Arvosteluperusteet Java-ohjelmointi-kurssikokeen 11.12.2006 tehtävään 1

Arvosteluperusteet laati: Jukka Stenlund

Arvostelussa pyrittiin painottamaan asioita, jotka olivat tulleet Java-ohjelmointi -kurssilla uusina asioina ja joita siis ei oltu käsitelty vielä Ohjelmoinnin perusteet -kurssilla. Niinpä vastauksista, joissa oltiin osattu peruskurssin asiat mutta ei Java-ohjelmointi -kurssin asioita, ei juuri annettu pisteitä.

Yleisesti ottaen vastaukset arvosteltiin siten, että lähtökohtana pidettiin täysiä pisteitä ja virheistä sakotettiin. Jois kuitenkin pariudu-metodi oli täysin oikein, annettiin vastauksesta vähintään neljä pistettä, vaikka muista virheistä kertyneet miinuspisteet olisivat vieneet pisteet alle neljän. Myös muista ansioista saattoi saada jonkun pisteen, vaikka virhesakot olisivat nollanneet pisteet.

Java-ohjelmointi -kurssilla opittuna uutuutena tehtävässä oli luokkakohtaisten (static) ja oliokohtaisten (ei-static) piirteiden ero: luokkaan oli ohjelmoitava luokkakohtainen laskurikenttä, jolla pidettiin kirjaa käytetyistä yksilöllisyysnumeroista sekä oliokohtainen identiteettikenttä, jolle oliota luotaessa annettiin yksilöllisyyslaskurin mukainen numero. Koska luokkaan tarvittiin vielä perimäkenttä, tuli luokkaan määritellä seuraavat kolme kenttää (nimet sai toki keksiä itse):

```
private static int laskuri=0;
private final int identiteetti;
private String perimä;
```

Identiteetti-kenttä tuli varustaa final-määreellä, koska tehtävänannossa erikseen painotettiin, että identiteetti ei voi koskaan muuttua.

Jos luokkaan oli kirjoitettu vain yksi int-tyyppinen kenttä, vähennettiin tästä kuusi pistettä. Seuraavassa luettelossa mainituista virheistä vähennettiin pisteitä ainoastaan, mikäli luokkaan oli kirjoitettu erilliset int-kentät laskuria ja yksittäisen olion identiteettiä varten. Näistä virheistä vähennettiin kuitenkin yhteensä korkeintaan kuusi pistettä:

- vikaa static/ei-static -erottelussa: -4
- final-määre puuttuu identiteettikentästä: -2
- final-määreellä varustetun kentän arvoa muutetaan koodissa: -3
- luotava olio ei saa yksilöllisyysnumeroa: -3
- yksilöllisyyslaskuria ei kasvateta olioiden luomisen myötä: -2
- yksilöllisyyslaskurille ei aseteta alkuarvoa: -1

Jos luokassa oli vain yksi int-kenttä yksilöllisyysmekanismia varten, ei edellä lueteltuja mahdollisia virheitä ollut mielekästä tutkia koodista erikseen.

Tehtävänannossa sanottiin suoraan, että kenttien on oltava näkyvyydeltään private. Mikäli kentille oltiin tästä huolimatta annettu joitakin muita näkyvyyksiä, sakotettiin tästä kolme pistettä, ellei näitä oltu selitetty vastauksessa.

Jos vastaaja oli ymmärtänyt tehtävänannon siten, että perimä on String-taulukko, joka arvotaan, ja että jälkeläisen perimä on yhdistelmä vanhempien perimistä, sakotettiin tästä neljä pistettä.

Jos pariudu-metodissa oli jotain perustavasti pielessä, sakotettiin neljä pistettä.

Kolme pistettä menetti seuraavista virheistä:

- ymmärretty aivan oikein, että perimä on yksittäinen String (ei taulukko), mutta jälkeläisen perimä muodostetaan vastoin tehtävänantoa vanhempien perimien yhdistelmänä: -3

- olion perimä-kentän arvo jää asettamatta: -3
- ei palauteta mitään pariudu-metodista tai toString-metodista: -3
- jokin kenttä, jonka olisi tarkoitus liittyä olioon, määritellään konstruktorin koodissa, jolloin siitä tulee konstruktorin sisäinen muuttuja: -3
- syntaksiltaan virheellinen lause String [] uusiPerimä = new String; -3
- lauseesta, jolla luodaan olio, on jäänyt new-ilmaus pois: -3

Seuraavista virheistä menetti kaksi pistettä:

- mate-metodi muuten oikein, mutta muutamassa kohdassa koodia lukee Genome kun pitäisi lukea Germ (englanninkielinen vastaus): -2
- kaikista metodikutsuista puuttuu kutsuttavan metodin nimen perään kuuluvat sulut: -2
- koodissa on yksittäinen syntaksiltaan virheellinen rivi, jonka merkitys ei auennut tarkastajalle: -2
- syntaksiltaan virheellinen rivi this.perimä=String perimä; -2
- toString-metodissa ei lainkaan käytetä kentää, jonka arvo olisi pitänyt liittää osaksi palautettavaa merkkijonoa: -2
- verrataan totuusarvoa kokonaislukuun (Javassa totuusarvoja ei voi käsitellä kokonaislukujen tapaan): -2
- koodissa kopioidaan String-olio taulukon tapaan käyttämällä syntaksia, joka toimii vain taulukoille: -2
- määritellään muttuja if-lauseen haarassa, vaikkei haara ole lohkossa (esimerkki alla): -2

```
if (Math.random()<0.5)
      Basilisko lapsi=new Basilisko(toinen.perimä);
else
      Basilisko lapsi=new Basilisko(this.perimä);
```

- määritellään muuttuja if-lauseen haaralohkossa ja yritetään palauttaa muuttujan arvo if-lauseen jälkeen, jolloin muuttujaan ei enää voida viitata, koska ollaan poistuttu lohkosta, jossa se on määritelty (esimerkki alla): -2

```
if (Math.random()<0.5) {
      Basilisko lapsi=new Basilisko(toinen.perimä);
}
else {
      Basilisko lapsi=new Basilisko(this.perimä);
}
return lapsi; //ei onnistu!
```

Seuraavista virheistä menetti yhden pisteen:

- useammasta kuin yhdestä metodikustusta puuttuu sulut (ei kuitenkaan kaikista): -1
- sisennys tosi pahasti pielessä: -1
- muuttujan tyyppi (int tai String) puuttuu muuttujan esittelystä: -1 (ei enempää, koska vaikuttaa lähinää huolimattomuusvirheeltä)
- metodissa käytetään muuttujaa, jota ei ole esitelty: -1
- toString-metodissa syntaksivirhe katenaatiolausekkeessa: -1
- importoidaan metodi: -1
- final-määre väärässä kohtaa (private int final): -1
- lainausmerkit muuttujan ympärillä: -1
- ensin aivan oikein new Basilisko(this.perimä), mutta sitten virheellisesti new Basilisko(toinen), kun olisi pitänyt olla new Basilisko(toinen.perimä) tai new Basilisko(toinen.mikäOnPerimä()), varmaankin huolimattomuusvirhe: -1

- pariudu-metodissa väärin muotoiltu (ei mene kääntäjästä läpi) random-ehto: -1
- toString-metodin katenaatiolause jotenkin virheellinen, mutta lause sisältää kuitenkin oikeat muuttujat: -1

Turhaa koodia, ei kuitenkaan virheellistä (ei sakotettu):

- toString-metodissa katenoidaan loppuun "" turhan päiten
- on varauduttu String-olion muuttumiseen, vaikka String-oliot ovat muuttumattomia

Tyylittömyyksiä, joista ei sakotettu:

- staattiseen kenttään viittaaminen this-viitteen avulla

Muita huomioita:

Joissakin vastauksissa pariudu-metodissa viitattiin this-olion perimään käyttämällä perimä-kenttää, kun taas parametriolion perimään viitattiin kutsumalla parametriolion mikäOnPerimä-metodia. Tästä tulee vaikutelma, että kirjoittaja luulisi, että private-näkyvyys rajoittaa näkyvyyden olion sisään, kun tosiasiassa kenttä on näkyvissä myös muihin olioihin, jotka ovat ilmentymiä samasta luokasta.

Esimerkkiratkaisu

Esimerkiksi seuraavasti tehtävän saattoi ratkaista täysin pistein.

```
public class Basilisko {
  private static int laskuri = 1;
  private final int identiteetti;
  private String perimä;

  public Basilisko(String perimä) {
    identiteetti = laskuri;
    laskuri++;
    this.perimä = perimä;
  }

  public Basilisko pariudu(Basilisko toinen) {
    String lapsenPerimä;
    if (Math.random() < 0.5)
      lapsenPerimä = this.perimä;
    else
      lapsenPerimä = toinen.perimä;
    return new Basilisko(lapsenPerimä);
  }

  public String mikäOnPerimä() {
    return this.perimä;
  }

  public String toString() {
    return "(" + identiteetti + "): " + perimä;
  }
}
```