T-106.213 Ohjelmoinnin peruskurssi L1:n tentti

Tenttipäivä: 09.12.2003, laatija: Juha Sorva

Kirjoita kunkin vastauspaperisi alkuun kurssitunnus, päiväys, oma nimesi, opiskelijanumerosi (selkeästi) ja allekirjoituksesi. Jos olet suorittanut kurssin projektityön jo ennen syyslukukautta 2003, merkitse myös kyseinen suoritusajankohta ensimmäisen vastauspaperin yläosaan.

Eri tehtäville tässä tehtäväpaperissa lueteltuja pistearvoja saatetaan vielä muuttaa arvostelun yhteydessä. 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 (12 pistettä/100)

Kirjoita lyhyehkö selostus siitä, miten oliot ja luokat eroavat toisistaan olio-ohjelmoinnissa. Voit kuvitella, että kirjoitat selostuksen olio-ohjelmointia aloittelevalle henkilölle, jonka päässä pyörivät peruskysymykset kuten "Luokat ilmeisesti on jotain kuvauksia, joita kirjoitetaan tuonne ohjelmakoodiin, mutta mitä ne oliot sitten ovat?".

Keskeisten asioiden tuominen esiin tiivistetysti riittää. Selvästi yli puolen sivun vastaus keskivertokäsialalla saattaa olla jo tarpeettoman pitkä.

Arvosteluperusteina käytetään ensisijaisesti esitettyjen asioiden oikeellisuutta ja oleellisuutta - toissijaisesti myös sitä, miten selkeästi asiat on esitetty.

Tehtävä 2 (20 pistettä/100)

Lue seuraavat väittämät huolellisesti ja kerro ovatko ne totta vai tarua. Anna kussakin kohdassa myös lyhyt perustelu/selitys vastauksellesi. Perustelun toivottu pituus on yksi virke per kohta - tarkoitus ei ole antaa tyhjentävää selontekoa kustakin aiheesta vaan lähinnä osoittaa, että et ole vastannut vain arvaamalla. Jos keksit väitteen todeksi näyttävän esimerkin tai sen taruksi osoittavan vastaesimerkin, niin esimerkki riittää perusteluksi.

Kustakin kohdasta saa oikealla vastauksella neljä pistettä. Jos perustelu puuttuu tai ei ole hyväksyttävä, jäävät pisteet kyseisestä kohdasta saamatta.

a)

Olkoon käytössä muuttujat `a` ja `b`. Seuraava Java-ohjelmanpätkä vaihtaa näiden muuttujien arvot keskenään.

```
a = b;
b = a;
```

b)

Ohjelmoija voi määritellä Java-kieltä käyttäen itse omia tietotyyppejä (engl. data type).

c)

Kenttämäärittely, jossa kentän tyyppinä on juuri se luokka, jossa kyseinen kenttä on määritelty on Java-kielessä kelvollinen ja voi olla käytännössä mielekäs. (Esimerkki tämänkaltaisesta määrittelystä annettu alla.)

```
public class X {
  private X kentta;
  // jne. muita kenttiä ja metodeita
}
```

d)

Oletetaan, että luokka L toteuttaa rajapintaluokan (engl. interface) R. Tämä tarkoittaa muun muassa sitä, että kun luokan L ilmentymälle kutsutaan jotakin R:n metodia, suoritetaan R:ssä metodille määritelty toteutus, ellei L:ssä ole määritelty metodille toisenlaista toteutusta.

e)

Jos Java-metodi M1 kutsuu toista metodia M2, ja M2 heittää epäonnistuessaan tarkastettavan poikkeuksen (engl. checked exception), ei metodia M1 ole mahdollista laatia käyttämättä siinä `try-catch`-rakennetta, jolla varaudutaan tähän epäonnistumismahdollisuuteen.

Tehtävä 3 (68 pistettä/100)

Tutustu huolellisesti ohessa kuvattuihin neljään luokkaan `Otus`, `Ominaisuus`, `OminaisuusVaihtoehdoin` ja `Paikka`, jotka kuvaavat jonkinmoisen pelin paikkoja, paikoissa olevia otuksia ja otusten ominaisuuksia. Luokkien dokumentaatio on puutteellinen, mutta koodi on sentään onneksi varsin siistiä ja Javan kieliopin mukaista (siinä ei siis ole sellaisia virheitä, jotka kääntäjä havaitsisi...).

Vastaa mainittuihin luokkiin liittyviin kysymyksiin, jotka on esitetty alla. Vaikka et osaisi vastata johonkin tiettyyn kohtaan, ei kannata joutua paniikkiin - kysymykset ovat erityyppisiä ja haastavuudeltaan eritasoisia.

Kuhunkin kohtaan tarvitaan vain varsin lyhyt vastaus, korkeintaan muutama lause ellei toisin ole mainittu. Kunkin kohdan pistearvo on neljä pistettä.

Kohdat a-d käsittelevät vain luokkia `Otus`, `Ominaisuus` ja `OminaisuusVaihtoehdoin`, joihin siis kannattanee tutustua ensin. Lopuissa kohdissa kuvioon tulee mukaan myös `Paikka`-luokka.

a)

Kirjoita `Otus`-luokan `kerroOminaisuus`-metodin julkisesta osuudesta 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)

Ajetaan tämä testaustarkoitukseen luotu käynnistysmetodi. Luettele kaikki ne oheisten luokkien metodien (myös konstruktorien) kutsut, jotka ohjelman suorituksen aikana tehdään. Luettele metodikutsut siinä järjestyksessä, missä niiden suorittaminen aloitetaan. Mainitse kustakin metodikutsusta metodin nimi sekä rivinumero, jolla kyseisen metodin määrittely ohjelmakoodissa alkaa (tämä siksi, että vastauksesi olisi yksiselitteisempi). Jos jotakin metodia kutsutaan lukuisia kertoja, sen tulee esiintyä useita kertoja myös luettelossasi.

Pelkkä metodi- ja konstruktorikutsujen luettelo rivinumeroineen riittää vastaukseksi. Kuitenkin jos olet epävarma, voi kannattaa hieman selostaa vastaustasi - tällä voit ehkä pelastaa itsellesi enemmän pisteitä, jos luettelossasi on virheitä mutta päättelysi on jotakuinkin oikein.

```
public static void main(String[] komentoriviparametrit) {
  String[] vaihtoehdot = { "nainen", "mies", "joku muu" };

  Ominaisuus[] ekanOminaisuudet = new Ominaisuus[3];
1 ekanOminaisuudet[0] = new OminaisuusVaihtoehdoin("sukupuoli", "mies", vaihtoehdot);
2 ekanOminaisuudet[1] = new Ominaisuus("väri", "vihreä");
3 ekanOminaisuudet[2] = new Ominaisuus("luonne", "iloinen");
4 Otus ekaOtus = new Otus("Olli", ekanOminaisuudet);

  Ominaisuus[] tokanOminaisuudet = new Ominaisuus[3];
5 tokanOminaisuudet[0] = new OminaisuusVaihtoehdoin("sukupuoli", "joku muu", vaihtoehdot);
6 tokanOminaisuudet[1] = new Ominaisuus("väri", "punainen");
7 tokanOminaisuudet[2] = new Ominaisuus("luonne", "suruinen");
8 Otus tokaOtus = new Otus("Molli", tokanOminaisuudet);

9 Ominaisuus ekanSukupuoli = ekaOtus.kerroOminaisuus("sukupuoli");
10 Ominaisuus tokanIka = tokaOtus.kerroOminaisuus("ikä");
  if (ekanSukupuoli != null && tokanIka != null) {
    System.out.println(ekanSukupuoli);
    System.out.println(tokanIka);
  }

}
```

c)

Etsi `Ominaisuus`-luokasta huonosti tehty ohjelmointiratkaisu ja perustele lyhyesti, miksi se on vähintäänkin arveluttava. Kiinnitä huomiota siihen, miten tehty ratkaisu vaikuttaa juuri tähän annettuun esimerkkitapaukseen.

d)

Millaisia (Java perus-API:stakin löytyviä) perusvälineitä apuna käyttäen otusluokka voitaisiin toteuttaa kätevämmin tai muuten paremmin, kuitenkin muuttamatta otusluokan julkista liityntää? Perustele lyhyesti.

Java API:n luokkien tai niiden metodien nimiä ei ole tarpeen esittää.

e)

Kuvaile lyhyesti millainen on juuri luodun, annetulla konstruktorilla alustetun paikkaolion `otukset`-kentän arvo.

f)

Mikä on `Paikka`-olion `otuksia`-kentän suurin mahdollinen arvo? Perustele lyhyesti, miksi kentän arvo ei koskaan ylitä tätä yläraja-arvoa?

g)

`Paikka`-luokassa on kaksi metodia nimeltä `poistaOtus`. Mitä termiä tällaisesta samannimisyydestä käytetään? (Suomen- tai englanninkielinen termi kelpaa.) Mikä merkitys näiden kahden metodin keskenään erilaisilla parametreilla on metodien kutsujan kannalta?

h)

Kuvaile, miten paikkaluokan `poistaOtus(int poistettava)`-metodi käyttäytyy, jos sille annetaan negatiivinen luku parametriksi? Siis: mitä se tekee ja palauttaa? Syntyykö jokin poikkeus (engl. exception)?

i)

Kuvaile lyhyesti, mikä on `asetaOtus`-metodin tarkoitus `Paikka`-luokan kokonaisuudessa?

j)

Yksityistä metodia ei voi kutsua luokan itsensä ulkopuolelta. Kerro lyhyesti, miksi `Paikka`-luokan `asetaOtus`-metodin on perusteltua olla tällainen yksityinen metodi?

k)

Ajetaan tämä testaustarkoitukseen luotu käynnistysmetodi. Luettele kaikki ne `Paikka`-luokan metodien (myös konstruktorien) kutsut, jotka ohjelman suorituksen aikana tehdään. Huomaa siis erityisesti, että sinun tarvitsee luetella vain `Paikka`-luokan metodien kutsut. Katso tarkemmat ohjeet vastauksen muotoilusta kohdasta b.

```
public static void main(String[] komentoriviparametrit) {
  Ominaisuus[] ekanOminaisuudet = new Ominaisuus[1];
  ekanOminaisuudet[0] = new Ominaisuus("väri", "punainen");
  Otus ekaOtus = new Otus("Minttu", ekanOminaisuudet);

  Ominaisuus[] tokanOminaisuudet = new Ominaisuus[1];
  tokanOminaisuudet[0] = new Ominaisuus("väri", "kalpea");
  Otus tokaOtus = new Otus("Ville", tokanOminaisuudet);

  Paikka koppi = new Paikka("pussauskoppi", 3);
  koppi.lisaaOtus(ekaOtus);
  koppi.lisaaOtus(tokaOtus);

  for (int monesko = 0; monesko < koppi.kerroOtustenMaara(); monesko++)
    System.out.println(koppi.kerroOtus(monesko).kerroNimi());

  System.out.println(koppi.poistaOtus(ekaOtus));
  for (int monesko = 0; monesko < koppi.kerroOtustenMaara(); monesko++)
    System.out.println(koppi.kerroOtus(monesko).kerroNimi());
}
```

l)

Kenties oletkin jo huomannut, että annetussa ohjelmakoodissa ei ole kaikki ihan kohdallaan. Kaksi metodeista on laadittu väärin, minkä vuoksi ne eivät toimi järkevästi kaikilla kelvollisillakaan parametriarvoilla. (Sitä, että osa metodeista "kaatuu" saadessaan `null`-arvoja parametriksi, ei tässä pidetä virheenä - ko. metodeille pitää antaa "kunnollisia olioarvoja".)

Etsi nämä kaksi erillistä virhettä (anna rivinumerot) ja kuvaile lyhyesti, mikä menee pieleen ja miksi.

Vinkki: etsi toimintavirheitä. Kyseiset virheet ovat sen verran selviä (kunhan ne ensin löytää), ettei dokumentaation puutteesta huolimatta liene epäselvää, että kyseisten metodien olisi tarkoitus toimia toisin.

m)

Korjaa edellisessä kohdassa mainitsemasi virheet eli kirjoita virheellisten rivien ohjelmakoodi uusiksi.

n)

Kuvaile lyhyesti miten `otukset`-kenttää `Paikka`-luokassa käytetään eli millä logiikalla `Otus`-olioita pidetään sen eri indekseillä tallessa. Miten otusten poistot paikasta vaikuttavat `otukset`-kenttään?

o)

Millaisia (Java perus-API:stakin löytyviä) perusvälineitä apuna käyttäen paikkaluokka voitaisiin toteuttaa kätevämmin tai muuten paremmin, kuitenkin muuttamatta paikkaluokan julkista liityntää? Perustele lyhyesti.

Java API:n luokkien tai niiden metodien nimiä ei ole tarpeen esittää.

p)

Kuvaile ja perustele lyhyesti yksi mielekäs tapa käyttää abstraktia luokkaa siten, että tämän aihealueen luokkamallia suunniteltaessa olisi voitu (tai ehkä annettua luokkamallia laajentamalla vieläkin voitaisiin?) saada aikaan jotenkin parempi luokkamalli.

Vinkki: tähän kohtaan ei ole ainoaa oikeaa vastausta. Lisäksi se, mikä on "parempi luokkamalli" riippuu tietysti siitä, millaisessa pelisovelluksessa näitä luokkia lopulta haluttaisiin käyttää. Esitä kuitenkin jokin yksi, omasta mielestäsi mahdollisimman realistinen tapa mallin parantamiseksi.

Pitkää perustelua ei tarvita - muutamalla lauseella voi tästäkin kohdasta selvitä.

q)

Mitä hyötyä tai haittaa voisi mielestäsi olla siitä, että `Otus`-luokkaan lisättäisiin otuksen sijaintia kuvaamaan kenttä `private Paikka sijainti;`

Muista käydä täyttämässä kurssikyselylomake kurssisivuilla. Kurssipalautteen antamisesta saa 50 harjoitustehtäväpistettä.