OHJ-1400 Olio-ohjelmoinnin peruskurssi

Tentissä ei saa käyttää ylimääräistä kirjallista materiaalia, laskimia, tietokoneita tai muita lunttausvälineitä.

Muutama sana tenttivastauksen kirjoittamisesta:

1. Vastauksessa oleta vastaavasi sellaisen ihmisen kysymykseen, joka tuntee kohtalaisen hyvin ohjelmistotekniikan aihealuetta muutoin paitsi tämän kysymyksen osalta. Muista että vastauksesi tarkoitus on vakuuttaa tarkastaja osaamisestasi.
2. Mieti etukäteen esim. ranskalaisilla viivoilla vastauksesi pääkohdat ja lajittele ne johdonmukaiseen järjestykseen — älä kirjoita yhteen pötköön kaikkea mieleen tulevaa.
3. **Muista vastata kaikkiin tehtävän kysymyslauseisiin, sillä täysiä pisteitä ei voi saada jos kaikkiin kysyttyihin asioihin ei ole vastattu.**

1. Selitä (n. max. 6-7 riviä/kohta) seuraavat olio-ohjelmoinnin käsitteet ja mitä hyötyä niistä saadaan olio-ohjelmoinnissa. *Älä* selitä niistä pelkkää syntaksia tms. vaan kerro etupäässä, mitä ko. käsitteet *tarkoittavat*.
   a) CRC-kortti *(CRC card)*
   b) Kapselointi *(encapsulation)*
   c) Luokan vastuualue *(responsibility)*
   d) Dynaaminen sitominen *(dynamic binding)*
   e) Abstrakti kantaluokka *(abstract base class)*
   f) Luokkahierarkia *(class hierarchy)*
2. Alla on muodostettu pareja kurssiin liittyvistä termeistä. Selitä ko. termipareista, miten termit liittyvät yhteen. Pidä huoli, että vastauksesta selviää myös, että ymmärrät mitä termit tarkoittavat. (Huomaa, että joissain termeissä yhtenemiskohtia voi olla useampi kuin yksi.)
   a) Nimiavaruus — Luokka
   b) Rakentaja — Periytyminen
   c) Vastuualue — Olion luominen dynaamisesti
   d) Virtuaalifunktio — Purkaja
   e) Olion elinkaari — Luokan vastuualue
   f) CRC-kortti — Luokan rajapinta

KÄÄNNÄ!

3. Seuraavassa on joukko väittämiä olio-ohjelmoinnista ja C++:sta. Mitkä väittämät ovat oikein, mitkä väärin? Perustele mielestäsi vääristä väittämistä parilla lauseella, *miksi/miten* väittämä on väärin ja miten asia todellisuudessa on.

   a) Vakio-osoittimen (esim. `const Kirja*`) läpi oliolle saa kutsua vain vakiojäsenfunktioita.

   b) Jos dynaamisesti new'llä luodun olion jättää tuhoamatta *delete*llä, se ei haittaa koska ohjelman lopussa käyttöjärjestelmä vapauttaa muistin kuitenkin.

   c) UML:n muodostusmissuhde (musta salmiakki, *composition*) tarkoittaa, että suhteeseen kuuluvat luokat muodostuvat samalla tavoin, ts. niillä on sama rajapinta.

   d) UML:ssä periytymisnuolen yhteydessä oleva lukumäärämerkintä ilmoittaa, montako aliluokkaa kantaluokasta on periytetty.

   e) Jos luokassa jäsenmuuttuja laitettaisiin `public`-puolelle, voisi sitä muuttaa olion ulkopuolelta ilman, että olio itse sitä huomaa.

   f) Kopiorakentaja on funktio, joka luo uuden olion ja alustaa sen kopiorakentajalle annetun olion kopioksi.

4. Seuraavassa on muutama pikkuessee. Pyri vastaamaan kuhunkin kaikki olennaiset asiat mukaan ottaen.

   a) C++:n `const`-mekanismi ja olio-ohjelmointi. Missä kaikkialla `const` tulee vastaan ja mitä hyötyä siitä saadaan olio-ohjelmoinnissa?

   b) Mitä hyötyä C++:n *purkajista* on dynaamisessa muistinhallinnassa? Entä tuovatko ne uusia vaaroja dynaamiseen muistinhallintaan?

   c) Keksi ja kuvaa mahdollisimman monta keskenään erilaista tilannetta, jossa oikeanlainen periytymisen käyttö lisää koodin yleiskäyttöisyyttä ja vähentää tarvetta saman koodinpätkän moneen kertaan kirjoittamiseen.