NTNU
Det skapende universitet

Institutt for Datateknikk og Informasjonsvitenskap

Eksamensoppgave i TDT4180 Menneske-Maskin-Interaksjon (MMI)

Faglig kontakt under eksamen: Dag Svanæs

Tlf.: 91897536

Eksamensdato: 5. August

Eksamenstid (fra-til): 1500 - 1900

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

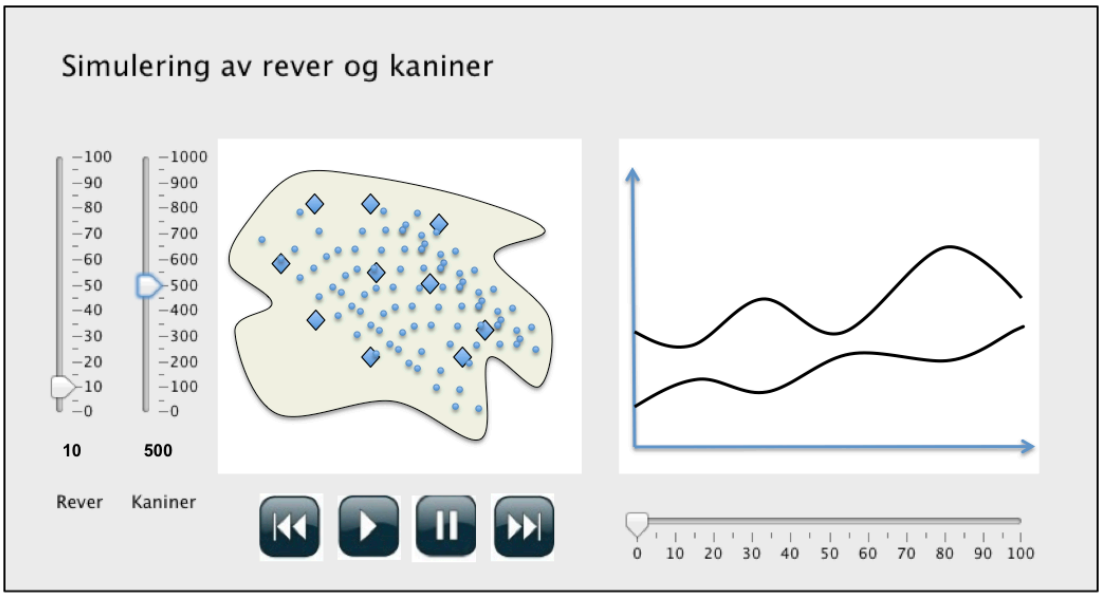
Annen informasjon:

Målform/språk: Bokmål

Antall sider: 9 (inklusive forsiden)

Antall sider vedlegg: 0

Oppgave 1. Grensesnittdesign (30%)

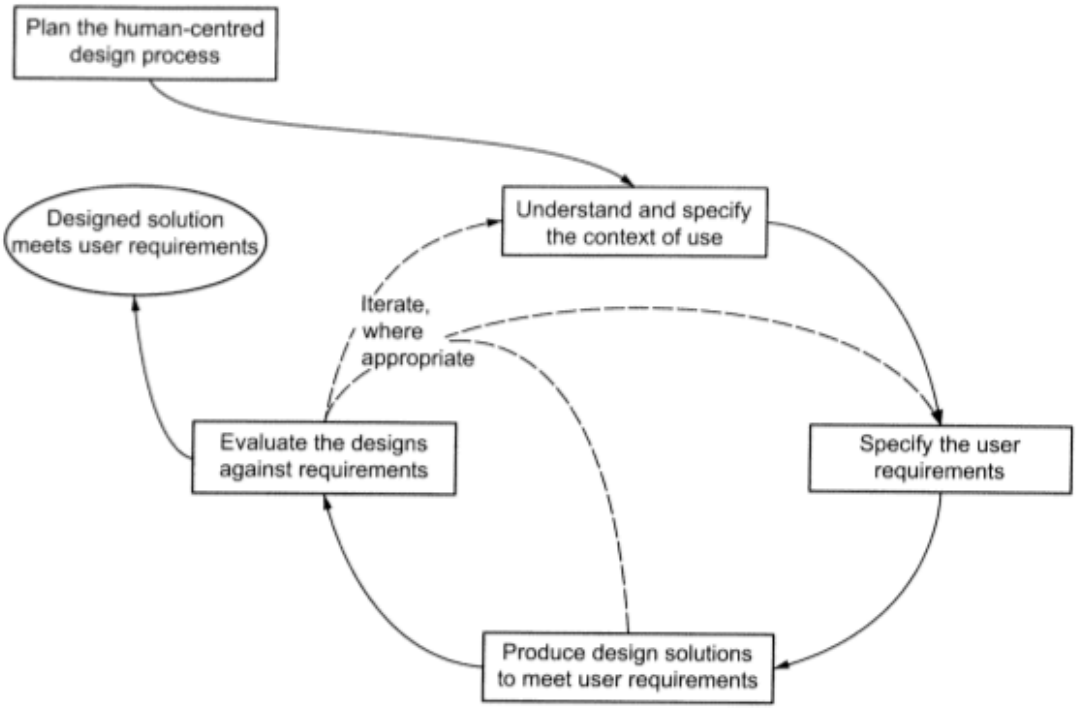


Figuren over viser simuleringsspillet POPULASJON til bruk i undervisning. Spillet gir brukeren mulighet til å teste ut hvordan to populasjoner av dyr (rever og kaniner) gjensidig påvirker hverandre over tid (0 til 100 uker). ”Rewind” knappen setter tiden tilbake til 0. ”Play” knappen starter en simulering. Den animeres, slik at ”Pause” knappen kan brukes til å stopp den. ”End” knappen flytter tidsmarkøren til uke 100. Både populasjoner og tid kan endres v.h.a. slidere. Populasjonene vises i de to JLabel objektene under JSlider objektene, men kan ikke endres her.

a) Hva menes med et visuelt gestalt, og hvorfor er gestaltteorien fra psykologien viktig i forhold til å utforme brukergrensesnitt.
b) Analyser skissen over i forhold til bruk av gestaltprinsipper.

Oppgave 2. Brukersentrert design (30%)

ISO standarden 9241-210 (2010) angir fasene i en brukersentrert systemutviklings-prosess som vist i figuren under:



a) Hva menes med begrepet ”context of use”, og hvorfor er dette et sentralt begrep i interaksjonsdesign?
b) Forklar denne figuren, og angi tre eksempler på brukersentrerte aktiviteter for hver av de fire fasene i iterasjonen.

Oppgave 3. Brukergrensesnittkonstruksjon (40%)

Skissen til programmet POPULASJON i oppgave 1 skal implementeres i JAVA/SWING. Du kan anta at det finnes en algoritme som beregner populasjonen av rever og kaniner i uken etter basert på verdiene av rever og kaniner.

a) Hvordan vil du bruke Model-View-Controller (MVC) til å skille presentasjon og data i din løsning av dette programmet?
b) Forklar hovedtrekkene i din løsning i forhold til valg av modell/modeller.
c) Beskriv modellen/modellene i detalj.
d) Vis klassediagrammet for løsningen din.
e) Tegn opp sekvensdiagrammet når det hele startes opp. Forklar.
f) Tegn opp sekvensdiagrammet når brukeren endrer verdien på antall rever. Forklar.
g) Tegn opp sekvensdiagrammet når brukeren trykke ”Play”. Forklar.
h) Tegn opp sekvensdiagrammet når brukeren endrer verdien på slideren for tid. Forklar.

Du trenger ikke forholde deg til layout eller hvordan komponentene tegnes ut.
Du trenger heller ikke forklare hvordan vinduene for uttegning av øya og uttegning av grafene skal implementeres. Du kan anta at disse implementeres som subklasser av JPanel, men du må angi hvilke metoder disse skal implementere.

For hver av komponentklassene så er relevante metoder og hjelpeklasser listet i appendikset bakerst. Merk: Det forlanges ikke at du skal anvende metoder eller klasser som ikke er listet i appendikset.

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 removePropertyChangeListener(PropertyChangeListener listener)

Remove a PropertyChangeListener from the listener list.
Parameters:
listener - The PropertyChangeListener to be removed

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 void addMouseListener(MouseListener l)

Adds the specified mouse listener to receive mouse events from this component.
Parameters:
l - the mouse listener

class JButton

An implementation of a push button

Methods:

public JButton(Icon icon)
Creates a button with an icon.
Parameters:
icon - the image that the button should display

public void addActionListener(ActionListener l)
Adds the specified action listener to receive action events from this JButton.

public void setIcon(Icon defaultIcon)
Sets the button's icon.

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.

class JLabel

A display area for a short text string. A label does not react to input events.

Methods:

public void setText(String text)

Defines the single line of text this component will display.

Class JSlider

A component that lets the user graphically select a value by sliding a knob within a bounded interval.

Methods:

public int getValue()

Returns the sliders value.

public void setValue(int n)

Sets the sliders current value.

public void addChangeListener(ChangeListener l)

Adds a ChangeListener to the slider.

Interface ChangeListener

Defines an object which listens for ChangeEvents.

public void stateChanged(ChangeEvent e)

Invoked when the target of the listener has changed its state.

Class ChangeEvent

ChangeEvent is used to notify interested parties that state has changed in the event source.

Methods:

public Object getSource()

The object on which the Event initially occurred.

Returns:
The object on which the Event initially occurred.