Olio-ohjelmointi: periytyminen ja virtuaalifunktiot

- Muodostinfunktio ei voi olla virtuaalinen, koska olion luokka ja luokkahierarkia on tunnettava muodostinta kutsuttaessa.
- Jos luokassa on virtuaalisia jäsenfunktioita, on myös tuhoajafunktion oltava virtuaalinen.
- Luokasta tulee monimuotoinen (=polymorfismi), kun luokassa on yksikin virtuaalinen jäsenfunktio.
- Ohjelman luokkia määriteltäessä tulisi aina miettiä, voidaanko jotain ohjelman luokkaa käyttää jossain monimuotoisena yliluokkana.
- Lähtökohtaisesti kannattaa tuhoajafunktio ja muut mahdollisesti aliluokissa korvattavat jäsenfunktiot määritellä virtuaalisiksi.
- Seuraavalla sivulla on esimerkki polymorfismista, yliluokan jäsenfunktion kutsumisesta ja virtuaalisen tuhoajafunktion toiminnasta.

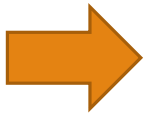
Olio-ohjelmointi: periytyminen ja virtuaalifunktiot

Tehtävä 1: Tee uusi projekti nimellä **VirtuaalinenKotitehtava (Non-Qt Project>Plain C++ Application)** ja kirjoita **main.cpp** tiedostoon alla olevat koodit.

- Seuraavalla sivulla selitetään ohjelman toimintaa. Tehtävässä kaikki koodi on **main.cpp** tiedostossa, suoraviivaisuuden vuoksi.

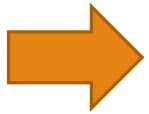
```
#include <iostream>
using namespace std;

class A
{
public:
   A()
   {
      cout << "A:n muodostin" << endl;
   }
   virtual ~A()
   {
      cout << "A:n tuhoaja" << endl;
   }
   virtual void f() const
   {
      cout << "A:n f()" << endl;
   }
   void gf()
   {
     cout << "A:n gf()" << endl;
   }
};
```



```
class B : public A
{
public:
   B()
   {
      cout << "B:n muodostin" << endl;
   }

   virtual ~B()
   {
       cout << "B:n tuhoaja" << endl;
   }
   virtual void f() const override
   {
     cout << "B:n f()" << endl;
   }
   void gf()
   {
     cout << ”B:n gf()" << endl;
   }
};
```



```
int main()
{
   A *object = new B();

   object->f();
   object->gf();

   delete object;

   return 0;
}
```

Tehtävä 1: Virtuaalisuus

- Ohjelmassa **VirtuaalinenKotitehtava** on kaksi luokkaa **A** ja **B**. Luokka **B** periin luokan **A**.
- Molemmilla luokilla on muodostin- ja tuhoajafunktiot, sekä jäsenfunktio **gf()** ja virtuaalinen jäsenfunktio **f()**.
- C++ -kielessä mahdollistetaan monipuolinen **polymorfisuus**: yliluokan tyyppiseen osoitinmuuttujaan voidaan viedä periytetyn olion osoite. Esimerkissä tämä on koodirivillä **A *object = new B();** Tämä on sallittu, koska **B** perii **A**:n.
- Kun ohjelmassa suoritetaan rivi **object->*f*();** kutsutaan luokan **B** jäsenfunktiota **f()**, koska luokkien välillä on perintäyhteys ja jäsenfunktio on *virtuaalinen*. Eli aliluokassa **B** on korvattu yliluokan **A** jäsenfunktio **f()**.
- Kun ohjelmassa suoritetaan rivi **object->g*f*();** kutsutaan yliluokan **A** jäsenfunktiota **gf()**, koska jäsenfunktio ei ole virtuaalinen.
- Jos muutat ohjelmassa koodirivin **A *object = new B();** muotoon **B *object = new B();** niin huomaat mitä ohjelmassa tapahtuu.
- Muuta koodirivi **B *object = new B();** muotoon **A *object = new B();** alla olevaa tehtävää varten

  - Jos otat pois tuhoajafunktioiden edessä olevat **virtual** sanat, niin huomaat ettei luokan **B** tuhoajafunktiota ajeta silloin ollenkaan. Tästä syystä luokkien tuhoajafunktiot on aina syytä asettaa virtuaalisiksi.