NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET
INSTITUTT FOR DATATEKNIKK OG INFORMASJONSVITENSKAP

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

EKSAMEN I FAG
TDT4180 MMI

Mandag 18. mai 2009
Tid: kl. 0900-1300

Bokmål

Sensuren faller 8. juni 2009

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

Kvalitetssikret: Hallvard Trætteberg

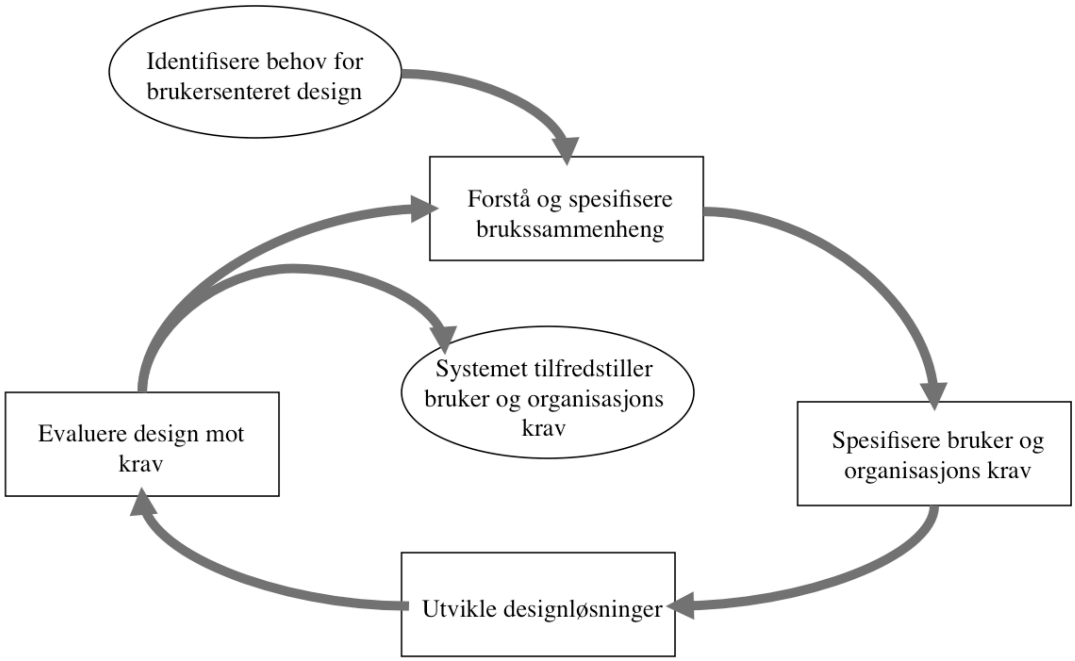
Oppgave 1 (25%) Grensesnittdesign

a) Forklar begrepet *mental modell*, og vis med et eksempel hvordan dette begrepet er relevant for utformingen av brukergrensesnitt.
b) Forklar begrepet *affordance*, og vis med et eksempel hvordan dette begrepet er relevant for utformingen av brukergrensesnitt.

Oppgave 2 (35%) Designprosessen

ISO 9241-11 definerer brukervennlighet/brukskvalitet (usability) som ”*anvendbarhet, effektivitet og subjektiv tilfredsstillelse for spesifikke brukere, med spesifikke mål, i spesifikke omgivelser*”.

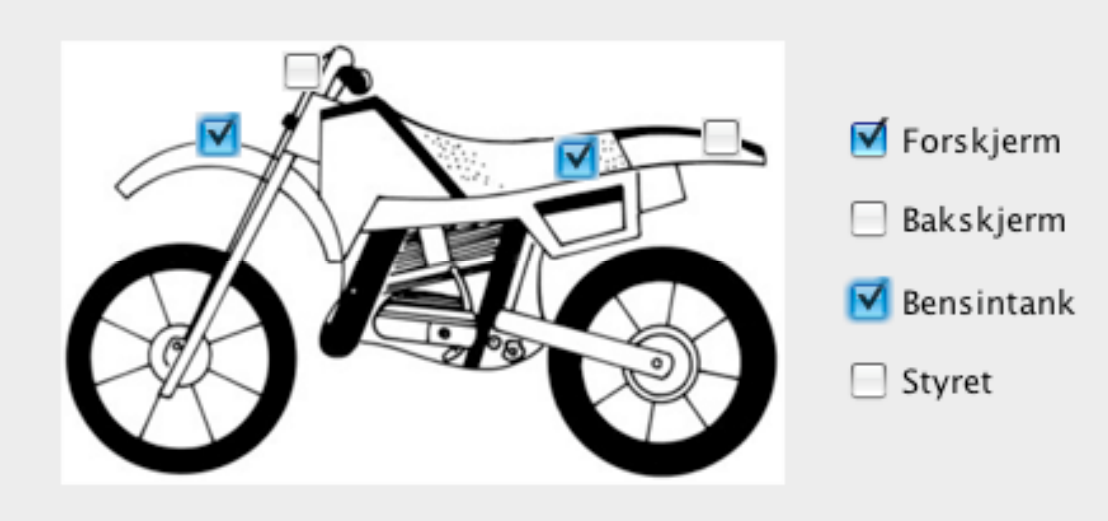
ISO 13407 beskriver en brukersentrert prosess bestående av fire faser i en sirkelbevegelse. Figuren under er hentet fra standarden og illustrerer dette.



a) Gi ett eksempel på en aktivitet/teknikk (noe man gjør) for hver av de fire fasene i figuren over.
b) Beskriv hvordan definisjonen av brukervennlighet i ISO 9241-11 kan brukes som rettesnor for hver av de fire aktivitetene du har valgt.

Oppgave 3 (40%) Grensesnittkonstruksjon

Et bil- og motorsykkelutleiefirma ønsker å lage et Java/Swing-basert skadeskjema for bil og motorsykkel. Denne oppgaven handler kun om skadeskjema for motorsykkel.



I figuren ser vi et eksempel på et skadeskjema for motorsykkel. Brukeren skal kunne krysse av for skade enten på selve motorsykkelen eller i listen til høyre. Kryssene (JCheckBox) skal i begge tilfeller oppdateres automatisk.

a) Hvordan vil du bruke Model-View-Controller (MVC) til å skille presentasjon og data i din løsning av skadeskjemaet? Forklar hovedtrekkene i din løsning i forhold til valg av modell/modeller.
b) Beskriv modellen/modellene i detalj.
c) Tegn opp hvilke instanser/objekter som eksisterer når skadeskjemaet kjører. Hva er deres relasjoner/referanser til hverandre? Bruk enkle firkanter for å angi instanser/objekter og piler for å angi relasjoner/referanser.
d) Tegn opp sekvensdiagrammet når skadeskjemaet startes opp. Forklar.
e) Tegn opp sekvensdiagrammet når brukeren klikker på forskjermen på figuren for å indikere at det er en skade og den tilsvarende sjekkboksen til høyre oppdateres. Forklar.
f) Det er ønskelig med et tekstfelt (JLabel) i tillegg som oppsummerer skadene tekstlig. For utfyllingen i eksempelet over ville det da stå: ”Du har rapportert skade på Forskjerm og Bensintank.”. Hvordan ville du implementere et slikt tillegg, og hvilke fordeler gir det at man valgte en MVC-arkitektur for implementasjonen.

Du trenger ikke forholde deg til selve figuren av motorsykkelen og hvordan denne tegnes ut.

Hint: Et JPanel kan godt være usynlig (samme farge og rammefarge som bakgrunnen) og brukes til å samle flere Swing-komponenter.

Appendiks: Relevante klasser, interface og tilhørende metoder

Det følgende er klippet og limt fra den offisielle dokumentasjonen på java.sun.com.

class PropertyChangeSupport

This is a utility class that can be used by objects that support bound properties. You can use an instance of this class as a variable in your object and delegate various work to it.

Methods:

public void addPropertyChangeListener(PropertyChangeListener listener)

Add a PropertyChangeListener to the listener list. The listener is registered for all properties.

Parameters:
listener - The PropertyChangeListener to be added

public void firePropertyChange(String propertyName,
Object oldValue,
Object newValue)

Report a bound property update to any registered listeners. No event is fired if old and new are equal and non-null.

Parameters:
propertyName - The name of the property that was changed.
oldValue - The old value of the property.
newValue - The new value of the property.

interface PropertyChangeListener

A "PropertyChange" event gets fired whenever an object changes a "bound" property. You can register a PropertyChangeListener with a source object so as to be notified of any bound property updates.

Methods:

public void propertyChange(PropertyChangeEvent evt)

This method gets called when a bound property is changed.

Parameters:
evt - A PropertyChangeEvent object describing the event source and the property that has changed.

class PropertyChangeEvent

A "PropertyChange" event gets delivered whenever an object changes a "bound" or "constrained" property. A PropertyChangeEvent object is sent as an argument to the PropertyChangeListener method.

Normally PropertyChangeEvents are accompanied by the name and the old and new value of the changed property. Null values may be provided for the old and the new values if their true values are not known.

An event source may send a null object as the name to indicate that an arbitrary set of if its properties have changed. In this case the old and new values should also be null.

Methods:

public String getPropertyName()

Gets the programmatic name of the property that was changed.

Returns:
The name of the property that was changed. May be null.

public Object getNewValue()

Gets the new value for the property, expressed as an Object.

Returns:
The new value for the property. May be null.

public Object getOldValue()

Gets the old value for the property, expressed as an Object.

Returns:
The old value for the property. May be null.

class JPanel

JPanel is a generic lightweight container.

Methods:

public Component add(Component comp)

Appends the specified component to the end of this container.

Class JCheckBox

An implementation of a check box -- an item that can be selected or deselected, and which displays its state to the user.

Methods:

public void addActionListener(ActionListener l)

Adds an ActionListener to the button.

Parameters:
l - the ActionListener to be added

public boolean isSelected()

Returns the state of the button. True if the toggle button is selected, false if it's not.

Returns:
true if the toggle button is selected, otherwise false

public void setSelected(boolean b)

Sets the state of the button. Note that this method does not trigger an actionEvent. Call doClick to perform a programatic action change.

Parameters:
b - true if the button should be selected, otherwise false

interface ActionListener

The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked.

Methods:

public void actionPerformed(ActionEvent e)

Invoked when an action occurs.

Class ActionEvent

A semantic event which indicates that a component-defined action occured. This high-level event is generated by a component (such as a Button) when the component-specific action occurs (such as being pressed). The event is passed to every ActionListener object that registered to receive such events using the component's addActionListener method.

The object that implements the ActionListener interface gets this ActionEvent when the event occurs. The listener is therefore spared the details of processing individual mouse movements and mouse clicks, and can instead process a "meaningful" (semantic) event like "button pressed".

Methods:

public Object getSource()

The object on which the Event initially occurred.

Returns:
The object on which the Event initially occurred.

MERK: Det er ikke nødvendig å benytte alle metoder/funksjoner.