OHJ-1450 Olio-ohjelmoinnin jatkokurssi *(Matti Rintala)*

Tentti 25.5.2010

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.

..... JOKAINEN TEHTÄVÄ omalle paperilleen! Nimi paperiin! Paperit eri pinoihin. .....

1. 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) Älykkäät osoittimet *(smart pointers)*
   c) Viipaloituminen *(slicing)*
   d) Metaohjelmointi *(metaprogramming)*
   e) Syväkopiointi *(deep copying)*
   f) Luokkafunktio *(static member function)*

2. Sopimussuunnittelu.

   a) Mistä sopimussuunnittelussa oikein on kyse? Mitä termejä sopimussuunnitteluun liittyy? Mitä ne tarkoittavat?
   b) Mitä hyötyä koit itse saaneesi sopimussuunnittelusta harjoitustyössä? Entä toiko se haittaa (olettaen, että itse työ olisi kuitenkin ollut muuten samanlainen)?
   c) Kuvaa (sanallisesti tai kaavalla, kuitenkin mahdollisimman täsmällisesti) seuraavien tuttujen operaatioiden rajapintaa sopimussuunnittelun kannalta. Alla `s` on tyyppiä `std::string`, `i` ja `j` ovat tyyppiä `int`.

      i. `i / j`
      ii. `i + j`
      iii. `s.clear()`
      iv. `s.resize(i, 'x')`
      v. `int main(int argc, char* argv[])` (*käyttöjärjestelmä* kutsuu `main`:a!)
      vi.
```
void tayta(int const* mika, int lkm, std::vector<int>& minne)
{
  if (*mika < 0) { throw std::out_of_range("Ei negatiivisia!"); }
  for (int i = 0; i != lkm; ++i) { minne[i] = *mika; }
}
```

..... KÄÄNNÄ! .....

..... JOKAINEN TEHTÄVÄ omalle paperilleen! Nimi paperiin! Paperit eri pinoihin.....

3. Kerro konkreettisella esimerkillä, miten ohjelmointi muuttuisi hankalammaksi, jos alla lueteltu asia poistettaisiin C++:sta (siis kukin kohta erikseen, poistetaan aina yksi asia). Pidä huoli, että vastauksesta käy ilmi, mitä ko. asia tarkoittaa. Keksitkö, miten ko. asian puutteen voisi kiertää?

   a) Avainsana `virtual`

   b) Poikkeukset

   c) Luokkamuuttujat

   d) Abstraktit kantaluokat (C++:n pure virtual, `virtual ... = 0`)

   e) Avainsana `friend`

   f) Templatet

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 tyhja
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     // Pelkka 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 sukunimea, ei voi poistaa!
         throw runtime_error("Ei sukunimea");
47     }

49     // Typistetaan pelkaksi etunimeksi
       nimi_->resize(katko, ' ');
51   }

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