

OHJ-1450 Olio-ohjelmoinnin jatkokurssi

Tentti 16.5.2008

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

Muutama sana tenttivastauksen kirjoittamisesta:

1. Mieti etukäteen esim. ranskalaisilla viivoilla vastauksesi pääkohdat ja lajittele ne johdonmukaiseen järjestykseen — älä kirjoita yhteen pötköön kaikkea mieleen tulevaa, se on varma tapa unohtaa olennaista.
2. Muista vastata kaikkiin tehtävän kysymyksiin, täysinä pisteitä ei voi saada jos kaikkiin kysytyihin asioihin ei ole vastattu.
3. Jos vastaus vaatii ohjelmakoodin kirjoittamista, sen ei tarvitse olla pilkulleen syntaksiltaan oikein.

..... Tehtävät 1. & 2. omalle paperilleen! Nimi paperiin! Paperit eri pinoihin.

1. Seuraavassa on joukko väittämiä olio-ohjelmoinnista ja C++:sta. Mitkä väittämät ovat oikein, mitkä väärin? Perustele mielestäsi vääristä väittämistä n. 3–6 rivillä, *miksi/miten* väittäminen on väärin ja miten asia todellisuudessa on.
 - a) Abstraktit kantaluokat (*abstract base classes*) ovat luokkia, joista ei enää voi periyttää aliluokkia.
 - b) C++:n toteutusmalleissa (*template*) tyyppiparametriksi käyvät mitkä tahansa tyypit, joilla ko. toteutusmallin koodi kääntyy.
 - c) Julistamalla itsensä toisen luokan ystäväksi (*friend*) luokka saa pääsyn tämän toisen luokan private-osaan.
 - d) Kun funktion paluutyyppi on luokka, palautetaan funktiosta osoitin, jonka päässä olevalle oliolle kutsutaan sijoitusoperaattoria paluuarvon taltiointiseksi.
 - e) Luokkafunktiot (*static member functions*) ovat luokan vastualueeseen liittyviä palveluita, joiden suorittaminen ei kuitenkaan kuulu minkään yksittäisen olion vastuulle.
 - f) Moniperiytyemisessä luokka periytyy kahdesta tai usemmasta keskenään vaihtoehdoisesta kantaluokasta.
2. Sopimussuunnittelu.
 - a) Mistä sopimussuunnittelussa oikein on kyse? Mitä ovat luokkainvariantit, esi- ja jälkiehdot?
 - b) C++:n vector-luokan indeksointi-operaattorin [] esiehtona on, että käytetty indeksi on pienempi kuin alkioden lukumäärä. Miten operaation jälkiehto muuttuu, jos esiehtoa lievennetään niin, että indeksiksi käy mikä tahansa kokonaisluku?
 - c) Kirjoita (sanallisesti tai kaavalla, kuitenkin mahdollisimman täsmällisesti) seuraavien tuttujen operaatioiden esi- ja jälkiehdot. Alla s on tyyppiä `std::string`, i ja j ovat tyyppiä `int` ja d on tyyppiä `double`.
 - i. `i+j`
 - ii. `i/j`
 - iii. `s.length()`
 - iv. `std::sqrt(d)` (neliöjuuren laskenta)
 - v. `int main(int argc, char* argv[])` (käyttöjärjestelmä kutsuu `main:a!`)

..... Tehtävät 3. & 4. omalle paperilleen! Nimi paperiin! Paperit eri pinoihin.

3. Alla on muodostettu pareja kurssiin liittyvistä termeistä. Selitä ko. termipareista, miten termit liittyvät yhteen. (Huomaa, että joissain termeissä yhtenemiskohtia voi olla useampi kuin yksi.)

- Sopimussuunnittelu — Poikkeukset
- Periytyminen — Olioiden kopiointi
- Funktio-olio — Geneerisyys
- Luokkahierarkia — Rajapintaluokat
- Pysyvyys- ja vaihtelevuusanalyysi — Templatet
- Poikkeukset — Periytyminen

4. Poikkeukset.

- Mitä ja mitkä ovat poikkeustakuut, mitä hyötyä niistä on ja miten ne helpottavat luotettavan ohjelman suunnittelemista?
- Kerro listauksen jäsenfunktioista, mitkä poikkeustakuut ne tarjoavat *ja miksi*.
- Pohdi mahdollisuuksia parantaa jäsenfunktioiden toteutuksia niin, että niiden poikkeustakuut parantuisivat. Jäsenmuuttujia tai niiden tyyppjä ei saa muuttaa. Tässä tehtävässä saa olettaa, että `string::length` ei heitä poikkeuksia ja että muut käytetyt `string:n` operaatiot jättävät merkkijonon ennalleen, jos heittävät poikkeuksen.

```

1 #include <string>
2 #include <stdexcept>
3
4 class Nimi
5 {
6     public:
7         Nimi() : etunimet_(0), sukunimi_(0)
8         {
9         }
10
11        ~Nimi()
12        {
13            delete etunimet_;
14            delete sukunimi_;
15        }
16
17        unsigned int etunimen_pituus() const
18        { // Lasketaan vain 1. etunimi
19            for (unsigned int i = 0;
20                 i < etunimet_>length(); ++i)
21            {
22                if (etunimet_>at(i) == ' ')
23                { // 1. etunimi loppui
24                    return i;
25                }
26            }
27
28            // On vain 1 etunimi
29            return etunimet_>length();
30        }
31
32        void vaihda_suku(std::string const& s)
33        {
34            delete sukunimi_;
35            // Tee kopio
36            sukunimi_ = new std::string(s);
37        }
38
39        void jata_1_etu()
40        {
41            if (etunimet_ == 0)
42            { // Ei etunimiä, ei voi poistaa!
43                throw std::runtime_error("Ei etuja");
44            }
45
46            // Typistetään 1. etunimen pituiseksi
47            etunimet_>resize(etunimen_pituus());
48        }
49
50        void lisää_etu(std::string const& s)
51        {
52            if (etunimet_ == 0) {
53                etunimet_ = new std::string(s);
54            } else {
55                etunimet_>push_back(' '); // Väli
56                etunimet_>append(s); // Etunimi
57            }
58        }
59
60        private:
61            std::string* etunimet_;
62            std::string* sukunimi_;
63        };

```