NORGES TEKNISK-
NATURVITENSKAPELIGE UNIVERSITET
INSTITUTT FOR DATATEKNIKK OG INFORMASJONSVITENSKAP

Faglig kontakt under eksamen:
Letizia Jaccheri
Tlf: 73593469 / 91897028

EKSAMEN I FAG SIF 8056 PROGRAMVAREARKITEKTUR
Tirsdag 29. mai 2001
Tid: kl. 0900-1300

Sensuren faller i uke 25

Hjelpemidler: A3 kalkulator ikke tillat. Alle trykte og håndskrevne hjelpemidler er tillatt.

Vektleggingen av oppgavene er indikert med prosent (veiledende). Under en oppgave teller alle deloppgavene likt.

Oppgave 1 (25%)

Diskuter relasjonen mellom kvalitetsattributtet brukervennlighet og programvarearkitekturen for et system. Angi også her eksempler, gjerne fra eCourse.

Oppgave 2 (25%)

Oppgaven gjelder følgende fire kvalitetsattributter: vedlikeholdbarhet (maintainability), ytelse (performance), gjenbruk (reuse) og sikkerhet (security).

- Hvilke av disse kan observeres mens systemet kjøres?
- Angi hvordan de fire forskjellige kvalitetsattributtene kan observeres, og hvis mulig hvilke kvantitative mål man kan gi for hver av dem.
- Forklar hvorfor en god beskrivelse av programvarearkitektur er viktig for å tilfredstille de kvalitetsatributtene som ikke er observerbare når systemet kjøres.

Oppgave 3 (50%)

I slutten av dokumentet, er en fragment fra en java servlet spesifikasjon gitt (fra eCourse kildekoden):

Besvar følgende spørsmål, og motiver dine svar:

1. eCourse er organisert etter model view controller arkitekturmønstret. Er FrontServlet en del av model, view eller controller?
2. Er FrontServlet en abstrakt eller konkret klasse? Hva er oppgave til denne klassen?
3. Gjenskap (reverse engineer) UML diagrammet som beskriver sammenhengen mellom FrontServlet og de andre klassene i koden over. For hver av klassene, beskriv om de hører til model, view eller controller.
4. FrontServlet er konstruert ut fra Façade-mønsteret. Beskriv fordelene og ulempene med dette mønsteret, og angi eksempler, helst fra eCourse systemet.
5. Foreslå en alternativ arkitektur for dette fragmentet som overvinner i det minste en av disse ulempene. Hvilke andre fordeler/ulemper har din arkitektur?

Exercise 1 (25%)

What is the relationship between the ”usability” quality attribute and the software architecture of a system. Give your definition of “usability” and of “software architecture”. Motivate with examples, preferably from the eCourse project.

Exercise 2 (25%)

Consider the following quality attributes: maintainability, performance, reuse, and security. Which of these attributes are observable at run time? Explain why a good software architecture description is crucial to achieve the quality attributes, which are not observable at run time, and explain why.

Exercise 3 (50%)

At the end of this document, there is a java servlet specification, which is extracted from eCourse source code.
Answer to the following questions and motivate your answers.

1. ECourse is organized according to the Model View Controller architectural pattern. Is FrontServlet part of the Model, View, or Controller?
2. Is FrontServlet an abstract or a concrete class? What is the purpose of this class?
3. Reconstruct (reverse engineering) the UML diagram describing the connections between FrontServlet and other classes in the system. Describe only the connections and the classes which can be extracted from this code fragment as you did not have access to eCourse documentation. For each of the classes above, describe if it is part of the Model, View, or Controller part.
4. FrontServlet is designed according to the Façade pattern. Describe the advantages and disadvantages of this pattern and give examples preferably from the eCourse system.
5. Provide one alternative architecture for this fragment, which overcomes at least one of the disadvantages above. Which are other advantages and disadvantages of your architecture?

```
public abstract class FrontServlet extends HttpServlet {
	protected ControllerManager controllerManager;
	protected EventGenerator eventGenerator;

	public void init() throws ServletException {
		controllerManager = initControllerManager();
		eventGenerator = initEventGenerator();
	}

	public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
		handleRequest(request, response);
	}

	public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
		PostRequestParser postRequestParser = new
PostRequestParser(tempFileDirectory, maxFileSize);
		postRequestParser.parametersToAttributes(request);
		handleRequest(request, response);
	}

	protected void handleRequest(HttpServletRequest request,
HttpServletResponse response) {
		try {
			HttpSession session = request.getSession(true);

			EApplicationEvent event =
eventGenerator.generateEvent(request);
			String nextUrl = sendEventToController(event,
session);
			updateSession(session,event);
			sendResponse(nextUrl, request, response);
		}
		catch (Exception e) {
			e.printStackTrace();
		}
	}

	protected String sendEventToController(EApplicationEvent event,
HttpSession session) throws Exception {
	ControllerInterface controllerInterface =
		controllerManager.getController(session);

	controllerInterface.handleEvent(event);
	}

	protected void sendResponse(String nextUrl, HttpServletRequest
request, HttpServletResponse response) throws Exception {
		request.getRequestDispatcher(nextUrl).forward(request,
response);
	}

	protected void updateSession(HttpSession session,
EApplicationEvent event) {}

protected abstract EventGenerator initEventGenerator();

	protected abstract ControllerManager initControllerManager();
}
```