TIE-02400 Ohjelmoinnin tekniikat

(Matti Rintala)

Tentti 23.9.2015

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. 6 riviä/kohta) seuraavat käsitteet ja mitä hyötyä/haittaa niistä on. ***Älä*** selitä niistä pelkkää syntaksia tms, vaan kerro etupäässä, mitä ko. käsitteet *tarkoittavat*.

   a) Yksikkötestaus *(unit testing)*
   b) Qt:n "signaalit"ja "slotit"*(signals and slots in Qt)*
   c) Keskeytyskohta *(breakpoint)*
   d) Luokkainvariantti *(class invariant)*
   e) Haarat versionhallinnassa *(branches)*
   f) C++:n nimiavaruus *(namespace)*

2. Periytyminen.

   a) Mitä on periytyminen? Luettele 4 mielestäsi *olennaisinta* periytymiseen liittyvää termiä ja selitä jokaisesta 2-3 lauseella, mitä termi tarkoittaa.
   b) Mitä ovat rajapintaluokat (*interface classes*) ja miten ne eroavat tavallisista luokista? Mihin niitä voi käyttää ja miksi niitä tarvitaan?
   c) Miten periytyminen vaikuttaa sopimussuunnittelussa? (esi- ja jälkiehdot, luokkainvariantti)

3. Modulaarisuus. Vastaa lyhyehkösti ja ytimekkäästi, mutta kuitenkin koko kysymykseen.

   a) Millä tavoin poikkeukset ja niitä käyttävä virheenkäsittely helpottaa modulaarisuutta eli sitä, että ohjelma voidaan jakaa toisistaan mahdollisimman riippumattomiin osiin?
   b) Mitä on sopimussuunnittelu? Miten sopimussuunnittelu auttaa modulaarisuutta? Mitä hyötyä siitä saadaan ohjelman jakamisessa riippumattomiin osiin?
   c) Entä miten periytyminen helpottaa modulaarisuuden saavuttamista?

KÄÄNNÄ!

4. Koodinlukutehtävä

a) Alla oleva listaus kuvaan yksinkertaisen luokan, joka käsittelee etu- ja sukunimiä. Siinä on kerrottu osa operaatioiden esi- ja jälkiehdoista, mutta osa puuttuu. Kirjoita puuttuvat osat.

b) Mitä virheitä niistä löydät valmiina annetuista esi- ja jälkiehdoista (koodiin verrattuna) ja miten ne korjaisit? *Huom*, saat koskea vai esi- ja jälkiehtoihin, et itse koodiin.

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.

```
#include <string>
#include <stdexcept>

class Nimi
{
public:
  // Esiehto: Ei ole
  // Jalkiehto: Olion etu- ja sukunimet
  //   alustettu tyhjiksi
  // Poikkeukset: Ei tule
  Nimi() : etunimet_(nullptr), sukunimi_(nullptr)
  {
  }

  // Esiehto: Ei ole
  // Jalkiehto: Olio siivottu
  // Poikkeukset: Ei tule
  ~Nimi()
  {
  }

  // Esiehto: ????
  // Jalkiehto: palauttaa n:nnen etunimen
  //   pituuden
  // Poikkeukset: ????
  unsigned int etunimen_pituus(int n) const
  { // Lasketaan etunimien pituuksia,
    // lopetetaan n:nnen jalkeen
    unsigned int pituus = 0;
    for (unsigned int i = 0;
         i < etunimet_->length(); ++i)
    {
      if (etunimet_->at(i) == ' ')
      { // Etunimi loppui. Palaa, jos n:s,
        // muuten siirrytaan seuraavaan
        if (--n == 0) { return pituus; }
        else { pituus = 0; }
      }
      else { ++pituus; }
    }
    // Tanne, koska viim. nimi ei lopu sanavaliin
    return pituus;
  }

  // Esiehto: s ei ole tyhja merkkijono
  // Jalkiehto: Sukunimi on vaihdettu annetuksi
  // Poikkeukset: ????
  void vaihda_sukunimi(std::string const& s)
  {
    delete sukunimi_; sukunimi_ = nullptr;
    // Tee kopio
    sukunimi_ = new std::string(s);
  }

  // Esiehto: ????
  // Jalkiehto: ????
  // Poikkeukset: ????
  std::string kokonimi()
  {
    return *etunimet_ + ' ' + *sukunimi_;
  }

  // Esiehto: ????
  // Jalkiehto: etunimi s lisatty muiden peraan
  // Poikkeukset: muistin loppuminen
  void lisaa_etunimi(std::string const& s)
  {
    if (etunimet_ == nullptr) {
      etunimet_ = new std::string(s);
    } else {
      etunimet_->push_back(' '); // Vali
      etunimet_->append(s); // Etunimi
    }
  }

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