Ohjelmistojen mallintaminen, syksy 2009, laskuharjoitus 4

1. Toisissa laskareissa oli tehtävänä laatia käyttötapausmalli Ohjelmoinnin harjoitustyön tuntikirjanpitoa tukevaan järjestelmään.

   Etsi luokkakandidaatit luennollakin esitettyä käsiteanalyysiä käyttäen, eli kerää tehtäväkuvauksessa esiintyvät substantiivit. Karsi sen jälkeen ylimääräiset (attribuutit, synonyymit ja asiaan kuulumattomat).

2. Tee alustava kohdealueen eli määrittelyvaiheen luokkakaavio tuntikirjanpitojärjestelmästä. Etene esim. samaan tapaan kuin luennon kampaamoesimerkissä.

3. Seuraavalla sivulla on annettu pieni Java-ohjelma. Takaisinmallinnetaan ohjelmakoodi UML:ksi. Eli piirrä ohjelmaa kuvaava luokkakaavio ja sekvenssikaavio, joka kuvaa Room-luokan main-metodin suoritusta.

   Huom: main()-metodi on luokan Room stattinen metodi, eli se suoritetaan "luokkatasolla". Sekvenssikaaviossa kannattaakin mallintaa omana laatikkona Room-luokka (joka siis suorittaa main():in) ja Room-olio, jonka Room-luokka luo ja jonka metodeja sensori ja lämmittimet käyttävät.

4. Palaamme toisista laskareista tutun LAL-järjestelmän pariin.

   Tehtävänä on etsiä LAL:in luokkakandidaatit samaan tapaan kuin tehtävässä 1. Eli kerää substantiivit ja karsi ylimääräiset.

5. Tehdään alustava kohdealueen luokkakaavio osalle LAL-järjestelmää. Keskitytään vain neljään ensimmäiseen tekstikappaleeseen (eli *Palkkaamme palvelukseemme työntekijöitä ...* ei tule enää mukaan) ja otetaan luokkakaavioon niiden joukosta löytyviä asioita.

   Ota edellisen tehtävän luokkalistasi ja karsi niistä ne luokat, jotka kuuluvat nyt tarkasteltavan osan ulkopuolelle. Mieti jäljelle jääneiden luokkien yhteyksiä sekä attribuutteja.

   Älä pyri täydelliseen ratkaisuun. Tärkeintä on, että ongelmakenttä rupeaa pikkuhiljaa jäsentymään.

```
public class Room {
      private int temperature = 0;
      int getTemperature() { return temperature; }
      void addTemperature(int t) { temperature = temperature + t; }
      public static void main(String[] args) {
            Room r = new Room();
            Thermostat t = new Thermostat(r);
            t.stabilizeAt(20);
      }
}

class Thermostat {
      private Sensor sensor;
      private Heater heaterA;
      private Heater heaterB;
      Thermostat(Room r) {
            sensor = new Sensor(r);
            heaterA = new Heater(r);
            heaterB = new Heater(r);
      }
      void stabilizeAt(int degrees) {
            while (sensor.getReading() < degrees) {
                  heaterA.heatTheRoom();
                  heaterB.heatTheRoom();
            }
      }
}

class Sensor {
      private Room room;
      Sensor(Room r) { room = r; }
      int getReading() { return room.getTemperature(); }
}

class Heater {
      private Room room;
      Heater(Room r) { room = r; }
      void heatTheRoom() { room.addTemperature(5); }
}
```