:

- Fordeler:
  - Lett å forstå dersom ikonene plasseres riktig på skjermbildet og ikon/tekst er lett forståelig.
- Ulemper: Tar skjermplass.

Menyer:

- Fordeler:
  - Standard måte å gjøre det på i de vanligste plattformene (Windows, MacOs,,).
- Ulemper:
  - Tar litt skjermplass. Krever litt flere musetrykk av brukeren. For erfarne brukere kan det legges in shortcuts (Ctrl C, Ctrl V,,).

Utifra dette så ser vi en mulig avveining mellom DM som gir mye ledig skjermplass, men krever ekstra forklaring på bruk, og Ikoner/menyer som tar litt plass, men er enklere å forstå for nybegynnere.

Karaktersetting: Ren oppramsing av metoder med pros/cons gir max. karakteren C. For B og A kreves at de viser at de har forstått applikasjonen og målgruppens egenart og argumenterer opp mot det.

b)

Schneiderman har 8 enkle regler for brukergrensesnitt:

1. Lag konsistente grensesnitt.
2. Gi trente brukere ”shortcuts”.
3. Gi informative tilbakemeldinger.
4. Lag tydelige arbeidssteg i dialogene.
5. Unngå feil og gi evt. enkle feilmeldinger.
6. La det være mulig å angre.
7. La brukeren ha kontroll.
8. Reduser behovet for å huske (korttidsminnet)

Angi kort hvordan hver av de 8 regelene kan anvendes for albumprogrammet.

Lag konsistente grensesnitt.

Sørg for at funksjoner som handler om det samme ser likt ut i alle deler av grensesnittet. F.eks. så bør måten å fjerne et bilde fra kameraet være likt måten å fjerne et bilde fra PCen.

Gi trente brukere ”shortcuts”.

I dette tilfelle kunne det være for f.eks. flytting av bilder. Det er vanlig å legge dette inn i nedtrekksmenyer. For dette konkrete tilfellet så er målgruppen ikke særlig trent. Det er derfor viktig at ”shortcuts” ikke kommer i veien for vanlig bruk, men som et ekstra for erfarne brukere.

Gi informative tilbakemeldinger.

Dette er ekstra viktig for eldre brukere. Prøv i så stor grad som mulig å unngå at feilsituasjoner oppstår. Dersom de oppstår, så bør det gis detaljert forklaring på hva som er problemet og hva som kan gjøres for å rette det opp.
Eksempel:

- Riktig: ”PCen har mistet kontakten med kameraet. Det kan skyldes at kabelen mellom kamera og PC ikke lenger er koblet ordentlig. Det kan også skyldes at kameraet er gått tomt for batteri. Når feilen er rettet opp vil programmet kunne fortsette”.
- Feil: ”Error 12 - USB communication error”.

Lag tydelige arbeidssteg i dialogene.

Dersom brukeren skal gjøre noe i sekvens så er det viktig at de vet hvor de er i prosessen. Eksempel kan være å sette inn og formatere en ny minnebrikke i kameraet. Det bør forklares steg for steg.

Unngå feil og gi evt. enkle feilmeldinger.

Se over.

La det være mulig å angre.

Alle handlinger bør kunne angres. For denne målgruppen er det viktig at de kan angre på det meste. Hvis mulig bør det f.eks. tas automatisk backup av bilder som slettes (en søplebøtte som så må tømmes eksplisitt).

La brukeren ha kontroll.

La aldri programmet ta initiativ til handlinger, slik f.eks. bindersen i Windows prøvde å gjøre. Spesielt for denne målgruppen så ville det sette en del brukere helt ut av spill.

Reduser behovet for å huske (korttidsminnet).

Sørg for at alle viktige valg til enhvertid er tydelig synlige på skjermen, og ikke gjemt bort i obskure undermenyer.

Karaktersetting: Her må de vise at de kan anvende disse prinsippene på dette konkrete eksemplet. Bare generell kunnskap om prinsippene gir max C.

Oppgave 2 (30%) Designprosessen

Albumprogrammet i oppgave 1 skal følge med som gratisprogram for et nytt digitalt kamera spesielt beregnet for pensjonister og eldre mennesker. Selve kameraet er utformet med få knapper og er gjort så enkelt som mulig.

Du har fått i oppdrag å finne ut hvilke funksjoner som skal tilbys i albumprogrammet. Mulighetene er mange, som f.eks. automatisk utlegging på hjemmeside, fjerning av røde øyne, tidslinje, tagging med GPS data og mye annet.

Hvordan vil du gå fram for å identifisere det riktige utvalget av funksjoner for denne målgruppen? Du har ressurser til å anvende tilsammen kun tre brukersentrerte metoder.

a) Hvilke tre metoder ville du ha valgt? Begrunn svaret.

Karaktersetting: For alle deloppgavene gjelder også her at ren oppramsing av metoder med pros/cons gir max. karakteren C. For B og A kreves at de viser at de tar utgangspunkt i applikasjonen og målgruppens egenart og argumenterer opp mot dette.

Det som kjennetegner situasjonen:

- Målgruppe med liten IT-erfaring.
- Tidlig fase av prosjekt, med fokus på kravinnhenting.
- En del rammebetingelser, som f.eks. et ferdig kamera, er gitt.
- Dette er en digitalisering av en allerede eksisterende praksis og teknologi (”gammeldagse” foto og fotoalbum)

Jeg ville ha fokusert på metoder som gir dialog med brukerne gjennom konkrete eksempler. Det er også viktig å få innsikt i hvordan brukergruppen bruker foto i dag.

1.
Jeg ville ha begynt med et dybdeintervju med noen utvalgte pensjonister om hvordan de bruker foto og fotoalbum idag. Jeg ville bedt de om å vise meg albumene sine og fortelle meg hvordan de gjør når de tar bilder og når de får bilder tilbake fra framkalling. Dette for å få innsikt i ”context-of-use”. Jeg ville også ha intervuet de om hvordan de bruker PC, internett, mobiltelefon og annen elektronikk i dag.

2.
Basert på det jeg lærte gjennom intervjuene så ville jeg ha laget noen enkle papirprototyper (skjermbilder) som jeg hadde vist fram i en fokusgruppe av pensjonister. Disse ville jeg ha supplert med noen enkle scenarier som jeg hadde laget for å illustrere bruken. Jeg ville så ha ønsket tilbakemelding på disse ideene og generelle krav til et slikt system.

3.
Etter å ha gjennomført fokusgruppen ville jeg ha laget en mer utfyllende prototyp som jeg hadde gjort en wizard-of-Oz test på noen brukere. Jeg ville ha brukt mye tid etter testene på prat rundt hva de tenker om noe slikt, og hva de anser som vesentlig og mindre vesentlig med en slik tjeneste.

b)
Angi hver av de tre valgte metodenes styrker og svakheter for denne oppgaven.

Styrken til intervjuene er at de gir dybdekunnskap om brukerne og deres hverdag. Ulempen er at man bare når et fåtall brukere.

Styrken til fokusgruppen er at man får tilbakemelding på konkrete forslag. Ulempen er at det er mange stemmer og det er lett å overse viktige detaljer. Det er også egentlig få brukere.

Styrken med Wizard-of-Oz test er at den er billig sammenlignet med en full test av kjørende prototyp. Den gir også tilbakemelding på brukervennlighet og egnetheten av konkrete løsninger. Ulempen er at det er en kunstig situasjon og at prototypen er meget ufullstendig.

c) For hver av de tre metodene, angi viktige ting å huske på med referanse til ISO9241-11 sin definisjon av brukervennlighet:
”*Anvendbarhet, effektivitet og tilfredsstillelse*
for bestemte brukere
med bestemte mål
i bestemte omgivelser.”

Intervju:
Det er viktig å plukke ut representative brukere. Å forstå hva de gjør i dag (mål med å ta vare på bilder og lage fotoalbum), og i hvilke omgivelser de gjør det (for seg selv, for å vise fram til andre,,,). Anvendbarhet går på å identifisere de sentrale funksjonene. Er effektivitet viktig? Hva kreves for å få god subjektiv tilfredstillelse?

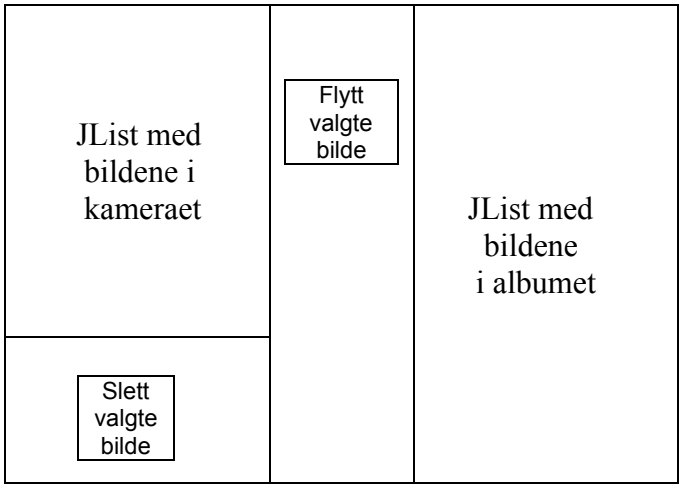
Fokusgruppe:
Har vi funnet representative brukere? Mål: Er scenariene realistiske? Omgivelser: Er de satt i realistiske omgivelser? Anvendbarhet: Er det dette de primært ønsker å kunne gjøre med programmet? Effektivitet: Hva er det viktig at går fort (hvis noe)? Hva bør være ”brukeropplevelsen”?

Wizard-of-Oz test:
Mye likt som for fokusgruppen. Det bør spørres om målgruppe, mål, og omgivelser i debriefingen etter testen.

Karaktersetting: På 2c er det viktig at de forstår at ISO9241-11 kan brukes som en slags veiledning når man anvender brukersentrerte metoder. De som ikke har sett koblingen får ikke bedre enn D her.

Oppgave 3 (40%) Grensesnittkonstruksjon

For å teste implementasjonen av programmet skal det lages en enkel prototyp i SWING. I stedet for ikoner brukes det lister med navnet på bildene (tekst). SWING-komponenten JList brukes i denne implementasjonen. Figuren under viser en skjematisk oversikt over denne testimplementasjonen. I denne første versjonen er bildene i albumet ikke ordnet i kategorier, altså kun en liste av alle bildenavnene.



Til venstre er en JList med listen av bildene i kameraet. Listen tillater valg av et element (single selection). Til høyre er en JList med listen av bildene i albumet på PCen. Det er to knapper, implementert som JButtons.

- Knappen under den venstre listen brukes til å slette valgte bilde i kameraet.
- Knappen mellom listene brukes til å flytte valgte bilde fra kameraet til albumet. Bildet skal da forsvinne fra kameraet. Dersom det er gjort et valg i høyre liste så skal bildet legges etter dette valget. Dersom ikke noe er valgt, så skal bildet legges først i listen.

For enkelhets skyld kan du bruke DefaultListModel som datamodell for hver av de to listene og JPanel som ytre ”innpakning” for alle elementene.

a)
Forklar den grunnleggende ideen bak ”Model-View-Controller”-prinsippet (MVC): Hva er hensikten med MVC, og hvordan er MVC realisert i Swing?

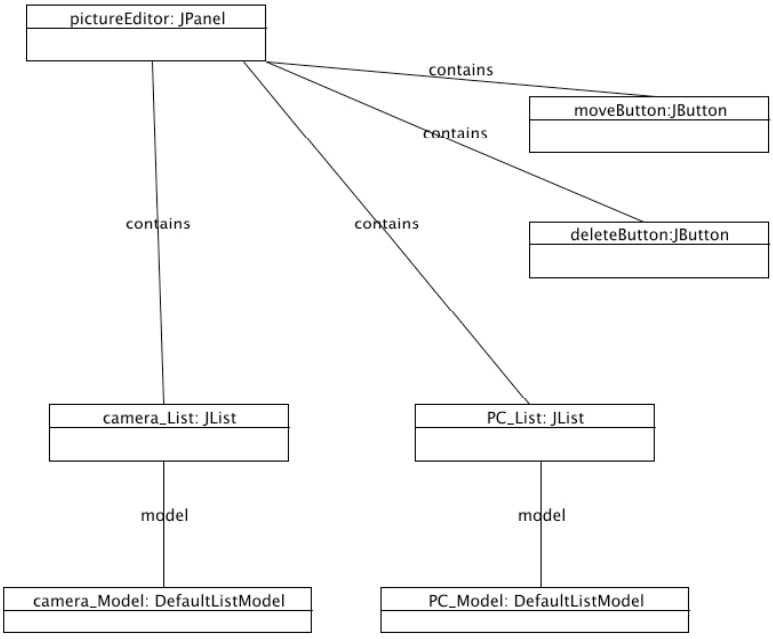
- MVC er en programvare arkitektur / teknikk for å skille data fra grensesnittelementer. Det kan sees på som en realisering av de to øverste lagene i en 3-lags arkitektur (grensesnitt, business logic, persistens). MVC stammer fra Smalltalk prosjektet, der man skilte mellom modellen, viewet og kontrolleren. Modellen innholder alle data som forandrer seg under kjøring. Viewet tar seg av presentasjon på skjerm, mens kontroller tar seg av input.

- Model, View og Controller objekter kobles under kjøring sammen slik at en modell kan ha flere View-Controller par koblet til seg. Ved å automatisere oppdatering av alle views når modellen endres, så innfører man en abstraksjon som gjør det mulig lett å legge til nye views.
- Oppdatering av views gjøres ved at modellen holder en liste med de View-Controllere som er koblet til den, og når det skjer en endring av modellen så sendes det beskjed til alle views om at det har skjedd en endring. Hvert enkelt view er så ansvarlig for å oppdatere seg selv ved å spørre modellen som sine data. Dette krever at all forandring av variable i modellen skjer gjennom egne "set" metoder slik at modellen kan sende de riktige oppdateringsmeldingene.

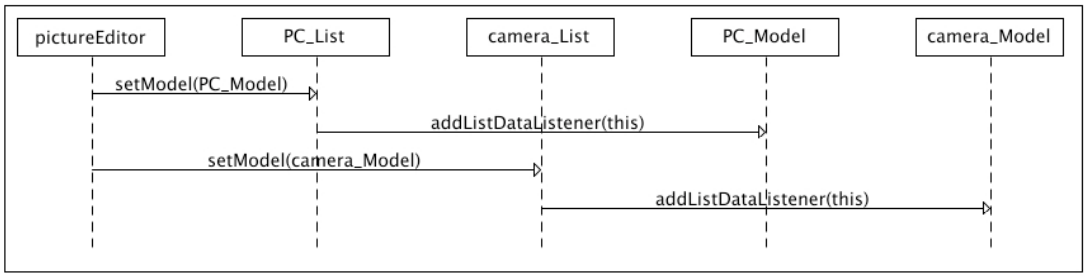
b)
Anvend MVC-prinsippet på testimplementasjonen.

- Hvilke objekter/instanser vil eksistere når programmet kjører? Tegn opp objektene og vis v.h.a. piler hvilke relasjoner som finnes mellom objektene.
- Vis v.h.a. et sekvensdiagram og forklar tekslig hva som skjer initielt når programmet startes opp.
- Vis v.h.a. et sekvensdiagram og forklar tekslig hva som skjer når brukeren har valgt et element i listen til venstre og trykker knappen for å slette valgte element.
- Vis v.h.a. et sekvensdiagram og forklar tekslig hva som skjer når brukeren har valgt et element i listen til venstre og trykker knappen for å flytte valgte element over til albumet.

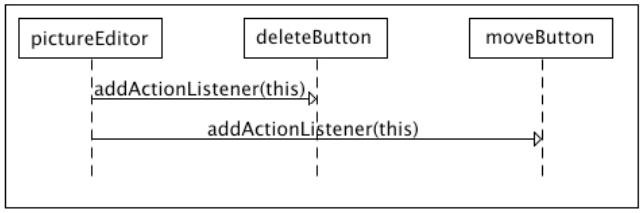
Oversikt over objekter:



Sekvensdiagram for oppstart:
For å få plass i bredden har jeg delt sekvensdiagrammet i to. Først det som kobler modellene til sine JList objekter. JPanel objektet sender ”setModel” til JList objektene. ”AddListDataListener” blir så sendt automatisk av JList objektene til sine respektive modellobjekter:

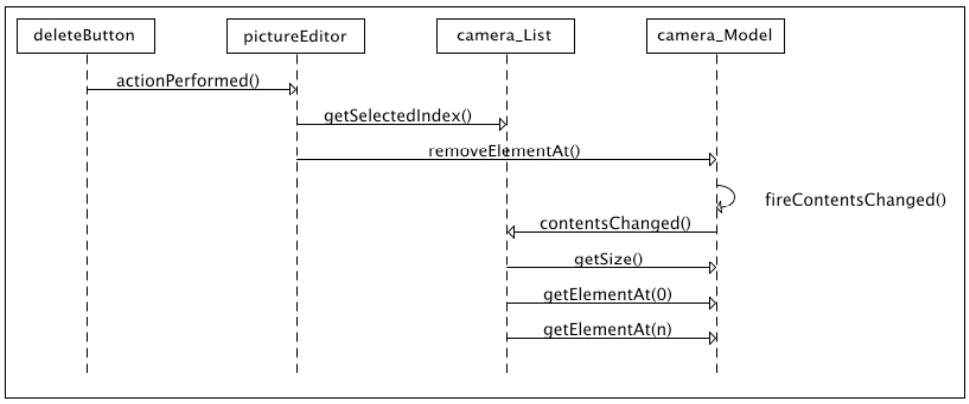


Deretter blir knappene knyttet til pictureEditor med ”call backs”. Dette fordrer at pictureEditor implementerer interfacet ActionListener:



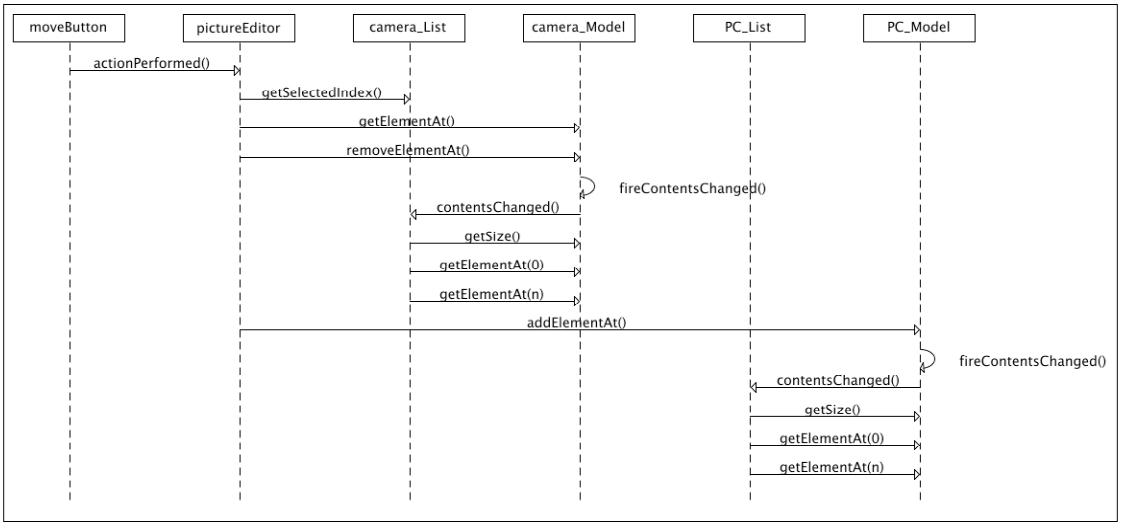
I tillegg til det som er vist her så settes begge listene i SINGLE_SELECTION mode ved oppstart og begge JList og JButton objektene legges inn i pictureEditor:JPanel med ”add”.

Sekvensdiagram for sletting av element:
Brukeren trykker på deleteButton knappen. Det starter en call-back til JPanel obektet. Det spør så camera_List om hvilket bilde som er valgt og sender beskjed til camera_Model om å slette det bildet. Dette skaper da automatisk et kall til seg selv om at noe er endret (fireContentsChanged). camera_List er det eneste objektet som har camera_Model som modell. Det får beskjed om at noe er endret (contentsChanged), og spør så modellen om størrelse på listen (getSize). Deretter itereres det over alle element 0..n i listen og henter nye data fra modellen som skal vises i lista på skjermen (getElementAt):



Sekvensdiagram for flytting:

Flytting er veldig lik sletting. I tillegg til sletting så blir valgte element i cameraList hentet ut fra modellen og lagt inn i PC_List sin modell. Oppdatering og samspill mellom modell og liste er lik:



---

c)

JButton har en metode ”void setEnabled(boolean b)” som gjør det mulig å sette en knapp i ”passiv” modus slik at den ikke kan ta imot brukerklikk. I denne implementasjonen så hadde det vært ønskelig at de to knappene var passive når det ikke var gjort valg i venstre liste. Forklar kort hvilke mekanismer JList tilbyr som kunne vært benyttet for å realisere denne funksjonaliteten.

Detaljene for besvarelse av denne oppgaven er ikke med i tillegget. Svaret er derfor på et litt generelt plan med utgangspunkt i det som er pensum i faget:

JList har også en s.k. selectionModel som tar hånd om seleksjon. Denne kan modifiseres slik at alle endringer av seleksjon fører til et kall. Endringskallet kan sendes enten til JPanel eller til selve knappen. Den som får endringshendelsen kan så sjekke om JList har noe innhold ved å spørre dens modell ”getSize()”. Dersom den har innhold så kalles setEnabled(true) ellers setEnabled(false) på knappen.

Relevante definisjoner fra SWING-dokumentasjonen.

public class JList extends JComponent
implements ListDataListener

A component that allows the user to select one or more objects from a list. A separate model, ListModel, represents the contents of the list.

Method Summary

void setModel(ListModel model)
Sets the model that represents the contents or "value" of the list and clears the list selection after notifying PropertyChangeListeners.

void setSelectionMode(int selectionMode)
Determines whether single-item or multiple-item selections are allowed.
…ListSelectionModel.SINGLE_SELECTION Only one list index can be selected at a time

boolean isSelectionEmpty()
Returns true if nothing is selected.

int getSelectedIndex()
Returns the first selected index; returns -1 if there is no selected item.
(If single selection mode, this method returns the index of the selected object, if any. –1 if no object selected).

public interface ListModel

This interface defines the methods components like JList use to get the value of each cell in a list and the length of the list. Logically the model is a vector, indices vary from 0 to ListDataModel.getSize() - 1. Any change to the contents or length of the data model must be reported to all of the ListDataListeners.

Method Summary

Object getElementAt(int index)
Returns the value at the specified index.

int getSize()
Returns the length of the list.

void addListDataListener(ListDataListener l)
Adds a listener to the list that's notified each time a change to the data model occurs.

void removeListDataListener(ListDataListener l)
Removes a listener from the list that's notified each time a change to the data model occurs.

public class DefaultListModel extends AbstractListModel

This class loosely implements the java.util.Vector API, in that it implements the 1.1.x version of java.util.Vector, has no collection class support, and notifies the ListDataListeners when changes occur.

All Implemented Interfaces:
ListModel

Method Summary

Object getElementAt(int index)
Returns the component at the specified index.

int getSize()
Returns the number of components in this list.

void add(int index, Object element)
Inserts the specified object as a component in this list at the specified index. (index = 0 inserts the element first in the list).

void removeElementAt(int index)
Deletes the component at the specified index

public abstract class AbstractListModel extends Object implements ListModel

The abstract definition for the data model that provides a List with its contents.

Method Summary

void addListDataListener(ListDataListener l)
Adds a listener to the list that's notified each time a change to the data model occurs.

protected void fireContentsChanged(Object source, int index0, int index1)
AbstractListModel subclasses must call this method after one or more elements of the list change.

public interface ListDataListener extends EventListener

Method Summary

void contentsChanged(ListDataEvent e)
Sent when the contents of the list has changed

public class JButton
extends AbstractButton

Constructor Summary

JButton(String text)
Creates a button with text.

public abstract class AbstractButton
extends Jcomponent

Method Summary

void addActionListener(ActionListener l)
Adds an ActionListener to the button.

void setEnabled(boolean b)
Enables (or disables) the button.

public interface ActionListener
extends EventListener

Method Summary

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

public class JPanel extends JComponent

JPanel is a generic lightweight container

Kommentarer:

- ActionEvent er ikke relevant for oppgaven, og er ikke listet her.
- EventListener er ikke relevant for oppgaven og er ikke listet.
- For de spesielt SWING-interesserte: I den faktiske implementasjon av SWING så bruker JList en egen intern klasse for å implementere ListDataListener interfacet. Resultatet er det samme som om den implementerte dette direkte.