T-106.1203 Ohjelmoinnin perusteet L:n tentti

Tenttipäivä: 24.10.2005, laatija: Juha Sorva

Huom! Tämä on L-peruskurssin tentti. Kurssin T-106.1200 Ohjelmoinnin perusteet T opiskelijoille järjestetään erilliset välikokeet!

Tentin arvosanarajat määritetään vasta tentin jälkeen. Tenttitulokset julkaistaan kuukauden sisällä tenttipäivästä kurssin WWW-sivuilla, ja oman suorituksen arvosteluun voi tutustua erillisessä tilaisuudessa, jonka aika ja paikka tullaan myös ilmoittamaan kurssisivuilla.

Tehtävä 1 (6 pistettä / 24)

Liitteessä 1 on annettu luokan `Henkilo` Java-ohjelmakoodi. Tutustu siihen huolellisesti. Luokan dokumentaatio on "puutteellista", mutta koodi on sentään Java-kieliopin mukaista (siinä ei siis ole sellaisia virheitä, joita normaali kääntäjä ei päästäisi läpi). Liitteen koodirivit on numeroitu tenttitehtävien muotoilemisen ja niihin vastaamisen kätevöittämiseksi.

Vastaa alla lueteltuihin kohtiin. Kuhunkin kohtaan tarvitaan vain varsin lyhyt vastaus, korkeintaan muutama lause ellei toisin ole mainittu. Kunkin kohdan pistearvo on yksi piste.

a) Rivit 6-9 ovat esimerkki eräänlaisesta Java-rakenteesta. Mistä rakenteesta on kyse ja mihin tällaisia rakenteita käytetään?

b) Esitä perusteltu mielipiteesi väitteestä:
"*Luokka* `Henkilo` *on erään tietotyypin (*data type*) määrittely.*"

c) Rivillä 4 on lähiomaisten käsittelyä varten kenttä (ilmentymämuuttuja; *field, instance variable*) `private Henkilo lahiomainen`. Esitä perusteltu mielipiteesi väitteestä: *"On arveluttavaa määritellä luokalle tähän tapaan kenttä siten, että kentän tyyppinä on juuri samainen luokka itse."*

d) Luokassa `Henkilo` on yksi sellainen virhe, joka estää osittain luokan mielekkään käytön. Missä tämä virhe on ja miten sen voisi korjata?

Tehtävä jatkuu toisella puolella paperia...

e) Suoritetaan liitteestä 1 löytyvä käynnistysmetodi (pääohjelmametodi; *main method*). Luettele kaikki ne henkilöluokan metodien ja konstruktorien kutsut, jotka tämän ohjelman suorituksen aikana tehdään. Luettele kutsut siinä järjestyksessä, missä niiden suorittaminen aloitetaan. Kirjaa kukin kutsu siten, että vastauksesta ilmenee yksiselitteisesti rivinumero, jolla kutsu tehdään sekä kutsuttu metodi tai konstruktori. Yksi luettelon kohta voisi siis olla vaikkapa:

Rivi 35: kutsutaan metodia kerroNimi

Jos jotakin metodia tai konstruktoria kutsutaan useita kertoja, täytyy sen vastaavasti esiintyä luettelossasi useita kertoja.

f) Esitä perusteltu mielipiteesi väitteestä:
*"Ohjelma voi kaatua rivillä 12 syntyvään poikkeukseen (*exception*), jos* `this`*-arvona on tyhjä olioarvo* `null`*. Toisin sanoen: kaatuminen on mahdollista jos* `this == null`*."*

Tehtävä 2 (6 pistettä / 24)

Liitteessä 2 on annettu luokan `Jono` Java-ohjelmakoodi. Tutustu siihen huolellisesti. Koodi on edellisen tehtävän tapaan Javan kieliopin mukaista. Vastaa alla lueteltuihin kohtiin.

taulukko

a) Kuvaile lyhyesti, millainen on juuri luodun `Jono`-olion `sisalto`-kentän arvo. varattu null arvoja Jono jono = new Jono(10)

b) Mikä on suurin mahdollinen arvo, jonka jonon kenttä `pituus` voi saada? Perustele lyhyesti, miksi ei ole mahdollista, että arvo kasvaa tätä suuremmaksi.

c) Kuvaile lyhyesti se logiikka, jolla jonottavat henkilöt varastoidaan `sisalto`-taulukon eri indekseille. Eli missä päin taulukkoa on jonossa ensimmäisenä oleva henkilö jne.?

d) Selosta lyhyesti, mikä on metodin `maaritaIndeksi` merkitys luokassa `Jono`?

e) Suoritetaan liitteestä 2 löytyvä käynnistysmetodi (pääohjelmametodi; *main method*). Mitä tulostuu?

f) Ideoi jokin tapa, jolla luokasta `Jono` voitaisiin laatia yleishyödyllisempi tekemällä siitä parametrisoitu tyyppi (*generic type, parameterized type*). Luettele lyhyesti, mitä muutoksia luokan ohjelmakoodiin pitäisi tällöin tehdä?

rajapintaluokat

Tehtävä 3 (6 pistettä / 24)

Liitteessä 3 on annettu luokkien `Ominaisuus` ja `OminaisuusVaihtoehdoin` ohjelmakoodi. Tutustu siihen huolellisesti. Koodi on edellisen tehtävän tapaan Javan kieliopin mukaista. Vastaa seuraaviin kohtiin.

a) Luokassa `OminaisuusVaihtoehdoin` on nimeltään ja parametreiltaan samanlainen `kelpaaArvoksi`-metodi kuin yliluokassa `Ominaisuus`. Millä termillä tätä ilmiötä voidaan kuvata? (Suomenkielisestä termistä täydet 1p, englanninkielisellä vain ½p.)

b) Esitä perusteltu mielipiteesi väitteestä:
"Koska luokalla `Ominaisuus` on aliluokk(i)a, on `Ominaisuus`-tyyppisen muuttujan arvon staattinen tyyppi (käännösaikainen tyyppi; compile-time type*) eri kuin sen dynaaminen tyyppi (ajonaikainen tyyppi;* runtime type*)."*

c) Luokassa `Ominaisuus` on kaksi erilaista konstruktoria. Millä termillä tätä ilmiötä voidaan kuvata? (Suomenkielisestä termistä täydet 1p, englanninkielisellä vain ½p.)

d) Luokan `Ominaisuus` metodi `toString` on toteutettu melko kummallisesti `while`-silmukkaa käyttäen. Laadi metodille järkevämpi, helpommin ymmärrettävä toteutus (joka kuitenkin saa aikaan saman asian kuin annettu versio). Vastaukseksi riitää uuden toteutuksen Java-koodi.

e) Etsi `Ominaisuus`-luokasta huonosti tehty ohjelmointiratkaisu ja perustele lyhyesti miksi se on vähintäänkin arveluttava. Koeta kiinnittää perustelussasi huomiota siihen, mitä ongelmia tästä ratkaisusta voisi syntyä juuri näiden esimerkkiluokkien kontekstissa. (Edellisessä kohdassa mainittu `toString`-metodin outo toteutustapa ei kelpaa vastaukseksi tähän kohtaan.)

f) Suoritetaan liitteestä 3 löytyvä käynnistysmetodi (pääohjelmametodi; *main method*). Luettele kaikki ne `Ominaisuus`- ja `OminaisuusVaihtoehdoin`-luokkien metodien ja konstruktorien kutsut, jotka tämän ohjelman suorituksen aikana tehdään. Kirjaa kutsut samaan tapaan kuin tehtävän 1 e-kohdassa .

Vika tehtävä löytyy toiselta puolelta paperia...

Tehtävä 4 (6 pistettä / 24)

Liitteessä 4 on annettu luokan `Tapahtuma` Java-ohjelmakoodi. Tutustu siihen huolellisesti. Koodi on jälleen Javan kieliopin mukaista. Vastaa alla lueteltuihin kohtiin.

a) Kirjoita metodin `kerro~~Ominaisuus~~` (Tieto) julkisesta osasta dokumentaatio. Toisin sanoen: kirjoita lyhyt kuvaus, josta selviävät metodin käytön kannalta oleelliset asiat. Jos osaat laatia Javadoc-dokumentaatiokommentteja, voit muotoilla vastauksesi sellaiseksi. Muu vapaamuotoinen kuvaus kelpaa kyllä vastaukseksi aivan yhtä hyvin.

b) Toteuta metodi `ilmoita` siten, että se lisää annetun henkilön tapahtumaan ilmoittautuneisiin henkilöihin, ellei maksimiosallistujamäärä ylittyisi. Jos tilaa ei ole, lisätään henkilö sen sijaan jonoon. Kuitenkin jos jonossakaan ei ole tilaa, ei ilmoittaminen onnistu, mistä merkiksi palautetaan `false`. Muissa tapauksissa palautetaan `true`.
(Muistutus: Listojen metodilla `add(uusiAlkio)` voi lisätä uuden alkion listaan ja metodi `size()` palauttaa listan koon `int`-lukuna.)

c) Keksi metodille `teeJotain` uusi nimi, joka kuvaa paremmin sitä, mitä se tekee. (Huom. listojen metodi `toArray` kopioi listan sisällön annettuun taulukkoon.)

d) Perustele lyhyesti, miksi rivin 46 silmukan jatkamisehdossa on mielekästä lukea `taulukko.length - 1` eikä vain `taulukko.length`.

e) Annetussa `Tapahtuma`-luokan toteutuksessa tapahtuman ominaisuuksista ("tiedoista") pidetään kirjaa taulukossa . Esitä jokin toinen tapa pitää kirjaa näistä ominaisuuksista ja perustele lyhyesti, miksi se voisi olla taulukon käyttöä parempi ratkaisu.

f) Annetussa `Tapahtuma`-luokan toteutuksessa tapahtumaan onnistuneesti ilmoittautuneista henkilöistä pidetään kirjaa listassa. Esitä jokin toinen tapa pitää kirjaa ilmoittautuneista ja perustele lyhyesti, miksi se voisi olla listan käyttöä parempi ratkaisu.

Muista vastata kurssipalautekyselyyn kurssisivuilla ennen marraskuun loppua! Kyselyyn vastaamisesta saa 200 harjoitustehtäväpistettä.

LIITE 1

```
 1: public class Henkilo {
 2:
 3:   private String nimi;
 4:   private Henkilo lahiomainen;
 5:
 6:   public Henkilo(String henkilonNimi, Henkilo henkilonLahiomainen) {
 7:     String nimi = henkilonNimi;
 8:     this.asetaLahiomainen(henkilonLahiomainen);
 9:   }
10:
11:   public String kerroNimi() {
12:     return this.nimi;
13:   }
14:
15:   public void asetaLahiomainen(Henkilo uusiOmainen) {
16:     this.lahiomainen = uusiOmainen;
17:   }
18:
19:   public Henkilo kerroLahiomainen() {
20:     return this.lahiomainen;
21:   }
22:
23:   public String kerroLahiomaisenNimi() {
24:     return this.kerroLahiomainen().kerroNimi();
25:   }
26:
27: }
28:
29:
30:
31:
32: public class Tehtava1E {
33:   public static void main(String[] komentoriviparametrit) {
34:     Henkilo heppu = new Henkilo("Max", null);
35:     System.out.println(heppu.kerroNimi());
36:     heppu.asetaLahiomainen(new Henkilo("Moritz", heppu));
37:     System.out.println(heppu.kerroLahiomaisenNimi());
38:   }
39: }
```

nim = null

jos yritetään pehku.kerroNimi eikä pehku olisi luotu ole luotu

Henkilö pehku = new Henkilö

kääntäjä ei päästäisi tätä läpi

pehku olisi tuntematon muuttuja: pehku = null

return null tai olio

null

kutsutaan konstruktoria

lahiomainen = new Henkilo("Moritz", heppu)

```
 1:
 2: /* Luokka Jono kuvaa henkilöjonoja. Jonon perään voidaan
 3:  * lisätä henkilöitä ja toisaalta jonosta poistetaan
 4:  * henkilöitä alkupäästä lähtien. Kullakin jonolla on
 5:  * tietty enimmäispituus. */
 6:
 7: public class Jono<Object>
 8:
 9:   private int pituus;
10:   private int alkuindeksi;
11:   private Henkilo[] sisalto;
12:
13:   public Jono(int enimmaiskoko) {
14:     this.sisalto = new Henkilo[enimmaiskoko];
15:     this.pituus = 0;
16:     this.alkuindeksi = 0;
17:   }
18:
19:   public boolean onTaysi() {
20:     return this.pituus == this.sisalto.length;
21:   }
22:
23:   public boolean onTyhja() {
24:     return this.pituus == 0;
25:   }
26:
27:   public boolean lisaaPeraan(Henkilo uusi) {
28:     if (this.onTaysi()) {
29:       return false;
30:     } else {
31:       int uudenIndeksi = this.maaritaIndeksi(this.pituus);
32:       this.sisalto[uudenIndeksi] = uusi;
33:       this.pituus++;
34:       return true;
35:     }
36:   }
37:
38:   public Henkilo poistaAlusta() {
39:     if (this.onTyhja()) {
40:       return null;
41:     } else {
42:       Henkilo poistettu = this.sisalto[this.alkuindeksi];
43:       this.sisalto[this.alkuindeksi] = null;
44:       this.alkuindeksi = this.maaritaIndeksi(1);
45:       this.pituus--;
46:       return poistettu;
47:     }
48:   }
49:
50:   private int maaritaIndeksi(int paikkaJonossa) {
51:     return (this.alkuindeksi + paikkaJonossa) % this.sisalto.length;
52:   }
53:
54: }
55:
56:
57:
58:
59:
60:
61:
62:
63:
64:
65:
66:
67:
```

```
68: public class Tehtava2E {
69:   public static void m(String[] komentoriviparametrit) {
70:     Jono jono = new J(3);
71:
72:     int monesko = 1;
73:     do {
74:       String nimi = "Heppu#" + monesko;
75:       Henkilo jonottaja = new Henkilo(nimi, null);
76:       if (jono.lisaaPeraan(jonottaja)) {
77:         System.out.println("Lisättiin: " + nimi);
78:       } else {
79:         System.out.println("Ei onnistunut!");
80:       }
81:     } while (!jono.onTyhja());
82:
83:     System.out.println("Loppu!");
84:   }
85: }
```

Lisättiin: Heppu#1
Lisättiin: Heppu#1
lisättiin: Heppu#1
"Ei onnistunut!"
"Ei onnistunut!"
"Ei onnistunut!"

LIITE 6

LIITE 3

```java
/* Luokka Ominaisuus kuvaa nimettyjä ominaisuuksia,
 * joilla on jokin merkkijonoarvo (esim. nimi: "väri", arvo: "vihreä").
 * Näillä ominaisuuksilla voi arvona olla mikä tahansa merkkijono
 * tai null-arvo.
 */
public class Ominaisuus {

  private String nimi;
  public String arvo;

  public Ominaisuus(String nimi, String alkuarvo) {
    this.nimi = nimi;
    this.arvo = alkuarvo;
  }

  public Ominaisuus(String nimi) {
    this(nimi, null);
  }

  public String kerroNimi() {
    return this.nimi;
  }

  /* Tällä metodilla voi ennen arvo-kenttään sijoittamista
   * tarkastaa, kelpaako annettu merkkijono tämän ominaisuuden
   * arvoksi.
   */
  public boolean kelpaaArvoksi(String kandidaatti) {
    return true; // kaikki jonot kelpaavat tavallisen
                 // ominaisuuden arvoiksi
  }

  public String toString() {
    String arvonKuvaus = "";
    boolean teeKuvaus = (this.arvo != null);
    while (teeKuvaus) {
      arvonKuvaus = this.arvo;
      teeKuvaus = false;
    }
    return this.nimi + ": " + arvonKuvaus;
  }
}
```

```java
/* OminaisuusVaihtoehdoin-luokka kuvaa sellaisia
 * ominaisuuksia, joiden arvona voi olla vain jokin tietyistä,
 * ominaisuuskohtaisista merkkijonoarvoista (tai null-arvo).
 */
public class OminaisuusVaihtoehdoin extends Ominaisuus {

  private String[] vaihtoehdot;

  public OminaisuusVaihtoehdoin(String nimi,
                                String alkuarvo,
                                String[] vaihtoehdot) {
    super(nimi);
    this.vaihtoehdot = vaihtoehdot;
    this.asetaArvo(alkuarvo);
  }

  public boolean kelpaaArvoksi(String kandidaatti) {
    if (kandidaatti == null) {
      return true; // ominaisuudella saa olla null-arvo
    }

    for (int indeksi = 0; indeksi < this.vaihtoehdot.length; indeksi++) {
      if (this.vaihtoehdot[indeksi].equals(kandidaatti)) {
        return true;
      }
    }
    return false;
  }

  public void asetaArvo(String uusiArvo) {
    if (this.kelpaaArvoksi(uusiArvo)) {
      this.arvo = uusiArvo;
    }
  }
}
```

```java
public class Tehtava3F {
  public static void main(String[] komentoriviparametrit) {
    String[] vaihtoehdot = { "nainen", "mies" };
    Ominaisuus[] ominaisuudet = new Ominaisuus[3];
    ominaisuudet[0] = new OminaisuusVaihtoehdoin("sukupuoli", "mies", vaihtoehdot);
    ominaisuudet[1] = new Ominaisuus("väri", "vihreä");
    ominaisuudet[2] = new Ominaisuus("luonne", "iloinen");
    for (int indeksi = 0; indeksi < ominaisuudet.length; indeksi++) {
      System.out.println(ominaisuudet[indeksi]);
    }
  }
}
```

Liite 4

```
 1: import java.util.List;
 2: import java.util.ArrayList;
 3:
 4: /* Luokka Tapahtuma kuvaa tapahtumia, joihin voi ilmoittautua.
 5:  * Kustakin tapahtumasta pidetään kirjaa toisaalta siitä,
 6:  * ketkä siihen ovat ilmoittautuneet, toisaalta siitä,
 7:  * ketkä ovat jonossa täyteen varatun tapahtuman
 8:  * peruutuspaikoille. Tapahtuman osallistujamäärällä
 9:  * ja jonottajien määrällä on ylärajat, jotka määrätään
10:  * konstruktoriparametreilla. Tapahtumaan liittyy
11:  * lisäksi erilaisia lisätietoja, joita kuvataan
12:  * Ominaisuus-olioina.
13:  */
14: public class Tapahtuma {
15:
16:   private Ominaisuus[] tiedot;
17:   private List<Henkilo> ilmoittautuneet;
18:   private int osallistujienEnimmaismaara;
19:   private Jono halukkaat;
20:
21:   public Tapahtuma(Ominaisuus[] tiedot, int osallistujienEnimmaismaara,
22:                    int jonottajienEnimmaismaara) {
23:     this.tiedot = tiedot;
24:     this.osallistujienEnimmaismaara = osallistujienEnimmaismaara;
25:     this.halukkaat = new Jono(jonottajienEnimmaismaara);
26:     this.ilmoittautuneet = new ArrayList<Henkilo>();
27:   }
28:
29:
30:   public Ominaisuus kerroTieto(String nimi) {
31:     for (int indeksi = 0; indeksi < this.tiedot.length; indeksi++) {
32:       if (nimi.equals(this.tiedot[indeksi].kerroNimi())) {
33:         return this.tiedot[indeksi];
34:       }
35:     }
36:     return null;
37:   }
38:
39:   public boolean ilmoita(Henkilo ilmoittautuja) {
40:     return false; // Toistaiseksi toteuttamatta!
41:   }
42:
43:   public Henkilo[] teeJotain() {
44:     Henkilo[] taulukko = new Henkilo[this.ilmoittautuneet.size()];
45:     this.ilmoittautuneet.toArray(taulukko);
46:     for (int indeksi1 = 0; indeksi1 < taulukko.length - 1; indeksi1++) {
47:       for (int indeksi2 = indeksi1 + 1; indeksi2 < taulukko.length; indeksi2++) {
48:         Henkilo arvo1 = taulukko[indeksi1];
49:         Henkilo arvo2 = taulukko[indeksi2];
50:         if (arvo1.kerroNimi().compareTo(arvo2.kerroNimi()) > 0) {
51:           taulukko[indeksi1] = arvo2;
52:           taulukko[indeksi2] = arvo1;
53:         }
54:       }
55:     }
56:     return taulukko;
57:   }
58: }
59:
60:
```

vaihtelaiset tukodmat ArrayList, linked List

parametrisoitu tyyppi

luonne suhteinen väri ihminen mies vihreä String nimi String arvo

tapahtumaan liittyvän tiedon nimi

palauttaa taulukon

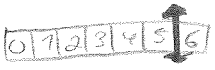
ilmoittautuneiden määrä

kopioi ilmoittautuneet henkilöt taulukkoon taulukko

nimi < n

nimi1 > nimi2

aakkosjärjestyksessä ensin



lengt = 7
lengt - 1 = 6

koska toinen for
~~Silmukka taulukon~~
Silmukan alkuarvo indeksi2 = indeksi1+1

kun on päästy taulukko.lengt - 1
indeksiin on jäljellä enää yksi