Tentti: T-106.1200 Ohjelmoinnin perusteet T

Tenttipäivä: 14.12.2007

Huom! Tämä on T-peruskurssin tentti. L- peruskurssilla on erillinen tenttipaperi.

Yleistä

Tehtävät 1 ja 2 muodostavat tentin pakollisen osan. Kummastakin näistä tehtävistä on saatava vähintään yksi piste päästäkseen läpi tentistä. Tehtäviä 3-5 ei edes arvostella, ellei tätä pakollista osaa läpäise. Toisaalta, jos molemmista pakollisista tehtävistä saa vähintään yhden pisteen, niin myös varmasti pääsee tentistä läpi vähintään arvosanalla 1. Arvosana määrittyy tällöin kaikkien tehtävien yhteenlasketun pistemäärän perusteella (joihin lisätään myös syksyn aikana luentopalautepaperien avulla kerätyt pisteet).

Arvosanojen pisterajat määrätään vasta tentin jälkeen. Tenttitulokset julkaistaan kuukauden sisällä tenttipäivästä kurssien WWW-sivuilla, ja oman suorituksen arvosteluun voi tutustua erillisessä tilaisuudessa, jonka aika ja paikka tullaan myös ilmoittamaan kurssisivuilla.

Tehtävä 1 (2 pistettä)

Tutustu huolellisesti tentin liitteessä annettuihin luokkiin. Luokka `Person` kuvaa henkilöitä, luokka `Group` henkilöryhmiä. Luokan `Test` avulla voi koekäyttää em. kahta luokkaa. Jotkut ohjelman osille annetut nimet (esim. `members2`) eivät ole kovin kuvaavia, joten joudut itse pohtimaan, mikä niiden merkitys ohjelmassa on.

a) Mitä ohjelma tulostaa, kun ajetaan luokan `Test` käynnistysmetodi?

b) Kuinka monta kertaa tällaisen ohjelma-ajon aikana suoritetaan luokassa `Group` oleva koodirivi numero 37?

Täysiin pisteisiin riittää periaatteessa oikean tulosteen ja suorituskertojen kirjaaminen vastauspaperiin. Kuitenkin voi kannattaa selittää ratkaisuasi sanallisesti muutamalla lauseella tentin arvostelijalle. Hieman väärä vastaus, jossa perusperiaatteet ovat oikein, on parempi kuin perustelematon väärä vastaus.

Tämän tehtävän tarkoitus on varmistaa, että jokainen kurssin läpäisevä opiskelija ymmärtää perusasiat Java-ohjelmakoodista ja ohjelman suorittamisesta. Tehtävä kuuluu tentin pakolliseen osaan ja siitä on saatava vähintään yksi piste läpäistäkseen tentin. Tehtävä arvostellaan asteikolla: 0 (hylätty) / 1 (hyväksytty) / 2 (erinomainen).

Tehtävä 2 (2 pistettä)

Kirjoita Java-kielinen metodi, joka:

- Saa parametrikseen merkkijonon (`String`).
- Laskee, montako kertaa annetussa merkkijonossa esiintyy sama merkki kahdesti peräkkäin. Palauttaa tällaisten "kaksoismerkkien" lukumäärän.

Esimerkkejä:

- Jos parametri on "laama on kiva", palautetaan 1, koska merkkijonossa on yksi kaksoismerkki (kaksi a-kirjainta peräkkäin).
- Jos parametri on "abcabcabc", palautetaan 0.
- Jos parametri on "suklaa-aarre", palautetaan 3.
- Jos parametri on "suklaa aarre", palautetaan myös 3.
- Jos parametri on "suklaaaarre", palautetaan 4. (Sama kirjain voi kuulua useaan kaksoismerkkiin.)

Voit käyttää apuna esimerkiksi luokan `String` metodia `charAt(int index)`, joka palauttaa parametrina annetun indeksin osoittaman merkin merkkijonosta.

Tämän tehtävän tarkoitus on varmistaa, että jokaisella kurssin läpäisevällä opiskelijalla on vähintäänkin auttavat taidot Java-ohjelmakoodin kirjoittamisessa. Tehtävä kuuluu tentin pakolliseen osaan ja siitä on saatava vähintään yksi piste läpäistäkseen tentin. Tehtävä arvostellaan asteikolla: 0 (hylätty) / 1 (hyväksytty) / 2 (erinomainen).

Tehtävä 3 (4 pistettä)

Esitä lyhyesti perusteltu mielipiteesi seuraavista väitteistä.

a) Väite: Luokasta kirjoitetussa dokumentaatiossa (esim. Javadoc-dokumentaatiossa) olisi hyvä kuvata luokan ohjelmakoodissa määritellyt muuttujat.

b) Oletetaan, että liitteen ohjelmakoodissa rivien 108 ja 109 väliin lisätään uusi rivi:

```
people[2] = new Person("Bob", 50);
```

Väite: Tämä rivin lisääminen vaikuttaa ohjelman tuottamaan tulosteeseen.

c) Väite: tavukoodin (*bytecode*) käyttö välikielenä mahdollistaa sen, että samaa Java-virtuaalikoneohjelmaa (JVM:ää) voi käyttää Java-ohjelmien ajamiseen erilaisilla tietokoneilla.

d) Oletetaan, että kaksi luokkaa A ja B toteuttavat saman rajapintamäärittelyn (eli rajapintaluokan eli rajapinnan, engl. *interface*). Väite: viittauksia kumman tahansa luokan ilmentymiin voi sijoittaa kumman tahansa luokan tyyppiseen muuttujaan tähän tapaan:

```
B b = new A();
```

Tehtävä 4 (2 pistettä)

Olkoon tutkittavana eräs ohjelma, jossa on eräässä metodissa peräkkäin rivit:

```
int size = this.array.length;
this.array[index].setData(data);
```

Oletetaan lisäksi, että on havaittu kyseisen ohjelman kaatuvan ajettaessa ajonaikaiseen poikkeukseen (*exception*). Tarkemmin ottaen ottaen ohjelma kaatuu `NullPointerException`-poikkeustilanteeseen, joka syntyy jälkimmäisellä yllä olevista riveistä. Arvioi, mistä virhe voisi johtua. Mistä päin ohjelmaa lähtisit etsimään virheen pohjimmaista syytä? Miksi voit keskittyä etsimään virhettä vain sieltä? Koko ohjelmaa ei ole tässä esitetty; kerro perusidea lyhyesti, korkeintaan muutamalla virkkeellä.

Tehtävä 5 (2 pistettä)

Alla olevaa rekursiivista koodia on tarkoitus käyttää eräässä ohjelmassa laskemaan, montako henkilöä, joiden nimi on annettu (`name`), löytyy annetusta henkilölistasta (`people`). On kuitenkin osoittautunut, että koodi ihan toimi.

Tutustu koodiin huolella ja korjaa siinä oleva virhe tai virheet. Säilytä koodin rekursiivinen luonne; älä käytä silmukoita.

```
public static int countPeople(String name, ArrayList<Person> people) {
  return countPeople(name, people, 0);
}

public static int countPeople(String name, ArrayList<Person> people, int start) {
  if (start == people.size()) {
    return 1;
  }
  int increment;
  if (people.get(start).getName().equals(name)) {
    increment = 1;
  } else {
    increment = 0;
  }
  return increment + countPeople(name, people, start + increment);
}
```

MUISTA VASTATA KURSSIPALAUTEKYSELYYN KURSSIN KOTISIVUILLA!
Kurssipalautteen antaminen on edellytys kurssin läpäisemiselle. Kysely on auki 21.12. asti.

```
1:
2: public class Group {
3:
4:
5:   private int size;
6:   private Person[] members;
7:   private Person[] members2;
8:
9:
10:  public Group(Person[] members) {
11:    this.size = members.length;
12:
13:    this.members = members;
14:    this.members2 = new Person[this.size];
15:    for (int index = 0; index < this.size; index++) {
16:      this.members2[index] = this.members[index];
17:    }
18:
19:    this.processMembers();
20:  }
21:
22:
23:  private void processMembers() {
24:    for (int index = 0; index < this.size - 1; index++) {
25:      int found = this.find(index);
26:      Person temporary = this.members2[index];
27:      this.members2[index] = this.members2[found];
28:      this.members2[found] = temporary;
29:    }
30:  }
31:
32:
33:  private int find(int start) {
34:    int bestSoFar = start;
35:    for (int index = start + 1; index < this.size; index++) {
36:      if (this.members[index].getAge() < this.members[bestSoFar].getAge()) {
37:        bestSoFar = index;
38:      }
39:    }
40:    return bestSoFar;
41:  }
42:
43:
44:  public int getSize() {
45:    return this.size;
46:  }
47:
48:
49:  public Person getMember(int index) {
50:    return this.members[index];
51:  }
52:
53:
54:  public Person getMember2(int index) {
55:    return this.members2[index];
56:  }
57:
58:
59: }
60:
61:
62:
63:
64:
65:
66:
67:
```

Jimmy 16
Richard 17
Alice 21
Kim 22
Dave 26

Jimmy 16
Alice 21
Richard 17
Kim 22
Dave 26

```
68:
69: public class Person {
70:
71:
72:   private String name;
73:   private int age;
74:
75:
76:   public Person(String name, int age) {
77:     this.name = name;
78:     this.age = age;
79:   }
80:
81:
82:   public String getName() {
83:     return this.name;
84:   }
85:
86:
87:   public int getAge() {
88:     return this.age;
89:   }
90:
91:
92: }
93:
94:
95:
96:
97: public class Test {
98:
99:   public static void main(String[] args) {
100:
101:    Person[] people = new Person[5];
102:    people[0] = new Person("Dave", 26);
103:    people[1] = new Person("Alice", 21);
104:    people[2] = new Person("Richard", 17);
105:    people[3] = new Person("Kim", 22);
106:    people[4] = new Person("Jimmy", 16);
107:
108:    Group group = new Group(people);
109:    for (int number = 0; number < group.getSize(); number++) {
110:      System.out.println(group.getMember(number).getName());
111:      System.out.println(group.getMember2(number).getName());
112:    }
113:
114:  }
115:
116:
117: }
118:
119:
120:
121:
122:
123:
124:
125:
126:
127:
128:
129:
130:
131:
132:
133:
134:
```

BOB

	1	2
	Jimmy	Jimmy
	Alice	Richard
	Richard	Alice
	Kim	Kim
	Dave	Dave

Dave
Jimmy
Alice
Richard
Richard
Alice
Kim
Kim
Jimmy
Dave