Kertaus - luokkakaavio ja oliokaavio

- Luokka, attribuutit, metodit
- Normaali yhteys
  - osallistumisrajoitteiden merkitseminen
  - yhteyden nimi, roolinimet
- Kompositio
  - käytetään vain jos olemassaoloriippuvuus
- Riippuvuus

Ohjelmistotekniikan menetelmät

luento 4, 22.11.2016

Perintä

Yleistys-erikoistus ja periminen

Perintä

Yleistys-erikoistus ja periminen

- Tähän mennessä tekemissämme luokkakaaviossa kaksi luokkaa ovat voineet liittyä toisiinsa muutamalla tapaa
- Yhteys ja kompositio liittyvät tilanteeseen, missä luokkien olioilla on rakenteellinen (= jollain lailla pysyvä) suhde, esim.:
  - Henkilö omistaa Auton (yhteys: normaali viiva)
  - Huoneet sijaitsevat Talossa (kompositio: musta salmiakki)
    - olemassaoloriippuvuus, eli salmiakin toisen pään olemassaolo riippuu salmiakkipäässä olevasta
    - Jos talo hajotetaan, myös huoneet häviävät, huoneita ei voi siirtää toiseen taloon
- Löyhempi suhde on taas riippuvuus, liittyy ohimenevämpiin suhteisiin, kuten tilapäiseen käyttösuhteeseen, esim:
  - AutotonHenkilö käyttää Autoa (katkoviivanuoli)
- Tänään tutustumme yhteen hieman erilaiseen luokkien väliseen suhteeseen, eli **yleistys-erikostussuhteeseen**, jonka vastine ohjelmoinnissa on periminen

Perintä

Yleistys-erikoistus ja periminen

- Ajatellaan luokkia Eläin, Kissa ja Koira
- Kaikki Koira-luokan oliot ovat selvästi myös Eläin-luokan oliota, samoin kaikki
- Kissa-luokan oliot ovat Eläin-luokan olioita
- Koira-oliot ja Kissa-oliot ovat taas täysin eriäviä, eli mikään koira ei ole kissa ja päinvastoin

Perintä

Yleistys-erikoistus ja periminen

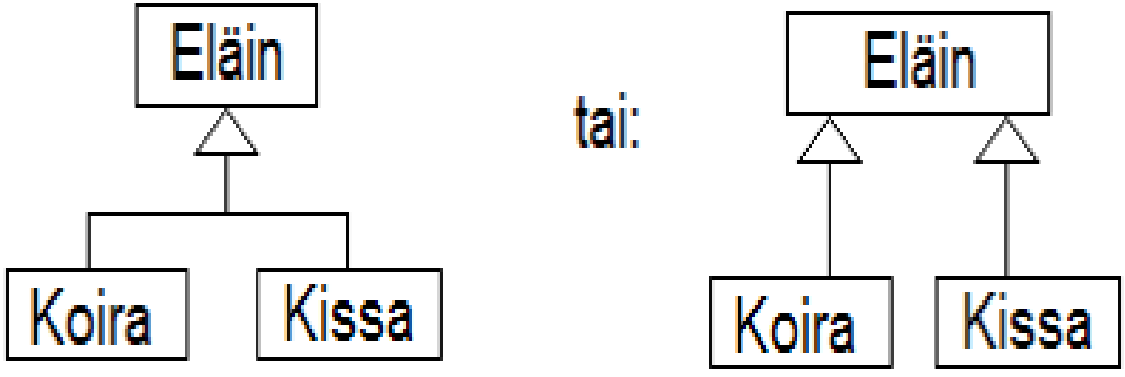
- Ajatellaan luokkia Eläin, Kissa ja Koira
- Kaikki Koira-luokan oliot ovat selvästi myös Eläin-luokan oliota, samoin kaikki
- Kissa-luokan oliot ovat Eläin-luokan olioita
- Koira-oliot ja Kissa-oliot ovat taas täysin eriäviä, eli mikään koira ei ole kissa ja päinvastoin
- Voidaankin sanoa, että luokkien Eläin ja Koira sekä Eläin ja Kissa välillä vallitsee yleistys-erikoistussuhde:
  - Eläin on **yliluokka** (engl. *Superclass*)
  - Kissa ja Koira ovat eläimen **aliluokkia** (engl. *Subclass*)
- Yliluokka Eläin siis määrittelee mitä tarkoittaa olla eläin
  - Kaikkien mahdollisten eläinten yhteiset ominaisuudet ja toiminnallisuudet
- Aliluokassa, esim. Koira tarkennetaan mitä muita ominaisuuksia ja toiminnallisuutta luokan olioilla eli Koirilla on kuin yliluokassa Eläin on määritelty

Perintä

Yleistys-erikoistus ja periminen

Aliluokat siis perivät (engl. *inherit*) yliluokan ominaisuudet ja toiminnallisuuden

- Luokkakaaviossa yleistyssuhde merkitään siten, että aliluokasta piirretään yliluokkaan kohdistuva nuoli, jonka päässä on iso ”valkoinen” kolmio
- Jos aliluokkia on useita, voivat ne jakaa saman nuolenpään tai molemmat omata oman nuolensa:



Perintä

Yleistys-erikoistus ja periminen

- Tarkkamuistisimmat huomaavat ehkä, että olemme jo törmänneet kurssilla yleistys-erikoisstussuhteeseen käyttötapausten yhteydessä
  - Sama valkoinen kolmiosymboli oli käytössä myös käyttötapausten yleistyksen yhteydessä
  - Yleistetty käyttötapaus opetustarjonnan ylläpito erikoistui kurssin perustamiseen, laskariryhmän perustamiseen ym..
  - Erikoistus myös mainittu edellisen yhteydessä

Luokkien välinen yleistys-erikoistussuhde eli yli- ja aliluokat toteutetaan ohjelmointikielissä siten, että aliluokka perii yliluokan

Perintä

Väripiste perii pisteen

Perintä

Väripiste perii pisteen

```
public class Piste{
    private int x, y;

    public void siirra(int dx, int dy) {
        x+=dx;
        y+=dy;
    }

    public String toSting(){
        return "(" + x + ")"
    }
}
```

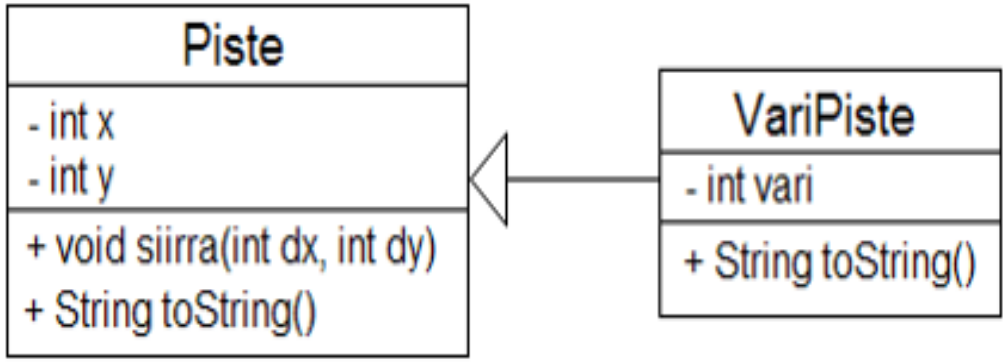
```
public class VariPiste extends Piste {
    private int vari;

    public String toSting(){
        return super.toString() + "c: " + vari;
    }
}
```

Perintä

Väripiste perii pisteen

```java
public class Piste{
    private int x, y;

    public void siirra(int dx, int dy) {
        x+=dx;
        y+=dy;
    }

    public String toSting(){
        return "(" + x + ")"
    }
}
```

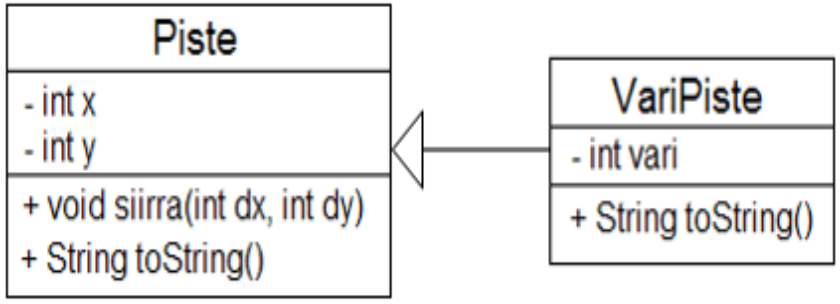
```java
public class VariPiste extends Piste {
    private int vari;

    public String toSting(){
        return super.toString() + "c: " + vari;
    }
}
```



Perintä

Väripiste perii pisteen

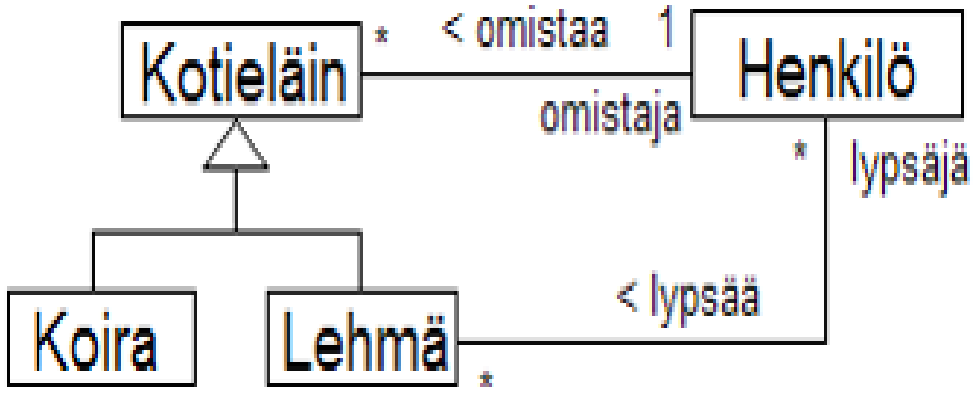
- Yliluokan Piste attribuutit x ja y sekä metodi `siirra()` siis periytyvät aliluokkaan VariPiste
  - Periytyviä attribuutteja metodeja ei merkitä aliluokan kohdalle
- Jos ollaan tarkkoja, Piste-luokan metodi toString periytyy myös VariPiste-luokalle, joka syrjäyttää (engl. *override*) perimänsä omalla toteutuksella
  - Korvaava `toString()`-metodi merkitään aliluokkaan VariPiste
- Eli kuviosta on pääteltävissä, että VariPisteellä on:
  - Attribuutit x ja y sekä metodi siirra perittynä
  - Attribuutti vari, jonka se määrittelee itse
  - Määritelty metodi toString joka syrjäyttää yliluokalta perityn
  - Koodista nähdään, että korvaava metodi käyttää yliluokassa määriteltyä metodia



Perintä

Mitä kaikkea periytyy?

- Luokat Koira ja Lehmä ovat molemmat luokan Kotieläin aliluokkia
- Jokaisella kotieläimellä on omistajana joku Henkilö-olio
- Koska omistaja liittyy kaikkiin kotieläimiin, merkitään yhteys Kotieläin- ja Henkilö-luokkien välille
  - **Yhteydet periytyvät aina aliluokille**, eli Koira-olioilla ja Lehmä-olioilla on omistajana yksi Henkilö-olio
- Ainoastaan Lehmä-olioilla on lypsäjiä
  - Yhteys lypsää tuleekin Lehmän ja Henkilön välille

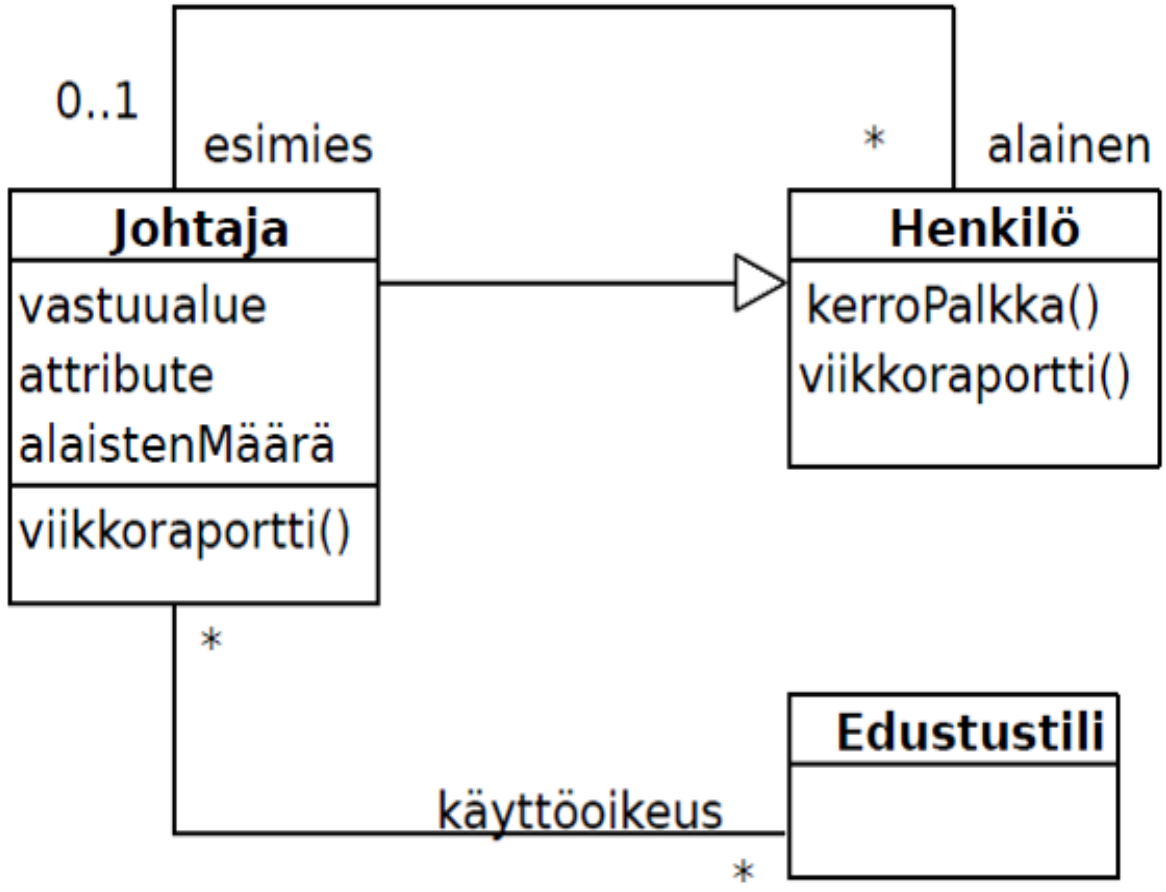


Perintä

Aliluokan ja yliluokan välinen yhteys

Uusi esimerkki:

- Yrityksen työntekijää kuvaa luokka Henkilö Henkilöllä on metodit kerroPalkka() ja viikkoraportti()
  - Henkilöllä on metodit kerroPalkka() ja viikkoraportti()
- Johtaja on Henkilön aliluokka
  - Johtajalla on alaisena useita henkilöitä
  - Henkilöllä on korkeintaan yksi johtaja esimiehenä
  - Johtajalla voi olla käyttöoikeuksia Edustustileihin
  - Edustustilillä on useita käyttöoikeuden omaavia johtajia
  - Johtajan viikkoraportti on erilainen kuin normaalin työntekijän viikkoraportti
- Tilannetta kuvaava luokkakaavio seuraavalla sivulla


0..1
esimies
*
alainen
Johtaja
vastuualue
attribute
alaistenMäärä
viikkoraportti()
Henkilö
kerroPalkka()
viikkoraportti()
*
Edustustili
käyttöoikeus
*

Perintä

Aliluokan ja yliluokan välinen yhteys

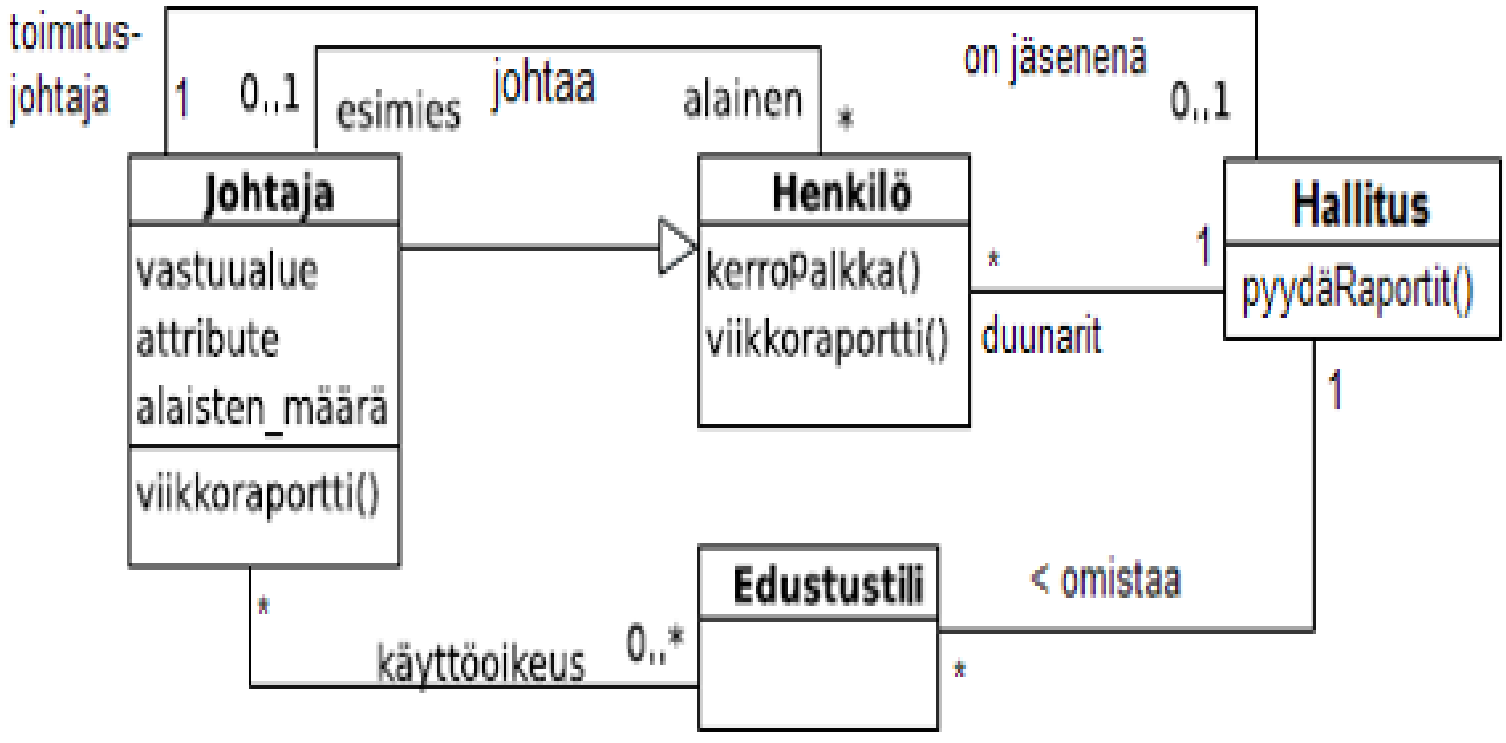
- Johtaja siis perii **kaiken** Henkilöltä
  - Henkilö on alainen-roolissa yhteydessä nollaan tai yhteen Johtajaan
  - Tästä seuraa, että myös Johtaja-olioilla on sama yhteys, eli myös johtajilla voi olla johtaja!
- Metodi `viikkoraportti` on erilainen johtajalla kuin muilla henkilöillä, siispä Johtaja-luokka korvaa Henkilö-luokan metodin omallaan
- Esim. Henkilö-luokan metodi `viikkoraportti`:
  - Kerro ajankäyttö työtehtäviin
- Johtaja-luokan korvaama metodi `viikkoraportti`:
  - Kerro ajankäyttö työtehtäviin
  - Laadi yhteenveto alaisten viikkoraporteista
  - Raportoi edustustilin käytöstä
- Yhteys käyttöoikeus Edustustileihin voi siis ainoastaan olla niillä henkilöillä, jotka ovat johtajia

Perintä

Aliluokan ja yliluokan välinen yhteys

Laajennetaan mallia:

- Yrityksen hallitus koostuu ulkopuolisista henkilöistä (joita ei sisällytetä malliin) ja yrityksen toimitusjohtajasta, joka siis kuuluu henkilöstöön
  - Hallitus on edustustilien omistaja
  - Hallitus "tuntee" toimitusjohtajaa lukuunottamatta kaikki työntekijät, myös normaalit johtajat ainoastaan Henkilö-olioina
- Hallitus pyytää työntekijöiltä viikkoraportteja
  - Viikkoraportin tekevät kaikki paitsi toimitusjohtaja


toimitus-
johtaja
1
0..1
esimies
johtaa
alainen
*
on jäsenenä
0..1
Johtaja
vastuualue
attribute
alaisten_määrä
viikkoraportti()
Henkilö
kerroPalkka()
viikkoraportti()
*
duunarit
1
Hallitus
pyydäRaportit()
1
*
käyttöoikeus
0..*
Edustustili
< omistaa
*

Abstraktit luokat

Abstraktit luokat

- Yliluokalla Kuvio on sijainti, joka ilmaistaan x- ja y-kordinaatteina sekä metodi `piirrä`
- Kuvion aliluokkia ovat Neliö ja Ympyrä
  - Neliöllä on sivun pituus ja Ympyrällä säde

Abstraktit luokat

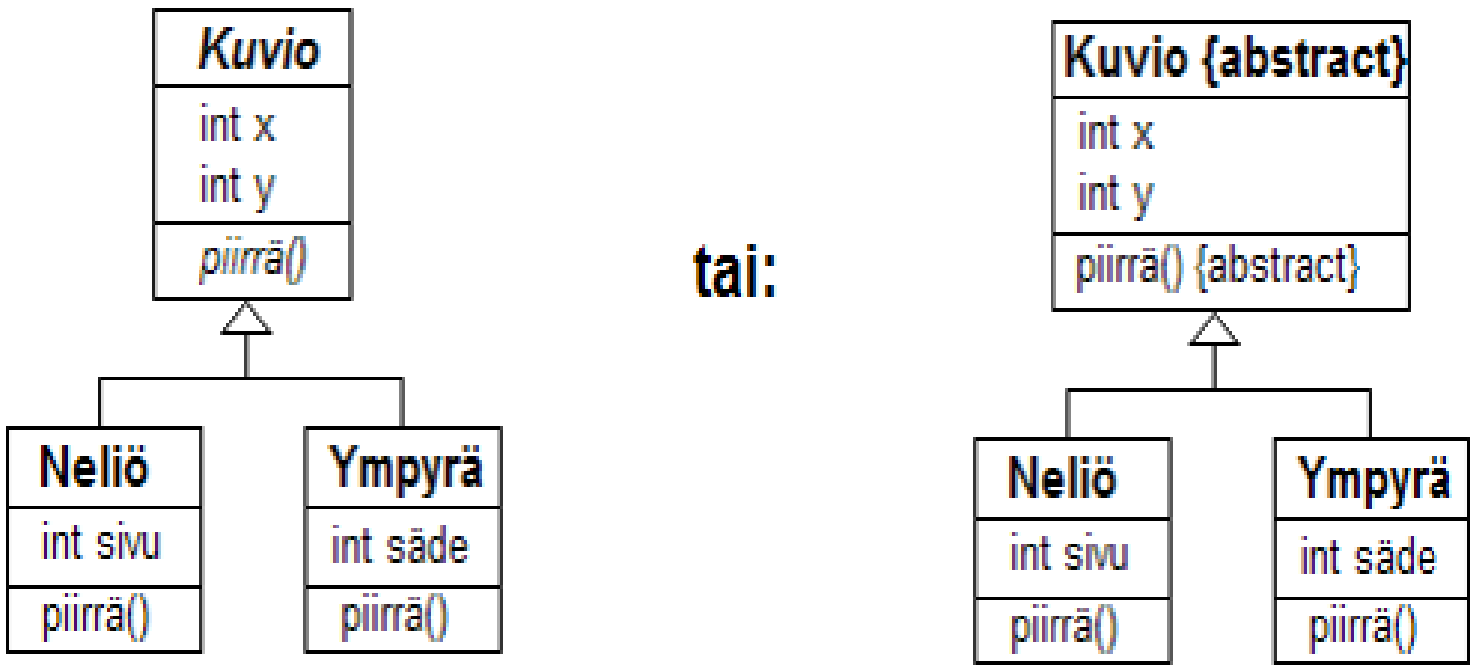
- Yliluokalla Kuvio on sijainti, joka ilmaistaan x- ja y-kordinaatteina sekä metodi `piirrä`
- Kuvion aliluokkia ovat Neliö ja Ympyrä
  - Neliöllä on sivun pituus ja Ympyrällä säde
- Kuvio on nyt pelkkä abstrakti käsite, Neliö ja Ympyrä ovat konkreettisia kuvioita jotka voidaan piirtää ruudulle kutsumalla sopivia grafiikkakirjaston metodeja
- Kuvio onkin järkevä määritellä **abstraktiksi luokaksi**, eli luokaksi josta ei voi luoda instansseja, joka ainoastaan toimii sopivana yliluokkana konkreettisille kuvioille

Abstraktit luokat

- Yliluokalla Kuvio on sijainti, joka ilmaistaan x- ja y-kordinaatteina sekä metodi `piirrä`
- Kuvion aliluokkia ovat Neliö ja Ympyrä
  - Neliöllä on sivun pituus ja Ympyrällä säde
- Kuvio on nyt pelkkä abstrakti käsite, Neliö ja Ympyrä ovat konkreettisia kuvioita jotka voidaan piirtää ruudulle kutsumalla sopivia grafiikkakirjaston metodeja
- Kuvio onkin järkevä määritellä **abstraktiksi luokaksi**, eli luokaksi josta ei voi luoda instansseja, joka ainoastaan toimii sopivana yliluokkana konkreettisille kuvioille
- Kuviolla on attribuutit `x` ja `y`, mutta metodi `piirrä` on abstrakti metodi, eli Kuvio ainoastaan määrittelee metodin nimen ja parametrien sekä paluuarvon tyypit, mutta metodille ei anneta mitään toteutusta
- Kuvion perivät luokat Neliö ja Ympyrä antavat toteutuksen abstraktille metodille

Abstraktit luokat

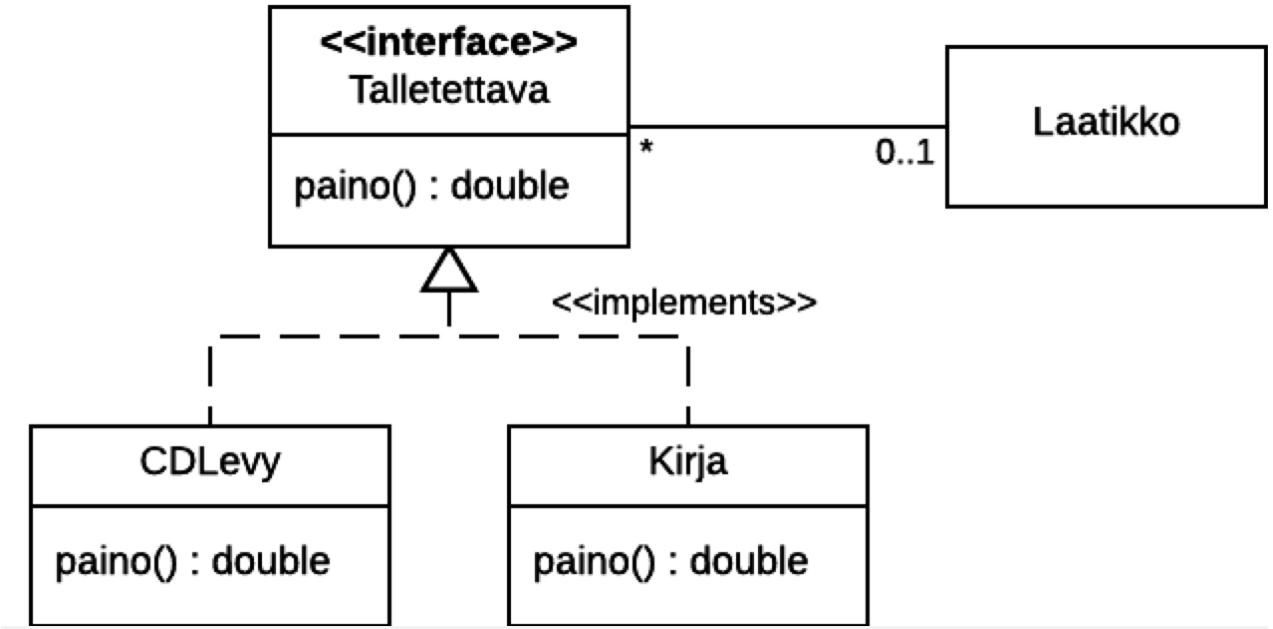
- Luokkakaaviossa on kaksi tapaa merkitä abstraktius
  - Abstraktin luokan/metodin nimi kursiivilla, tai
  - liitetään abstraktin luokan/metodin nimeen tarkenne abstract



Rajapinta UML-kaaviossa

Muistutuksena viime viikolta...

- Kuvataan lähes samalla tavalla kuin perintä
- Rajapinta kuvataan luokkana, johon liitetään tarkenne `<<interface>>`
  - Rajapinnalle on merkitty metodi *paino*, jonka se määrittelee
- Rajapinta ja toteuttavat luokat yhdistetään katkoviivalla, voidaan käyttää tarkennetta «implements»
- Rajapinnan toteuttaville luokille on merkitty myös metodi *paino*, sillä molemmat niistä toteuttavat sen omalla tavallaan



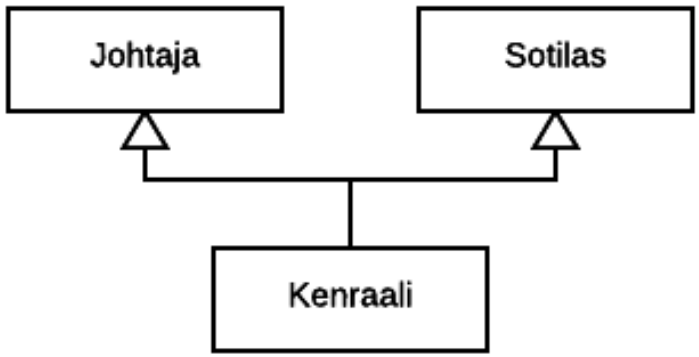
Moniperintä

Lisää perintää, monesti!

Moniperintä

Lisää perintää, monesti!

- Joskus tulee esiin tilanteita, joissa yhdellä luokalla voisi kuvitella olevan useita yliluokkia
- Esim. kenraalilla on sekä sotilaan, että johtajan ominaisuudet

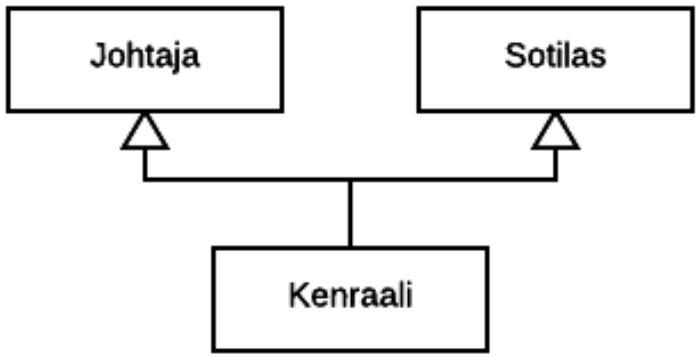


- Kyseessä moniperintä (engl. *multiple inheritance*)

Moniperintä

Lisää perintää, monesti!

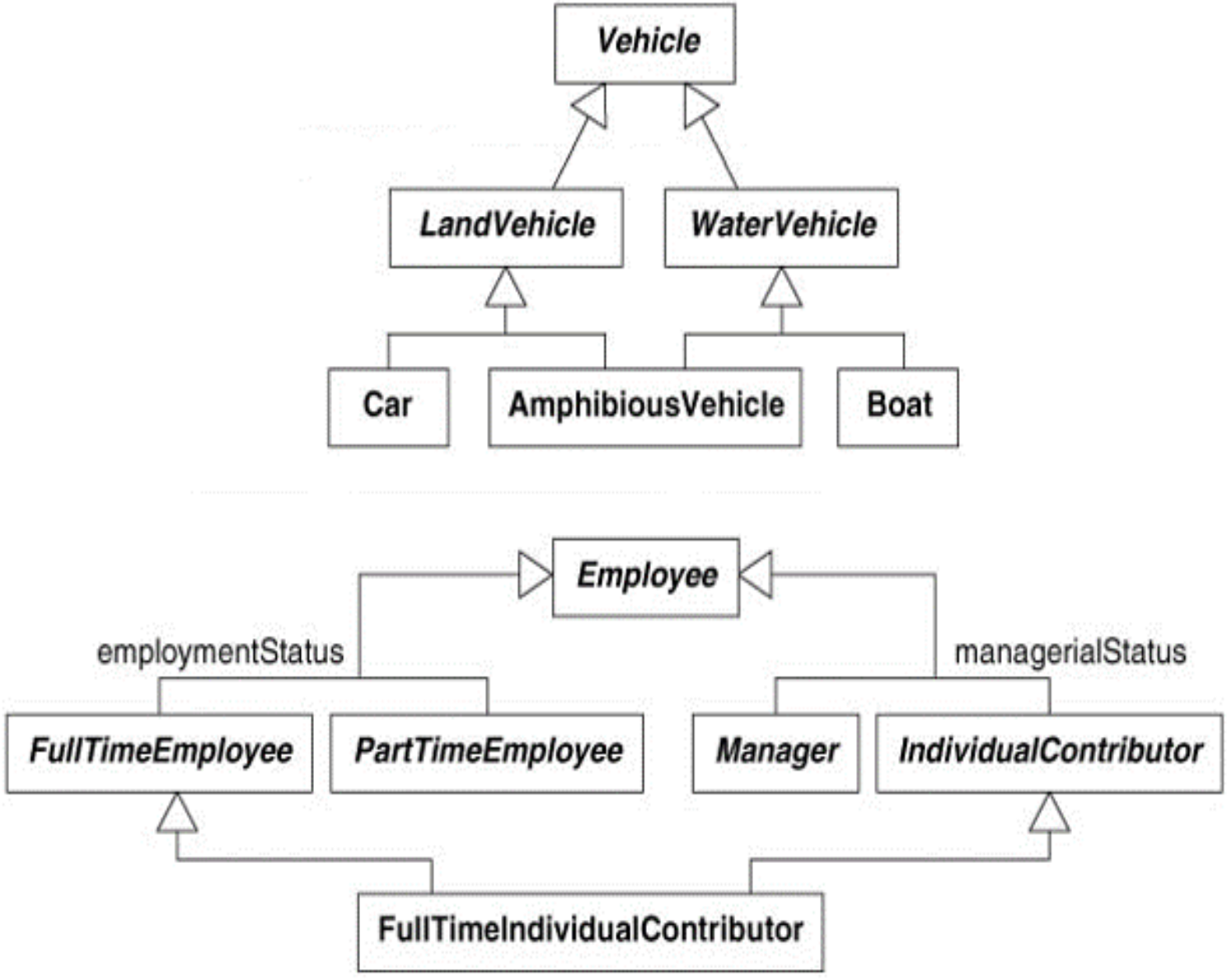
- Joskus tulee esiin tilanteita, joissa yhdellä luokalla voisi kuvitella olevan useita yliluokkia
- Esim. kenraalilla on sekä sotilaan, että johtajan ominaisuudet



- Kyseessä moniperintä (engl. *multiple inheritance*)
- Seuraavalla sivulla lisää esimerkkejä
  - Kulkuneuvo jakautuu maa- ja merikulkuneuvoksi - Auto on maakulkuneuvo, vene merikulkuneuvo, amfibio sekä maa- että merikulkuneuvo
  - Työntekijät voi jaotella kahdella tavalla: Pää- ja sivutoimiset ja Johtajat ja normaalit (Yksittäinen työntekijä voi sitten olla esim. päätoiminen johtaja)

Moniperintä

Lisää perintää, monesti!



Moniperintä

Moniperinnän ongelmat

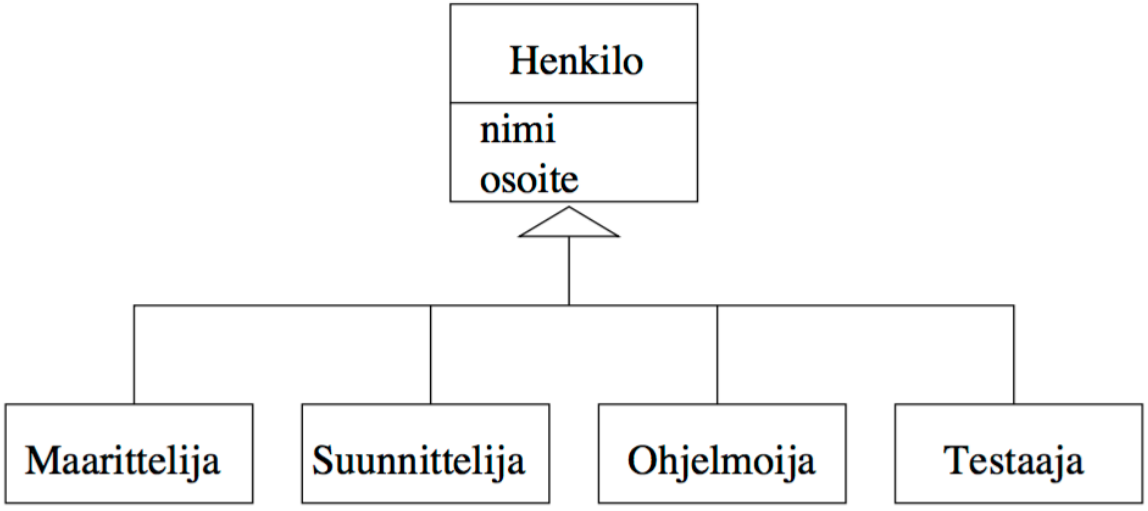
- Moniperintä on monella tapaa ongelmallinen asia ja useat kielet, **kuten Java eivät salli moniperintää**
  - C++ sallii moniperinnän
  - "moderneissa"kielissä kuten Python, Ruby ja Scala on olemassa ns. mixin-mekanismi, joka mahdollistaa "hyvinkäyttäytyvän"moniperintää vastavan mekanismin
- Useimmiten onkin viisasta olla käyttämättä moniperintää ja yrittää hoitaa asiat muin keinoin
- Näitä muita keinoja (jos unohdetaan mixin-mekanismi) ovat:
  - Moniperiytymisen korvaaminen yhteydellä, eli käytännössä "liittämällä"olioon toinen olio, joka laajentaa alkuperäisen olion toiminnallisuutta
  - Javan rajapinnan toimivat joissain tapauksessa moniperinnän korvikkeena varsinkin kun rajapinnat tukevat Java 8:n ilmestymisen jälkeen metodien oletusarvoisia toteutuksia

Esimerkki periytymisen virheellisestä käyttöyrityksestä

- Ohjelmistoyrityksessä työskentelee henkilöitä erilaisissa tehtävissä:
  - Määrittelijöinä, Suunnittelijoina, Ohjelmoijina ja Testaajina
- Yritys toteuttaa omia tarpeitaan varten henkilöstöhallintajärjestelmän
- Ensimmäinen yritys mallintaa yrityksen työntekijöitä alla
- Vaikuttaa loogiselta: esim. testaaja on Henkilö...

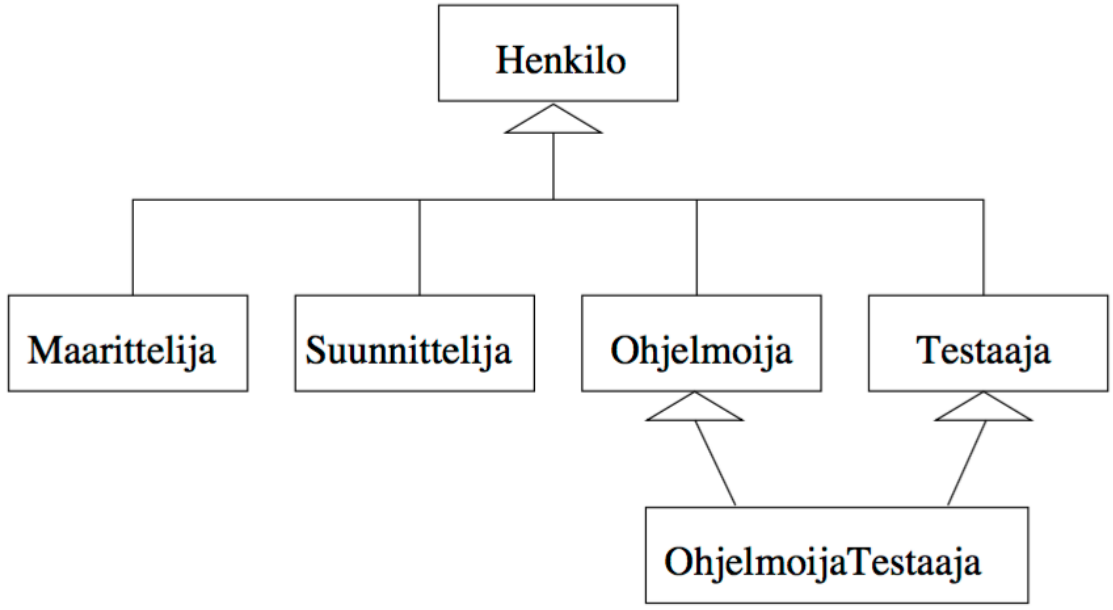
Esimerkki periytymisen virheellisestä käyttöyrityksestä

- Ohjelmistoyrityksessä työskentelee henkilöitä erilaisissa tehtävissä:
  - Määrittelijöinä, Suunnittelijoina, Ohjelmoijina ja Testaajina
- Yritys toteuttaa omia tarpeitaan varten henkilöstöhallintajärjestelmän
- Ensimmäinen yritys mallintaa yrityksen työntekijöitä alla
- Vaikuttaa loogiselta: esim. testaaja on Henkilö...

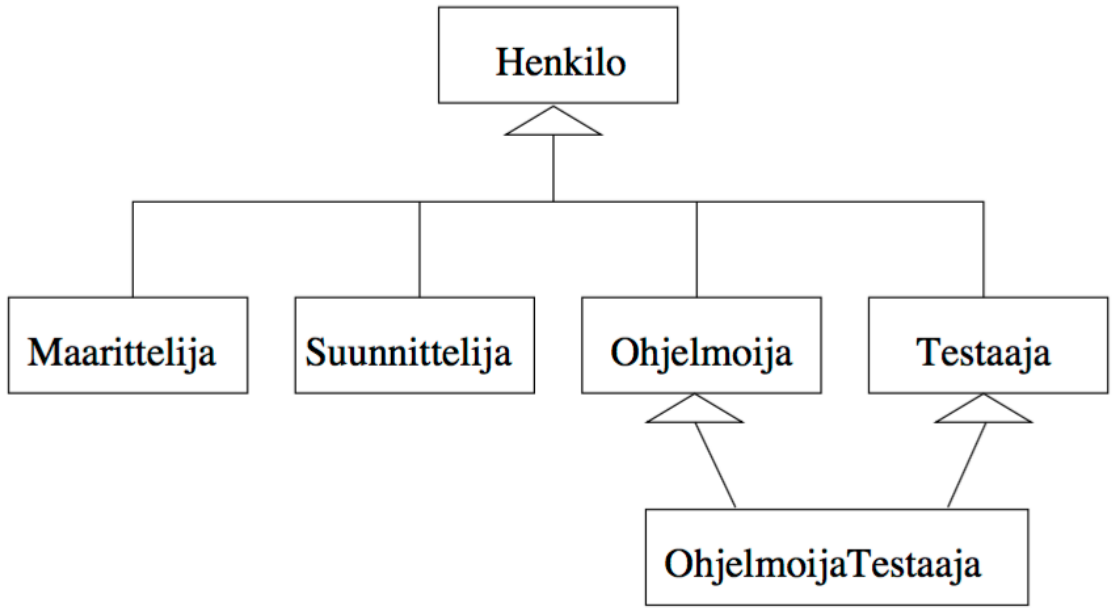


- Entä jos työntekijällä on useita tehtäviä hoidettavanaan?
  - Esim. ohjelmoiva testaaja
- Yksi vaihtoehto olisi mallintaa tilanne käyttämällä *moniperintää*:

- Entä jos työntekijällä on useita tehtäviä hoidettavanaan?
  - Esim. ohjelmoiva testaaja
- Yksi vaihtoehto olisi mallintaa tilanne käyttämällä *moniperintää*:



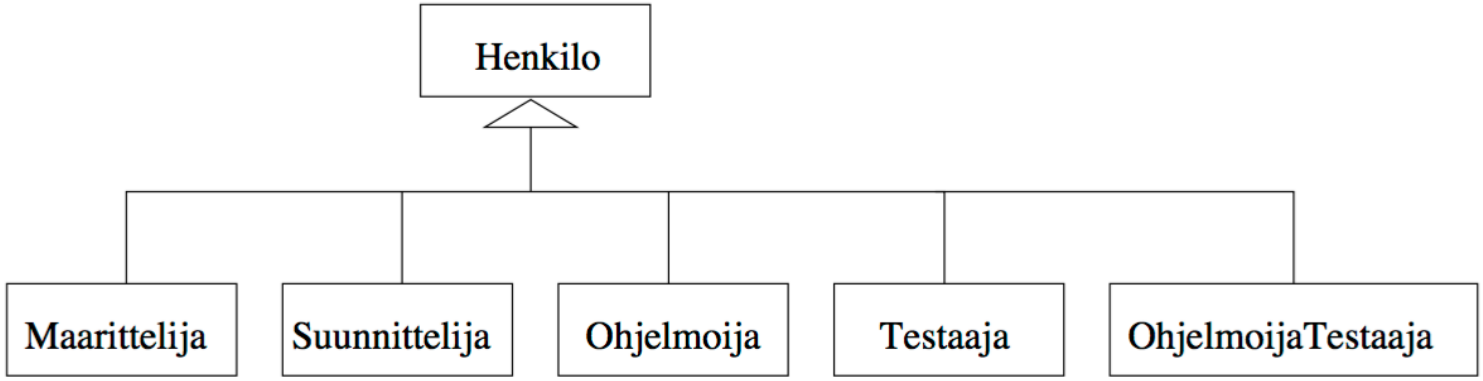
- Entä jos työntekijällä on useita tehtäviä hoidettavanaan?
  - Esim. ohjelmoiva testaaja
- Yksi vaihtoehto olisi mallintaa tilanne käyttämällä *moniperintää*:



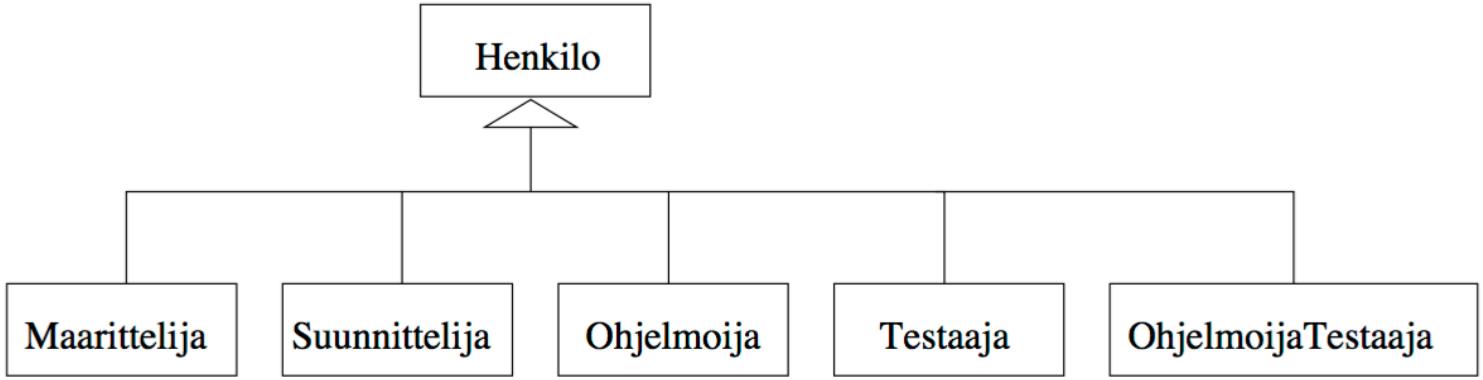
- Tämä on huono idea muutamastakin syystä
  - Jokaisesta työtehtäväkombinaatiosta pitää tehdä oma aliluokka
  - jos kaikki kombinaatiot otetaan huomioon, yhteensä luokkia tarvittaisiin 10 kappaletta
  - Kuten mainittua, esim. Javassa ei ole moniperintää

- Jos toteutuskieli ei tue moniperintää, yksi vaihtoehto on jokaisen työyhdistelmän kuvaaminen omana suoraan Henkilön alla olevana aliluokkana

- Jos toteutuskieli ei tue moniperintää, yksi vaihtoehto on jokaisen työyhdistelmän kuvaaminen omana suoraan Henkilön alla olevana aliluokkana



- Jos toteutuskieli ei tue moniperintää, yksi vaihtoehto on jokaisen työyhdistelmän kuvaaminen omana suoraan Henkilön alla olevana aliluokkana



- *Ratkaisu on erittäin huono*: nyt esim. OhjelmoijaTestaaja ei peri ollenkaan Ohjelmoija- eikä Testaaja-luokkaa
- Seurauksena se, että samaa esim. Ohjelmoija-luokaan liittyvää koodia joudutaan toistamaan moneen paikkaan
- Yksi suuri ongelma tässä ja edellisessä ratkaisussa on miten hoidetaan tilanne, jossa henkilö siirtyy esim. suunnittelijasta ohjelmoijaksi
  - Esim. Javassa olio ei voi muuttaa luokkaansa suoritusaikana: Suunnittelijaksi luodut pysyvät suunnittelijoina!

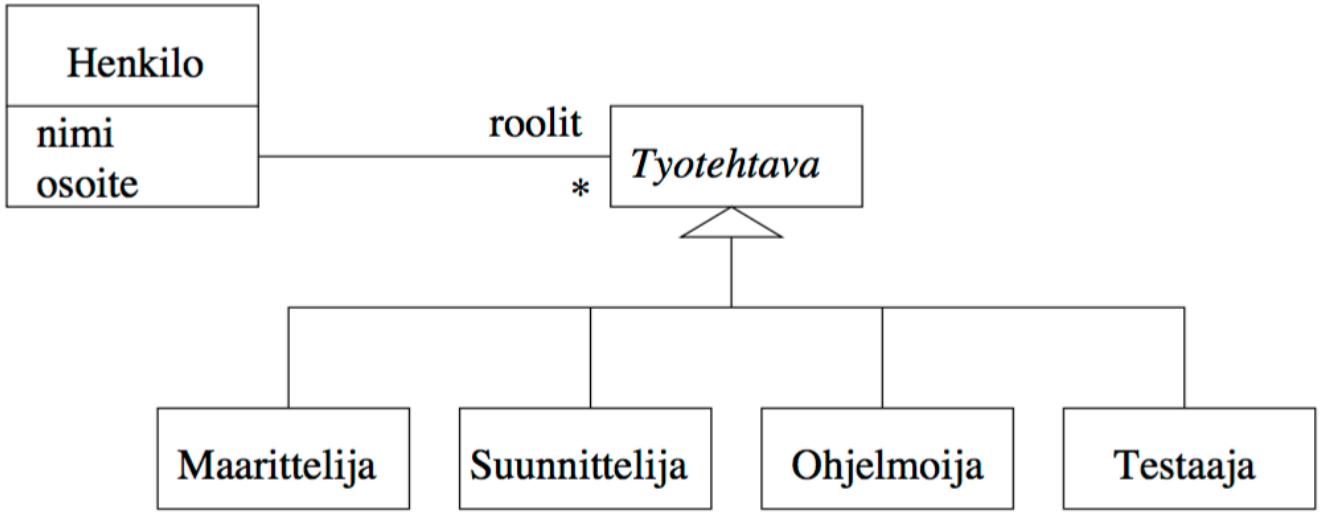
Roolin mallintaminen omana luokkana

- Henkilön työtehtävää voidaan ajatella henkilön *rooliksi* yrityksessä
  - Vaikuttaa siltä, että henkilön eri roolien mallintaminen ei kunnolla onnistu periytymistä käyttäen
- Parempi tapa mallintaa tilannetta on pitää luokka Henkilö kokonaan erillisenä ja liittää työtehtävät, eli henkilön roolit, siihen erillisinä luokkina

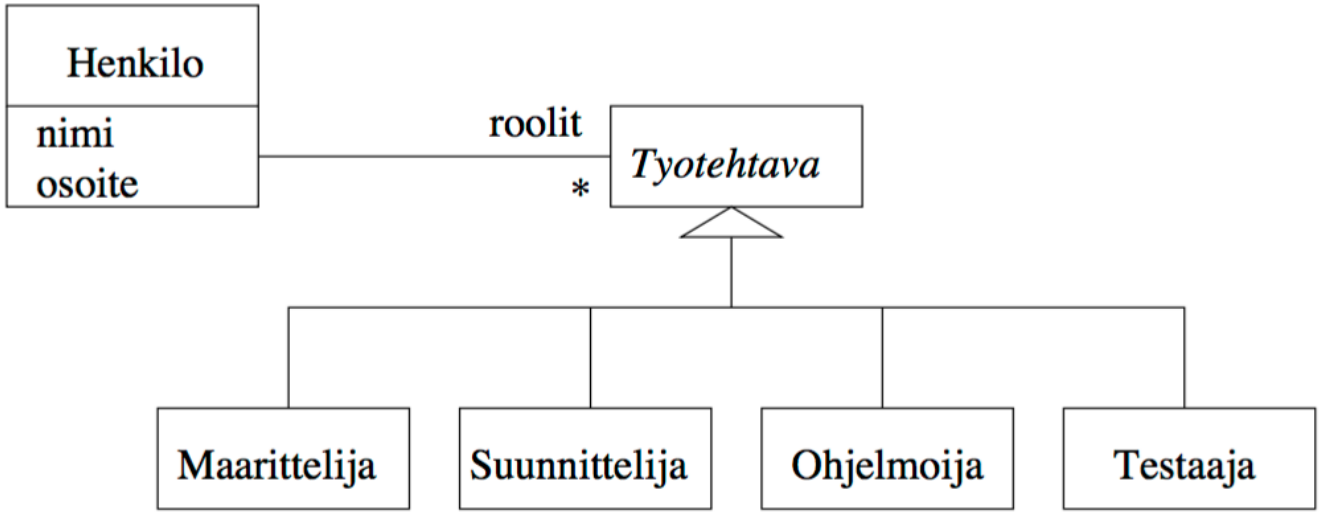
Roolin mallintaminen omana luokkana

- Henkilön työtehtävää voidaan ajatella henkilön *rooliksi* yrityksessä
  - Vaikuttaa siltä, että henkilön eri roolien mallintaminen ei kunnolla onnistu periytymistä käyttäen
- Parempi tapa mallintaa tilannetta on pitää luokka Henkilö kokonaan erillisenä ja liittää työtehtävät, eli henkilön roolit, siihen erillisinä luokkina
- Ratkaisu seuraavalla sivulla
  - Luokka Henkilö kuvaa siis "henkilöä itseään"ja sisältää ainoastaan henkilön "persoonaan"liittyvät tiedot kuten nimen ja osoitteen
  - Henkilöön liittyy yksi tai useampi Työtehtävä eli työntekijärooli
  - Työntekijäroolit on mallinnettu periytymishierarkian avulla, eli jokainen henkilöön liittyvä rooli on jokin konkreettinen työntekijärooli, esim. Ohjelmoija tai Testaaja

Roolin mallintaminen omana luokkana



Roolin mallintaminen omana luokkana



- Oikeastaan kaikki ongelmat ratkeavat tämän ratkaisun myötä
  - Henkilöön voi liittyä nyt kuinka monta roolia tahansa
  - Henkilön rooli voi muuttua: poistetaan vanha ja lisätään uusi rooli

Määrittelyvaiheen luokkakaavion laatiminen

Alustavan luokkamallin muodostaminen

Määrittelyvaiheen luokkakaavion laatiminen

Alustavan luokkamallin muodostaminen

- Kuten jo muutamaan kertaan on mainittu, olioperustaisessa ohjelmistokehityksessä pyritään muodostamaan koko ajan tarkentuva luokkamalli, joka *simuloi* sovelluksen kohdealuetta
  - Ensin luodaan **määrittelyvaiheen luokkamalli** sovelluksen käsitteistöstä
  - Suunnitteluvaiheessa tarkennetaan edellisen vaiheen luokkamalli **suunnitteluvaiheen luokkamalliksi**

Määrittelyvaiheen luokkakaavion laatiminen

Alustavan luokkamallin muodostaminen

- Kuten jo muutamaan kertaan on mainittu, olioperustaisessa ohjelmistokehityksessä pyritään muodostamaan koko ajan tarkentuva luokkamalli, joka *simuloi* sovelluksen kohdealuetta
  - Ensin luodaan **määrittelyvaiheen luokkamalli** sovelluksen käsitteistöstä
  - Suunnitteluvaiheessa tarkennetaan edellisen vaiheen luokkamalli **suunnitteluvaiheen luokkamalliksi**
- Tarkastellaan seuraavaksi miten alustava, määrittelyvaiheen luokkamalli voidaan muodostaa
  - Tunnistetaan sovellusalueen käsitteet ja niiden väliset suhteet
  - Eli karkeasti ottaen tehtävänä on etsiä todellisuutta simuloiva luokkarakenne

Määrittelyvaiheen luokkakaavion laatiminen

Alustavan luokkamallin muodostaminen

- Kuten jo muutamaan kertaan on mainittu, olioperustaisessa ohjelmistokehityksessä pyritään muodostamaan koko ajan tarkentuva luokkamalli, joka *simuloi* sovelluksen kohdealuetta
  - Ensin luodaan **määrittelyvaiheen luokkamalli** sovelluksen käsitteistöstä
  - Suunnitteluvaiheessa tarkennetaan edellisen vaiheen luokkamalli **suunnitteluvaiheen luokkamalliksi**
- Tarkastellaan seuraavaksi miten alustava, määrittelyvaiheen luokkamalli voidaan muodostaa
  - Tunnistetaan sovellusalueen käsitteet ja niiden väliset suhteet
  - Eli karkeasti ottaen tehtävänä on etsiä todellisuutta simuloiva luokkarakenne
- Menetelmästä käytetään nimitystä **käsiteanalyysi** (engl. *conceptual modeling*)
  - Järjestelmän sovellusalueen käsitteistöä kuvaavaa luokkamallia kutsutaan **kohdealueen luokkamalliksi** (engl. problem domain model)

Määrittelyvaiheen luokkakaavion laatiminen

Alustavan luokkamallin muodostaminen

- Menetelmän voi ajatella etenevän seuraavien vaiheiden kautta
  1. Kartoita luokkaehdokkaat
  2. Karsi luokkaehdokkaita
  3. Tunnista olioiden väliset yhteydet
  4. Lisää luokille attribuutteja
  5. Tarkenna yhteyksiä
  6. Etsi ”rivien välissä” olevia luokkia
  7. Etsi yläkäsitteitä
  8. Toista vaiheita 1-7 riittävän kauan
- Yleensä aloitetaan vaiheella 1 ja sen jälkeen edetään sopivalta tuntuvassa järjestyksessä
- On harvinaista, että ensimiettimältä päädytään *lopulliseen* ratkaisuun

Määrittelyvaiheen luokkakaavion laatiminen

Luokkaehdokkaiden kartoitus

- **(1)** Laaditaan lista tarkasteltavan sovelluksen kannalta keskeisistä asioista, kohteista ja ilmiöistä, joita ovat esim:
  - Toimintaan osallistujat
  - Toiminnan kohteet
  - Toimintaan liittyvät tapahtumat, materiaalit ja tuotteet ja välituotteet
  - Toiminnalle edellytyksiä luovat asiat
- Kartoituksen pohjana voi käyttää esim.
  - kehitettävästä järjestelmästä tehtyä vapaamuotoista tekstuaalista kuvausta tai
  - järjestelmän halutusta toiminnallisuudesta laadittuja käyttötapauksia
- **Luokkaehdokkaat** ovat yleensä järjestelmän toiminnan kuvauksessa esiintyviä **substantiiveja**

Määrittelyvaiheen luokkakaavion laatiminen

Luokkaehdokkaiden kartoitus

Etsitään käytettävissä olevista kuvauksista kaikki substantiivit ja otetaan ne alustaviksi luokkaehdokkaiksi:

- Tarkasteltavana ilmiönä on elokuvalipun varaaminen. Lippu oikeuttaa paikkaan tietyssä näytöksessä. Näytöksellä tarkoitetaan elokuvan esittämistä tietyssä teatterissa tiettyyn aikaan. Samaa elokuvaa voidaan esittää useissa teattereissa useina aikoina. Asiakas voi samassa varauksessa varata useita lippuja yhteen näytökseen.

Määrittelyvaiheen luokkakaavion laatiminen

Luokkaehdokkaiden kartoitus

Etsitään käytettävissä olevista kuvauksista kaikki substantiivit ja otetaan ne alustaviksi luokkaehdokkaiksi:

- Tarkasteltavana ilmiönä on elokuvalipun varaaminen. Lippu oikeuttaa paikkaan tietyssä näytöksessä. Näytöksellä tarkoitetaan elokuvan esittämistä tietyssä teatterissa tiettyyn aikaan. Samaa elokuvaa voidaan esittää useissa teattereissa useina aikoina. Asiakas voi samassa varauksessa varata useita lippuja yhteen näytökseen.

Määrittelyvaiheen luokkakaavion laatiminen

Luokkaehdokkaiden kartoitus

Löydettiin seuraavat substantiivit:

- elokuvalippu
- varaaminen
- lippu
- paikka
- näytös
- elokuva
- esittäminen
- teatteri
- aika
- asiakas
- varaus

Määrittelyvaiheen luokkakaavion laatiminen

Ehdokkaiden karsiminen

- **Näin pitkällä listalla on varmasti ylimääräisiä luokkakandidaatteja**
- **(2)** Karsitaan sellaiset jotka eivät vaikuta potentiaalisilta luokilta
- Kysymyksiä, joita karsiessa voi miettiä
  - Onko käsitteellä merkitystä järjestelmän kannalta
  - Onko kyseessä jonkin muun termin synonyymi
  - Onko kyseessä jonkin toisen käsitteen attribuutti
  - Liittyykö käsitteeseen tietosisältöä?
  - On tosin olemassa myös tietosisällöttömiä luokkia...
  - Onko käsitteeseen liittyvä tieto sellaista, että järjestelmän on *muistettava* tieto pitkiä aikoja
- Huomattavaa on, että kaikki luokat eivät yleensä edes esiinny sanallisissa kuvauksissa, vaan ne löytyvät vasta jossain myöhemmässä vaiheessa

Määrittelyvaiheen luokkakaavion laatiminen

Ehdokkaiden karsiminen

Löydettiin seuraavat substantiivit:

- ~~elokuvalippu~~ ← Termin lippu synonyymi
- ~~varaaminen~~ ← Tekemistä
- lippu
- paikka
- näytös
- elokuva
- ~~esittaminen~~ ← Tekemistä
- teatteri
- ~~aika~~ ← Attribuutti (näytöksen)
- asiakas
- varaus

Määrittelyvaiheen luokkakaavion laatiminen

Alustava yhteyksien tunnistaminen

- Kun luokat on alustavasti tunnistettu, kannattaa ottaa paperia ja kynä ja piirtää alustava luokkakaavio, joka koostuu vasta luokkalaatikoista
- **(3)** Tämän jälkeen voi ruveta miettimään minkä luokkien välillä on yhteyksiä
- Aluksi yhteydet voidaan piirtää esim. pelkkinä viivoina ilman yhteys- ja roolinimiä tai osallistumisrajoitteita
- Tekstuaalisessa kuvauksessa olevat **verbit ja genetiivit** viittaavat joskus olemassaolevaan **yhteyteen**
  - Lippu *oikeuttaa* paikkaan tietyssä näytöksessä
  - Näytöksellä *tarkoitetaan* elokuvan esittämistä tietyssä teatterissa tiettyyn aikaan
  - Samaa elokuvaa *voidaan* esittää useissa teattereissa useina aikoina.
  - Asiakas voi samassa varauksessa *varata* useita lippuja yhteen näytökseen

Määrittelyvaiheen luokkakaavion laatiminen

Alustava yhteyksien tunnistaminen

Huom: Kaikki verbit eivät ole yhteyksiä! Yhteydellä tarkoitetaan pysyvää suhdetta, usein verbit ilmentävät ohimeneviä asioita

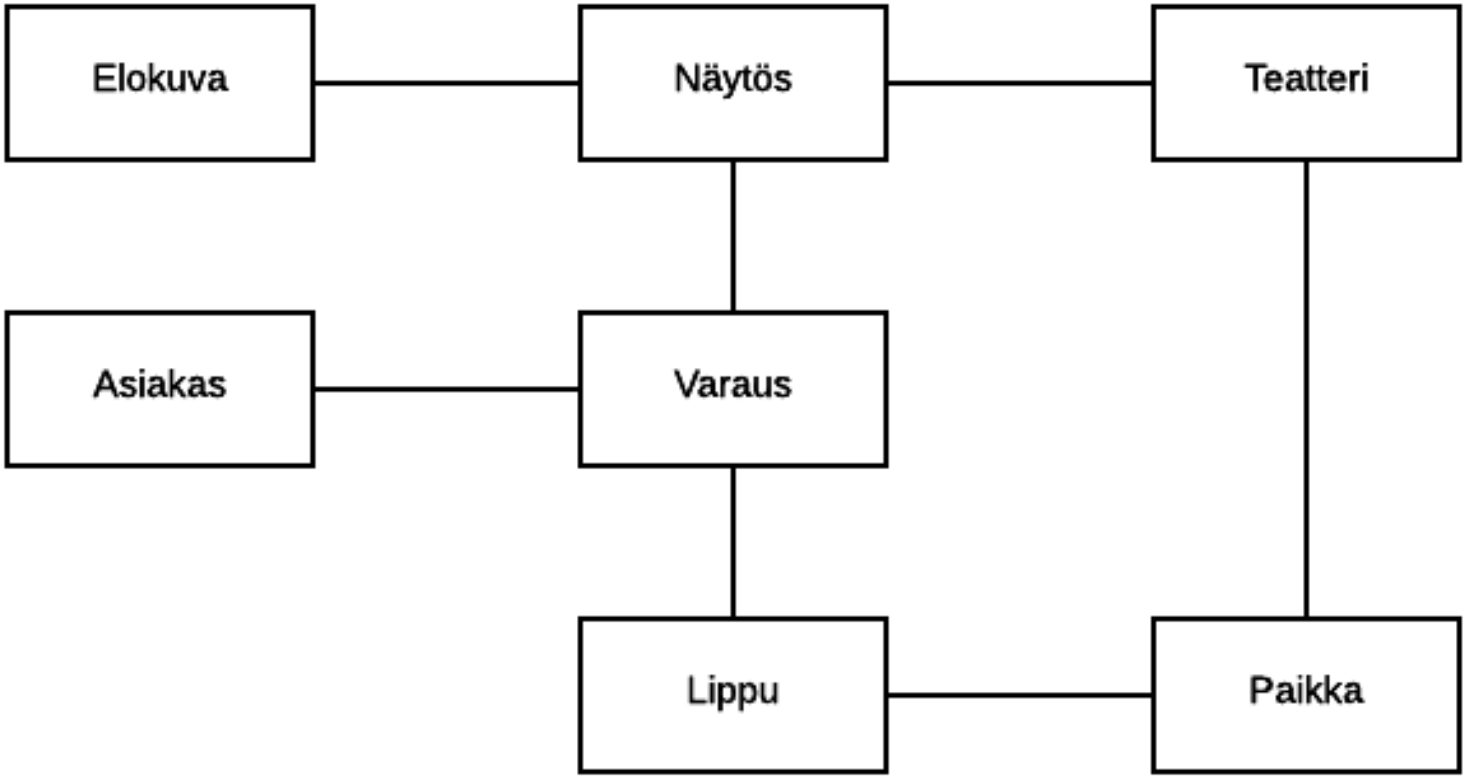
Löydettiin seuraavat yhteydet:

- Lippu – paikka
- Näytös – elokuva
- Näytös – teatteri
- Elokuva – teatteri
- Asiakas – varaus
- Asiakas – lippu
- Lippu – varaus

Määrittelyvaiheen luokkakaavion laatiminen

Alustava yhteyksien tunnistaminen

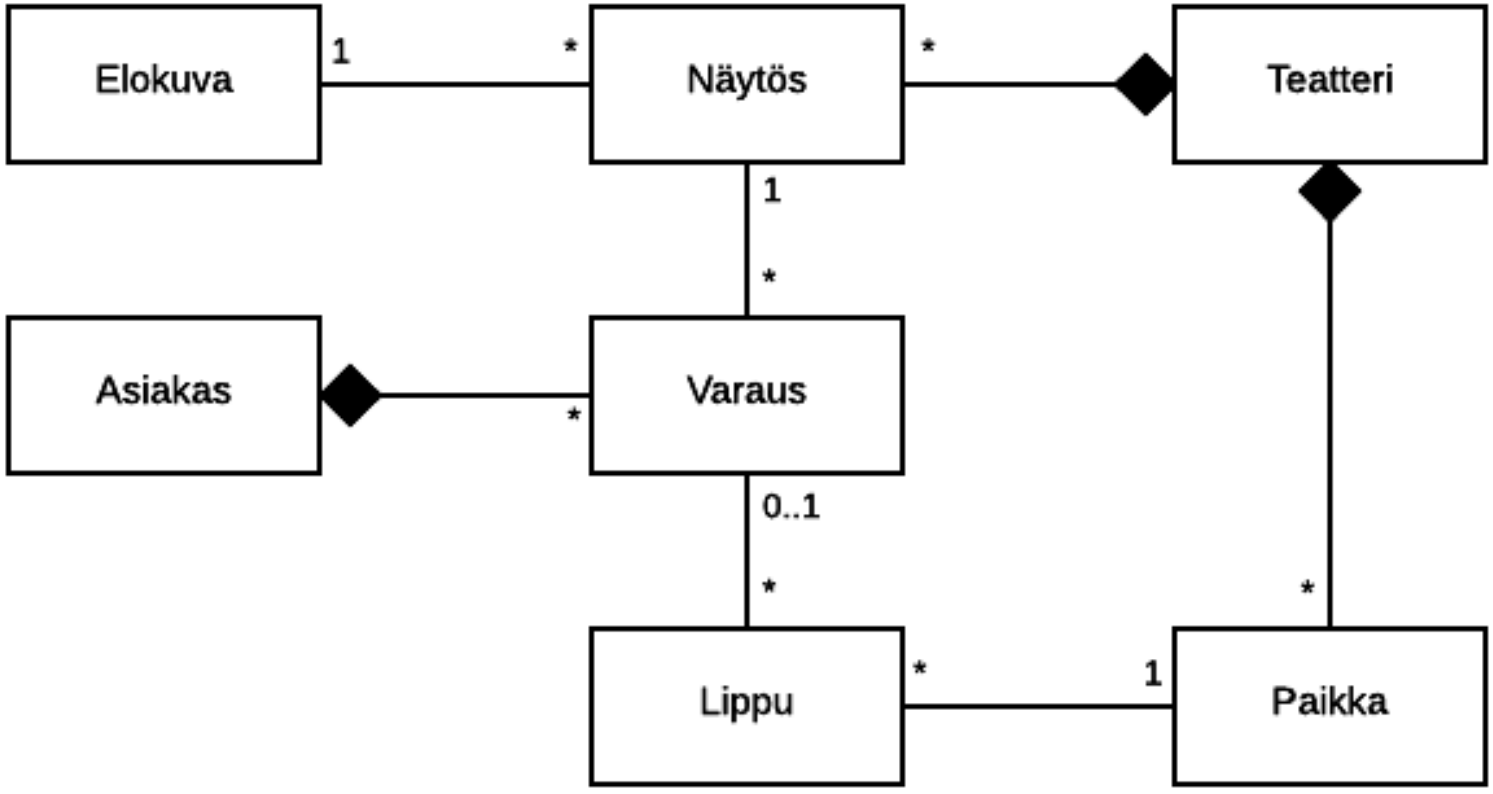
- **Seuraavaksi jonkinlainen hahmotelma.** Osa yhteyksistä siis edellisen listan ulkopuolelta, ”maalaisjärjellä” pääteltyjä:



Määrittelyvaiheen luokkakaavion laatiminen

Alustava yhteyksien tunnistaminen

- Kun yhteys tunnistetaan ja vaikuttaa tarpeelliselta, tarkennetaan yhteyden laatua ja kytkentärajoitetta
- Ei ole olemassa oikeaa etenemisstrategiaa...



Määrittelyvaiheen luokkakaavion laatiminen

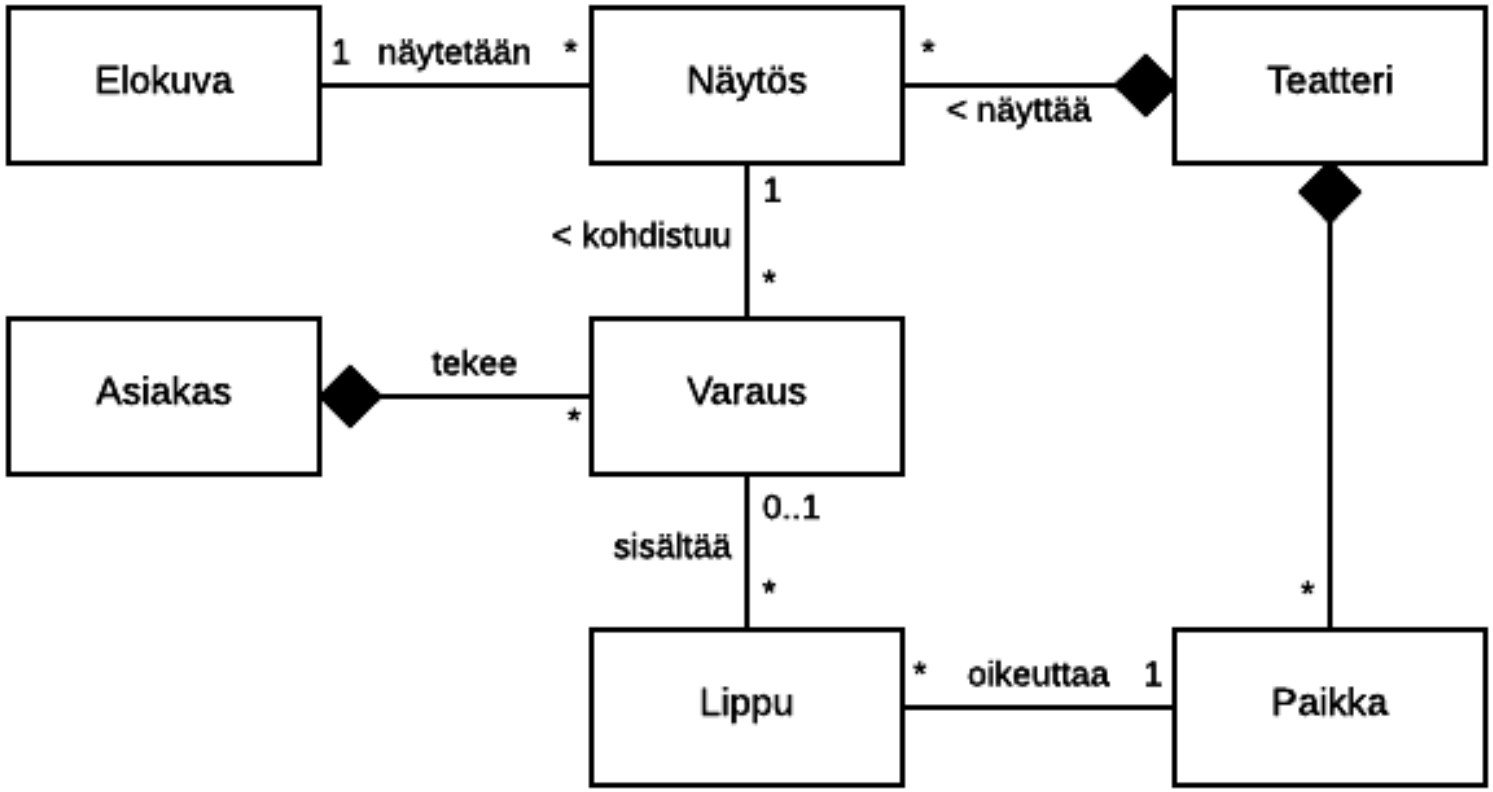
Yhteyksien tarkentaminen ja attribuuttien etsiminen

- **(4)** Attribuuttien löytäminen edellyttää yleensä lisätietoa, esim. asiakkaan haastatteluista
- Määrittelyvaiheen aikana tehtävää kohdealueen luokkamallia ei ole välttämättä tarkoituksenmukaista tehdä kaikin osin tarkaksi

Määrittelyvaiheen luokkakaavion laatiminen

Yhteyksien tarkentaminen ja attribuuttien etsiminen

- **(4)** Attribuuttien löytäminen edellyttää yleensä lisätietoa, esim. asiakkaan haastatteluista
- Määrittelyvaiheen aikana tehtävää kohdealueen luokkamallia ei ole välttämättä tarkoituksenmukaista tehdä kaikin osin tarkaksi
- **(5)** Tarkennetaan myös yhteyksiä



Määrittelyvaiheen luokkakaavion laatiminen

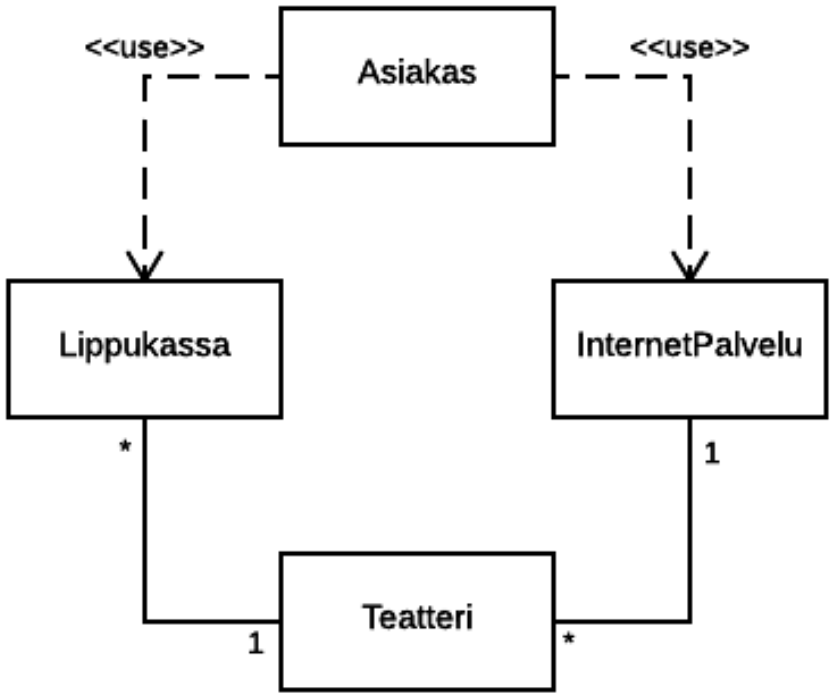
Mikä ei ole yhteys ja mikä on?

- Oletetaan, että lipunvaraustapahtuman tekstuaaliseen kuvaukseen liittyisi myös seuraava:
  - Asiakas tekee lippuvarauksen elokuvateatterin internetpalvelun kautta. Elokuvateatterissa on useita lippukassoja. Asiakas lunastaa varauksensa lippukassalta viimeistään tuntia ennen esitystä.
- Tästä kuvauksesta löytyy kaksi uutta luokkakandidaattia:
  - internetpalvelu
  - lippukassa
- Tekstuaalisen kuvauksen perusteella teatterilla on yhteys internetpalveluun sekä lippukassoihin
- Ovatko nämä *rakenteellisia* yhteyksiä, jotka voidaan merkitä myös luokkakaavioon?

Määrittelyvaiheen luokkakaavion laatiminen

Mikä ei ole yhteys ja mikä on?

- Tekstuaalinen kuvaus antaa viitteen, että asiakkaalla on yhteys sekä internetpalveluun että lippukassaan
- **Näitä ei kuvata** luokkamallissa yhteytenä sillä kyse ei ole rakenteisesta, pysyvälaatuisesta yhteydestä
- Jos se, että asiakas käyttää lippukassan palvelua halutaan merkata luokkakaavioon, tulee yhteyden sijasta käyttää riippuvuutta



Määrittelyvaiheen luokkakaavion laatiminen

Mikä ei ole yhteys ja mikä on?

- Edellisen kalvon kaltaisessa tilanteessa ei olisi mielekästä merkitä Asiakkaasta riippuvuutta Lippukassaan tai Internetpalveluun
- Luokka Asiakas on järjestelmässä oikean elokuvateatterin asiakkaan representaatio, joka ei kuitenkaan itsessään "suorita"mitään toimenpiteitä

Määrittelyvaiheen luokkakaavion laatiminen

Mikä ei ole yhteys ja mikä on?

- Edellisen kalvon kaltaisessa tilanteessa ei olisi mielekästä merkitä Asiakkaasta riippuvuutta Lippukassaan tai Internetpalveluun
- Luokka Asiakas on järjestelmässä oikean elokuvateatterin asiakkaan representaatio, joka ei kuitenkaan itsessään "suorita"mitään toimenpiteitä
- Hieman samaan tapaan yliopiston kuhunkin opiskelijaan liittyy OODI-järjestelmässä oma "olio", kuitenkaan tuo OODI:ssa oleva olio ei tee mitään, esim. käy luennoilla tai suorita kursseja
- Luokkamallin yhteyksissä siis ei tule kuvata toiminnallisuutta, tyyliin Asiakas tekee varauksen Internetpalvelussa tai Opiskelija käy kurssin Luennolla vaan olioiden välisiä suhteita (jotka voivat olla toiminnan seuraus):

Määrittelyvaiheen luokkakaavion laatiminen

Mikä ei ole yhteys ja mikä on?

- Edellisen kalvon kaltaisessa tilanteessa ei olisi mielekästä merkitä Asiakkaasta riippuvuutta Lippukassaan tai Internetpalveluun
- Luokka Asiakas on järjestelmässä oikean elokuvateatterin asiakkaan representaatio, joka ei kuitenkaan itsessään "suorita"mitään toimenpiteitä
- Hieman samaan tapaan yliopiston kuhunkin opiskelijaan liittyy OODI-järjestelmässä oma "olio", kuitenkaan tuo OODI:ssa oleva olio ei tee mitään, esim. käy luennoilla tai suorita kursseja
- Luokkamallin yhteyksissä siis ei tule kuvata toiminnallisuutta, tyyliin Asiakas tekee varauksen Internetpalvelussa tai Opiskelija käy kurssin Luennolla vaan olioiden välisiä suhteita (jotka voivat olla toiminnan seuraus):

Suurin osa varsinaisen toiminnallisuuden suorittavista olioista tulee vasta suunnittelutason luokkamalleihin.

Määrittelyvaiheen luokkakaavion laatiminen

Yläkäsitteiden etsiminen

- Seuraavaksi täytyy **(7)** etsiä yläkäsitteet
- **Lyhyesti:** jos tekisimme yleistä lippupalvelujärjestelmää, olisi **lippu** todennäköisesti yläkäsite, joka erikoistuu esim. elokuvalipuksi, konserttilipuksi, ym...

Määrittelyvaiheen luokkakaavion laatiminen

Yläkäsitteiden etsiminen

- Seuraavaksi täytyy **(7)** etsiä yläkäsitteet
- **Lyhyesti:** jos tekisimme yleistä lippupalvelujärjestelmää, olisi **lippu** todennäköisesti yläkäsite, joka erikoistuu esim. elokuvalipuksi, konserttilipuksi, ym...
- Määrittelyvaiheen aikana tehtävään sovelluksen kohdealueen luokkamalliin ei vielä liitetä mitään metodeja
  - Metodien määrittäminen tapahtuu vasta ohjelman suunnitteluvaiheessa
  - Palaamme aiheeseen myöhemmin
  - Suunnitteluvaiheessa luokkamalli tarkentuu muutenkin monella tapaa

Kirjasto

Toinen esimerkki määrittelyvaiheen luokkakaavion laatimisesta

- Tavoitteena on määritellä ja suunnitella tietojärjestelmä, jonka avulla hallitaan kirjaston lainaustapahtumia. Asiakasta haastattelemalla on kerätty lista järjestelmältä toivotusta toiminnallisuudesta
- Lista toiminnallisuuksista löytyy seuraavalta sivulta

Kirjasto

Toinen esimerkki määrittelyvaiheen luokkakaavion laatimisesta

- Tavoitteena on määritellä ja suunnitella tietojärjestelmä, jonka avulla hallitaan kirjaston lainaustapahtumia. Asiakasta haastattelemalla on kerätty lista järjestelmältä toivotusta toiminnallisuudesta
- Lista toiminnallisuuksista löytyy seuraavalta sivulta
- Voisimme muodostaa listan perusteella alustavan käyttötapausmallin kirjaston vaatimuksista
- Emme tänään kuitenkaan mene käyttötapuksiin vaan muodostamme vaatimuslistan perusteella kirjastoa mallintavan määrittelyvaiheen luokkakaavion

Kirjasto

Lista toivotusta toiminnallisuudesta

- Kirjasto lainaa alussa vain kirjoja, myöhemmin ehkä muitakin tuotteita, kuten CD- ja DVD-levyjä
- Yksittäistä kirjanimikettä voi olla useampia kappaleita
- Kirjastoon hankitaan uusia kirjoja ja kuluneita tai hävinneitä kirjoja poistetaan
- Kirjastonhoitaja huolehtii lainojen, varausten ja palautusten kirjaamisesta
- Kirjastonhoitaja pystyy ylläpitämään tietoa lainaajista sekä nimikkeistä
- Nimikkeen voi varata jos yhtään kappaletta ei ole paikalla
- Varaus poistuu lainan yhteydessä tai erikseen peruttaessa
- Lainaajat voivat selata valikoimaa kirjastossa olevilla päätteillä
- Kirjauduttuaan päätteelle omalla kirjastokortin numerolla on lainaajan myös mahdollista selailla omia lainojaan
- Kirjaston päätteiden tarjoama toiminnallisuus on asiakkaiden käytössä myös Web-selaimen avulla

Kirjasto

Etsitään substantiivit

Kirjasto

Etsitään substantiivit

- **Kirjasto** lainaa alussa vain **kirjoja**, myöhemmin ehkä muitakin tuotteita, kuten **CD- ja DVD-levyjä**
- Yksittäistä **kirjanimikettä** voi olla useampia **kappaleita**
- Kirjastoon hankitaan uusia kirjoja ja kuluneita tai hävinneitä kirjoja poistetaan
- Kirjastonhoitaja huolehtii **lainojen**, **varausten** ja **palautusten** kirjaamisesta
- Kirjastonhoitaja pystyy ylläpitämään tietoa **lainaajista** sekä **nimikkeistä**
- Nimikkeen voi varata jos yhtään kappaletta ei ole paikalla
- Varaus poistuu lainan yhteydessä tai erikseen peruttaessa
- Lainaajat voivat selata **valikoimaa** kirjastossa olevilla **päätteillä**
- Kirjauduttuaan päätteelle omalla **kirjastokortin** numerolla on lainaajan myös mahdollista selailla omia lainojaan
- Kirjaston päätteiden tarjoama **toiminnallisuus** on **asiakkaiden** käytössä myös **Web-selaimen** avulla

Kirjasto

Mietitään mitkä ovat oleellisia järjestelmän kannalta ja päädytään seuraaviin

Kirjasto

Mietitään mitkä ovat oleellisia järjestelmän kannalta ja päädytään seuraaviin

- Kirjasto
- Kirjanimike
  - kirjaston valikoimassa oleva kirja
- Kirja
  - edustaa yhtä fyysistä kopiota kirjanimikkeestä, eli tiettyä kirjanimikettä kirjastossa voi olla useampia kappaleita
- Lainaaja
- Kirjastonhoitaja
- Laina
- Varaus

Kirjasto

Hylätyksi tulivat

- CD ja DVD
  - Otetaan mukaan järjestelmään vasta jos todetaan tarpeelliseksi
- Kappale
  - Kirjan synonyymi
- Palautus
  - Toiminnallisuutta, tapahtuma missä laina "poistuu"
- Kirjastokortti
  - Oletetaan että lainaajalla on aina tasan yksi kortti
  - Saattaisi myös olla tilanteita, joissa oletettaisiin toisin. Asia syytä varmistaa ohjelmiston tilaajalta
- Pääte ja Web-selain
  - Epäoleellisia asioita tietosisällön osalta
- Toiminnallisuus
  - Epämääräinen käsite
- Asiakas
  - Lainaajan synonyymi

Kirjasto

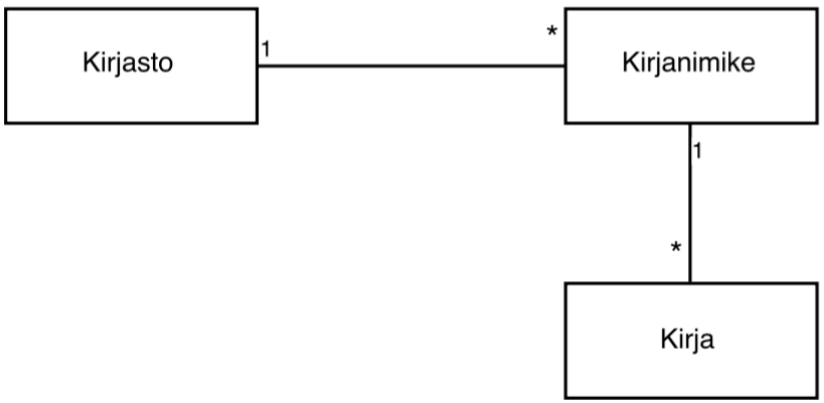
askel 1

- Kirjaston valikoimassa on useita **Kirjanimikkeitä**
- Jokaista kirjanimikettä voi olla useampi fyysinen kappale eli **Kirja**

Kirjasto

askel 1

- Kirjaston valikoimassa on useita **Kirjanimikkeitä**
- Jokaista kirjanimikettä voi olla useampi fyysinen kappale eli **Kirja**



Kirjasto

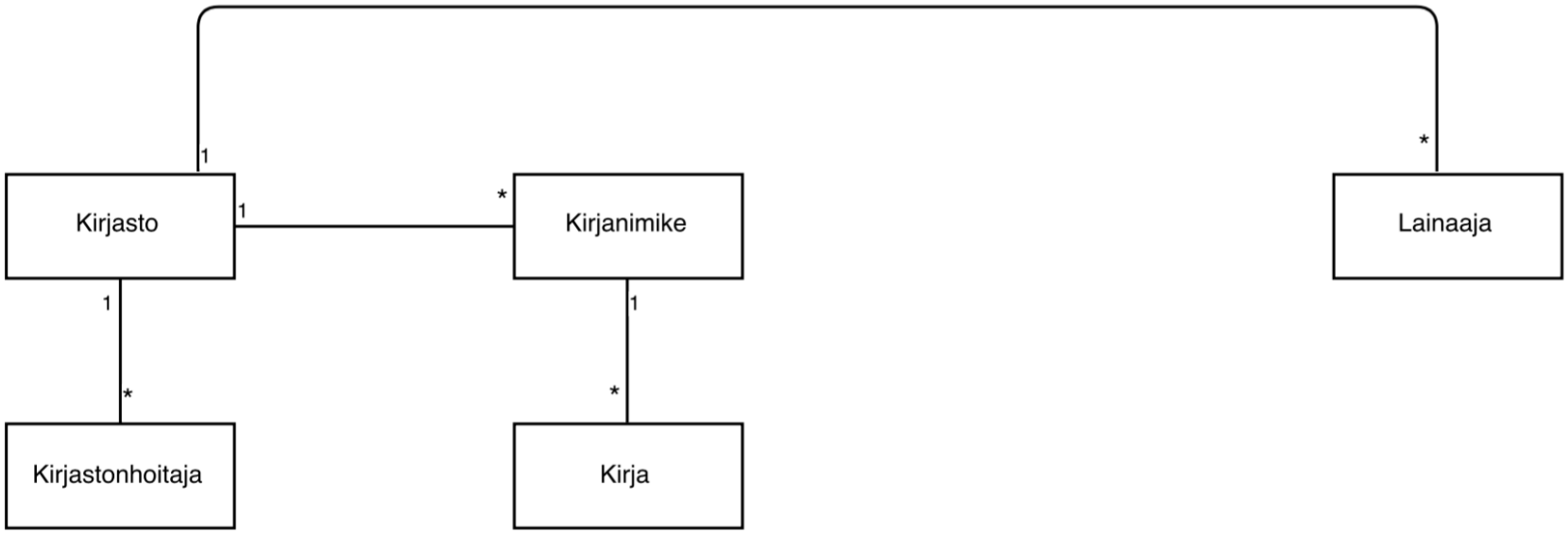
askel 2

- Kirjastossa on useita **Kirjastonhoitajia** ja **Lainaajia**

Kirjasto

askel 2

- Kirjastossa on useita **Kirjastonhoitajia** ja **Lainaajia**



Kirjasto

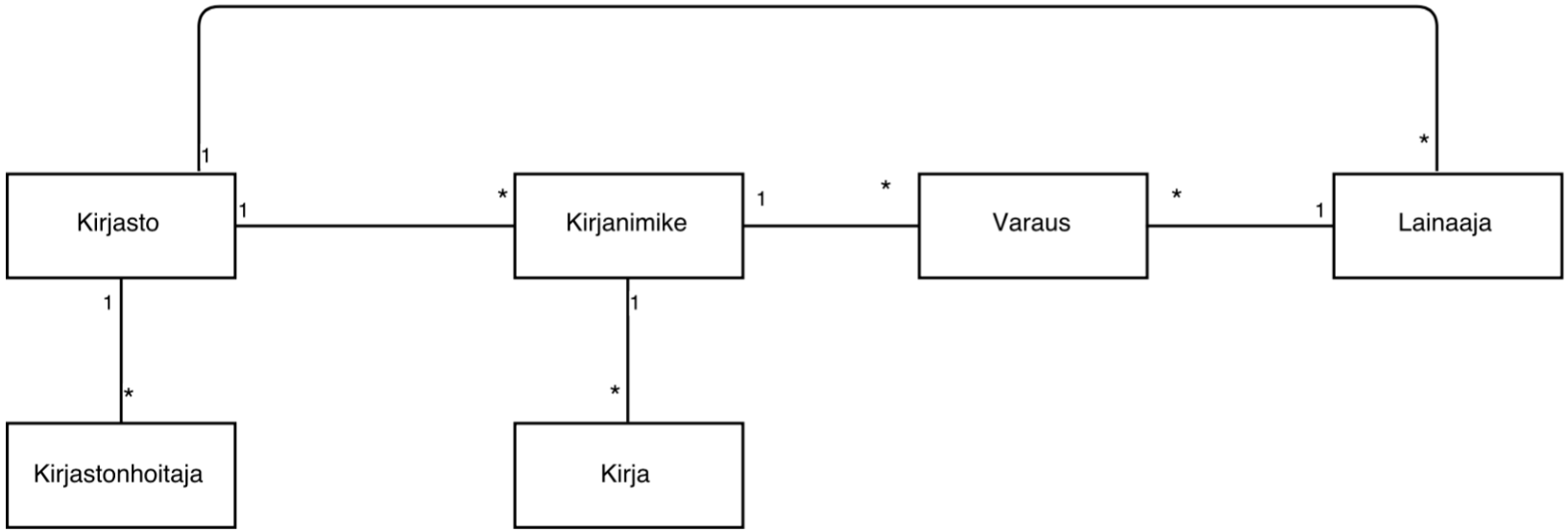
askel 3

- **Varaus** kohdistuu tiettyyn kirjanimikkeeseen (eivät siis yksittäisiin kirjoihin)
- Tiettyyn kirjanimikkeeseen voi liittyä useita varauksia
- Varaukseen liittyy aina yksi lainaaja ja lainaajalla voi olla useita varauksia

Kirjasto

askel 3

- **Varaus** kohdistuu tiettyyn kirjanimikkeeseen (eivät siis yksittäisiin kirjoihin)
- Tiettyyn kirjanimikkeeseen voi liittyä useita varauksia
- Varaukseen liittyy aina yksi lainaaja ja lainaajalla voi olla useita varauksia



Kirjasto

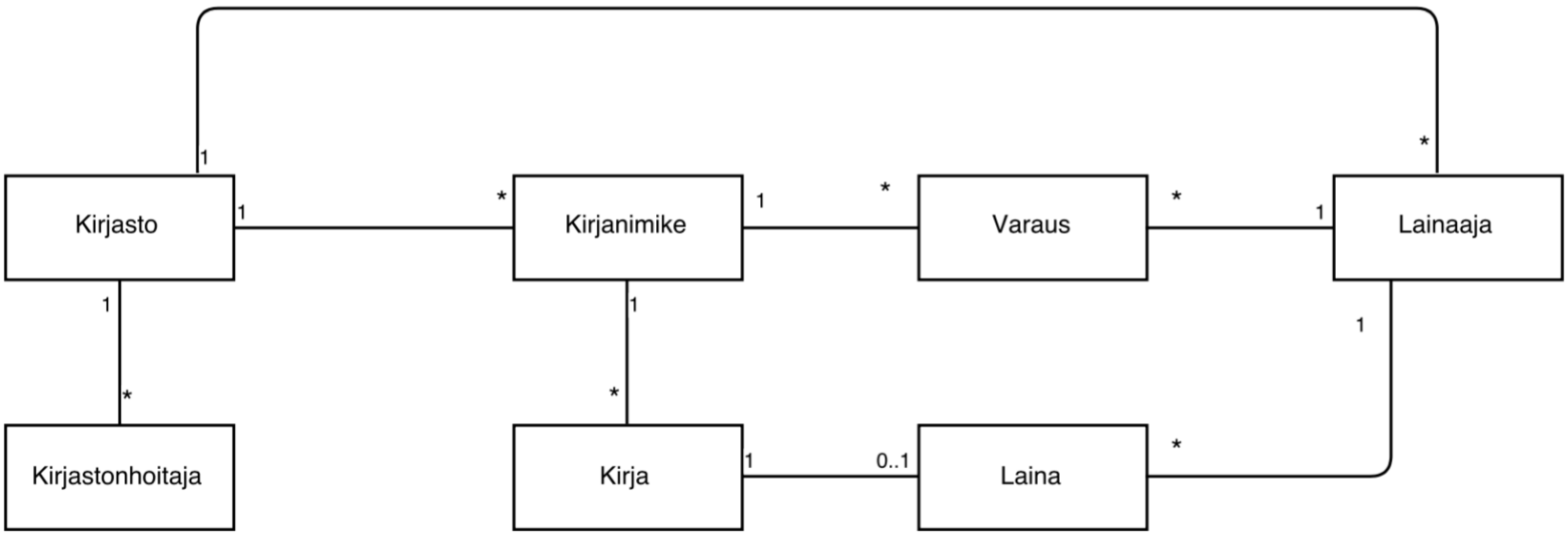
askel 4

- **Laina** taas liittyy yhteen fyysiseen kirjaan
- Kirja on joko lainassa tai ei ole, eli kirjaan liittyy 0 tai 1 lainaa
- Lainaajaan liittyy aina yksi lainaaja
- Lainaajalla voi olla yhtä aikaa useita lainoja

Kirjasto

askel 4

- **Laina** taas liittyy yhteen fyysiseen kirjaan
- Kirja on joko lainassa tai ei ole, eli kirjaan liittyy 0 tai 1 lainaa
- Lainaajaan liittyy aina yksi lainaaja
- Lainaajalla voi olla yhtä aikaa useita lainoja



Huomioita luokkakaavion muodostamisesta

- Substantiiveja saa yleensä siivota aika rankalla kädellä, läheskään kaikki eivät ole "hyviä"luokkia
- Määrittelyvaiheen luokkakaaviota tehdessä ei kannata ajatella liikaa toiminnallisuutta eli sitä kuka tekee ja mitä
  - Toiminnallisuutta ajatellaan tarkemmin vasta suunnitteluvaiheessa
- Olioiden välisissä suhteissa huomio kannattaa kiinnittää pysyvämpiluonteisiin yhteyksiin

Huomioita luokkakaavion muodostamisesta

- Substantiiveja saa yleensä siivota aika rankalla kädellä, läheskään kaikki eivät ole "hyviä"luokkia
- Määrittelyvaiheen luokkakaaviota tehdessä ei kannata ajatella liikaa toiminnallisuutta eli sitä kuka tekee ja mitä
  - Toiminnallisuutta ajatellaan tarkemmin vasta suunnitteluvaiheessa
- Olioiden välisissä suhteissa huomio kannattaa kiinnittää pysyvämpiluonteisiin yhteyksiin
  - Esim. edellisessä tekstissä ilmenee että Kirjastonhoitaja huolehtii lainojen, varausten ja palautusten kirjaamisesta. Luokkakaavion kannalta tässä ei ole oleellista kuka varauksen tekee vaan se että nimikkeeseen voi liittyä varauksia
  - Tekeminen siis välillä implikoi, että olioilla on pysyvämpiluonteinen yhteys
  - Kirjastohoitaja-oliot eivät todennäköisesti ohjelmakoodissa tule olemaan niitä joka tekevät varauksen, kirjastohoitaja on lähinnä kirjastohoitajan tietoja tallettava entiteetti. Toiminnasta, esim. varauksen tekemisestä vastaa todennäköisesti joku muu olio

Huomioita luokkakaavion muodostamisesta

- Tekstuaalisista kuvauksista eivät läheskään aina tule esille kaikki tärkeät luokat
- Tosimaailmassa ei välttämättä ole mitään tekstuaalista kuvausta
- Käsiteanalyysiä voi tehdä esim. käyttötapausten tekstuaalisille kuvauksille tai asiakkaan "puheelle"

Huomioita luokkakaavion muodostamisesta

- Tekstuaalisista kuvauksista eivät läheskään aina tule esille kaikki tärkeät luokat
- Tosimaailmassa ei välttämättä ole mitään tekstuaalista kuvausta
- Käsiteanalyysiä voi tehdä esim. käyttötapausten tekstuaalisille kuvauksille tai asiakkaan "puheelle"
- Määrittelyvaiheen luokkakaavion tekemiseen ei kannata tuhlata hirveästi aikaa, on nimittäin (lähes) varmaa, että suunnittelu- ja toteutusvaiheessa luokkamallia muutetaan
- Lopputulosta tärkeämpi on usein itse prosessi

Huomioita luokkakaavion muodostamisesta

- Tekstuaalisista kuvauksista eivät läheskään aina tule esille kaikki tärkeät luokat
- Tosimaailmassa ei välttämättä ole mitään tekstuaalista kuvausta
- Käsiteanalyysiä voi tehdä esim. käyttötapausten tekstuaalisille kuvauksille tai asiakkaan "puheelle"
- Määrittelyvaiheen luokkakaavion tekemiseen ei kannata tuhlata hirveästi aikaa, on nimittäin (lähes) varmaa, että suunnittelu- ja toteutusvaiheessa luokkamallia muutetaan
- Lopputulosta tärkeämpi on usein itse prosessi
- Suunnittelu- ja toteutusvaiheessa järjestelmään lisätään muutenkin luokkia hoitamaan esim. seuraavia tehtäviä
  - Käyttöliittymän toteutus, tietokantayhteyksien hallinta, sovelluksen ohjausoliot

Huomioita luokkakaavion muodostamisesta

- Tekstuaalisista kuvauksista eivät läheskään aina tule esille kaikki tärkeät luokat
- Tosimaailmassa ei välttämättä ole mitään tekstuaalista kuvausta
- Käsiteanalyysiä voi tehdä esim. käyttötapausten tekstuaalisille kuvauksille tai asiakkaan "puheelle"
- Määrittelyvaiheen luokkakaavion tekemiseen ei kannata tuhlata hirveästi aikaa, on nimittäin (lähes) varmaa, että suunnittelu- ja toteutusvaiheessa luokkamallia muutetaan
- Lopputulosta tärkeämpi on usein itse prosessi
- Suunnittelu- ja toteutusvaiheessa järjestelmään lisätään muutenkin luokkia hoitamaan esim. seuraavia tehtäviä
  - Käyttöliittymän toteutus, tietokantayhteyksien hallinta, sovelluksen ohjausoliot
- Määritteluvaiheen luokkakaavio toimii pohjana tietokantasuunnittelulle
  - Osa olioistahan (esimerkissämme lähes kaikki) on tallennettava

Huomioita luokkakaavion muodostamisesta

Mallinnuksen eteneminen

- Isoa ongelmaa kannattaa lähestyä pienin askelin, esim:
  - Yhteydet ensin karkealla tasolla, tai
  - Tehdään malli pala palalta, lisäten siihen muutama luokka yhteyksineen kerrallaan

Huomioita luokkakaavion muodostamisesta

Mallinnuksen eteneminen

- Isoa ongelmaa kannattaa lähestyä pienin askelin, esim:
  - Yhteydet ensin karkealla tasolla, tai
  - Tehdään malli pala palalta, lisäten siihen muutama luokka yhteyksineen kerrallaan
- Mallinnus iteratiivisesti etenevässä ohjelmistokehityksessä
  - Ketterissä menetelmissä suositaan iteratiivista lähestymistapaa ohjelmistojen kehittämiseen
    - kerralla on määrittelyn, suunnittelun ja toteutuksen alla ainoastaan osa koko järjestelmän toiminnallisuudesta
  - Jos ohjelmiston kehittäminen tapahtuu ketterästi, kannattaa myös ohjelman luokkamallia rakentaa iteratiivisesti
  - Eli jos ensimmäisessä iteraatiossa toteutetaan ainoastaan muutaman käyttötapauksen kuvaama toiminnallisuus, esitetään iteraation luokkamallissa vain ne luokat, jotka ovat merkityksellisiä tarkastelun alla olevan toiminnallisuuden kannalta
  - Luokkamallia täydennetään myöhempien iteraatioiden aikana niiden mukana tuoman toiminnallisuuden osalta