Vårt system kan kjùres ved å skrive

STUD1 konto fredo 37 (holdeplass)

Holdeplass er frivillig. Dersom man kun sender linjenr finner systemet den nñrmeste holdeplassen. Systemet returnerer de 3 neste avgangene uavhengig av retning.

1. a. Collaboration diagram

De endringer vi har gjort i Collaboration diagrammet er at vi har lagt til elementene smsInput, smsoutput, dynoutput , dyninput . DynOutput og dynInput kommuniserer med trafikanten. Bruker kommuniserer med systemet via smsInput og smsOutput.

1 b. Composite structure diagram

I det nye kompositt struktur har vi gjort en del endringer. Vi har lagt til inputMediator og outputMediator, inn til og ut fra hvert av de elementene. Brukersesjon har blitt viktigere i prosjekt 3, og ikke System som var mye viktigere i prosjekt 2 enn nå. Dette er fordi System bare skal ta imot meldinger fra Bruker via PATS og videre sender melding til Brukersesjon. BrukerSesjon tar i mot forespùrsel fra System og lagrer all informasjon, deretter kontakter BrukerSesjon de andre elementene for å hente de nùdvendige informasjon. System skal i utgangspunktet kunne håndtere flere brukere dersom det er flere brukere om gangen, men pga. tidsmessige grunner har begrenset oss til at en bruker sender inn en forespùrsel om gangen. Men uansett har vi en mulighet i systemet vårt at System kan håndtere flere brukere om gangen dersom man senere har tid til å utvide programmet / systemet vårt. BrukerSesjon er nå hovedelement i systemet i prosjekt 3, siden alle forespùrsler skal gis til BrukerSesjon fùrst også sende BrukerSesjon hver av forespùrsel videre til de andre elementene.
RuteDynamisk sender en forespùrsel til Trafikanten og mottar informasjon tilbake fra dem. Etter at BrukerSesjon har fått tilbake melding fra de elementene, sender BrukerSesjon svar til System som igjen sender meldingen til PATS, og PATS sender svaret til brukeren.

Vi har ºfjernetº RuteStatisk i JavaFrame implementasjonen siden vi fant ut at det ikke var mulig for oss å få tak i informasjon på ruteStatisk fraTrafikanten.

2. a. Tilstandsmaskiner

Vi har også forandret på våre tilstandsdiagrammer. Fùrste tilstand er VentSMS som får sms fra Bruker og sender den videre til VentBrukersesjon. Neste tilstandsdiagram, Brukersesjon håndteres så forespùrselen fùr den sender svar tilbake og brukeren får den informasjonen han vil ha.

I tilstandsmaskinen Brukersesjon setter vi fùrst variablene i VentSjekkSyntaks. Hvis linjenummer er 37 så går den enten videre til tilstanden VentFinnPosisjon for å finne posisjonen dersom posisjonen brukeren bare har tastet inn linjenr. Hvis brukeren allerede har nevnt en holdeplass i meldingen, sjekker den holdeplass ved hjelp av tilstanden VentSjekkHoldeplass. Etter å ha sjekket holdeplass, og denne eksister, går den videre til VentFinnNeste3Avganger for å finne neste tre avganger. Hvis brukerens ùnskede holdeplass ikke eksisterer sender programmet en melding tilbake om dette. Etter å ha funnet nñrmeste holdeplass går en også til tilstanden VentFinnNeste3avganger (Dersom det skulle oppstå en feil vil det genereres en beskjed til brukeren).

3. a. Implementasjon

Vi brukte Java 1.5 for implementasjon av programmet trafikanten pluss. Vi har også brukt noe kode fra eksempel programmet lñreren har gitt oss.
I programmet vårt har vi ikke implementert ruteStatisk, da denne informasjonen automatisk vil bli funnet ved kall til Trafikanten.

3. b. Dokumentasjon på databasedesign

Vi lastet ned classes12.zip fra oracle sin hjemmeside, hvor det inneholdt pakker som oracle.sql, javax.swing, oracle.jdbc.driver.
Disse pakkene brukes blant annet til for å kunne laste ned Oracle JDBC driveren, og oppkopling mot databasen.

Vi brukte Oracle som relasjonsdatabasehåndteringssystem SQL for å utfùre spùrringer.
Vi måtte gjùre oppkopling mot databasen to ganger. Den ene gangen for å finne nñrmeste holdeplass, og den andre gangen var sùk for å finne stasjons ID.