Tampereen yliopisto
Tampere University

Luokkahierarkiat

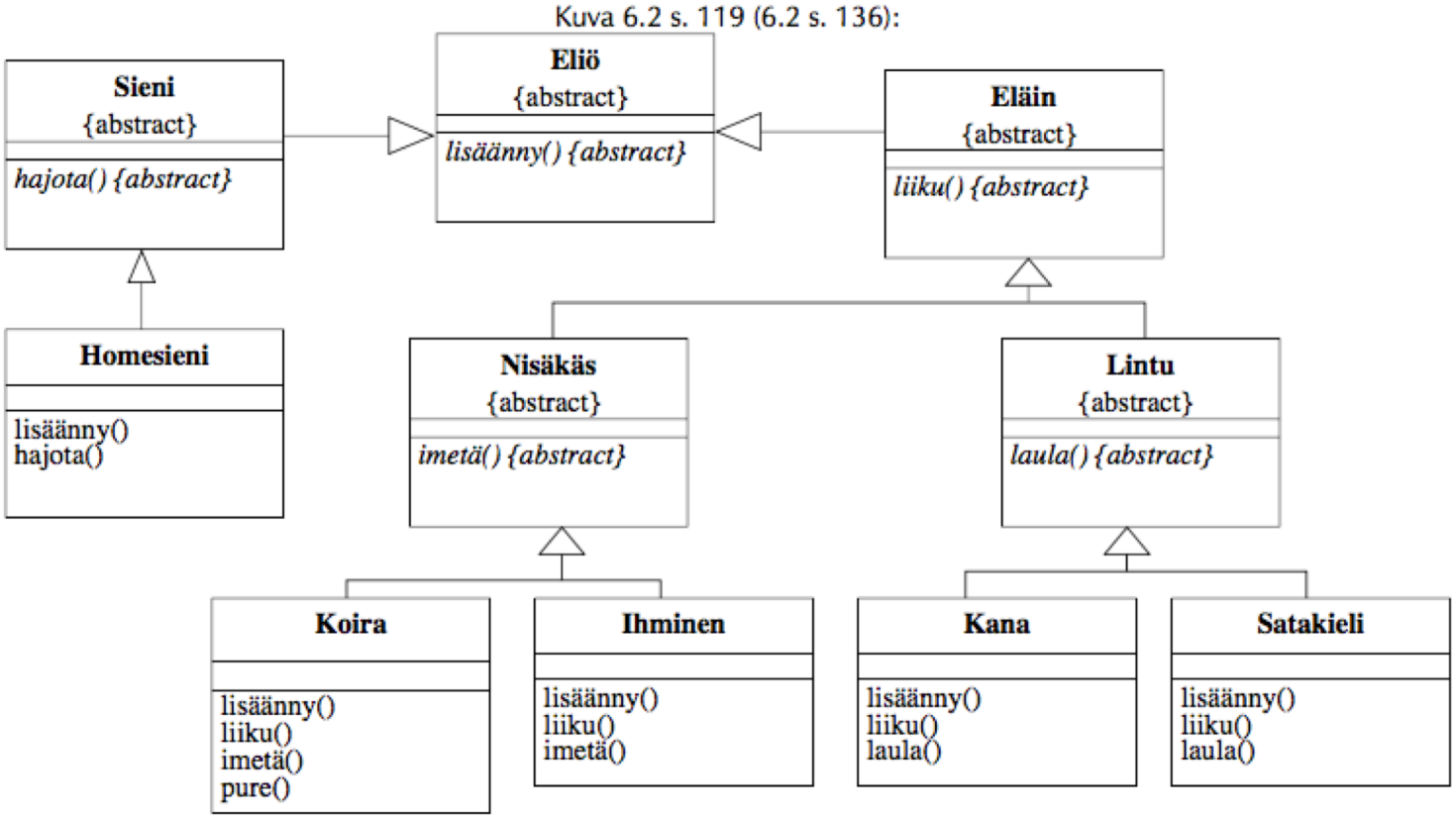
24.9.2019

Tampereen yliopisto
Tampere University

Periytyminen

- Asioiden luokittelu ja kategorisointi luonnollista ihmiselle
- Käytetty laajalti tieteissä, kielissä jne.
- Olio-ohjelmointi:
  - Ryhmittely yhteisten rajapintojen perusteella
  - Ryhmittely yhteisen toteutuksen perusteella
- Useimmissa kielissä molemmilla sama mekanismi: periytyminen (*inheritance*)

Luokkahierarkia



Luokkahierarkiat

- Erilaisia luokkien rooleja:
  - Rajapintaluokat
    - vain puhtaita virtuaalifunktioita
  - Abstraktit kantaluokat
    - ainakin yksi puhdas virtuaalifunktio
  - Konkreettiset luokat
    - ei puhtaita virtuaalifunktioita

Luokkahierarkiat

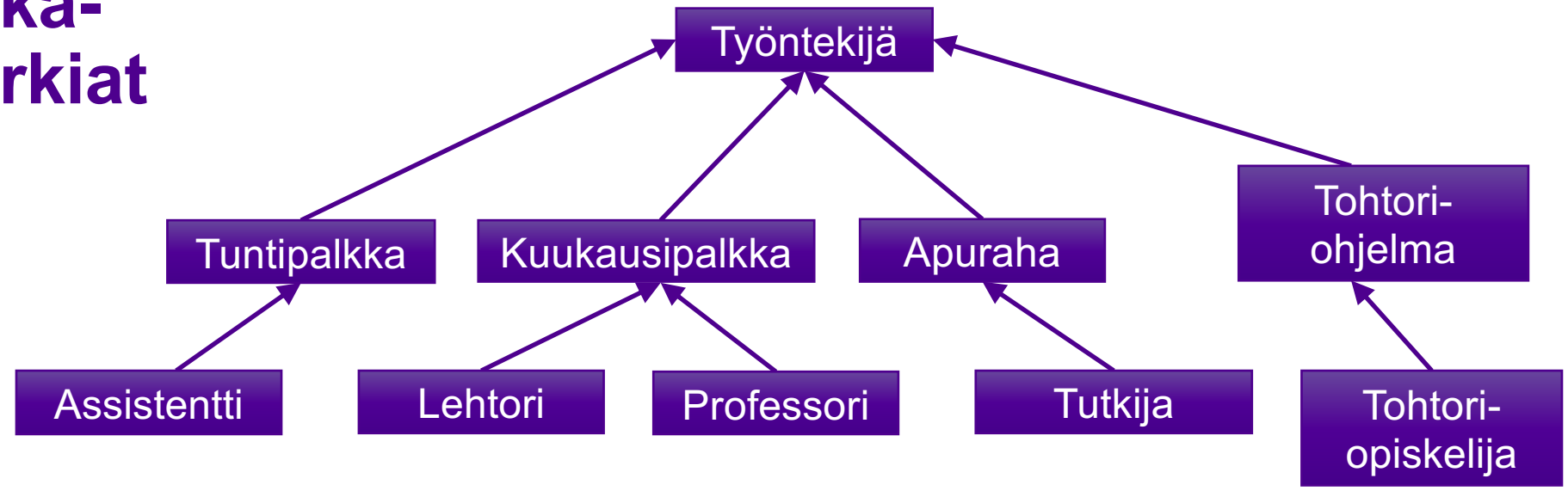
- Hierarkian yläosan luokat
  - “yläkäsitteitä”
  - Määräävät alempien luokkien rajapintaa
  - Polymorfismi: olioihin viittaaminen “yhteisellä nimellä”

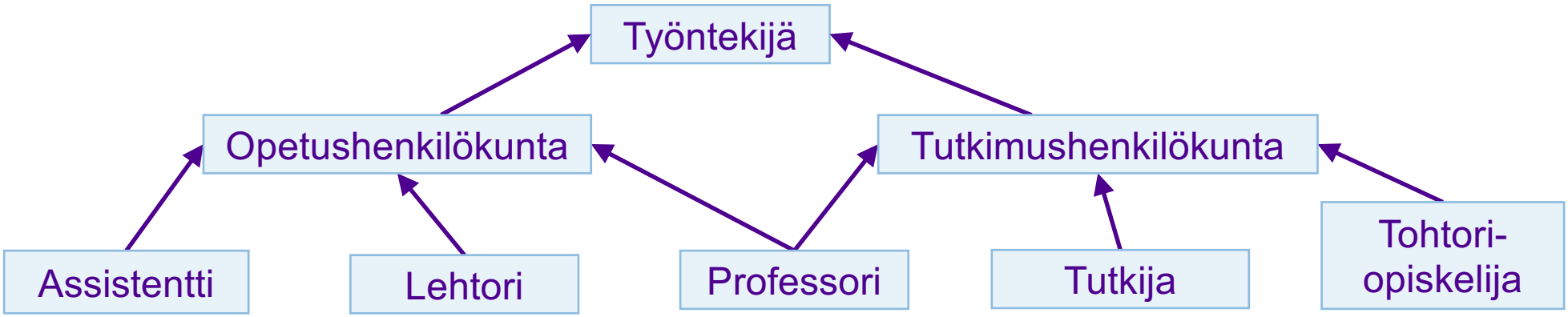
Luokkahierarkiat

- Hierarkian alaosan luokat
  - Palveluiden toteutus
  - Erikoistaminen (*specialization*)
  - Dynaaminen sitominen

Tampereen yliopisto
Tampere University

Luokka-hierarkiat





Periytyminen ja uudelleenkäyttö

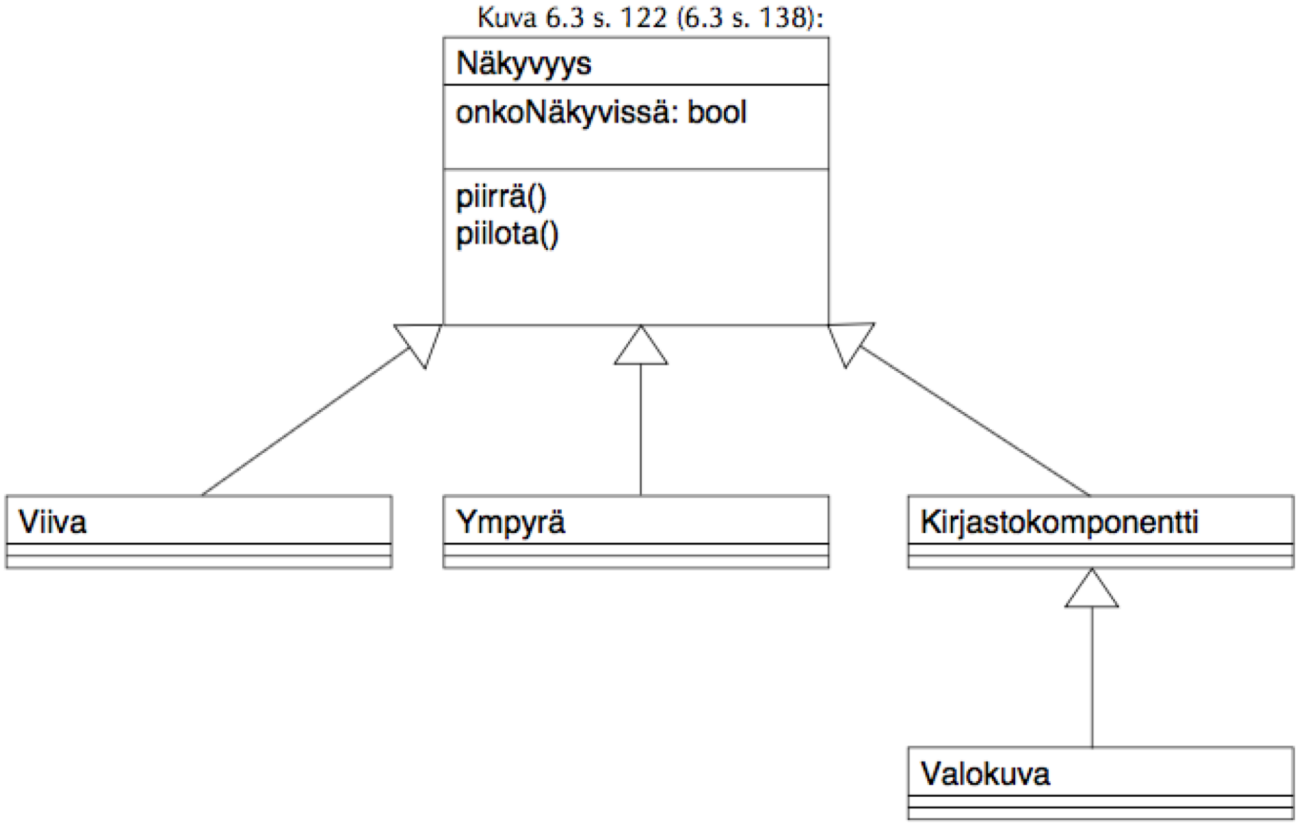
- Usein luokilla yhteisiä ominaisuuksia
- Yleistys (*generalization*)
  - Yhteiset osat omaan luokkaansa
  - Osat periyttämällä mukaan varsinaisiin luokkiin

Periytyminen ja uudelleenkäyttö

- Aliluokan ei tarvitse kirjoittaa uudelleen kantaluokan jo toteuttamia palveluita
  - Ohjelmakoodin uudelleenkäyttö ⇒ Ei koodin toistoa
- Vaarana koodin pirstoutuminen

Periytyminen ja uudelleenkäyttö

Kuva 6.3 s. 122 (6.3 s. 138):



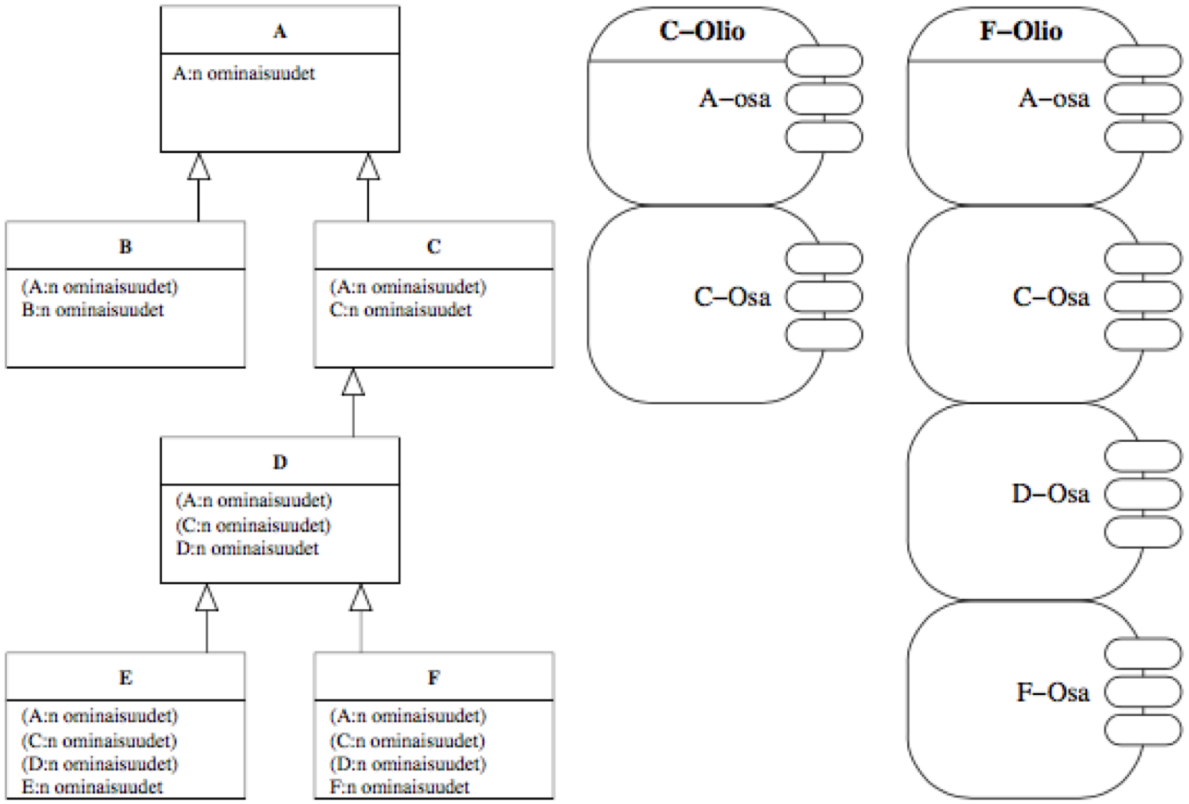
C++: Periytymisen perusteet

- Aliluokka “perii” kantaluokan kaikki ominaisuudet, voi myös lisätä uusia
  - kaikkiin kantaluokan ominaisuuksiin ei pääse käsiksi (suoraan)
- Moniperiytyminen
- Näkyvyysmääreet (public, private, protected) periytymisessä
  - ominaisuuksien näkyvyys
  - periytymistyyppi

Tampereen yliopisto
Tampere University

Periytyminen ja oliot

Kuva 6.4 s. 123 (6.4 s. 139):



C++: Periytymisen syntaksi

```
class A {
    // Luokan A ominaisuudet
};
class B : public A {
    // Luokan B A:han lisäämät ominaisuudet
};
class C : public A {
    // Luokan C A:han  lisäämät ominaisuudet
};
class D : public C {
    // Luokan D C:hen lisäämät ominaisuudet
};
```

C++: Periytymisen syntaksi

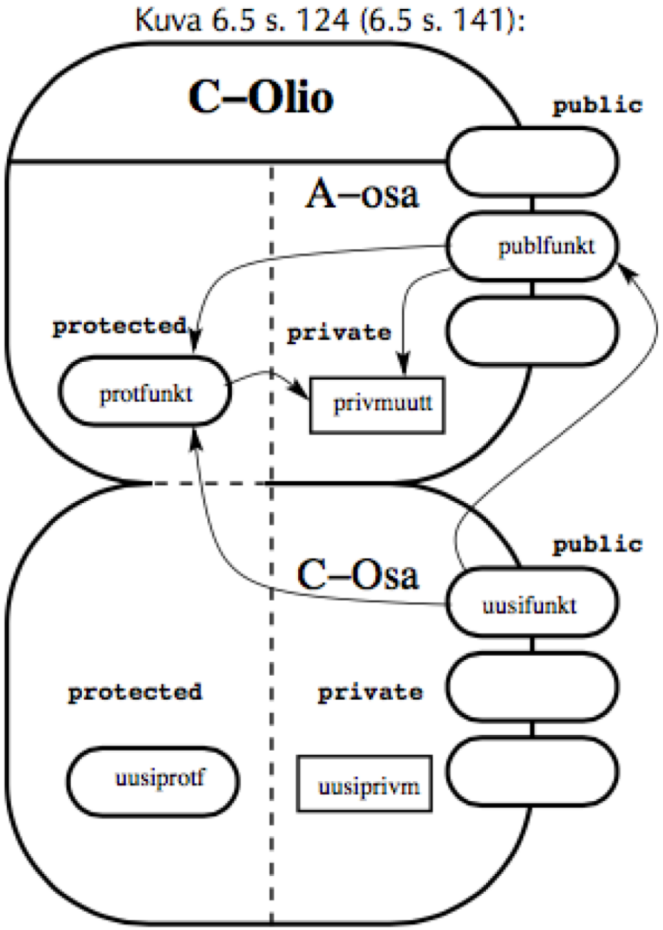
```
// ...
class E : public D {
    // Luokan E D:hen lisäämät ominaisuudet
};
class F : public D {
    // Luokan F D:hen lisäämät ominaisuudet
};
```

Tampereen yliopisto
Tampere University

Periytyminen

Periytyminen ja näkyvyys

Kuva 6.5 s. 124 (6.5 s. 141):



Tampereen yliopisto
Tampere University

Periytyminen ja rakentajat

- Kantaluokka ja aliluokan “lisäosa”
- Aliluokka ei pääse kantaluokan private-osiin
- Vastuualuejako:
  - Kantaluokan rakentaja – kantaluokan alustus
  - Aliluokan rakentaja – kantaluokan rakentajan kutsuminen, aliluokan lisäosan alustus
- Rakentamisjärjestys hierarkiassa ylhäältä alas

Kantaluokan rakentaja

```
class Lokiviesti
{
public:
    Lokiviesti(string const& viesti);
    // ...
private:
    string viesti_;
};

Lokiviesti::Lokiviesti(string const& viesti) : viesti_(viesti)
{
}
```

Aliluokan rakentaja

```
class PaivattyLokiviesti : public Lokiviesti {
public:
    PaivattyLokiviesti(Paivays const& pvm, string const& viesti);
    // ...
private:
    Paivays pvm_;
};

PaivattyLokiviesti::PaivattyLokiviesti(Paivays const& pvm,
                                       string const& viesti) :
                                       Lokiviesti(viesti), pvm_(pvm)
{
}
```

Tampereen yliopisto
Tampere University

Olioiden luominen

```
Lokiviesti viesti("Kävin leffassa.");


PaivattyLokiviesti pvmviesti(tanaan(),
"Huono oli.");
```

Periytyminen ja purkajat

- “Kerrosrakenne” kuten rakentajillakin
- Vastuualuejako:
  - Aliluokan purkaja – aliluokan lisäosan siivoaminen
  - Kantaluokan purkaja – kantaluokkaosan siivoaminen
- Purkujärjestys hierarkiassa alhaalta ylös
- Kutsutaan automaattisesti
- Kantaluokan purkaja aina *virtuaalinen*

Aliluokan olion ja kantaluokan suhde

- Kantaluokan rajapinta $\subset$ aliluokan rajapinta

$\Rightarrow$ Aliluokan olio “kelpaa” kantaluokan olioksi

  - Osoittimen päässä
  - Viitteen päässä

- Is-a: “*aliluokan olio on tyypiltään myös kantaluokan olio*”

Aliluokan olion ja kantaluokan suhde

```
class Kantaluokka { … };
class Aliluokka : public Kantaluokka { … };
void funktio ( Kantaluokka& kantaolio );

Kantaluokka* k_p = 0;
Aliluokka aliolio;
k_p = &aliolio;
funktio(aliolio);
```

Tampereen yliopisto
Tampere University

Periytymisen käyttö laajentamiseen

- Aliluokka lisää omia palveluitaan
- Kantaluokan palvelut sellaisenaan
  - mahdollistaa koodin uudelleenkäytön
- “Älä käytä periytymistä turhaan”
- Vaihtoehtona usein palveluiden lisääminen alkuperäiseen luokkaan

Luokan `Kirja` esittely

```
class Kirja {
public:
    Kirja(std::string const& nimi, std::string const& tekija);
    virtual ~Kirja();
    std::string annaNimi() const;
    std::string annaTekija() const;
private:
    std::string nimi_;
    std::string tekija_;
};
```

Luokan `Kirja` toteutus

```
Kirja::Kirja(string const& n, string const& t) : nimi_(n), tekija_(t) {
    cout << "Kirja " << nimi_ << " luotu" << endl;
}

Kirja::~Kirja() {
    cout << "Kirja " << nimi_ << " tuhottu" << endl;
}

string Kirja::annaNimi() const {
    return nimi_;
}
…
```

Luokan `Kirjastonkirja` esittely

```
class KirjastonKirja : public Kirja
{
public:
    KirjastonKirja(std::string const& nimi, std::string const& tekija,
                   Paivays const& palpvm);
    virtual ~KirjastonKirja();
    bool onkoMyohassa(Paivays const& tanaan) const;
private:
    Paivays palpvm_;
};
```

Luokan `Kirjastonkirja` toteutus

```
KirjastonKirja::KirjastonKirja(string const& nimi,
                               string const& tekija, Paivays const& palpvm) :
    Kirja(nimi, tekija), palpvm_(palpvm) {
    cout << "Kirjastonkirja " << nimi << " luotu" << endl;
}

KirjastonKirja::~KirjastonKirja() {
    cout << "Kirjastonkirja " << annaNimi() << " tuhottu" << endl;
}

bool KirjastonKirja::onkoMyohassa(Paivays const& tanaan) const {
    return palpvm_.paljonkoEdella(tanaan) < 0;
}
…
```