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.

Palauta kaikki *nimetyt* vastauspaperit *omiin pinoihinsa*!

..... Tehtävät 1. & 2. **omalle paperilleen**! Nimi paperiin! .....

1. 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) Luokan rajapinta on sitä parempi, mitä laajempi se on.
   b) Jos dynaamisesti new'llä luodun olion jättää tuhoamatta *delete*llä, se ei haittaa koska ohjelman lopussa käyttöjärjestelmä vapauttaa muistin kuitenkin.
   c) Tapahtumasekvenssit ovat UML:n notaatio, jolla erotellaan toisistaan ohjelman mahdolliset ja mahdottomat tapahtumat.
   d) Jos luokassa jäsenmuuttuja laitettaisiin `public`-puolelle, voisi sitä muuttaa olion ulkopuolelta ilman, että olio itse sitä huomaa.
   e) UML:ssä periytymisnuolen yhteydessä oleva lukumäärämerkintä ilmoittaa, montako aliluokkaa kantaluokasta on periytetty.
   f) Määreellä `protected` merkitään luokassa ne jäsenfunktiot ja -muuttujat, joiden halutaan näkyvän aliluokille muttei muualle.
2. Seuraavassa on muutama pikkuessee. Pyri vastaamaan kuhunkin kaikki olennaiset asiat mukaan ottaen.
   a) C++:n `const`-mekanismi ja olio-ohjelmointi.
   b) Miten periytyminen auttaa *oliosuunnittelun* tasolla? **Huomaa,** että kyse on suunnittelusta, ei koodaamisesta!
   c) Olioiden elinkaari (luominen ja tuhoutuminen) olio-ohjelmoinnissa ja C++:ssa.

..... *KÄÄNNÄ!* .....

..... Tehtävät 3. & 4. **omalle paperilleen**! Nimi paperiin! .....

3. 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) Kapselointi *(encapsulation)*
   b) Ennakkoesittely *((forward) declaration)*
   c) Käyttötapaus *(use case)*
   d) Luokkahierarkia *(class hierarchy)*
   e) Luokan vastuualue *(responsibility)*
   f) Abstrakti kantaluokka *(abstract base class)*

4. Alla on luokkakaavio, jossa on neljä luokkaa ja niiden välisiä yhteyksiä.

   a) Kirjoita kaavion perusteella luokkien mahdolliset esittelyt (siis otsikkotiedoston "`class X {...};`", ei jäsenfunktioiden toteutusta). **Perustele** lyhyesti esittelyihin, miksi olet päätynyt mihinkin ratkaisuun.
   b) Salliiko luokkakaavio sellaisen tilanteen, että teoskokoelma koostuisi useista toisista teoskokoelmista? Perustele vastauksesi.
   c) Miten luokkakaavio muuttuisi, jos halutaan että teoskokoelmiin liittyy ylimääräinen kommentti, ja useat teoskokoelmat voivat jakaa tämän kommentin keskenään? Entä miten muuttuvat luokkien esittelyt? **Perustele** jälleen lyhyesti ratkaisusi.