TIE-02400 Ohjelmoinnin tekniikat

(Matti Rintala)

Tentti 25.11.2015

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. Periytyminen Seuraavassa on joukko väittämiä periytymisestä (C++:ssa). Mitkä väittämät ovat oikein, mitkä väärin? *Lisäksi* perustele mielestäsi vääristä väittämistä parilla lauseella, *miksi/miten* väittämä on väärin ja miten asia todellisuudessa on.

   a) Periytymisen yhteydessä aliluokka perii kantaluokalta kaiken muun toteutuksen paitsi kantaluokan private-osan.

   b) Dynaaminen sitominen (*dynamic binding*) tarkoittaa sitä, että jäsenfunktion kutsun yhteydessä kutsuttava koodi valitaan vasta ajonaikana.

   c) Periytymisen "aliluokan olio kuuluu myös kantaluokkaan" tarkoittaa, että aliluokan olioiden toiminnallisuus on sama kuin kantaluokan olioiden.

   d) Määreellä `protected` merkitään luokassa ne jäsenfunktiot ja -muuttujat, joiden halutaan näkyvän aliluokille muttei muualle.

   e) Luokkahierarkioiden tarkoituksena on esittää, mitkä luokat käyttävät minkäkin toisen luokan palveluita.

   f) Kun luokka periytyy rajapintaluokasta, se saa käyttöönsä rajapintaluokan jäsenfunktioiden toteutukset.

2. Sopimussuunnittelu *(Contract Programming)*.

   a) Mistä sopimussuunnittelussa on kyse, ts. miten se helpottaa ohjelmointia?

   b) Mitkä ovat sopimussuunnittelun keskeiset käsitteet ja mitä ne *tarkoittavat*?

   c) Kirjoita (sanallisesti tai kaavalla, kuitenkin mahdollisimman täsmällisesti) seuraavien funktioiden esi- ja jälkiehdot. Tehtävässä `v` on globaali muuttuja (hyi) tyyppiä `std::vector<int>`.

   i.
```
int f1(int i) {
    if (i<0) { return -i; } else { throw std::exception(); }
}
```
   ii.
```
unsigned int f2(string s) { return s.length(); }
```
   iii.
```
int f3(unsigned int i) { return v[i-1]; }
```
   iv.
```
void f4() { v.push_back(v[0]); }
```
   v.
```
void f5(int i) { v.push_back(v.at(i)); }
```
   vi.
```
int f6(int i) { return v[i]/i; }
```

KÄÄNNÄ!

3. Termit

Selitä (max. 7 riviä/kohta) seuraavat käsitteet ja mitä hyötyä/haittaa niistä on. ***Älä*** selitä niistä pelkkää syntaksia tms, vaan kerro etupäässä, mitä ko. käsitteet *tarkoittavat*.

a) Poikkeushierarkia (*exception hierarchy*)
b) Qt:n "slotit" (*Qt slots*)
c) Testitapaus (yksikkötestauksessa) (*test case (in unit testing)*)
d) Rajapintaluokka (*interface class*)
e) Qt:n "layoutit" (*Qt layouts*)
f) Kutsujälki debuggerissa (*stack trace in debugger*)

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. ***Perustele*** *valintasi mahdollisimman hyvin.*.

Tässä tehtävässä saa olettaa, että vektorin `size()`, `begin()` ja `erase()` eivät heitä poikkeuksia. Lisäksi `at()` ja `push_back()` jättävät vektorin ennalleen, jos heittävät poikkeuksen.

```
#include <vector>

class Lukuja
{
public:
  Lukuja() : data_()
  { }
  ~Lukuja()
  { }

  void lisaa(int x)
  { data_.push_back(x); }

  int montako_loytyy(int x) const
  {
    int lkm = 0;
    for (unsigned int i=0;
         i<data_.size(); ++i)
    {
      if (data_.at(i) == x)
      {
        ++lkm;
      }
    }
    return lkm;
  }

  void edesta_loppuun()
  {
    int luku = data_.at(0);
    data_.erase(data_.begin());
    data_.push_back(luku);
  }

  void tuplaa_viim_n_kertaa(int n)
  {
    int viim = data_.at(data_.size()-1);
    for (int i=0; i<n; ++i)
    {
      data_.push_back(viim);
    }
  }

  void tuplaa_viim_jos_ainoa()
  {
    int viim = data_.at(data_.size()-1);
    if (montako_loytyy(viim) == 1)
    {
      tuplaa_viim_n_kertaa(1);
    }
  }
private:
  std::vector<int> data_;
};
```