TIE-02400 Ohjelmoinnin tekniikat

(Matti Rintala)

Tentti 23.9.2015

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

Muutama sana tentivastauksen kirjoittamisesta:
1. Mieti etukäteen esim. ranskalaisilla viivoilla vastauksesi pääkohdat ja lajittele ne johdonmukaisen järjestyksen — älä kirjoita yhteen pötköön kaikkea mieleen tulevaa, se on varma tapa unohtaa olennaista.
2. Muista vastata kaikkiin tehtävän kyvymyksiin, täysin pisteitä ei voi saada jos kaikkiin kysymyksiin asioihin ei ole vastattu.
3. Jos vastaus vaatii ohjelmaa on kirjoittamista, sen ei tarvitse olla pilkulleen syntaksillaan 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) Keskeytysohjaus (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 rajoittaluokat (interface classes) ja miten ne eroavat tavallisista luokista? Mihin niitä voi käyttää ja miksi niitä tarvitaan?

c) Miten periytyminen vaikuttaa sopimussuunnittelussa? (esi- ja jälkiriehdot, luokkainvariantti)


a) Millä tavoin poikkeukset ja niitä käyttävä virheenkäsitellyt helpottaa modulaarisuutta eli sitä, että ohjelma voidaan jakaa toisistaan mahdollisimman riippumatomiin osiin?

b) Mitä on sopimussuunnittelu? Miten sopimussuunnittelu auttaa modulaarisuutta? Mitä hyötyä siitä saadaan ohjelman jakamisessa riippumatomiin osiin?

c) Entä miten periytyminen helpottaa modulaarisuuden saavuttamista?

..............................KÄÄNNÄ!..............................
4. Koodinlukutehtävä


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.

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

class Nimi
{
public:
7   // Esiehto: Et ole
// Jalkitehto: Olen etu- ja sukunimet
9   // alustettu tyhjäksi
// Poikkeus: Et tule
11 Nimi() : etunimet_(nullptr), sukunimi_(nullptr) // Tee kopio
             { return pituus(); }
13
19   // Esiehto: Et ole
// Jalkitehto: Olto silvottu
21 // Poikkeus: Et tule
Nimi() {}
23
29   // Esiehto: ????
// Jalkitehto: palauttaa n:nennen etunimen pituuden
31 // Poikkeus: ????
33 unsigned int etunimen_pituus(int n) const
35 // Lopetetaan n:nennen jälkeen
37 unsigned int pituus = 0;
39 for (unsigned int i = 0;
41 // Tonne, koska viim. nimi ei lopu sanavallin
43 )
45 // Esiehto: s et ole tyhjä merkkijono
// Jalkitehto: Sukunimi on vaihdettu annetuksi
47 // Poikkeus: ????
49 void vaihda_sukunimi(std::string const &s)
51 { delete sukunimi_; sukunimi_ = nullptr;
53 } sukunimi_ = new std::string(s);
55 // Esiehto: ????
// Jalkitehto: ????
57 // Poikkeus: ????
59 std::string kokonimi()
61 { return *etunimet_ + ' ' + *sukunimi_; }
63 // Esiehto: ????
// Jalkitehto: etunimi s lisättä muiden perään
// Poikkeus: muistin loppuminen
65 void lisaa_etunimi(std::string const &s)
67 { if (etunimet_ == nullptr) {
69 etunimet_ = new std::string(s);
71 } else {
73 etunimet_->push_back(' '); // Vali
75 if (--n == 0) { return pituus();
77 } else { pituus = 0; }
79 } else { ++pituus; }
};
}```