Ohjelmistoarkkitehtuurit
Loppukoe syksy 2010

Tehtävistä 1-5 saa max 6 pistettä, tehtävästä 6 max 4 pistettä. Ei oheiskirjallisuutta.

1. Asiakastiedon hallintajärjestelmässä on Client- ja Account-komponentteja. Samalla asiakkaalla voi olla useita tilejä, ja sama tili voi olla usean asiakkaan hallinnassa. Kun jonkin tilin kattosaldo ylittyy, tilin Client-komponentit asetetaan tarkkailutilaan, ja kaikille ko. asiakkaan tileille tulee nostoesto. Lisäksi halutaan varautua siihen, että järjestelmää voidaan helposti laajentaa siten, että asiakkaan tilan muuttumisesta voidaan informoida myös muita komponentteja, joiden rajapintaa ei voida muuttaa. Anna UML-komponenttikaavio tarvittavista suunnitteluratkaisuista rajapintoineen. Käytä Tarkkailija- ja Sovitin-suunnittelumalleja, ja merkitse missä kohtaa kaaviota niitä on käytetty.

2. Henkivakuutusten hallintajärjestelmä on tarkoitettu asiakaspisteessä toimivalle virkailijalle. Se sisältää mm. seuraavat toiminnallisuudet: asiakas- ja vakuutustietojen ylläpito, asiakaskorttien tunnistus, vakuutusmaksujen ennakkolaskenta, asiakkaiden ja vakuutusten tilastointiraportit, käyttäjien hallinnointi. Kaikki vakuutustoimintaan liittyvä tieto on talletettu tietokantaan. Anna järjestelmän kerrosarkkitehtuuri. Sijoita kerroksiin seuraavat komponentit tai alijärjestelmät:

   AWT (Javan yleinen grafiikkatuki), **InsBase** (tietokannan vakuutusabstraktiokerros), **DB** (tietokanta), **DBMS** (tietokannan hallintajärjestelmä), **InsLogic** (yleinen tuki vakuutussovelluksille), **InsCard** (kortinlukijan abstraktio), **CardReader** (kortinlukijan ajuri), **LifeInsLogic** (henkivakuutusten logiikkaosa), **InsGraphics** (yleinen graafinen tuki vakuutussovelluksille), **LifeInsGUI** (järjestelmän GUI), **Swing** (Javan GUI-kirjasto), **InsReporter** (yleinen tuki vakuutustietojen raportoinnille), ja **LifeReporter** (henkivakuutusten raportointikomponentti) ja **Linux** (käyttöjärjestelmä).

3. Selitä lyhyesti, mitkä ovat tietovuoarkkitehtuurin edut ja potentiaaliset ongelmat.

4. Kirjoita lyhyt (n. 1-2 käsikirjoitussivua) essee aiheesta "Ketterä arkkitehtuurin arviointi". Rakenna essee seuraavan jaottelun mukaan (kustakin kohdasta noin 1 kappale tekstiä):
   - ATAMin yleinen luonne ja tarkoitus
   - ATAM-prosessi
   - miten ATAM-prosessia voisi keventää (ajan- ja resurssien käyttö)
   - mitä ongelmia keventämisestä voisi olla

5. Yritys tekee tietylle sovellusalueelle järjestelmiä, joihin sisältyy skriptausmahdollisuus: järjestelmien osatoimintoja on mahdollista räätälöidä kuvaamalla ne skriptikielellä käyttäjän toimesta. Skripteissä tehdään tyypillisesti pientä aritmetiikkaa ja kuvataan tietty toimintalogiikka, jonka ohjaamana kutsutaan yrityksen ko. sovellusalueelle tarjoaman tuoterunkoalustan yksittäisiä palveluja. Skripteissä käytetään myös nimettyjä järjestelmämuuttujia, joiden kautta päästään käsiksi järjestelmän tietosisältöön ja tilaan. Toimintalogiikka kuvataan yksinkertaisilla ohjausrakenteilla (ehdollisuus ja toisto). Alla on kuvattu eräässä yrityksen tuotteessa käytetty skriptikieli.

KÄÄNNÄ

```
Script ::= "SCRIPT" Statement
Statement ::= identifier "=" Expression |
    "IF"  Expression "DO" Statement |    // looginen ehto kuvataan aritm. lausekkeella
    "WHILE" Expression "DO" Statement |    // tosi: arvo >0, epätosi: arvo <=0
    "CALL" PlatformServiceCall |
    "[" DeclarationList StatementList "]"
Expression ::= Expression Op Expression  | Primary
Op ::= "+" | "-" | "*" | "/"
Primary ::= integer | float | VariableReference
VariableReference ::= "WATER_LEVEL" | "COFFEE_LEVEL"
PlatformServiceCall ::= "START_WARMING" | "STOP_WARMING" | "WAIT" Expression
DeclarationList ::= {Declaration ";"}
Declaration ::= "DECL" identifier
StatementList ::= Statement {";" Statement}   // { } tarkoittaa iteraatiota 0 tai enemmän kertaa
```

Yritys päättää rakentaa tuoterunkoalustan näille tuotteille. Osa tuota alustaa tulee olemaan kehys, jonka avulla voidaan helposti kehittää tuotteen tarvitsema skriptin tulkinnan toteutus omana komponenttinaan, joka kutsuu alustan API:ssa olevia palveluja skriptin mukaisesti. Voidaan olettaa, että komponentti saa skriptin olioesityksen kääntäjältä, joka lukee skriptitekstin (tähän ei tarvitse kiinnittää tässä huomiota). Skriptikielen muunneltavuus-vaatimukset ovat:

(1) Kieleen sisältyy tuotteesta riippuva joukko järjestelmämuuttujia (PlatformVariable) ja alustapalveluja (PlatformService). Viimeksi mainittuihin voidaan liittää palvelusta riippuen joukko parametreja, jotka ovat aritmeettisia lausekkeita (esim. yllä WAIT-palvelu).

(2) Kieleen voidaan liittää uuden tyyppisiä tuotteesta riippuvia ohjauslauseita perusohjaus-rakenteiden (yllä olevat ehto- ja toistolauseet) lisäksi.

(3) Kieleen voidaan lisätä uuden tyyppisiä tuotteesta riippuvia aritmeettisia (binäärisiä) operaatioita neljän perusoperaation lisäksi.

Tehtävä: Suunnittele kehys, jonka erikoistuksena saadaan uudelle tuotteelle sen tarvitsema skriptikielen tulkintakomponentti, ja anna kehyksen arkkitehtuuri UML luokkakaaviona. Kehys perustuu Tulkki-suunnittelumallin soveltamiseen skriptin olioesitykseen. Merkitse kaavioon näkyviin variaatiopisteet kuvaamalla, mihin kohtiin voi tulla tuotekohtaisia luokkia. Merkitse myös näkyviin rajapinnat, joiden välityksellä tulkki kommunikoi alustan kanssa, ja selitä miten rajapintoja käytetään.