# *OHJ-1400 Olio-ohjelmoinnin peruskurssi*

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

Muutama sana tenttivastauksen kirjoittamisesta:

1. Vastauksessa oleta vastaavasi sellaisen ihmisen kysymykseen, joka tuntee kohtalaisen hyvin ohjelmistotekniikan aihealuetta muutoin paitsi tämän kysymyksen osalta. Muista että vastauksesi tarkoitus on vakuuttaa tarkastaja osaamisestasi.
2. Mieti etukäteen esim. ranskalaisilla viivoilla vastauksesi pääkohdat ja lajittele ne johdonmukaiseen järjestykseen — älä kirjoita yhteen pötköön kaikkea mieleen tulevaa.
3. Jos vastaus vaatii ohjelmakoodin kirjoittamista, sen **ei** tarvitse olla pilkulleen syntaksiltaan oikein.

..... Tehtävät 1. & 2. **omalle paperilleen**! Nimi paperiin! .....

1. Seuraavassa on joukko väittämiä olio-ohjelmoinnista ja C++:sta. Mitkä väittämät ovat oikein, mitkä väärin? Perustele mielestäsi vääristä väittämistä parilla lauseella, *miksi/miten* väittämä on väärin ja miten asia todellisuudessa on.

   a) Jos dynaamisesti new'llä luodun olion jättää tuhoamatta *delete*llä, se ei haittaa koska ohjelman lopussa käyttöjärjestelmä vapauttaa muistin kuitenkin.

   b) Luokan vastuualueella tarkoitetaan sitä osaa ohjelmasta, joka käyttää luokan olioita ja on näin niistä vastuussa.

   c) Ennakkoesittelyllä tarkoitetaan sitä, että luokan nimi esitellään ohjelmassa ennen luokan määrittelyä.

   d) Jos kaksi oliota kuuluu samaan luokkaan, C++:ssa olio pääsee käsiksi toisen olion private-osaan.

   e) Tapahtumasekvenssit ovat UML:n notaatio, jolla erotellaan toisistaan ohjelman mahdolliset ja mahdottomat tapahtumat.

   f) Jos luokassa jäsenmuuttuja laitettaisiin `public`-puolelle, voisi sitä muuttaa olion ulkopuolelta ilman, että olio itse sitä huomaa.

   g) UML:n muodostusmissuhde (musta salmiakki, *composition*) tarkoittaa, että suhteeseen kuuluvat luokat muodostuvat samalla tavoin, ts. niillä on sama rajapinta.

   h) Jos olio jäsenmuuttujina on toisia olioita, nämä jäsenmuuttujaoliot tuhotaan ja niiden purkaja suoritetaan, vaikka isäntäoliolle ei olisikaan kirjoitettu purkajaa.

2. *Tausta:* **Abstraktio** on yksi tärkeimmistä asioista ohjelmoinnissa. Se tarkoittaa joidenkin "epäolennaisten" seikkojen piilottamista, jotta asiasta saadaan tiivistettyä tilanteen kannalta oleellinen. Esim. C++:n funktio on abstraktio, koska siitä näytetään ulospäin vain kutsumisen kannalta olennaiset asiat ja itse toteutus piilotetaan "epäolennaisena".
   *Tehtävä:* Missä kaikialla abstraktion vaikutus näkyy *olio-ohjelmoinnissa?* Yritä löytää mahdollisimman monta asiaa ja kerro jokaisesta funktioesimerkin tapaan, **miten abstraktion vaikutus näkyy.** (Vihje: abstraktio on erittäin olennainen olio-ohjelmoinnissa, se väijyy miltei joka nurkan takana.)

..... *KÄÄNNÄ!* .....

..... Tehtävät 3. & 4. **omalle paperilleen!** Nimi paperiin! .....

3. Vastaa tiiviisti (mieti mikä on *olennaista* = abstraktio) seuraaviin (max. n. 8-9 riviä per kohta):
   a) Millä tavoin `const`-sanan käyttö auttaa C++:ssa ja *olio-ohjelmoinnissa*? Mihin kaikkeen sitä käytetään ja mitä hyötyjä sillä saavutetaan?
   b) Mitä on periytyminen olio-ohjelmoinnissa? Mitä konkreettista hyötyä siitä saadaan ohjelmoinnissa? Entä onko periytymisen käytöstä haittapuolia?
   c) Millaisia tapoja C++:ssa on välttää muistivuotoja ja muita dynaamiseen muistinhallintaan liittyviä vaaroja? Mitkä näistä tavoista ovat olio-ohjelmoinnin "ansiota" ja miksi? Entä tuovatko oliot C++:ssa uusia vaaroja dynaamiseen muistinhallintaan?

4. Vastaa toisella paperilla olevan ohjelmalistauksen perusteella alla oleviin tehtäviin:
   a) Piirrä kuva (vapaamuotoinen), josta käy ilmi, mitä olioita ja muuttujia ohjelmassa on olemassa rivin 53 suorittamisen jälkeen. Kuvasta tulee selvitä myös muuttujien ja olioiden jäsenmuuttujien arvot. Osoittimista tulee selvitä, mihin olioon ne osoittavat. Olioiden nimien ja tyyppien (=luokkien) tulee myös selvitä kuvasta (nimi vain niistä olioista, joilla on nimi).
   b) Ohjelmassa on vikaa (muistivuoto/-vuotoja). Mitä lisäyksiä ohjelmaan täytyy tehdä (ja mihin kohtaan), jotta ne korjaantuvat?
   c) Ohjelmassa on myös muutama muu "hyvän C++-olio-ohjelmointitavan" vastainen asia. Millaista huomautettavaa löydät?

```
1  #include <iostream>
   using namespace std;
3
   class Base
5  {
   public:
7    Base(int newvalue);
     ~Base();
9    virtual int foo(int a);
     int count;
11 private:
     int value;
13 };
   Base::Base(int newvalue) : value(newvalue)
15 {
     count = 0;
17 }
   Base::~Base()
19 {}
   int Base::foo(int a)
21 {
     count = count+100; return 10*value + a;
23 }

25 class Deriv : public Base
   {
27 public:
     Deriv(int newvalue);
29   virtual int foo(int a);
   private:
31   int value;
     Base* bp;
33 };
   Deriv::Deriv(int newvalue)
35   : Base(newvalue - 2), value(newvalue),
       bp(new Base(newvalue - 1))
37 {
   }
39 int Deriv::foo(int a)
   {
41   return 100*value + a;
   }
43
   int footwo(Base& b1, Deriv d2)
45 {
     return b1.foo(1) + d2.foo(2);
47 }

49 int main()
   {
51   Deriv d(9);
     Deriv* dp = new Deriv(5);
53   cout << footwo(d, *dp) << endl;
     cout << d.count << endl;
55   return 0;
   }
```