Luokkakaavion tarkoitus

- Järjestelmän tietosisällön kuvaaminen:
  - tiedot ja niiden väliset kytkennät
  - järjestelmän tiedot kuvaavat kohdealueiden ilmiöitä, joten luokkakaavion tulisi määritellä kohdealueen rakenne

Luokkakaavion tarkoitus

- Ohjelman rakenteen kuvaaminen:
  - tiedot ja niiden väliset kytkennät
  - tietojen yhteys toimintaan
  - Olio-ohjelman idea on usein simuloida vastaavaa reaalimaailman (kohdealueen) ilmiötä, joten luokat löytyvät reaalimaailmaa analysoimalla.
  - Kohdealueen rakenteen mukaiset luokat muodostavat tällöin ohjelman luokkarakenteen ytimen.

Luokkakaavion laatiminen

- Kokonaisvaltainen lähestymistapa:
  - pyritään löytämään kerralla koko kohdealuetta kuvaava malli
  - hankalaa, jos kohdealue on laaja
  - ensin karkea yleiskuva, sitten lisää yksityiskohtia
- Osista kokonaisuuteen:
  - jaetaan kokonaisuus osiin ja tehdään osakohtaisia malleja, jotka sitten yhdistetään kokonaismalliksi
  - osa voisi olla käyttötapaus
  - yksityiskohdista yleiskuvaan

Luokkakaavion laatiminen

- Kartoita luokkaehdokkaita.
- Karsi ehdokkaita.
- Tunnista olioiden väliset yhteydet.
- Täsmennä luokkakuvauksia määrittelemällä attribuutit.
- Määrittele yhteyksiin liittyvät osallistumis-rajoitteet.
- Liitä luokkiin palvelut.
- Varmista palvelujen ja tietosisällön yhteensopivuus.

Kartoita luokkaehdokkaita

- Laadi luettelo tarkasteltavan ilmiön kannalta keskeisistä kohteista tai ilmiöistä, jotka voisivat tulla kyseeseen luokkina tai olioina:
  - osallistujat,
  - toiminnan kohteet,
  - toimintaan liittyvät tapahtumat,
  - materiaalit,
  - tuotteet ja välituotteet,
  - toiminnalle edellytyksiä luovat asiat.

Kartoita luokkaehdokkaita

- Kartoituksen pohjana voi käyttää **vapaa-muotoista tekstikuvausta** tarkasteltavasta ilmiöstä.
  - Kuvauksesta alleviivataan luokkaehdokkaita ja kerätään ne luetteloon.
  - Luokkaehdokkaat esiintyvät kuvauksessa usein substantiiveina.
  - Verbit voivat ilmaista yhteyksiä.
  - Alustavaa karsintaa voi tehdä sen perusteella, onko asia lainkaan oleellinen mallinnettavan ilmiön kannalta.

Karsi ehdokkaita

- Löydetyt ehdokkaat käydään läpi ja arvioidaan, voisiko ehdokas tulla kyseeseen luokkana.
  - Liittyykö luokan ilmentymiin tietosisältöä, joka on välttämätöntä järjestelmän kannalta (yleensä on oltava useita attribuutteja)?
  - Tarvitaanko tietoa ilmentymien olemassaolosta?
  - Onko asia riittävän tärkeä kohdealueen kannalta?
  - Karsitaan synonyymit.
- Karsintaa ja ehdokkaiden kartoitusta voidaan joutua tekemään iteratiivisesti. Ensimmäinen karsintakierros ei välttämättä tuota lopullista tulosta.

Tunnista yhteydet

- Yhteyksiä (ilmentymien välillä) voi etsiä vapaamuotoisesta kuvauksesta:
  - *verbit*
  - *genetiivit*
  - *muut ilmaukset, jotka kuvaavat kytkentää*
- Yhteyksienkin suhteen tulisi miettiä
  - *onko yhteys oleellinen tarkasteltavan ilmiön kannalta?*
  - *onko se rakenteellinen?*
- Asia pitäisi esittää vain kertaalleen:
  - *johdettavissa olevat yhteydet?*
  - *karsitaan tai merkitään*
- Älä piilota yhteyksiä attribuuteiksi!

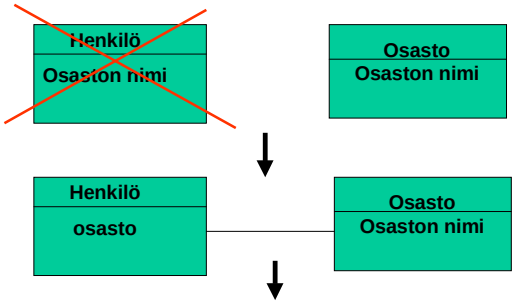
Määrittele attribuutit

- Attribuutteja saattaa löytyä vapaamuotoisesta kuvauksesta.
- Yleensä niiden löytäminen edellyttää lisäselvityksiä kohdealueesta, esimerkiksi toiminnan osapuolten haastatteluja.
- Attribuuttien kohdalla pitäisi myös selvittää, mihin niitä tarvitaan.
- Ja uudelleen:
  - **Älä piilota yhteyksiä attribuutteihin!**

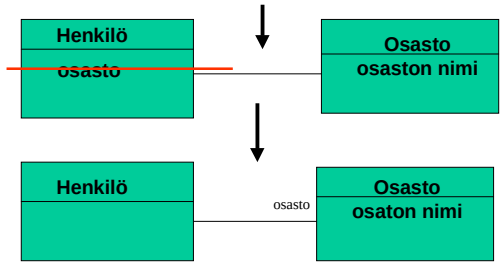
Määrittele osallistumisrajoitteet

- Osallistumisrajoitteiden avulla ilmaistaan rakenteellisia **sääntöjä**.
- Säännöt eivät välttämättä tule esiin vapaamuotoisessa kuvauksessa, vaan edellyttävät tarkempaa kohdealueen analysointia.

Yhteyden piilottaminen



Sama asia kahteen kertaan



- **Puutarhatonttu esimerkki**

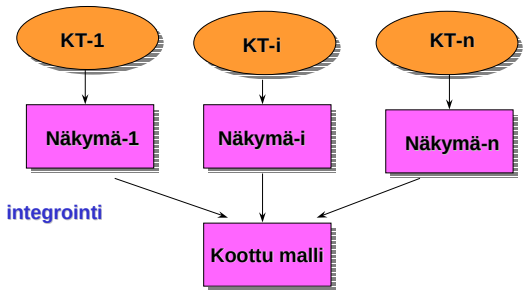
Käyttötapaukset ja tietosisältö

- Käyttötapaukset määrittelevät toiminnalliset vaatimukset järjestelmälle, mutta ne asettavat vaatimuksia myös järjestelmän tietosisällölle, koska toiminnot edellyttävät tietojen olemassaoloa.
- Eräs tapa järjestelmän sisältöluokkien suunnitteluun on tehdä suunnittelu käyttötapauksittain ja sitten yhdistää käyttötapauskohtaiset mallit.

Käyttötapaukset ja tietosisältö

- Lähtökohtana käyttötapaukset:
  - Mitä tietoja ja tietokokonaisuuksia tarvitaan käyttötapauksen hoitamiseksi?
- Jokaisen käyttötapauksen perusteella laaditaan käyttötapauskohtainen näkymä tarvittavista tiedoista, eli pelkästään käyttötapauksen tarpeet huomioon ottava luokkakaavio.
- Näkymät yhdistetään sisällön kokonaismallin aikaansaamiseksi.

Käyttötapaukset ja tietosisältö



Käyttötapaukset ja tietosisältö

- Käyttötapauksittain sisältömallia muodostettaessa saadaan ainakin periaatteessa malliin mukaan kaikki käyttötapauksen tarvitsemat tiedot.
- Käyttötapausmallin ja tietosisältömallin yhteensopivuuden varmistamiseksi voidaan käyttää riippuvuusmatriisia.

- **Kampaamoesimerkki**

Käyttötapaukset ja tietosisältö

- Riippuvuusmatriiseja:
  - Luokat ja yhteydet / käyttötapaukset
    - *Matriisin alkiona riippuvuustieto:*
    - *luo, muuttaa, poistaa, käyttää*
  - Matriisit voidaan esittää myös attribuuttitasolla, jolloin näkyisi attribuuttien käsittely. Tällöin matriisit on kuitenkin syytä pilkkoa luokkakohtaisiksi.

Käyttötapaukset ja tietosisältö

	Käyttötapaukset																					
Olioluokat	Uusi artikkeli	Uusi artikkeliversio	Tiedustelu artikkelin tilasta	Lausunnonantajan valinta	Lausunnon saapuminen	Muistutus lausunnosta	Puuttumaan jääneyt lausunt	Valituksen käsittely	Julkaisupäätöksen kirjaus	Palautus korjattavaksi	Julkaistavaksi hyväksyminn	Hylkääminen	Viimeistellyn saapuminen	Oikovedoksen lähettäminen	Korjausten vastaanotto	Erpainosten tilaus	Henkilötietojen rekisteröint	Sisällysluettelo	Yvosikertatiedot	www-sivut	Raportit
Article	L	M	K	K	K	K	K	K	M	M	M	M	M	M	M	K		K	K	K	K
Article version	L	L	K	M	K	K	K	M	M	M	M										K
Person	X	X		X		K	K			K	K	K	K	K	K		X				K
Reference			K	L	M	M	M	M													K
Journal																		L	K	K	K

K= Käyttää, L = Luo, M= Muuttaa, P= Poistaa, X=Luo tai muuttaa

Käyttötapaukset ja tietosisältö

- Jokaiselle luokalle ja yhteydelle täytyisi löytyä käyttötapaus, jolla voidaan luoda luokan ilmentymiä ja kytkeä olioita ko. yhteyteen.
- Jokaiselle luokalle ja yhteydelle täytyisi löytyä käyttötapaus, joka hyödyntää luokan tietoja ja yhteyksiä.
- Jos ilmentymät eivät ole ikuisia, pitää löytyä käyttötapaus, jolla hävitetään ilmentymiä.

Käyttötapaukset ja tietosisältö

- Jos ilmentymän tietosisältö ei ole muuttumaton, pitää löytyä käyttötapaus, jolla tietosisältöä voidaan muuttaa.
- Jos osapuoli voidaan irroittaa yhteydestä, täytyy tähän tarkoitukseen löytyä käyttötapaus.
- Toisaalta jokaisen käyttötapauksen täytyy jollain tavoin liittyä sisältöluokkiin.

Luokkahierarkia

- Luokkia määrittelemällä luodaan luokitusjärjestelmä.
- Eri menetelmät asettavat erilaisia vaatimuksia luokitusjärjestelmälle:
  - joissakin edellytetään, että luokat ovat erillisiä (ei yhteisiä ilmentymiä)
  - toisissa sallitaan päällekkäisyys (yhteiset ilmentymät)

Luokkahierarkia

- Luokkahierarkiassa luokka voidaan määritellä toisen luokan alaluokaksi
- esim.
  - **luokka nainen on luokan henkilö alaluokka**
  - **luokka johtaja on luokan henkilö alaluokka**
  - **luokat auto, laiva ja lentokone ovat luokan kulkuväline alaluokkia**

Luokkahierarkia

- Jos luokka A on luokan B alaluokka, siitä seuraa, että jokainen A:n ilmentymä on myös B:n ilmentymä.
  - **jokainen johtaja on myös henkilö**
- Tästä taas seuraa, että kaikki attribuutit, palvelut ja yhteydet, jotka on määritelty luokan B ilmentymille, liittyvät myös A:n ilmentymiin.

Luokkahierarkia

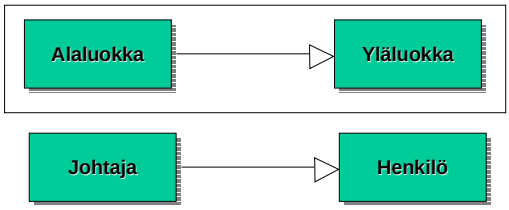
- Jos henkilölle on määritelty attribuutti Nimi, niin myös johtajalla on automaattisesti Nimi-attribuutti.
- Jos henkilö on määritelty osapuoleksi työsuhde-yhteyteen, myös johtaja voi olla osapuolena työsuhde-yhteydessä.
- Tätä ilmiötä kutsutaan periytymiseksi (inheritance).

Luokkahierarkia

- Periytymisessä yläluokkaan liitetyt attribuutit, palvelut ja yhteydet periytyvät alaluokalle.
- **Huom.** periytyminen on luokkien välistä määrittelyjen periytymistä - ilmentymät eivät peri mitään toisilta ilmentymiltä.

Luokkahierarkia

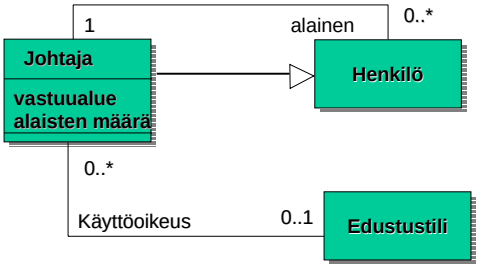
- Yläluokan ja alaluokan välinen riippuvuussuhde kuvataan kaaviossa:



Luokkahierarkia

- Alaluokkaan voidaan liittää yläluokalta perittyjen attribuuttien, palvelujen ja yhteyksien lisäksi omia attribuutteja, yhteyksiä ja palveluja.
  - **Johtajalla on ominaisuudet vastuualue ja alaisten määrä, joita ei ole henkilöllä yleisesti.**
  - **Vain johtajalla voi olla oikeus käyttää edustustiliä.**
  - **Johtajalla voi olla alaisia.**

Luokkahierarkia



Luokkahierarkia

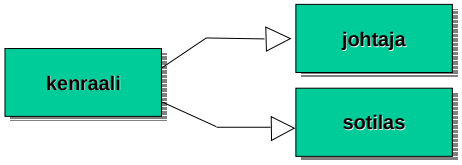
- Alaluokkaan voidaan liittää uusia palveluita, joita yläluokalla ei ole.
- Alaluokassa voidaan myös **syrjäyttää** (override) yläluokassa määritelty palvelu. Syrjäyttäminen on sisällön uudelleenmäärittelyä.
- Henkilöllä voisi olla palvelu viikkoraportti:
  - kerro ajankäyttö työtehtäviin
- Johtajan viikkoraportti-palvelun sisältö voisi olla:
  - kerro ajankäyttö työtehtäviin
  - laadi yhteenveto alaisten viikkoraporteista
  - raportoi edustustilin käyttö

Luokkahierarkia

- Syrjäyttäminen on erityisesti olio-ohjelmoinnissa hyödyllinen tekniikka.
- Olio-ohjelmoinnin merkittävimmät hyödyt tulevat esiin juuri syrjäyttämismahdollisuuden kautta:
  - muokattavat kirjastopalvelut
  - yksinkertaisemmat ohjelmat

Luokkahierarkia

- Tilannetta, jossa luokka on useamman kuin yhden luokan välitön alaluokka, kutsutaan moniperiytymiseksi (multiple inheritance).
  - Kaikki olio-ohjelmointikielet eivät tarjoa moniperintää (esim. Javassa ei ole, C++:ssa on).

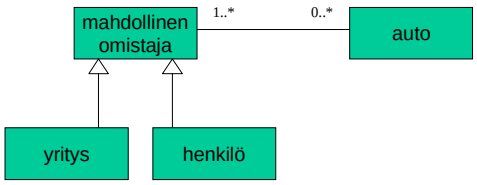


Luokkahierarkia

- Järjestelmää mallinnettaessa luokkahierarkiasta on eniten hyötyä:
  - sääntöjen ilmaisemisessa
  - kuvauksen ekonomisuudessa (ei tarvitse toistaa samoja asioita useassa luokassa)

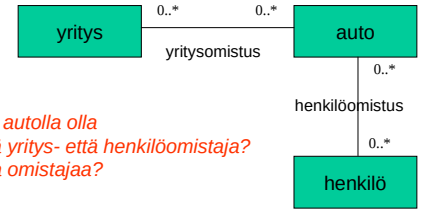
Luokkahierarkia

- Esimerkki säännöistä: Autolla täytyy olla omistaja. Omistaja voi olla yritys tai henkilö.



Luokkahierarkia

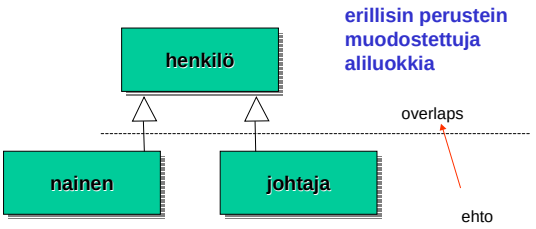
- Ellei yläluokkaa käytettäisi:



Voiko autolla olla
- sekä yritys- että henkilöomistaja?
- nolla omistajaa?

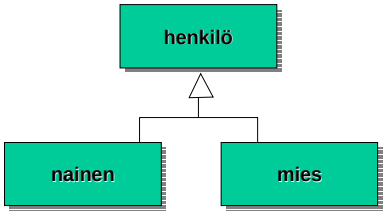
Luokkahierarkia

- Aliluokkia voidaan muodostaa monin erilaisin perustein.



Luokkahierarkia

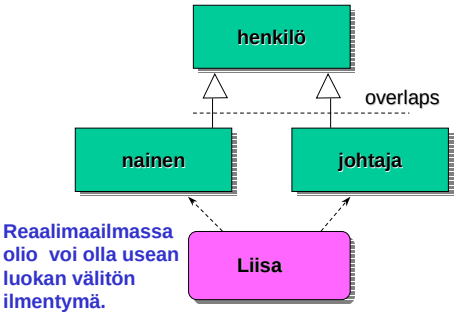
- Samaan luokitteluperusteeseen perustuvat poissulkevat luokat.



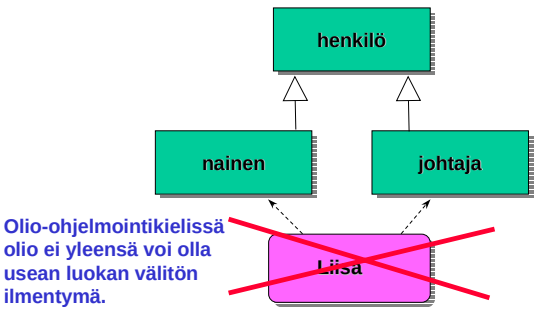
Luokkahierarkia

- Kun reaalimaailmassa luokitellaan ilmiöitä, voidaan käyttää samanaikaisesti useita erilaisia luokitteluperusteita:
  - *sukupuoli, kengännumero, virka-asema, auton omistus, jne.*
- Monissa oliomenetelmissä ja olio-ohjelmointi-kielissä ollaan paljon rajoittuneempia.

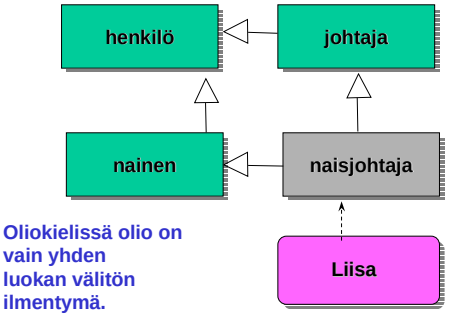
Luokkahierarkia

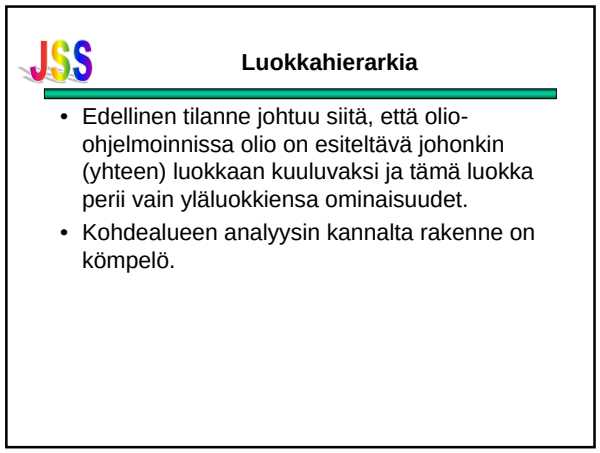


Luokkahierarkia



Luokkahierarkia




JSS
Luokkahierarkia
• Edellinen tilanne johtuu siitä, että olio-ohjelmoinnissa olio on esiteltävä johonkin (yhteen) luokkaan kuuluvaksi ja tämä luokka perii vain yläluokkiensa ominaisuudet.
• Kohdealueen analyysin kannalta rakenne on kömpelö.