OHJ-1450 Olio-ohjelmoinnin jatkokurssi

Tentti 12.11.2007

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, täysiä pisteitä ei voi saada jos kaikkiin kysyttyihin asioihin ei ole vastattu.
3. Jos vastaus vaatii ohjelmakoodin kirjoittamista, sen **ei** tarvitse olla pilkulleen syntaksiltaan oikein.

1. Selitä (max. 7 riviä/kohta) seuraavat olio-ohjelmoinnin ja C++:n käsitteet ja **mitä hyötyä/haittaa niistä on olio-ohjelmoinnissa.** (Älä selitä niistä pelkkää syntaksia tms, vaan mitä ko. käsitteet *tarkoittavat*.)
   a) Poikkeushierarkia *(exception hierarchy)*
   b) Luokkamuuttuja (*static data member*)
   c) Suunnittelumalli *(design pattern)*
   d) Rajapintaluokka (*interface class*)
   e) Pysyvyys- ja vaihtelevuusanalyysi *(commonality and variability analysis)*
   f) Sovelluskehys *(framework)*
2. Sopimussuunnittelu.
   a) Mistä sopimussuunnittelussa oikein on kyse? Mitä ovat luokkainvariantit, esi- ja jälkiehdot?
   b) Kirjoita (sanallisesti tai kaavalla, kuitenkin mahdollisimman täsmällisesti) seuraavien tuttujen operaatioiden esi- ja jälkiehdot. Alla `v` on vektori tyyppiä `std::vector<int>`, `i` ja `j` ovat tyyppiä `int` ja `X` on jokin luokka.
      i. `i/j`
      ii. `v[i]`
      iii. `v.at(i)`
      iv. `new X`
      v. `double sqrt(double d)` (neliöjuuren laskenta)
      vi. `int main(int argc, char* argv[])` (käyttöjärjestelmä kutsuu `main`:a)

KÄÄNNÄ!

3. Vastaa seuraaviin kysymyksiin (max. 15 riviä/kohta):

   a) Miten olioiden luominen ja tuhoutuminen C++:ssa eroaa kielistä, joissa on roskienkeruu (esim. Java)? Luettele kummankin hyviä ja huonoja puolia.

   b) Mitä hyötyä dynaamisesta sitomisesta (*dynamic binding*) on ohjelman *suunnittelun* kannalta, (älä keskity yhteen ainoaan luokkaan vaan ohjelman kokonaisuutena)? Entä mitä haittavaikutuksia dynaamisella sitomisella voi olla?

   c) Mitä on **viipaloituminen** *(slicing)* olio-ohjelmoinnissa? Mitä ongelmia se tuo C++:aan ja millaisia keinoja on sen estämiseen?

4. Poikkeukset.

   a) Mitä ja mitkä ovat poikkeustakuut, mitä hyötyä niistä on ja miten ne helpottavat luotettavan ohjelman suunnittelemista?

   b) Kerro listauksen jäsenfunktioista, mitkä poikkeustakuut ne tarjoavat *ja miksi*.

   c) Pohdi mahdollisuuksia parantaa jäsenfunktioiden toteutuksia niin, että niiden poikkeustakuut parantuisivat. Jäsenmuuttujia tai niiden tyyppejä 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>
   #include <stdexcept>
3  using namespace std;

5  class Nimi
   {
7  public:
     Nimi() : nimi_(0)
9    {
       // Aluksi tyhjä
11     nimi_ = new string;
     }
13
     ~Nimi()
15   {
       delete nimi_;
17   }

19   int etunimen_pituus() const
     {
21     for (int i = 0;
            i < nimi_->length(); ++i)
23     {
         if (nimi_->at(i) == ' ')
25       { // Etunimi loppui
           return i;
27       }
       }
29     // Pelkkä etunimi
       return nimi_->length();
31   }
```

```
33   void vaihda_nimi(string const& uusi)
     {
35     delete nimi_;
       // Tee kopio
37     nimi_ = new string(uusi);
     }
39
     void poista_sukunimi()
41   {
       int katko = etunimen_pituus();
43
       if (katko == nimi_->length())
45     { // Ei sukunimeä, ei voi poistaa!
         throw runtime_error("Ei sukunimea");
47     }

49     // Typistetään pelkäksi etunimeksi
       nimi_->resize(katko, ' ');
51   }

53   void vaihda_sukunimi(string const& uusi)
     {
55     poista_sukunimi();
       nimi_->push_back(' '); // Lisää väli
57     nimi_->append(uusi); // ja sukunimi
     }
59
   private:
61   string* nimi_;
   };
```