OHJ-1450 Olio-ohjelmoinnin jatkokurssi (Matti Rintala)

Tentti 14.5.2012

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. Termit

   Selitä (max. 7 riviä/kohta) seuraavat olio-ohjelmoinnin/C++:n käsitteet ja mitä **hyötyä/haittaa** niistä on olio-ohjelmoinnissa. ***Älä*** selitä niistä pelkkää syntaksia tms, vaan kerro etupäässä, mitä ko. käsitteet *tarkoittavat*.

   a) Pysyvyys- ja vaihtelevuusanalyysi *(commonality and variability analysis)*
   b) Viipaloituminen *(slicing)*
   c) Metaohjelmointi *(metaprogramming)*
   d) Suunnittelumalli *(design pattern)*
   e) Laiska kopiointi *(copy-on-write)*
   f) Poikkeushierarkia *(exception hierarchy)*

2. Termipareja

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

   - Periytyminen — Olioiden kopiointi
   - Abstrakti kantaluokka — Virtuaalifunktiot
   - Kopiorakentaja — Funktion paluuarvo
   - Funktio-olio — Geneerisyys
   - Sopimussuunnittelu — Poikkeukset
   - Osoittimet — Olioiden omistus/elinkaari

3. Sopimussuunnittelu ja poikkeustakuut

   a) Mistä sopimussuunnittelussa oikein on kyse, mitä siihen liittyy ja mitä hyötyä siitä on ohjelman suunnittelussa?
   b) Miten sopimussuunnittelu voi hyödyttää myös ohjelman testausvaihetta?
   c) Millaisia vaikutuksia periytymisellä on sopimussuunnitteluun?
   d) Mitä ja mitkä ovat poikkeustakuut, mitä hyötyä niistä on ja miten ne helpottavat luotettavan ohjelman suunnittelemista?

KÄÄNNÄ!

4. Koodinlukutehtävä

   a) Alla oleva listaus kuvaan yksinkertaisen luokan, joka käsittelee etu- ja sukunimiä. Siinä on kerrottu operaatioiden esi- ja jälkiehdot. Mitä virheitä niistä löydät (koodiin verrattuna) ja miten ne korjaisit? *Huom,* saat koskea vai esi- ja jälkiehtoihin, et itse koodiin

   b) Nyt vastaavasti esi- ja jälkiehdot on kiinnitetty alkuperäisiksi, mutta koodia saa muokata. Millaisia muutoksia koodiin pitäisi tehdä, jotta alkuperäiset esi- ja jälkiehdot riittäisivät? Koodia ei ole pakko kirjoittaa, yksityiskohtainen sanallinen kuvauskin riittää. Voi myös olla, että joissain tapauksissa korjaus ei ole edes mahdollinen (kerro tästäkin).

   c) Kerro jokaisesta operaatiosta, minkä *poikkeustakuun* se tarjoaa ja miksi.

   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
   class Nimi
5  {
   public:
7    // Esiehto: -
     // Jalkiehto: Olion etu- ja sukunimet
9    //   alustettu tyhjiksi
     // Poikkeukset: -
11   Nimi() : etunimet_(0), sukunimi_(0)
     {
13   }

15   // Esiehto: -
     // Jalkiehto: Olio siivottu
17   // Poikkeukset: -
     ~Nimi()
19   {
       delete etunimet_;
21     delete sukunimi_;
     }
23
     // Esiehto: -
25   // Jalkiehto: palauttaa n:nnen etunimen
     //   pituuden
27   // Poikkeukset: -
     unsigned int etunimen_pituus(int n) const
29   { // Lasketaan etunimien pituuksia,
       // lopetetaan n:nnen jalkeen
31     unsigned int pituus = 0;
       for (unsigned int i = 0;
33          i < etunimet_->length(); ++i)
       {
35       if (etunimet_->at(i) == ' ')
         { // Etunimi loppui. Palaa, jos n:s,
37         // muuten siirrytaan seuraavaan
           if (--n == 0) { return pituus; }
39         else { pituus = 0; }
         }
41       else { ++pituus; }
       }
43     // Tanne, koska viim. nimi ei lopu sanavaliin
       return pituus;
45   }

47   // Esiehto: s ei ole tyhja merkkijono
     // Jalkiehto: Sukunimi on vaihdettu annetuksi
49   // Poikkeukset: muistin loppuminen
     void vaihda_sukunimi(std::string const& s)
51   {
       delete sukunimi_; sukunimi_ = 0;
53     // Tee kopio
       sukunimi_ = new std::string(s);
55   }

57   // Esiehto: -
     // Jalkiehto: Palauttaa merkkijonon, jossa
59   //   etu+suku sanavalilla erotettuna
     // Poikkeukset: -
61   std::string kokonimi()
     {
63     return *etunimet_ + ' ' + *sukunimi_;
     }
65
     // Esiehto: s:ssa ei sanavaleja
67   // Jalkiehto: etunimi s lisatty muiden peraan
     // Poikkeukset: muistin loppuminen
69   void lisaa_etunimi(std::string const& s)
     {
71     if (etunimet_ == 0) {
         etunimet_ = new std::string(s);
73     } else {
         etunimet_->push_back(' '); // Vali
75       etunimet_->append(s); // Etunimi
       }
77   }

79 private:
     std::string* etunimet_;
81   std::string* sukunimi_;
   };
```