Tentti: T-106.1200 Ohjelmoinnin perusteet T

Tenttipäivä: 03.01.2007

Huom! Tämä on T-peruskurssin tentti.
Kurssilla T-106.1203 Ohjelmoinnin perusteet L 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. Arvosanojen pisterajat määrätään vasta tentin jälkeen.

Tehtävä 1 (2 pistettä)

Tutustu tarkasti oheisessa liitteessä annettuihin luokkiin. Luokka `ParkingLot` kuvaa parkkipaikkakäytössä olevia alueita, joissa on kussakin tietty määrä paikkoja (*space*) autoille. Jokaisella paikalla on parkkimittari, joka osaa pitää kirjaa ajasta, jonka auto on ollut parkissa kyseisellä paikalla. Parkkimittareita kuvaa luokka `ParkingMeter` ja autoja luokka `Car`. Luokan `Test` avulla voi koekäyttää em. luokkia.

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

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).

Täysiin pisteisiin riittää oikea tuloste. Voit myös selittää ratkaisuasi sanallisesti, jos haluat selventää ajatustenjuoksuasi tentin arvostelijalle.

Tehtävä 2 (2 pistettä)

Kirjoita Java-kielinen metodi, joka:

- saa parametrikseen merkkijonoja (`String`) sisältävän listan
- palauttaa sellaisten parametrilistassa olevien merkkijonojen lukumäärän, joiden pituus on suurempi kuin kaikkien listassa olevien merkkijonojen keskimääräinen pituus.

Esimerkiksi, jos metodin parametrina saamassa listassa on merkkijonot "a", "muu", "ananasakäämä", "lordi" ja "humppa", niin niiden pituuksien keskiarvo on 5,4. Kaksi merkkijonoista on tätä pidempi, joten metodin tulee palauttaa luku kaksi.

(Liukulukujen käytöstä johtuvia pyöristysvirheitä ei tarvitse tässä tehtävässä erikseen huomioida.)

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 perusteltu mielipiteesi seuraavista väitteistä, kustakin ***korkeintaan parilla virkkeellä***.

a) Väite: Java-muuttujan käyttöalueen (*scope*) määrää yksiselitteisesti se, millaista näkyvyysmäärettä (esim. `public`) sen määrittelyn yhteydessä käytetään.

b) Olkoon tutkittavana metodi:

```
public static int doStuff(int number, int another) {
  if (number < another) {
    return 1;
  } else {
    return 2 + doStuff(number - 1, another + 1);
  }
}
```

Väite: kun tätä metodia kutsutaan parametriarvoilla 6 ja 1, syntyy ns. loputon rekursio, joka johtaa ohjelman käytössä olevien muistiresurssien loppumiseen.

c) Väite: Bytecode-välikielen käyttö mahdollistaa sen, että samaa Java-virtuaalikoneohjelmaa (JVM:ää) voi käyttää Java-ohjelmien ajamiseen erilaisilla tietokoneilla.

d) Väite: tyyppimuunnoksen (*cast*) ideana on muuttaa arvon dynaaminen tyyppi (eli ajonaikainen tyyppi, *runtime type*) erilaiseksi kuin sen staattinen tyyppi.

Tehtävä 4 (2 pistettä)

Kerro lyhyesti, mitä ovat vapaasti heitettävät poikkeukset (*unchecked exceptions*)? Miten ne eroavat muista poikkeuksista? Anna pieni esimerkki, joka osoittaa, että ymmärrät vapaasti heitettävien poikkeusten idean. (Pelkkä esimerkki ei riitä pisteiden saamiseen.).

Tehtävä 5 (2 pistettä)

a) Esitä lyhyesti perusteltu mielipiteesi väitteestä: Olio-ohjelmointi (eli olioita ja luokkia hyväksi käyttävä tietokoneohjelmointi) on tuore keksintö.

b) Määrittele korkeintaan parilla virkkeellä, mikä on mm. Java-ohjelman suoritukseen liittyvä käsite ohjelmalaskuri (*program counter*).

```java


public class ParkingLot {

  private ParkingMeter[] meters;   // container
  private int count;               // stepper


  public ParkingLot(int size) {
    this.meters = new ParkingMeter[size];
    for (int space = 0; space < this.getSize(); space++) {  // space: stepper
      this.meters[space] = new ParkingMeter();
    }
    this.count++;
  }


  public int getSize() {
    return this.meters.length;
  }


  public boolean hasCar(int space) {
    return this.getCar(space) != null;
  }


  public int getCarCount() {
    return this.count;
  }


  public Car getCar(int space) {
    if (this.hasIndex(space)) {
      return this.meters[space].getCar();
    } else {
      return null;
    }
  }


  private boolean hasIndex(int space) {
    return space >= 0 && space < this.getSize();
  }


  public int park(Car car) {
    return this.park(car, this.findSpace());
  }


  private int findSpace() {
    for (int space = 0; space < this.getSize(); space++) {  // space: stepper
      if (!this.hasCar(space)) {
        return space;
      }
    }
    return -1;
  }


  public int park(Car car, int space) {
    if (this.find(car) < 0 && this.hasIndex(space) &&
        this.meters[space].start(car)) {
      this.count++;
      return space;
    } else {
      return -1;
    }
  }


  public int depart(Car car) {
    return this.depart(this.find(car));
  }


  private int find(Car car) {
    for (int space = 0; space < this.getSize(); space++) {  // space: stepper
      if (this.getCar(space) == car) {
        return space;
      }
    }
    return -1;
  }


  public int depart(int space) {
    if (this.hasCar(space)) {
      this.count--;
      return this.meters[space].free();
    } else {
      return -1;
    }
  }


  public void advanceOneHour() {
    for (ParkingMeter meter : this.meters) {  // meter: most-recent holder
      meter.advanceHours(1);
    }
  }


  public String getDescription() {
    String description = "";                    // description: gatherer
    for (ParkingMeter meter : this.meters) {    // meter: most-recent holder
      description += meter.getDescription() + "\n";
    }
    return description;
  }

}
```