NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET
INSTITUTT FOR DATATEKNIKK OG INFORMASJONSVITENSKAP

Faglig kontakt under eksamen:
Hallvard Trætteberg, Tlf: 73 59 34 43

EKSAMEN I FAG
TDT4180 MMI

Onsdag 28. mai 2008
Tid: kl. 0900-1300

Bokmål

Sensuren faller 18. juni 2008

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

Oppgave 1 (30%) Grensesnittdesign

a) Jacob Nielsen har utarbeidet en 10 punkts liste for gode brukergrensesnitt. Hans første regel er ” Visibility of system status” (synlighet av systemets status). Hva menes med dette? Gi to eksempler.
b) Et annet av hans punkter er ”Recognition rather than recall” (gjenkjennelse heller enn å huske). Hva mener han med dette? Gi to eksempler.
c) Gestaltprinsippene fra psykologien sier noe om menneskelig persepsjon. Hvorfor er denne teorien viktig for design av grafiske brukergrensesnitt? Gi to eksempler.

Oppgave 2 (30%) Designprosessen

Du arbeider for et IT-firma som skal lage en ny løsning for håndholdte billettlesere for konduktører på tog. I det nye systemet så skal alle togbilletter ha strekkoder som kan leses av en integrert strekkodeleser i en håndholdt enhet.

Figuren over viser eksempel på en slik håndholdt enhet og en overfylt togvogn. Enheten har strekkodeleser integrert i toppen, trykkfølsom skjerm og noen ekstra knapper. Konduktørene skal kunne ha denne i lommen, og bruke den ved kontroll av billetter.

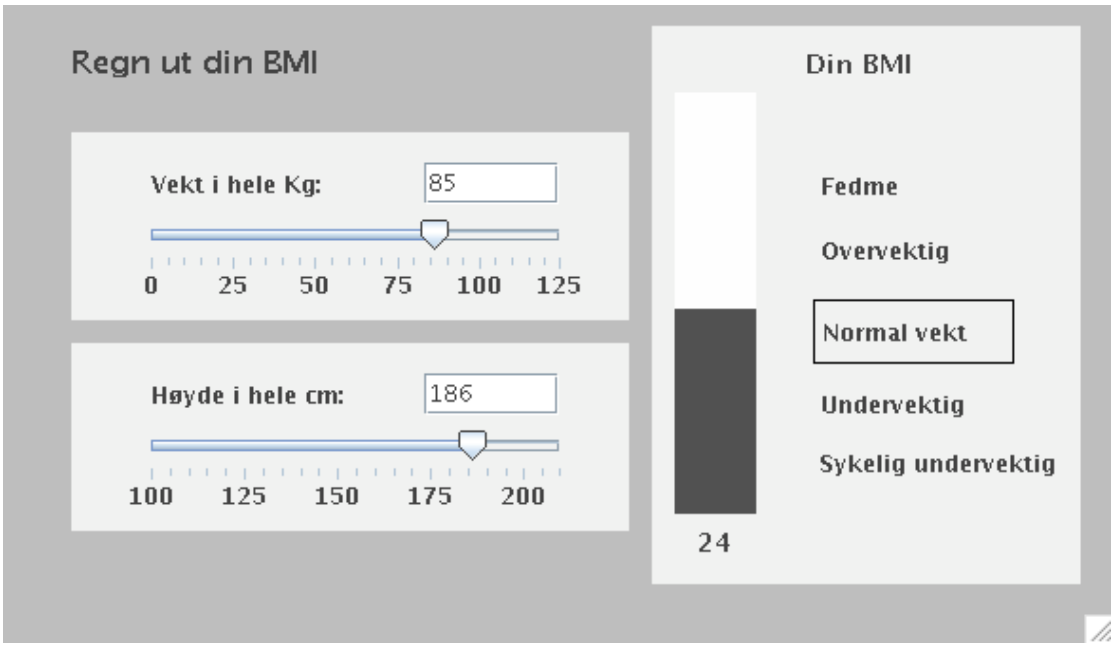
Første versjon av systemet skal leveres om et år. Du har fått i oppdrag å planlegge en siste brukbarhetstest av systemet som skal gjøres før systemet skal settes i drift hos togselskapet som har bestilt løsningen.

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

Bruk denne definisjonen som utgangspunkt for planleggingen av testen.

a) Hvilke konsekvenser har denne definisjonen av brukervennlighet generelt for hva brukbarhetstester skal måle, og for hvordan de skal gjennomføres?
b) Beskriv hvordan du vil forberede, gjennomføre og analysere brukbarhetstesten i dette konkrete tilfellet.

Oppgave 3 (40%) Grensesnittkonstruksjon



Du har fått i oppgave å implementere en Java/Swing applet for å regne ut BMI (Body Mass Index) på en webside for slanking. Bildet over viser hvordan BMI-kalkulatoren skal se ut. Brukeren skal kunne legge inn sin vekt og høyde og automatisk få regnet ut sin BMI.

For både vekt og høyde så kan brukeren enten dra i en skyvespak (”slider”) eller taste inn verdien direkte i et tekstfelt. I begge tilfelle vil skjermbildet oppdateres automatisk med de riktige verdiene.

Formelen for BMI er:

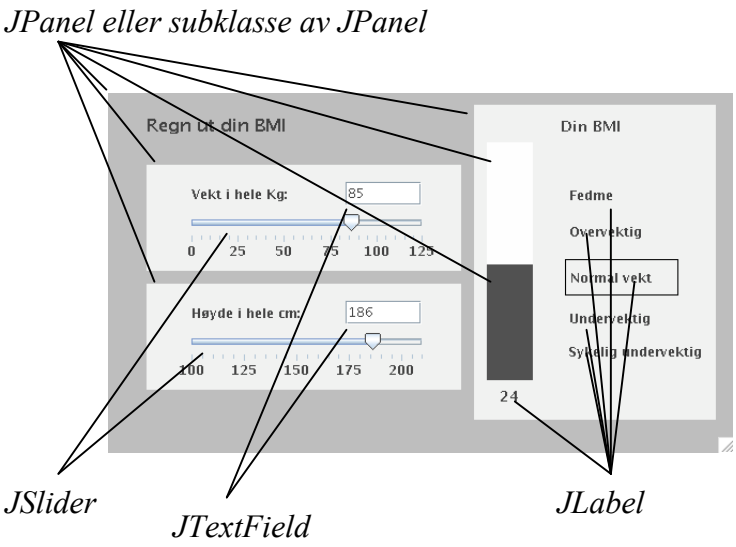
$$\text{BMI} = \frac{vekt\ (\text{kg})}{hoyde^2 (\text{m})}$$

Eksempel

$$19{,}6 = \frac{60\,(\text{kg})}{1{,}75^2\,(\text{m})}$$

* BMI på under 16,5 = sykelig undervektig
* BMI på mellom 16,5 og 18,5 = undervektig.
* BMI på mellom 18,5 og 24,9 = normal vekt.
* BMI på mellom 25 og 29,9 = overvektig.
* BMI på 30 og over = fedme.

Figuren under viser bruk av SWING-komponenter i implementasjonen. I tillegg til det som er angitt på figuren så er de faste tekstene implementert som *JLabel*.



Klassen *JTextField* behandler i utgangspunktet ikke tall, men konvertering mellom *String* og *int* kan gjøres enkelt i Java v.h.a innebygde metoder i klassen *Integer.* (Du trenger ikke forholde deg til feilsituasjoner ved at brukeren skriver inn tekst som ikke er gyldige heltall)

static String toString(int i)
Returns a String object representing the specified integer.

Eksempel: s = Integer.toString(i);

static int parseInt(String s)
Parses the string argument as a signed decimal integer.

Eksempel: i = Integer.parseInt(s);

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.**

Søylen til høyre er laget v.h.a. en JPanel som forandrer høyde. Angivelse av BMI-kategori (”Fedme”,,) gjøres v.h.a. skifte av rammefarge rundt teksten. Du trenger ikke gå i detalj om hvordan disse grensesnittelementene implementeres.

a) Hvordan vil du bruke Model-View-Controller til å skille presentasjon og data i din løsning av BMI-kalkulatoren? Forklar hovedtrekkene i din løsning i forhold til valg av modell/modeller.
b) Tegn opp hvilke instanser/objekter som eksisterer når BMI-kalkulatoren kjører. Hva er deres relasjoner/referanser til hverandre? Bruk enkle firkanter for å angi instanser/objekter og piler for å angi relasjoner/referanser.
c) Tegn opp sekvensdiagrammet når BMI-kalkulatoren startes opp. Forklar.
d) Tegn opp sekvensdiagrammet når brukeren legger inn en ny verdi i tekstfeltet for vekt. Forklar.

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 JTextField

JTextField is a lightweight component that allows the editing of a single line of text.

Methods:

public String getText()

Returns the text contained in this JTextField

public void setText(String t)

Sets the text of this JTextField to the specified text.

public void addActionListener(ActionListener l)

Adds the specified action listener to receive action events from this JTextField.

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 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 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.

MERK: Det er ikke nødvendig å benytte ActionEvent eller ChangeEvent i implementasjonen.