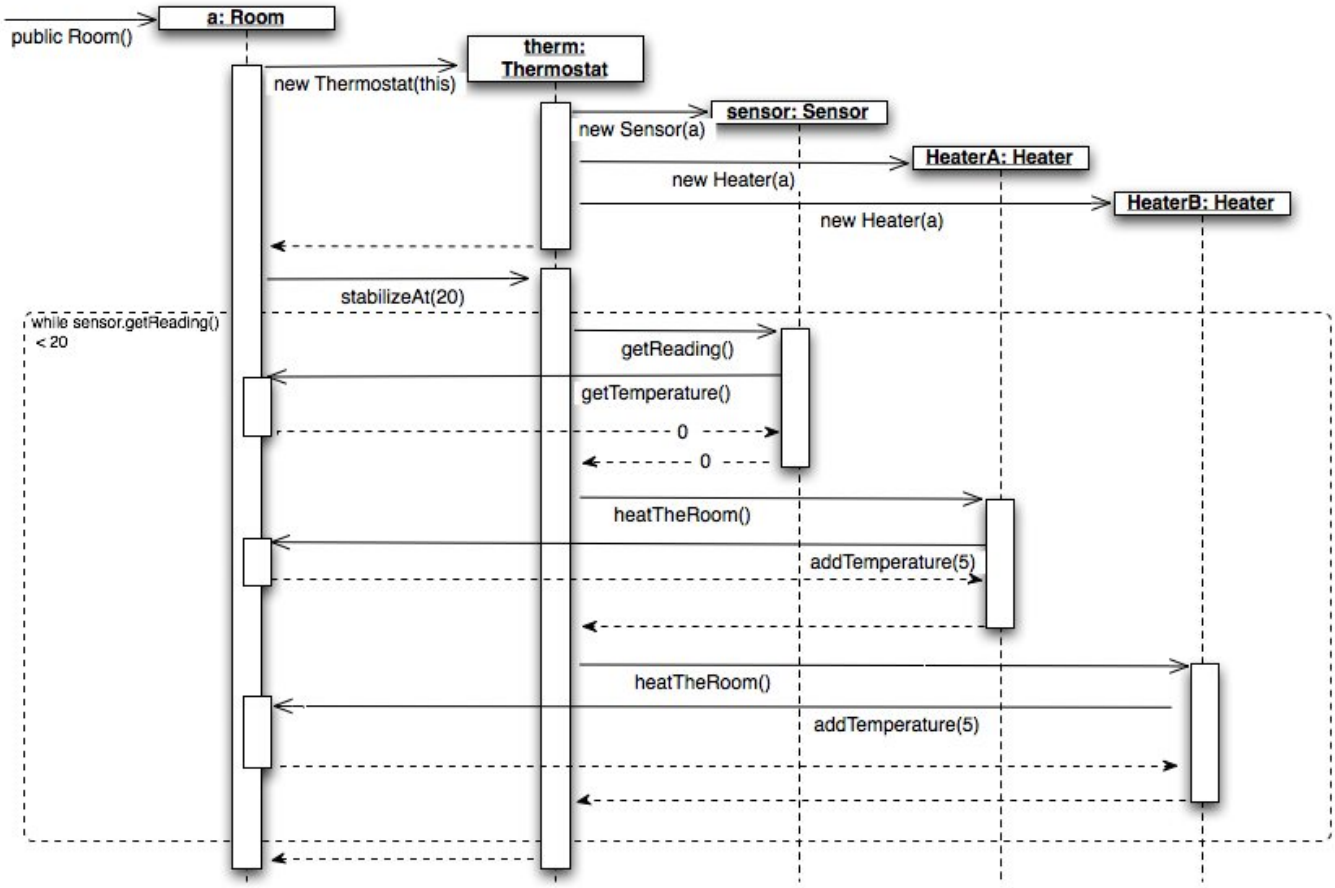
Ratkaisut harjoitusviikon 5 tehtäviin

Jarmo Vestola, Tommi Voss, Kari Piukka, Jonne Kohvakka, Perttu Määttä

Helsingin yliopisto
Tietojenkäsittelytieteen laitos
Ohjelmistotekniikan menetelmät -kurssi
Kevät 2007
Ohjaaja Jari Suominen
Harjoitusryhmä 1
Opintopiiri 3

Tehtävä 2

Pienestä Java-ohjelmasta tehty sekvenssikaavio, joka kuvaa Room-luokan konstruktorin suoritusta. Luodut oliot ovat laatikoissa ylärivillä ja niiden olemassaoloa kuvaa ajan kanssa alaspäin etenevä katkoviiva, paksu palkki kertoo olion aktiivisuuden. Nuoliin liittyvä teksti kuvaa metodin tai konstruktorin kutsua parametreineen ja katkoviivalla merkitty iso laatikko kuvaa silmukan suorituksen ehtoa.



Tehtävä 3

Eclipsellä tehty ja JUnitilla ajettu yksikkötestiluokan ***testitapaukset*** koodi:.

```
import junit.framework.TestCase;
import java.util.Date;

public class LALTest extends TestCase {


          public void testaaEriAikoja() { //MaksettuHinta kaikissa 20.0
                    // 60 tuntia
                    assertEquals(4.0, LAL.palautus(20.0, new Date(2007, 4, 20, 11,
0), new Date(2007, 4, 22, 23, 0)), 0.01);
                    // 11 pv = 264 h
                    assertEquals(20.0, LAL.palautus(20.0, new Date(2007, 4, 11, 0,
0), new Date(2007, 4, 22, 0, 0)), 0.01);
                    // 10 pv = 240 h
                    assertEquals(20.0, LAL.palautus(20.0, new Date(2007, 4, 12, 0,
0), new Date(2007, 4, 22, 0, 0)), 0.01);
                    // 239 h 59 min 59 s
```

```
                        assertEquals(18.0, LAL.palautus(20.0, new Date(2007, 4, 12, 16,
0, 0), new Date(2007, 4, 22, 15, 59, 59)), 0.01);
                        // 23 h 59 min 59 s
                        assertEquals(0.0, LAL.palautus(20.0, new Date(2007, 4, 21, 16,
0, 0), new Date(2007, 4, 22, 15, 59, 59)), 0.01);
            }

            public void testaaEriHintoja() { //Peruutushetki kaikissa 24 h ennen
lähtöhetkeä
                        // laskettu summa alle 0,5 euroa
                        assertEquals(0.0, LAL.palautus(4.99, new Date(2007, 4, 21, 16,
0), new Date(2007, 4, 22, 16, 0)), 0.01);
                        // laskettu summa yli 0,5 euroa
                        assertEquals(0.5, LAL.palautus(5.0, new Date(2007, 4, 21, 16,
0), new Date(2007, 4, 22, 16, 0)), 0.01);
            }

            public void testaaPoikkeukset() {
                        try { //hinta negatiivinen
                                    assertEquals(0.5, LAL.palautus(-5.0, new Date(2007,
4, 21, 16, 0), new Date(2007, 4, 22, 16, 0)), 0.01);
                        } catch(IllegalArgumentException e) { }

                        try { // peruutushetki lähtöhetken jälkeen
                                    assertEquals(0.0, LAL.palautus(50.0, new Date(2007,
4, 22, 16, 0), new Date(2007, 4, 20, 16, 0)), 0.01);
                        } catch(IllegalArgumentException e) { }
            }

}
```

Tehtävä 4

LAL-tietojärjestelmän ***Palautus-metodin täydennys***-koodi:

```
import java.util.Date;

public class LAL {

            /**
             * Tämä metodi laskee, paljonko perutusta lipusta palautetaan rahaa.
             * Palautettava määrä on 10 prosenttia lipusta maksetusta hinnasta kutakin
             * vuoron lähtöön jäljellä olevaa täyttä vuorokautta kohden. Alle 50
sentin
             * summia ei kuitenkaan palauteta.
             *
             * Esimerkki: lipusta maksettiin 20 euroa ja lippu peruttiin 60 tuntia
ennen
             * lähtöä. Lipusta maksetaan takaisin 0,2 * 20 euroa = 4 euroa
             *
             * @param maksettuHinta                    lipusta ostohetkellä maksettu
hinta euroina
             * @param peruutushetki                    aika, jolloin peruutus tehtiin
             * @param lähtöhetki            peruttuun lippuun liittyvän vuoron
lähtöaika
             * @return
            asiakkaalla palautettava summa euroina
             * @throws IllegalArgumentException jos maksettu hinta on negatiivinen tai
perumishetki on lähtöhetken jälkeen
             */

            public static double palautus(double maksettuHinta,Date peruutushetki,
Date lähtöhetki)
            throws IllegalArgumentException {
```

```
                        long millisekunnit = lähtöhetki.getTime() -
peruutushetki.getTime();
                        int tunnit = (int)(millisekunnit / (1000 * 60 * 60)); //
Peruutushetken ja lähtöajan välinen aika tunneissa
                        double palautetaan;

                        //Jos maksettu hinta negatiivinen tai peruutushetki lähtöhetken
jälkeen niin heitetään poikkeus
                        if(maksettuHinta < 0.0 || tunnit < 0.0)
                                    throw new IllegalArgumentException();

                        //Palautetaan 0, jos palautushetki on alle vuorokauden
                        else if(tunnit < 24)
                                    return 0.0;
                        //Ei palauteta maksettua hintaa suurempaa summaa
                        else if(tunnit >= 24*10)
                                    return maksettuHinta;
                        else
                                    palautetaan = maksettuHinta * 0.1 * (tunnit / 24);

                        //Ei palauteta alle 0,5 euron summia
                        if(palautetaan < 0.5)
                                    return 0.0;

                        return palautetaan;

//                      return -1.0;
            }

}
```