NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET
INSTITUTT FOR DATATEKNIKK OG INFORMASJONSVITENSKAP

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

EKSAMEN I FAG
TDT4180/IT2401 – MMI

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

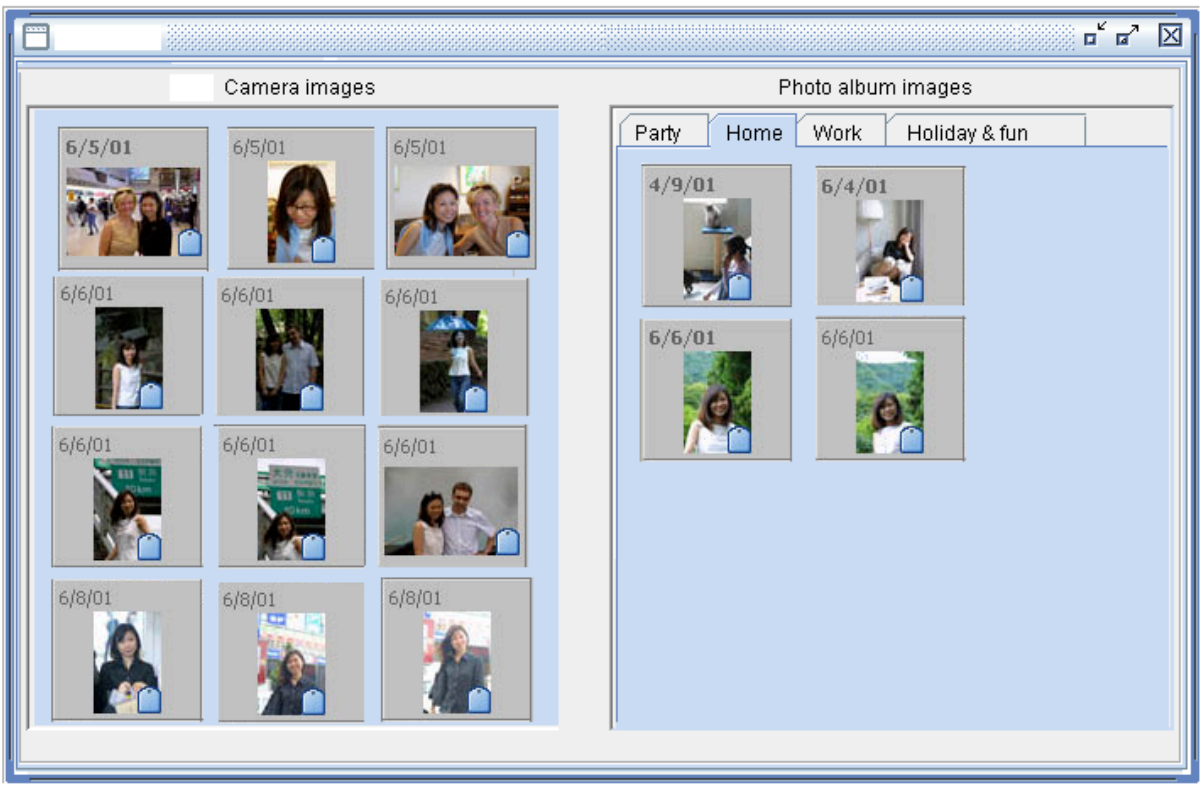
Bokmål

Sensuren faller 13. juni 2007

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

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?

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.

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.

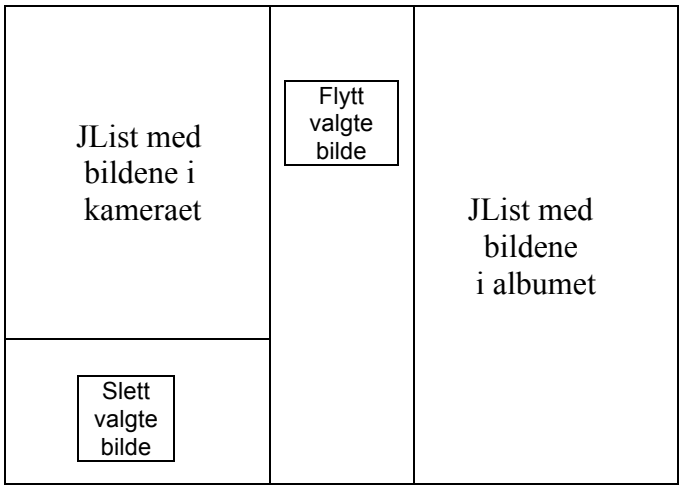
Hvordan vil du gå fram for å identifisere det riktige utvalget av funksjoner for denne målgruppen? Du har ressurser til å anvende tilsammen kun tre brukersentrerte metoder.

a) Hvilke tre metoder ville du ha valgt? Begrunn svaret.
b) Angi hver av de tre valgte metodenes styrker og svakheter for denne oppgaven.
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.”

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?

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.

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.

- **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.