Ohjelmistotekniikan menetelmät

Luento 5, 14.08.2017

Kertaus: määrittelyvaiheen luokkakaavion muodostaminen

1. Etsi luokkaehdokkaat tekstikuvauksista (substantiivit)
2. Karsi luokkaehdokkaita (poista tekemistä tarkoittavat sanat)
3. Tunnista olioiden välisiä yhteyksiä (verbit ja genetiivit)
4. Lisää luokille attribuutteja
5. Tarkenna yhteyksiä (kytkentärajoitteet, kompositiot)
6. Etsi "rivien välissä olevia luokkia
7. Etsi yläkäsitteitä
8. Toista vaiheita 1-7 riittävän kauan

- Aloitetaan vaiheella 1, sen jälkeen edetään muihin vaiheisiin peräkkäin, rinnakkain tai/ja sekalaisessa järjestyksessä toistaen
- Lopputuloksena alustava toteutusriippumaton sovelluksen kohdealueen luokkamalli
- Malli tulee tarkentumaan ja täsmentymään suunnitteluvaiheessa
- Siksi ei edes kannata tähdätä ”täydelliseen” malliin

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

Kirjasto

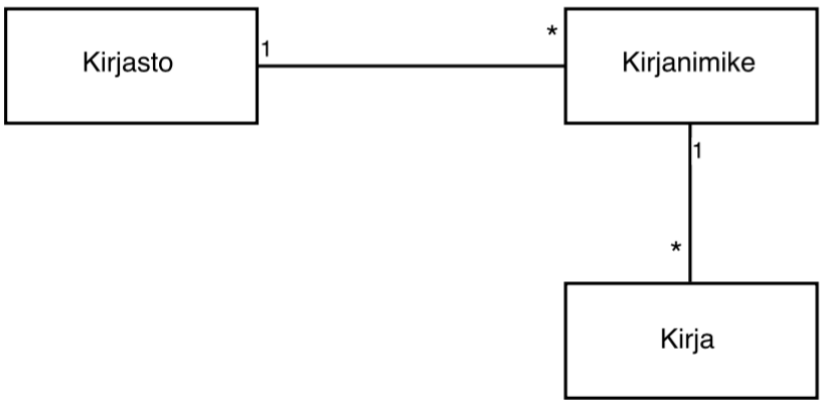
Hylätyksi tulivat

- 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

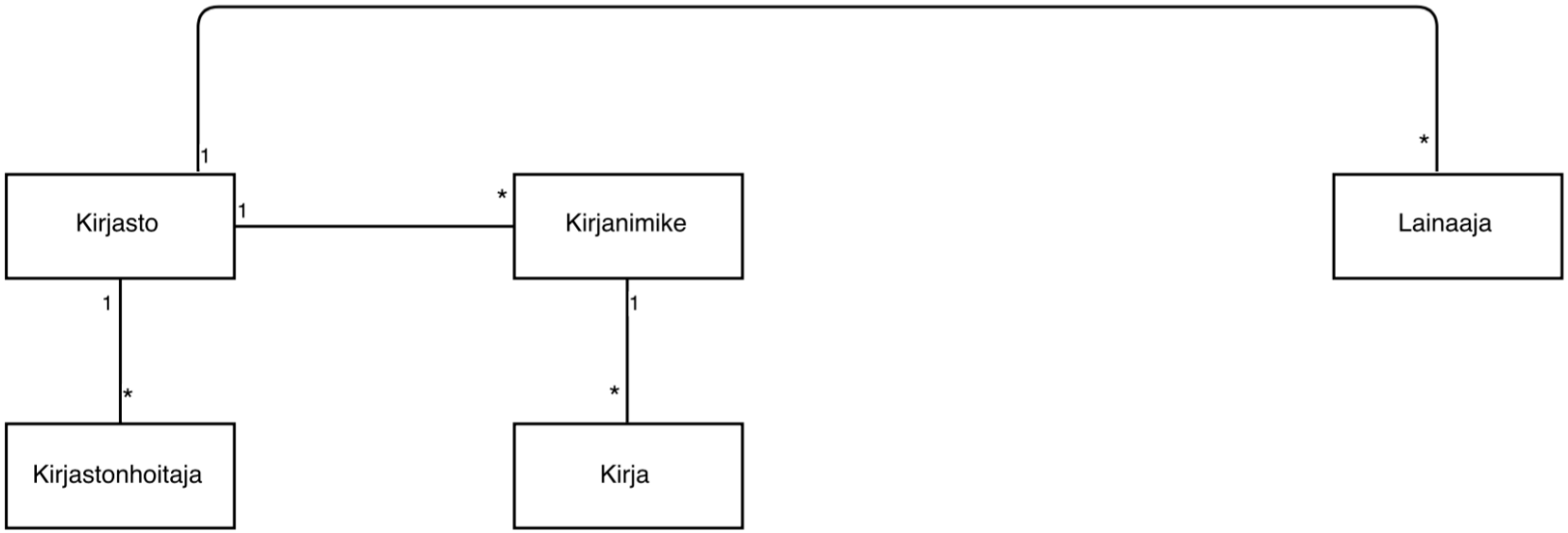
- Kirjaston valikoimassa on useita **Kirjanimikkeitä**
- Jokaista kirjanimikettä voi olla useampi fyysinen kappale eli **Kirja**



Kirjasto

Askel 2

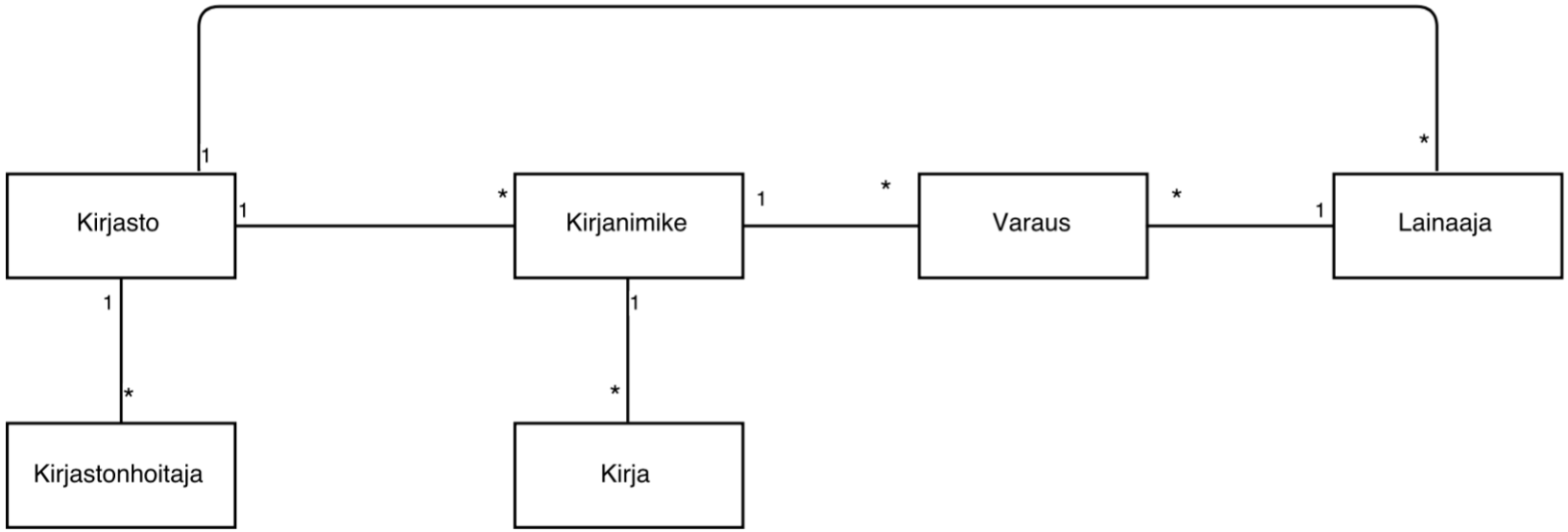
- Kirjastossa on useita **Kirjastonhoitajia** ja **Lainaajia**



Kirjasto

Askel 3

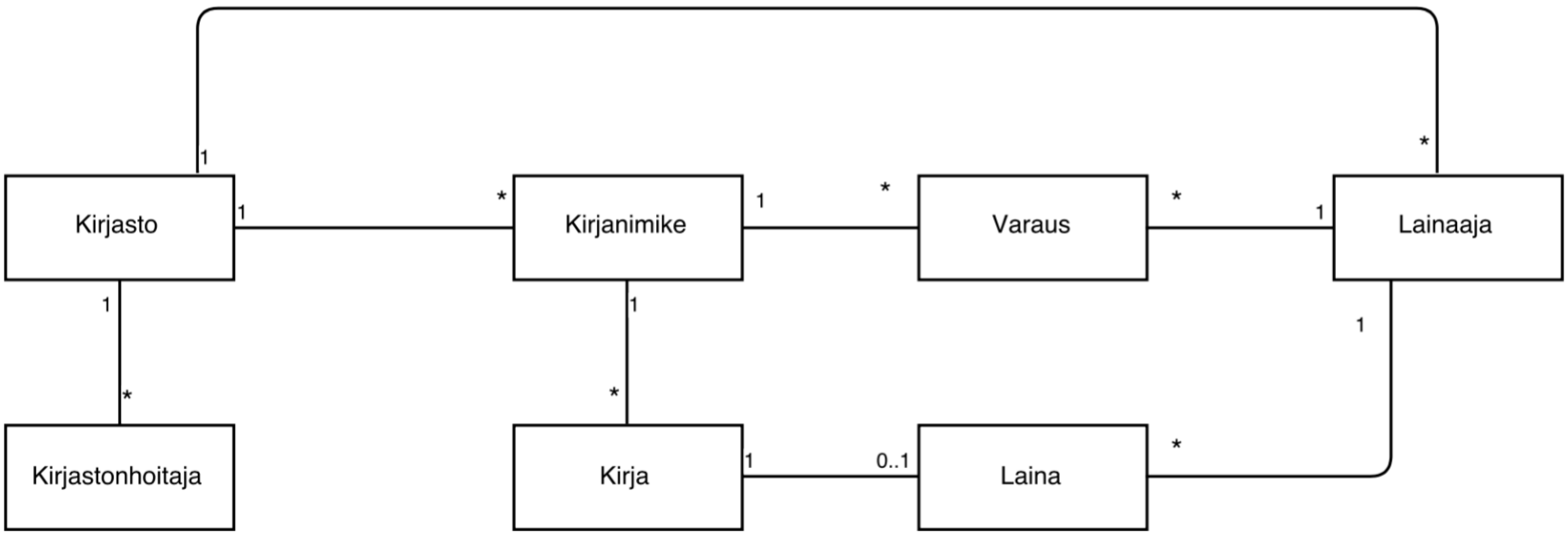
- **Varaus** kohdistuu tiettyyn kirjanimikkeeseen (eivät siis yksittäisiin kirjoihin)
- Tiettyyn kirjanimikkeeseen voi liittyä useita varauksia
- Varaukseen liittyy aina yksi lainaaja ja lainaajalla voi olla useita varauksia



Kirjasto

Askel 4

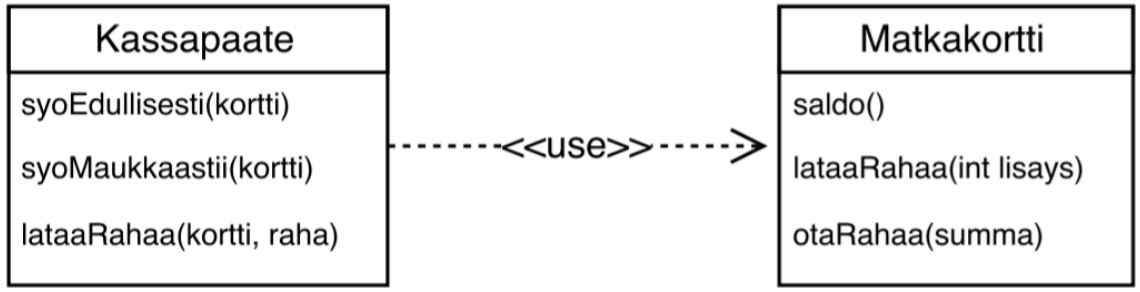
- **Laina** taas liittyy yhteen fyysiseen kirjaan
- Kirja on joko lainassa tai ei ole, eli kirjaan liittyy 0 tai 1 lainaa
- Lainaajaan liittyy aina yksi lainaaja
- Lainaajalla voi olla yhtä aikaa useita lainoja



Olioiden yhteistyön mallintaminen

Olioiden yhteistyön mallintaminen

- Luokkakaaviosta käy hyvin esille ohjelman *rakenne*
  - mitä luokkia on olemassa
  - miten oliot liittyvät toisiinsa
- Entä ohjelman toiminta?
  - Luokkakaaviossa voi olla metodien nimiä
  - Pelkät nimet eivät kuitenkaan kerro juuri mitään!



- Tarve kuvata esim. skenaario "ostetaan 3 euroa sisältävällä maksukortilla edullinen lounas"

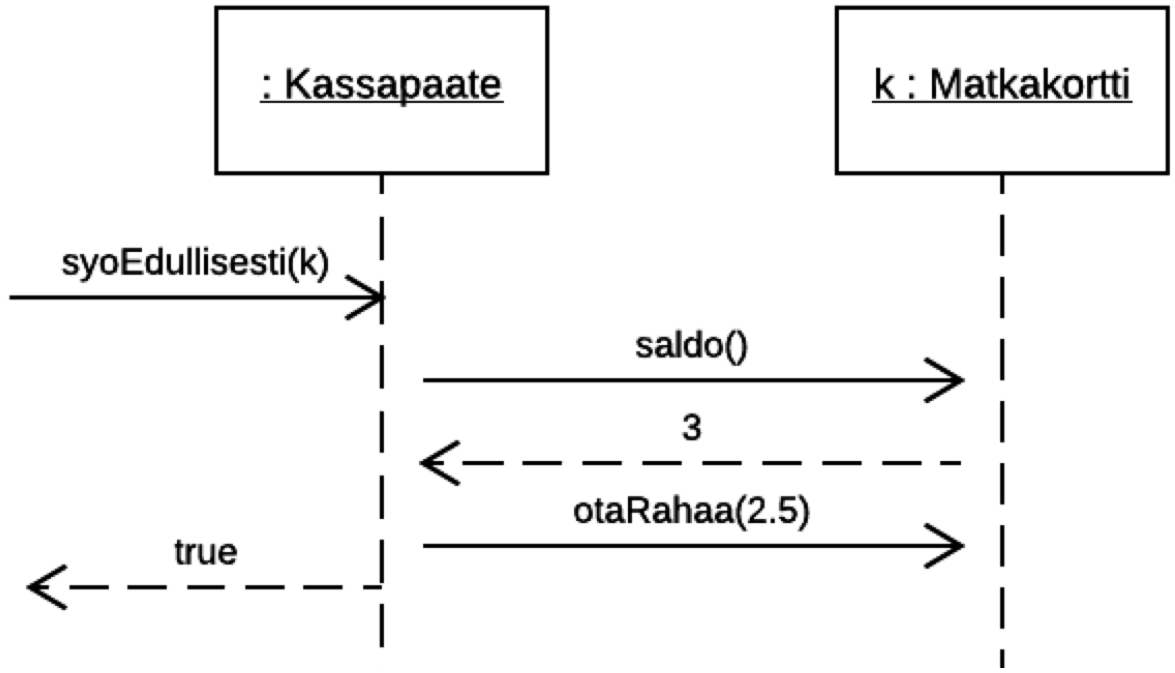
Olioiden yhteistyön mallintaminen

Koska järjestelmän toiminnan kulmakivenä on järjestelmän sisältämien olioiden yhteistyö, tarvitaan menetelmä yhteistyön kuvaamiseen

- UML tarjoaa kaksi menetelmää, joita kohta tarkastelemme:
  - sekvenssikaavio
  - kommunikaatiokaavio
- Huomionarvoista on, että luokkakaaviossa tarkastelun pääkohteena olivat luokat ja niiden suhteen.
- Yhteistyötä mallintaessa taas fokuksessa ovat oliot eli luokkien instanssit
  - Luokkahan ei tee itse mitään, ainoastaan oliot voivat toimia

Sekvenssikaavio

- "*Ostetaan 3 euroa sisältävällä maksukortilla edullinen lounas*
  - Lukemalla koodia (ks. mallivastaus ohpe viikko 5) huomataan, että kassapääte kysyy ensin kortin saldon ja huomatessaan sen riittävän, vähentää kortilta edullisen lounaan hinnan
- Tilanteen kuvaava sekvenssikaavio:



Sekvenssikaavio

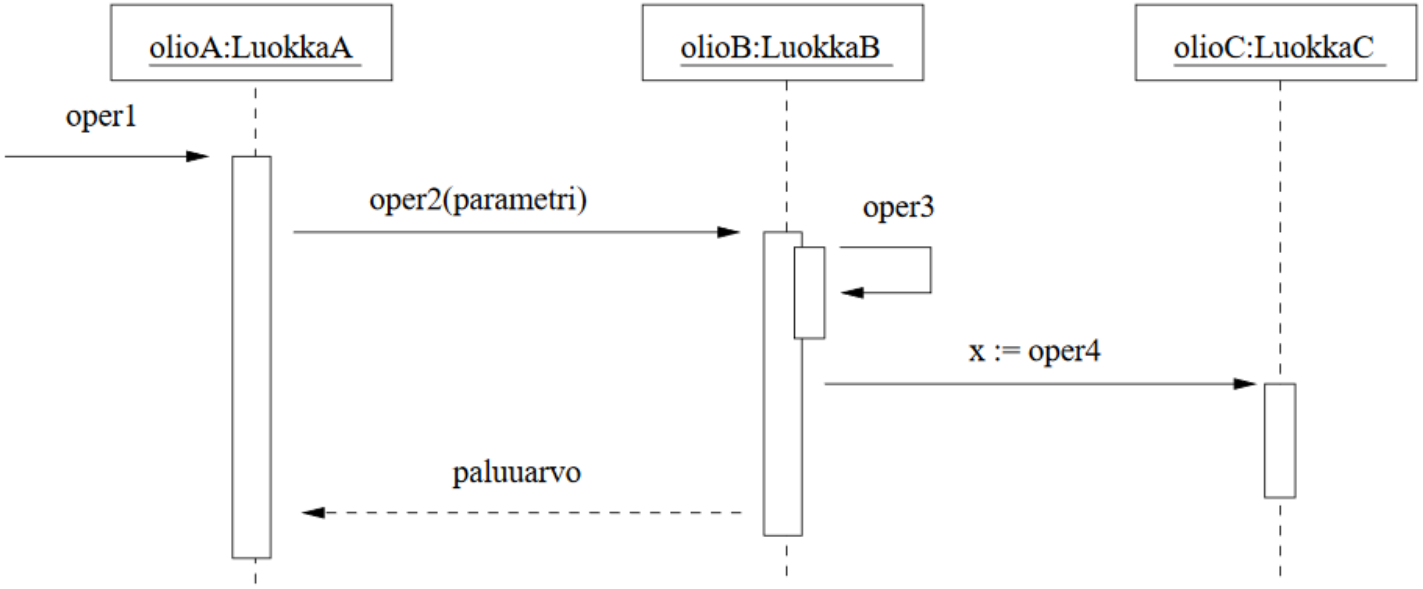
- Sekvenssikaaviossa kuvataan tarkasteltavan skenaarion aikana tapahtuva olioiden vuorovaikutus
- Oliot esitetään kuten oliokaaviossa, eli laatikkoina, joissa alleviivattuna olion nimi ja tyyppi
- Sekvenssikaaviossa oliot ovat (yleensä) ylhäällä rivissä
- Aika etenee kaaviossa alaspäin
- Jokaiseen olioon liittyy katkoviiva eli elämänviiva (engl. lifeline), joka kuvaa sitä, että olio on olemassa
- Metodikutsu piirretään nuolena, joka lähtee kutsuvasta oliosta ja kohdistuu kutsuttavan olion elämänlankaan
- Tyypillisesti yksi sekvenssikaavio kuvaa järjestelmän yksittäisen toimintaskenaarion

Sekvenssikaavio

- Esimerkissä toiminta alkaa sillä, että joku (esim. pääohjelma, tässä tapauksessa nuoli on merkitty tulevan tyhjästä) kutsuu Kassapaate-olion metodia `syoEdullisesti`
- Metodikutsun seurauksena kassapääte kutsuu maksukortin metodia `saldo`, joka palauttaa kortilla olevan rahamäärän
  - Kortin palauttama saldo on merkitty katkoviivalla
  - Tämän jälkeen kassapääte kutsuu kortin metodia `otaRahaa`, parametrilla 2.5 eli veloittaa kortilta edullisen lounaan hinnan
- Kun hinta on veloitettu, Kassapääte palauttaa operaation onnistumisen merkiksi true metodin syoEdullisesti kutsujalle
  - Metodin paluuarvo on jälleen merkitty katkoviivalla

Sekvenssikaavio

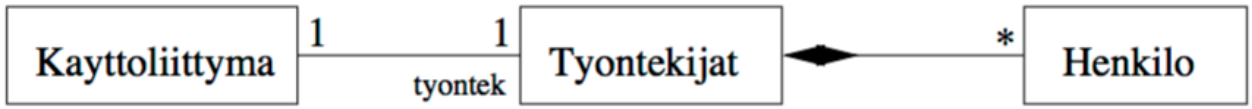
- alla oleva kuva esittelee lisää sekvenssikaavioiden syntaksia
  - Joskus on hyödyllistä piirtää **aktivaatiopalkki**, joka merkitsee ajan jolloin olio on aktiivisena, eli sen suoritus on kesken
  - Aktivaatiopalkkia harvemmin jaksetaan piirtää paperille
  - Merkinnällä x:=oper4 tarkoitetaan, että metodia oper4() kutsutaan ja sen palautusarvo otetaan talteen muuttujaan x



Sekvenssikaavio

Henkilöstohallinta ja palkanmaksu

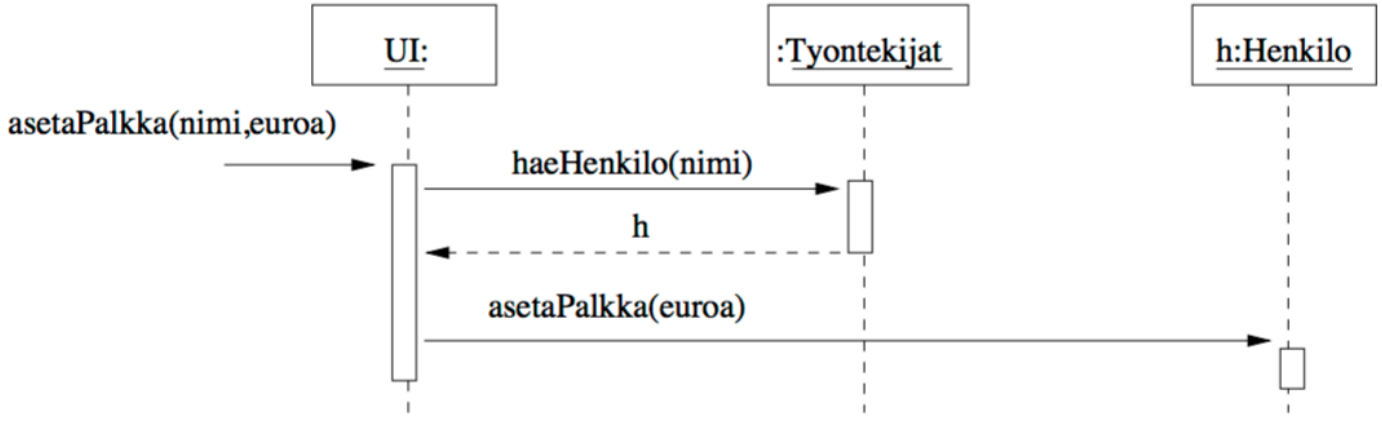
- Tarkastellaan Henkilöstönhallintajärjestelmää
  - Työntekijät-olio pitää kirjaa työntekijöistä, jotka Henkilö-oliota
  - Kayttoliittyma-olio hoitaa korkeamman tason komentojen käsittelyn



- Tarkastellaan operaatiota lisaaPalkka(nimi, palkka)
  - Lisätään parametrina annetulle henkilölle uusi palkka
- Suunnitellaan, että operaatio toimii seuraavasti:
  - Ensin käyttöliittymä hakee Tyontekijat-oliolta viitteen Henkilo-olion
  - Sitten käyttöliittymä kutsuu Henkilo-olion palkanasetusmetodia
- Seuraavalla sivulla operaation suoritusta vastaava sekvenssikaavio
  - Havainnollistuksena myös osa luokan Kayttoliittyma koodista

Sekvenssikaavio

Henkilöstohallinta ja palkanmaksu




```
public class Kayttoliittyma{
   private Tyontekijat tyontek;   // viite luokan Tyontekijat olioon

   public void asetaPalkka(String nimi, int euroa){
      Henkilo h = tyontek.haeHenkilo(nimi);
      h.asetaPalkka(euroa);
   }
}
```

Sekvenssikaavio

Henkilöstohallinta ja palkanmaksu

- Tässä oli oikeastaan jo kyse oliosuunnittelusta
  - Alunperin oli ehkä päätetty luokkarakenne
  - Tiedettiin, että tarvitaan toiminto, jolla lisätään henkilölle palkka
  - Suunniteltiin, miten palkan asettaminen tapahtuu olioiden yhteistyönä
  - Suunnittelu tapahtui ehkä sekvenssikaaviota hyödyntäen
  - Siitä saatiin helposti aikaan koodirunko
- Sekvenssikaaviot ovatkin usein käytössä oliosuunnittelun yhteydessä
  - Kuten kohta näemme, voidaan niitä käyttää myös määrittelyssä kuvaamaan käyttötapauksen kulkua
- Huomaa parametrin **h** käyttö edellisen sivun kuvassa
  - *haeHenkilo*-metodikutsun paluuarvo on h
  - Kyseessä on sama h, joka on sekvenssikaaviossa esiintyvän olion nimi!

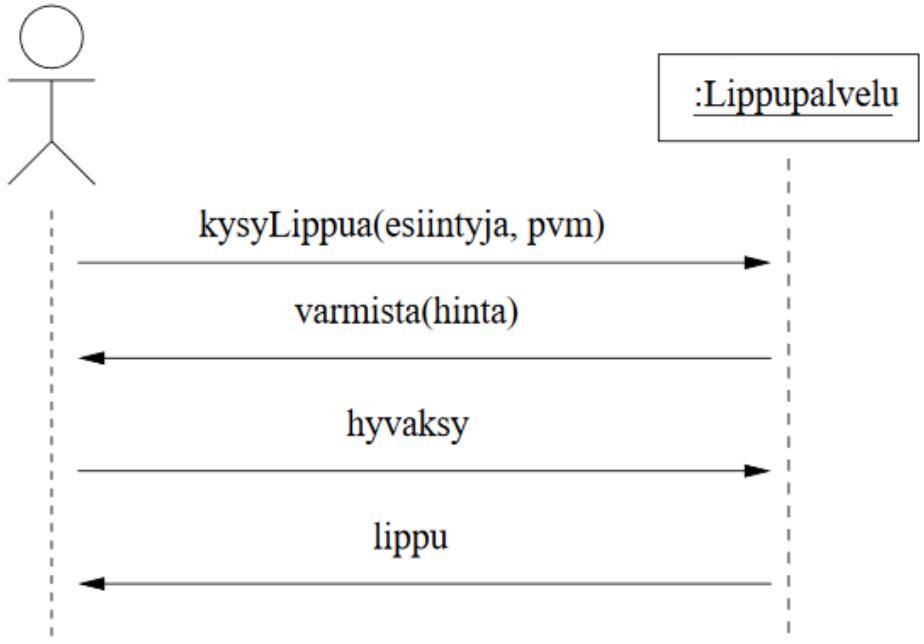
Sekvenssikaavio

Käyttötapausten kulun kuvaaminen sekvenssikaaviona

- Tarkastellaan alkeellista lippupalvelun tietojärjestelmää ja sen käyttötapausta **Lipun varaus, tilanne missä lippuja löytyy**
- Käyttötapauksen kulku:
  1. Käyttäjä kertoo tilaisuuden nimen ja päivämäärän
  2. Järjestelmä kertoo, minkä hintainen lippu on mahdollista ostaa
  3. Käyttäjä hyväksyy lipun
  4. Käyttäjälle annetaan tulostettu lippu
- Käyttötapauksen kulun voisi kuvata myös sekvenssikaavion avulla ajatellen koko järjestelmän yhtenä oliona
  - tällöin kyseessä *Järjestelmätason sekvenssikaavio*

Sekvenssikaavio

Käyttötapauksen Lipun varaus kuvaava sekvenssikaavio

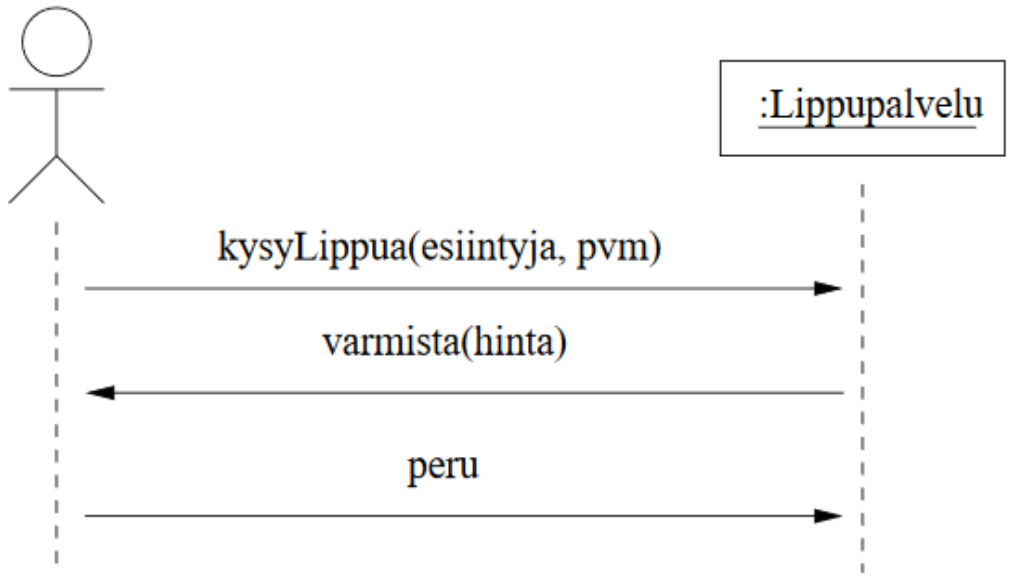


- Huom: Olioiden aktivaatiopalkit on jätetty kuvaamatta, sillä niille ei ole tarvetta esimerkissä

Sekvenssikaavio

Vaihtoehtoinen skenaario

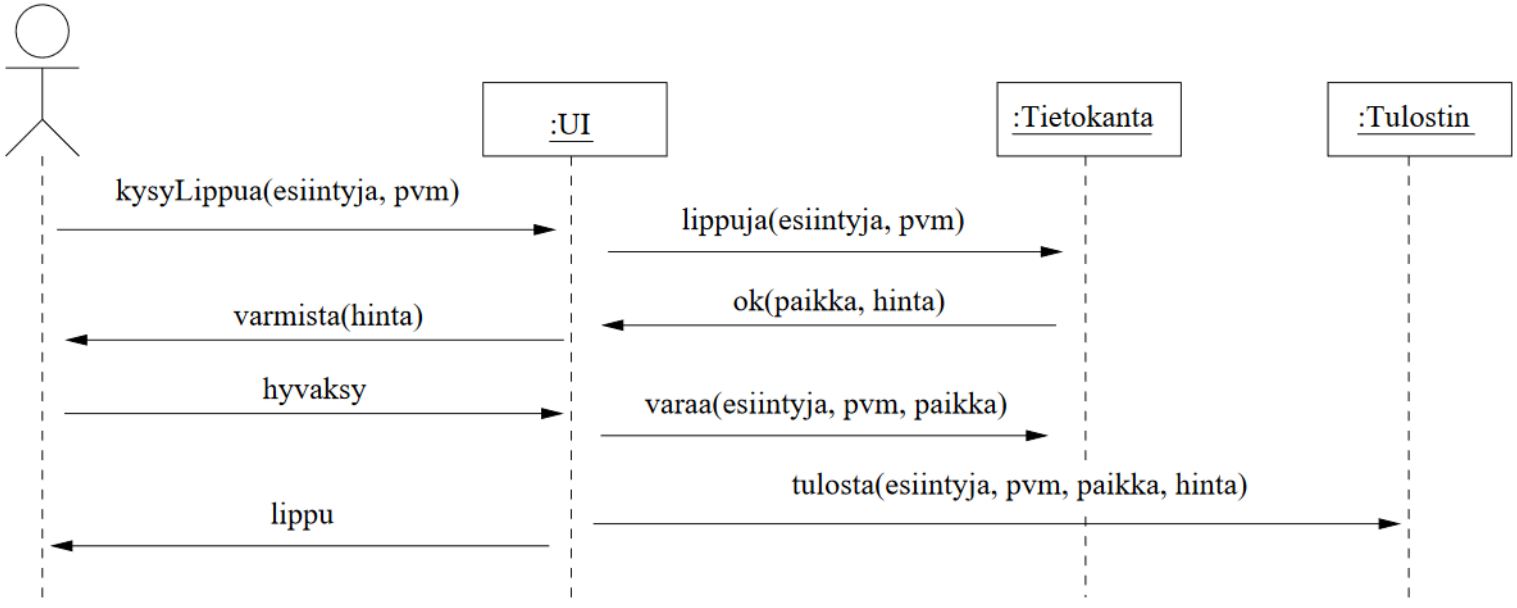
- Kuten kohta huomaamme, on myös yhteen sekvenssikaavioon mahdollista sisällyttää valinnaisuutta
- Toinen, usein selkeämpi vaihtoehto on kuvata vaihtoehtoiset skenaariot omina kaavioinaan
- Alla järjestelmätason sekvenssikaaviona tilanne, jossa asiakas hylkää tarjotun lipun



Sekvenssikaavio

Järjestelmätasolta suunnittelutasolle

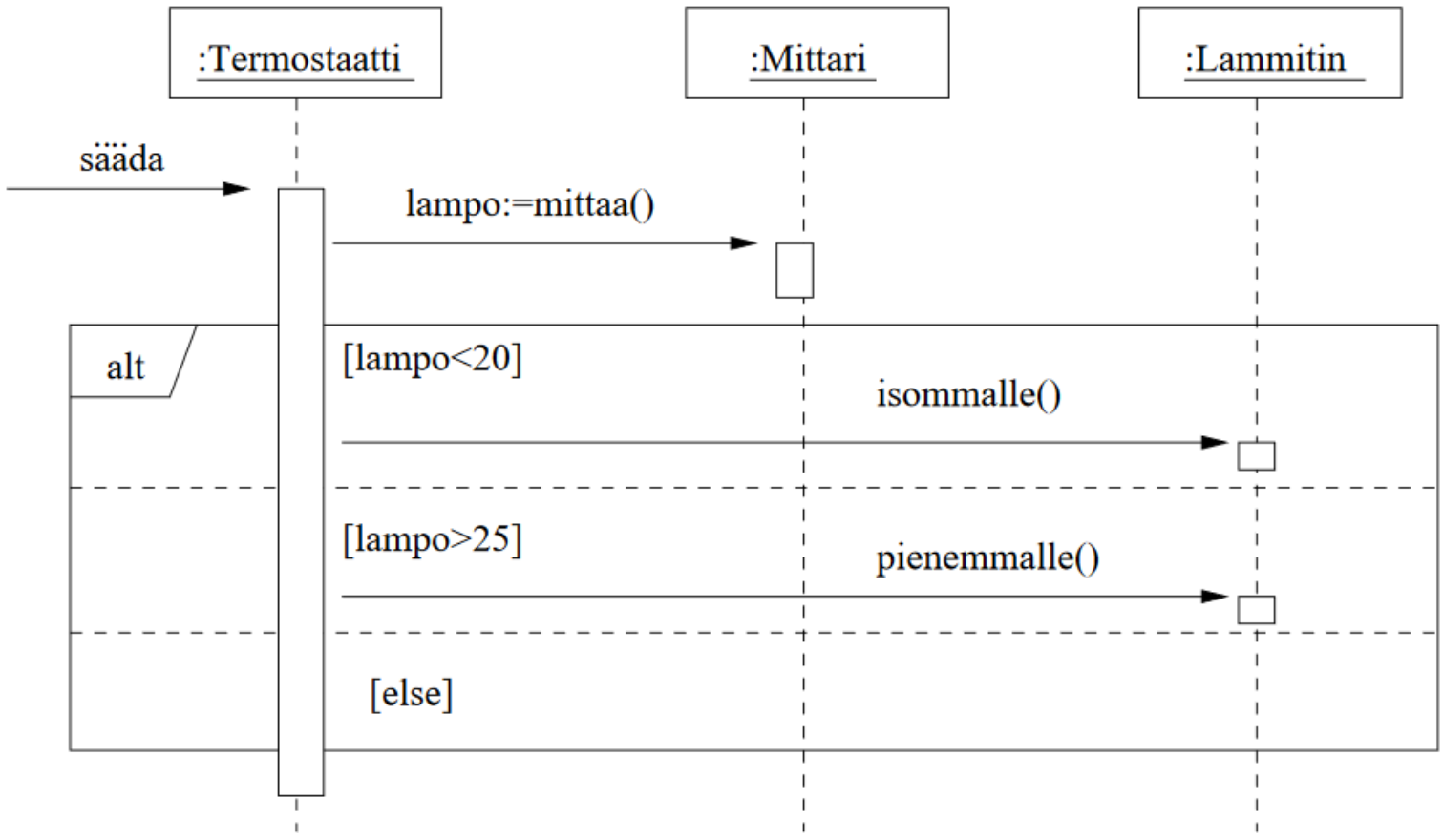
- Järjestelmätason sekvenssikaaviosta käy selkeästi ilmi käyttäjän ja järjestelmän interaktio
- Järjestelmän sisälle ei vielä katsota
- Seuraava askel on siirtyä suunnitteluun ja tarkentaa miten käyttötapauksen skenaario toteutetaan suunniteltujen olioiden yhteistyönä



Sekvenssikaavio

Valinnaisuus

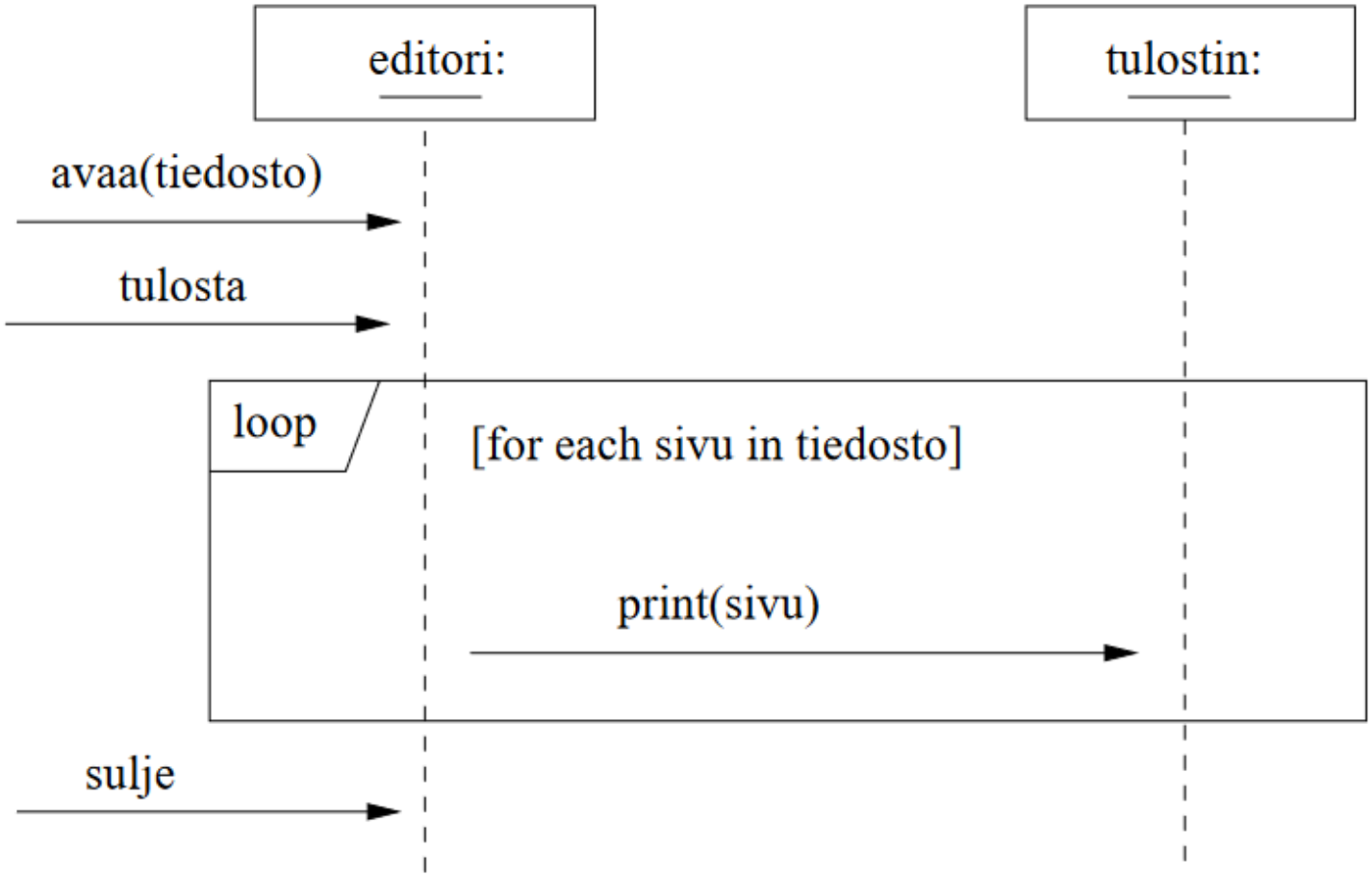
- Kaavioihin voidaan liittää lohko, jolla kuvataan valinnaisuutta
  - Vähän kuin if-else
  - Eli parametrina saadun arvon perusteella valitaan jokin kolmesta katkoviivan erottamasta alueesta



Sekvenssikaavio

Toisto

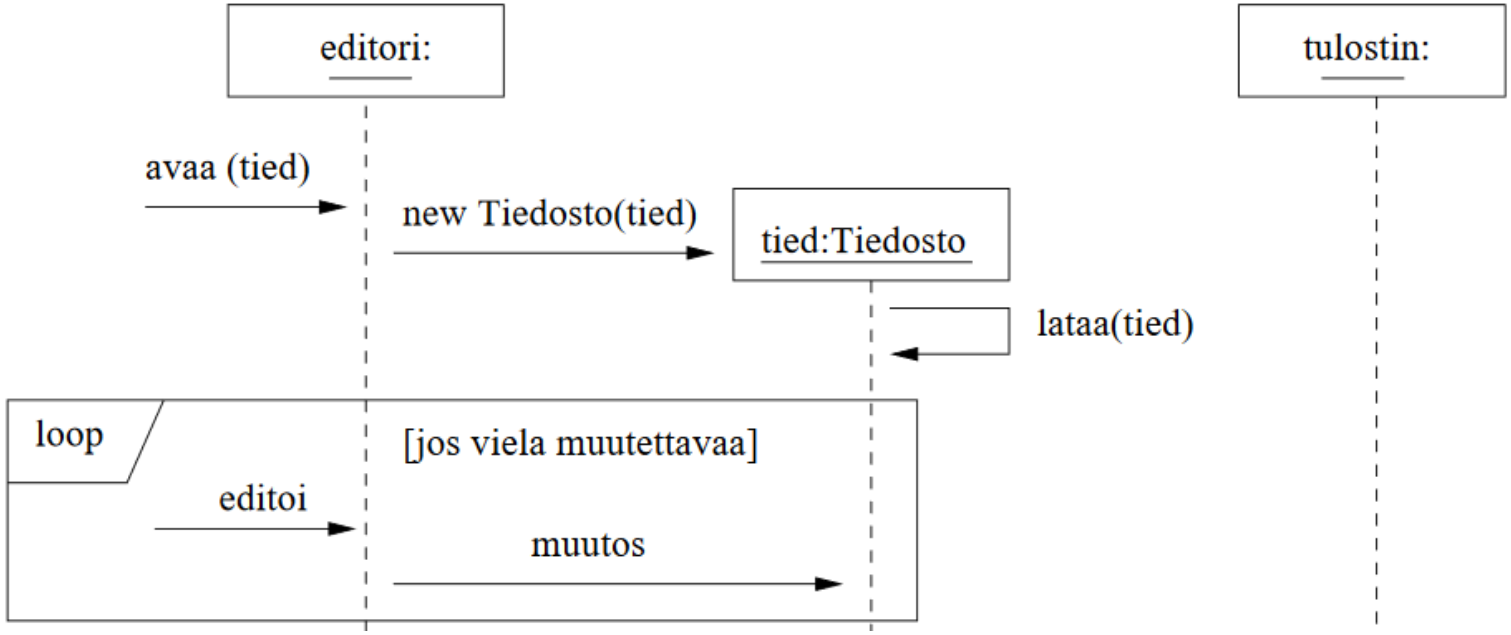
- Myös toistolohko mahdollinen (vrt. for tai while)
  - Huomaa miten toiston määrä on ilmaistu [ja] -merkkien sisällä
  - Voidaan käyttää myös vapaamuotoisempaa ilmausta, kuten "tulostetaan kaikki sivut erikseen"



Sekvenssikaavio

Olioiden luominen

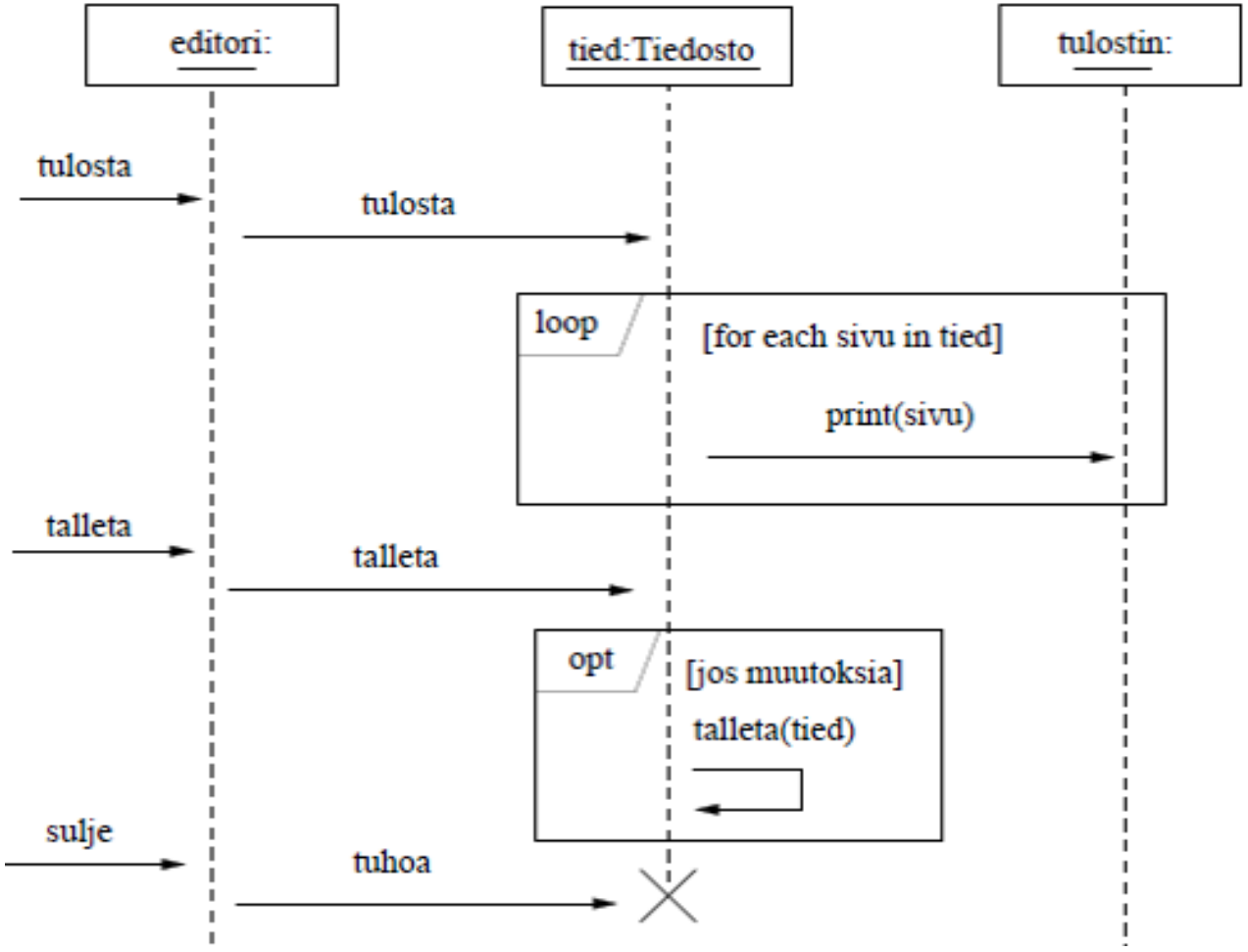
- Esimerkki luentomonisteesta, sivulta 65
- **Huom:** Uusi olio ei aloita ylhäältä vaan vasta siitä kohtaa milloin se luodaan



Sekvenssikaavio

Olioiden tuhoaminen

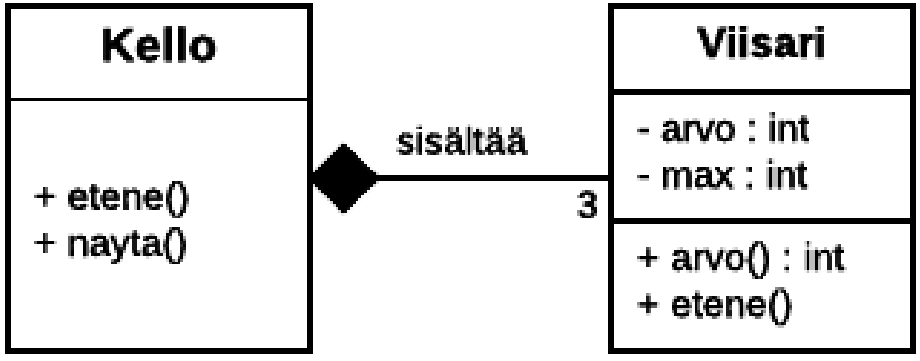
- Esimerkistä nähdään miten olion tuhoutuminen merkitään
- Mukana myös valinnainen (opt) lohko, joka suoritetaan jos ehto on tosi



Sekvenssikaavio

Takaisinmallinnus

- Takaisinmallinnuksella (engl. *reverse engineering*) tarkoitetaan mallien tekemistä valmiina olevasta koodista
  - Erittäin hyödyllistä, jos esim. tarve ylläpitää huonosti dokumentoitua koodia
- Seuraavilla sivulla löytyy Javalla toteutettu kello, joka nyt takaisinmallinnetaan
- **Luokkakaavio on helppo laatia:**



- Luokkakaaviosta ei vielä saa kuvaa kellon toimintalogiikasta joten tarvitaan sekvenssikaavioita

Sekvenssikaavio

Kello

```
public class Kello {
  private Viisari tunti;
  private Viisari minuutti;
  private Viisari sekunti;

  public Kello() {
    sekunti  = new Viisari(60);
    minuutti = new Viisari(60);
    tunti    = new Viisari(24);
  }

  public void etene(){
    sekunti->etene();
    if ( sekunti->aika()==0 ) {
      minuutti->etene();
      if ( minuutti->aika()==0 )
        tunti->etene();
    }
  }

  public void nayta(){
    System.out.print( tunti->aika() );    System.out.print(":");
    System.out.print( minuutti->aika() ); System.out.print(":");
    System.out.print( sekunti->aika() );
  }
}
```

Sekvenssikaavio

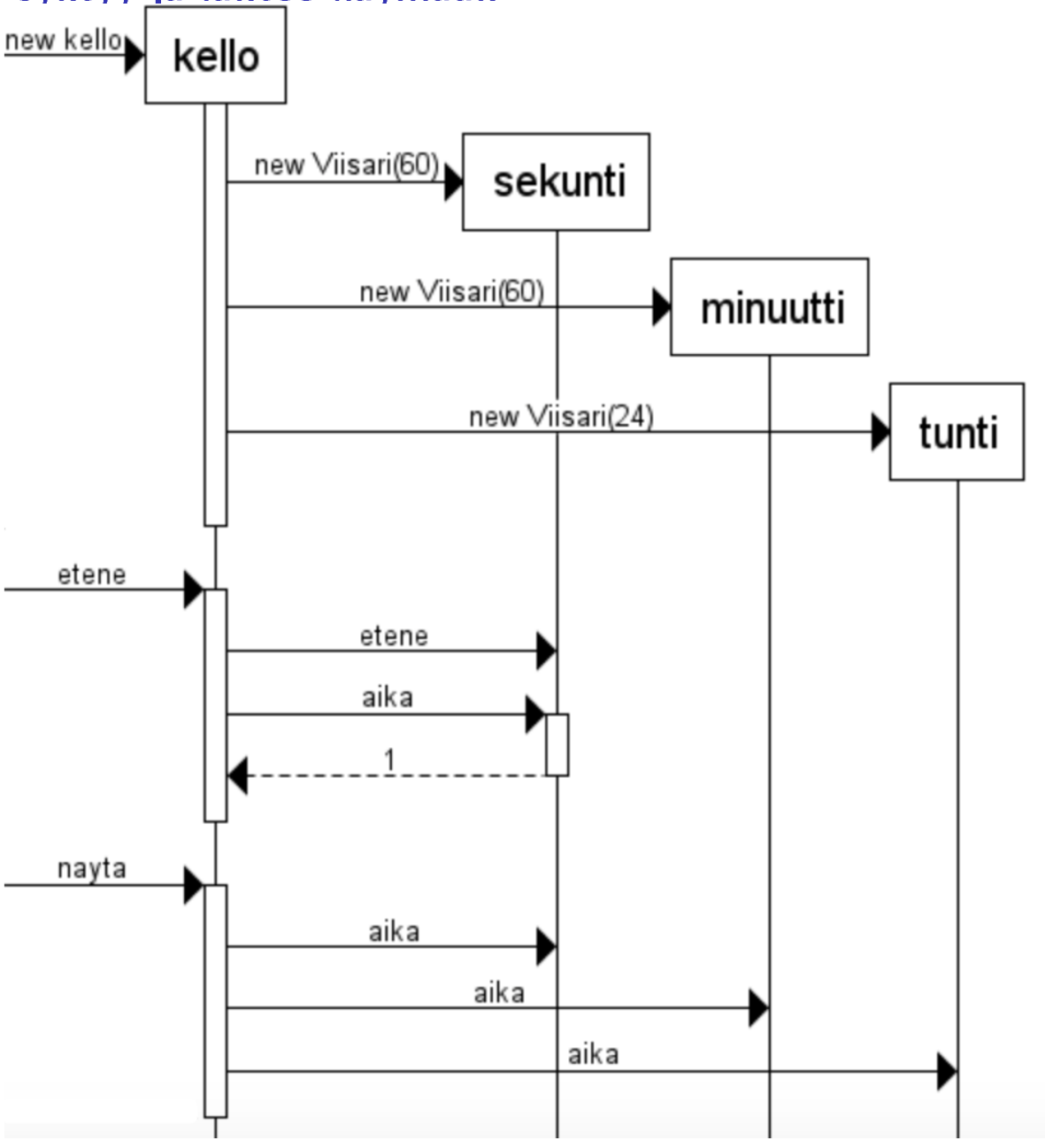
Viisari

```
public class Viisari{
  private int max;      // arvo, jonka saavutettuaan viisari pyörähtää nollaan
  private int arvo;

  public Viisari( int m ){ arvo = 0;  max = m; }

  public void etene(){
    arvo++;
    if ( arvo==max ) arvo = 0;
  }

  public int aika(){ return arvo; }
}
```

- kuvataan ensin kellon syntyminen, ensimmäisen sekunnin eteneminen ja ajan näyttäminen
- kaaviosta on jätetty pois Java-standardikirjaston out-oliolle suoritetut print()-metodikutsut

Sekvenssikaavio

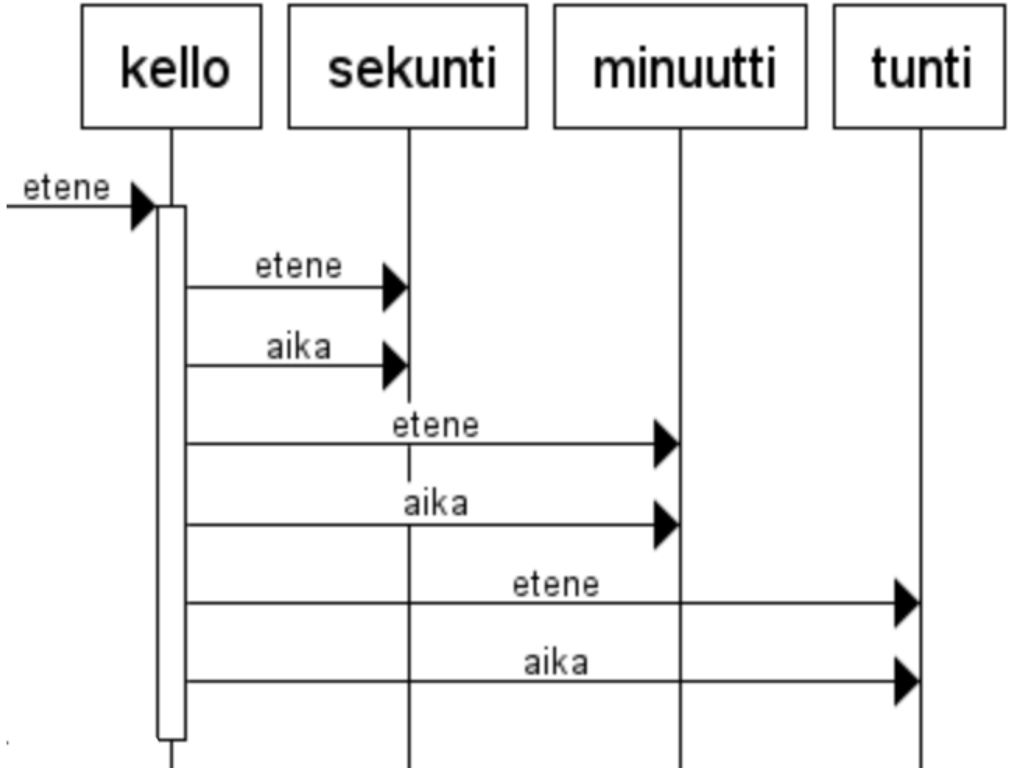
Kello syntyy ja lähtee käymään



Sekvenssikaavio

Kellon eteneminen tasatunnilla

- Tasatunnilla jokainen viisari etenee:



Luennolla tehtävä esimerkki: kurssin arvostelu

```
public class Paaohjelma {
    public static void main(String[] args) {
        Kurssi otm = new Kurssi();
        Opiskelija o1 = new Opiskelija("Arto", 45);
        Opiskelija o2 = new Opiskelija("Matti", 27);
        Opiskelija o3 = new Opiskelija("Maija", 55);

        otm.lisaaOpiskelija(o1);
        otm.lisaaOpiskelija(o2);
        otm.lisaaOpiskelija(o3);

        otm.arvosteleOpiskelijat();
        otm.tulostaKurssinTulokset();
    }
}

public class ArvosanaLaskin {
    public int selvitaArvosana(int pisteet) {
        if ( pisteet<30 )      return 0;
        else if ( pisteet<35 ) return 1;
        else if ( pisteet<40 ) return 2;
        else if ( pisteet<45 ) return 3;
        else if ( pisteet<50 ) return 4;

        return 5;
    }
```

```
public class Kurssi {
    private ArrayList<Opiskelija> opiskelijat;
    private ArvosanaLaskin arvosanaLaskin;

    public Kurssi(){
        opiskelijat = new ArrayList<Opiskelija>();
        arvosanaLaskin = new ArvosanaLaskin();
    }

    public void lisaaOpiskelija(Opiskelija opisk){
        opiskelijat.add(opisk);
    }

    public void arvosteleOpiskelijat(){
        for ( Opiskelija opisk : opiskelijat ) {
            int pisteet = opisk.getPisteet();
            int arvosana = arvosanaLaskin.selvitaArvosana( pisteet );
            opisk.setArvosana(arvosana);
        }
    }

    public void tulostaKurssinTulokset(){
        for ( Opiskelija opisk : opiskelijat )
            opisk.tulostaTiedot();
    }
}
```

```java
public class Opiskelija {
    private String nimi;
    private int pisteet;
    private int arvosana;

    public Opiskelija(String nimi, int pisteet) {
        this.nimi = nimi;
        this.pisteet = pisteet;
    }

    public int getPisteet(){
        return pisteet;
    }

    public void setArvosana(int arvosana){
        this.arvosana = arvosana;
    }

    public void tulostaTiedot(){
        System.out.println( nimi+" "+arvosana );
    }
}
```

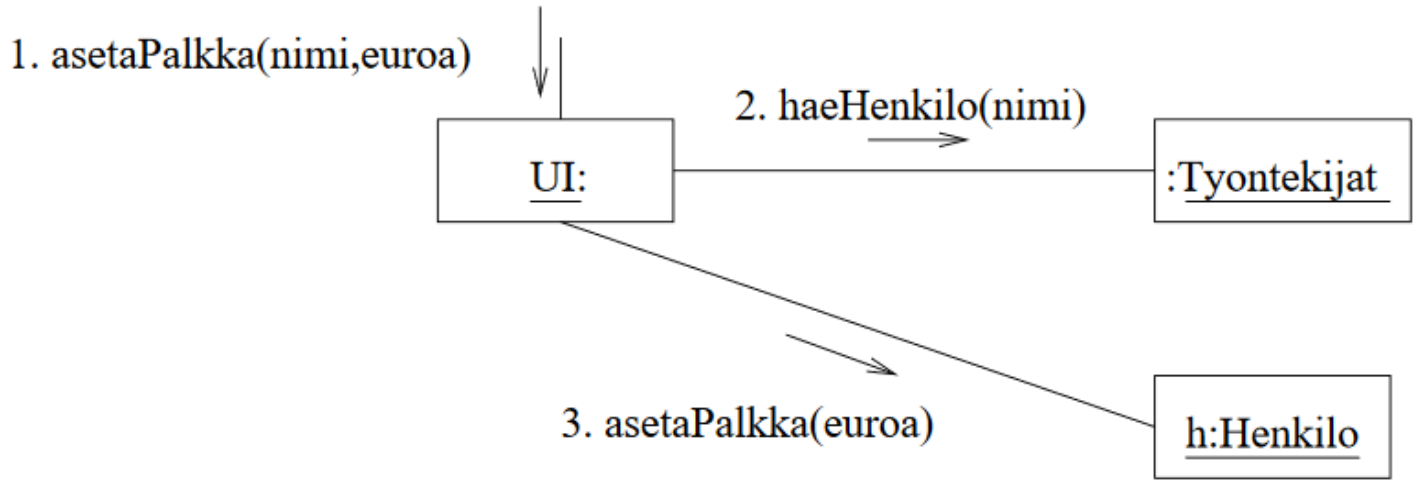
Kommunikaatiokaavio

Vaihtoehtoinen tapa kuvata yhteistoimintaa

Kommunikaatiokaavio

Vaihtoehtoinen tapa kuvata yhteistoimintaa

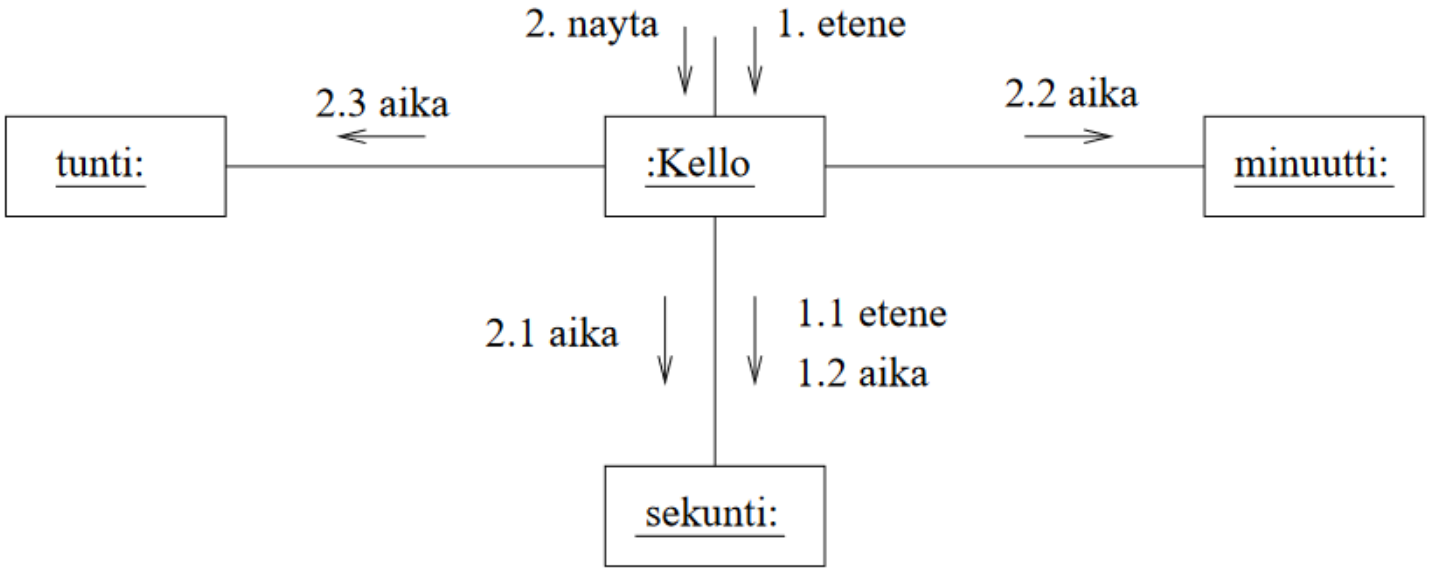
- Toinen tapa olioiden yhteistyön kuvaamiseen on kommunikaatiokaavio (engl. *communication diagram*)
- Alla muutaman sivun takainen esimerkki, jossa henkilölle asetetaan palkka
- Metodien suoritusjärjestys ilmenee numeroinnista, olioiden sijoittelu on vapaa



Kommunikaatiokaavio

Vaihtoehtoinen tapa kuvata yhteistoimintaa

- Viestien järjestyksen voi numeroida juoksevasti: 1, 2, 3, ...
- Tai allaolevan Kellon toimintaa kuvaavan esimerkin tyyliin hierarkkisesti:
  - Kellolle kutsutaan metodia *etene*, tällä numero 1
  - Eteneminen aiheuttaa sekuntiviisarille suoritetut metodikutsut *etene* ja *aika*, nämä numeroitu 1.1 ja 1.2
  - Seuraavaksi kellolle kutsutaan metodia *näytä*, numero 2
  - Sen aiheuttamat metodikutsut numeroitu 2.1, 2.2, 2.3, ...



Kommunikaatiokaavio

Yhteenveto olioiden yhteistoiminnan kuvaamisesta

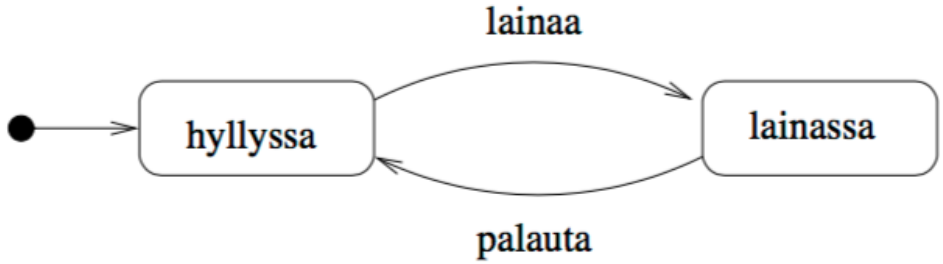
- Sekvenssikaavioita käytetään useammin kuin kommunikaatiokaavioita
  - Sekvenssikavio lienee luokkakaavioiden jälkeen eniten käytetty UML-kaaviotyyppi
- Sekä sekvenssi- että kommunikaatiokaavioilla tärkeä asema oliosuunnittelussa
- Kaaviot kannattaa pitää melko pieninä ja niitä ei kannata tehdä kuin järjestelmän tärkeimpien toiminnallisuuksien osalta
  - Kommunikaatiokaaviot ovat yleensä hieman pienempiä, mutta toisaalta metodikutsujen ajallinen järjestys ei käy niistä yhtä hyvin ilmi kuin sekvenssikaavioista
- On epäselvää missä määrin sekvenssikaavioiden valinnaisuutta ja toistoa kannattaa käyttää
- Sekvenssikaaviot on alunperin kehitetty tietoliikenneprotokollien kuvaamista varten

Lisää kaavioita

- Tähän mennessä olemme käsitelleet
  - luokkakaavioita
  - oliokaavioita
  - sekvenssikaavioita
  - kommunikaatiokaavioita
- Luokka- ja sekvenssikaaviot ovat varmasti kaksi ylivoimaisesti tärkeintä ja käytetyintä kaaviotyyppiä
- UML:ssa 13 erilaista kaaviotyyppiä
- Tutustumme tänään vielä kahteen kaaviotyyppiin
  - tilakaavioihin
  - aktiviteettikaavioihin
- Ensi viikolla tutustumme lyhyesti komponenttikaavioihin ja pakkauskaaviohin
- Kokonaan kurssin ulkopuolelle jää 5 eri kaaviotyyppiä

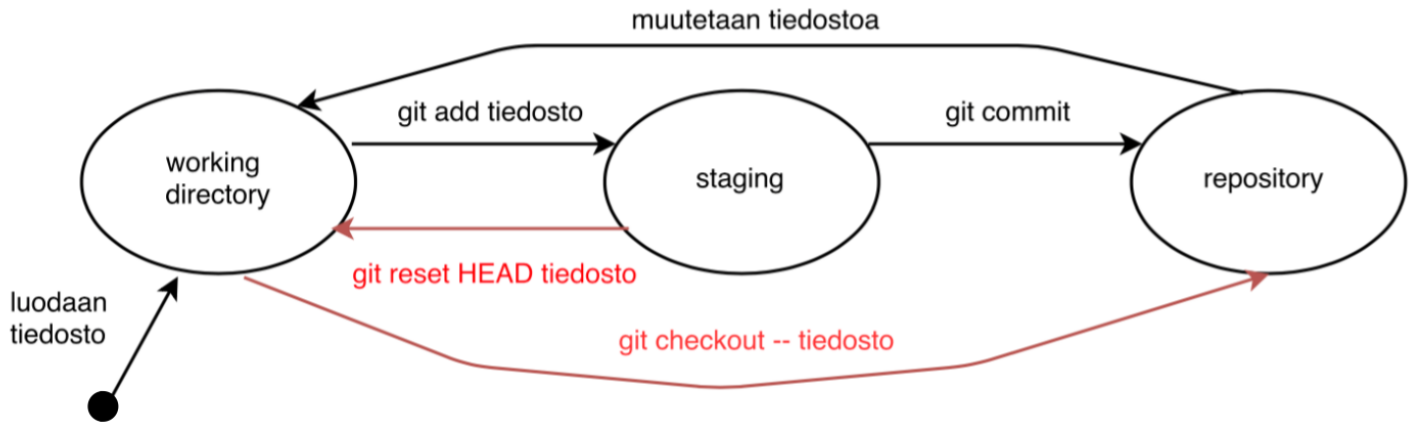
Tilakaavio

- Yksittäisten olioiden käyttäytymistapa voi olla erilainen eri tilanteissa
  - Luennon alun kirjastoesimerkin Kirja-oliot käyttäytyvät eri tavalla ollessaan lainassa kuin ollessaan hyllyssä
- Olion käyttäytyminen siis riippuu sen tilasta
  - Kun kirja on lainassa, ei sille voi suorittaa operaatiota lainaa
  - Kun kirja palautetaan, vaihtuu sen tila jälleen sellaiseksi, että uusi lainaus on mahdollista
- UML:n *tilakaavioiden* (engl. state machine diagram) avulla on mahdollista kuvailla olion tilasta riippuvaa käyttäytymistä
- Kirjalla *alkutila* sekä kaksi normaalia tilaa *hyllyssä* ja *lainassa*
- Tilojen välillä on *siirtymiä*. Siirtymän saa yleensä aikaan jokin tapahtuma tai heräte, esim. oliolle suoritettu metodikutsu



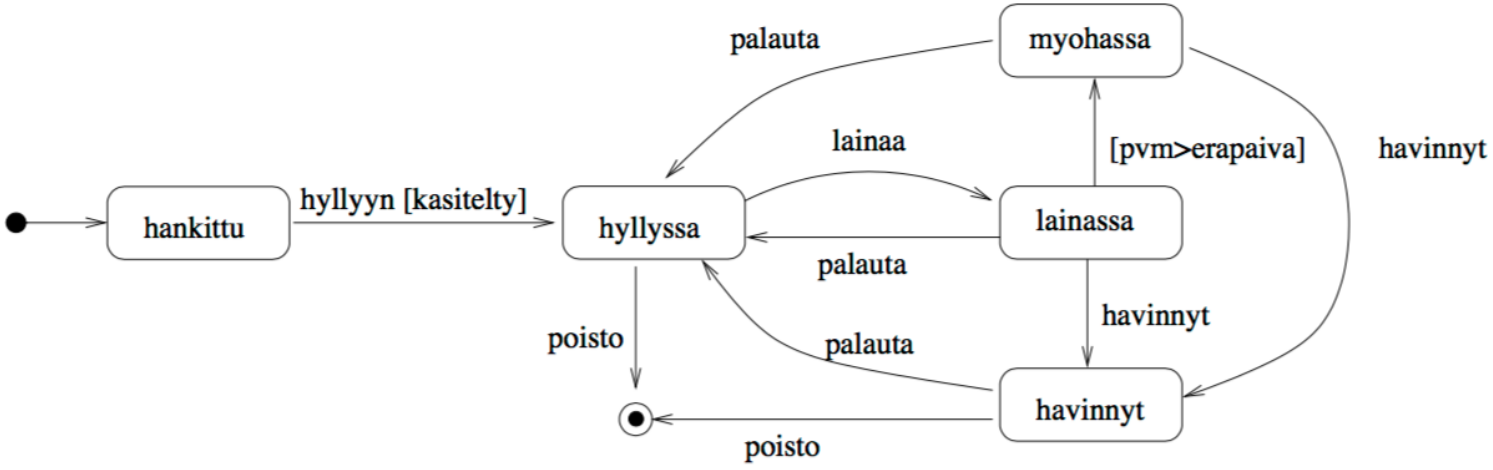
Tilakaavio

- Olemme jo nähneet kurssilla yhden tilakaavion
- Viikon 3 laskareissa kuvattiin versionhallinnassa olevan tiedoston tilaa
- Aluksi tiedosto on siis tilassa *working directory*
- Komento *git add* vie sen tilaan *staging*
- Komento *git commit* tallettaa tiedoston repositorioon, eli vie sen tilaan *repository*
- Tiloista staging ja working repository on myös muutosten perumista vastaavat siirtymät



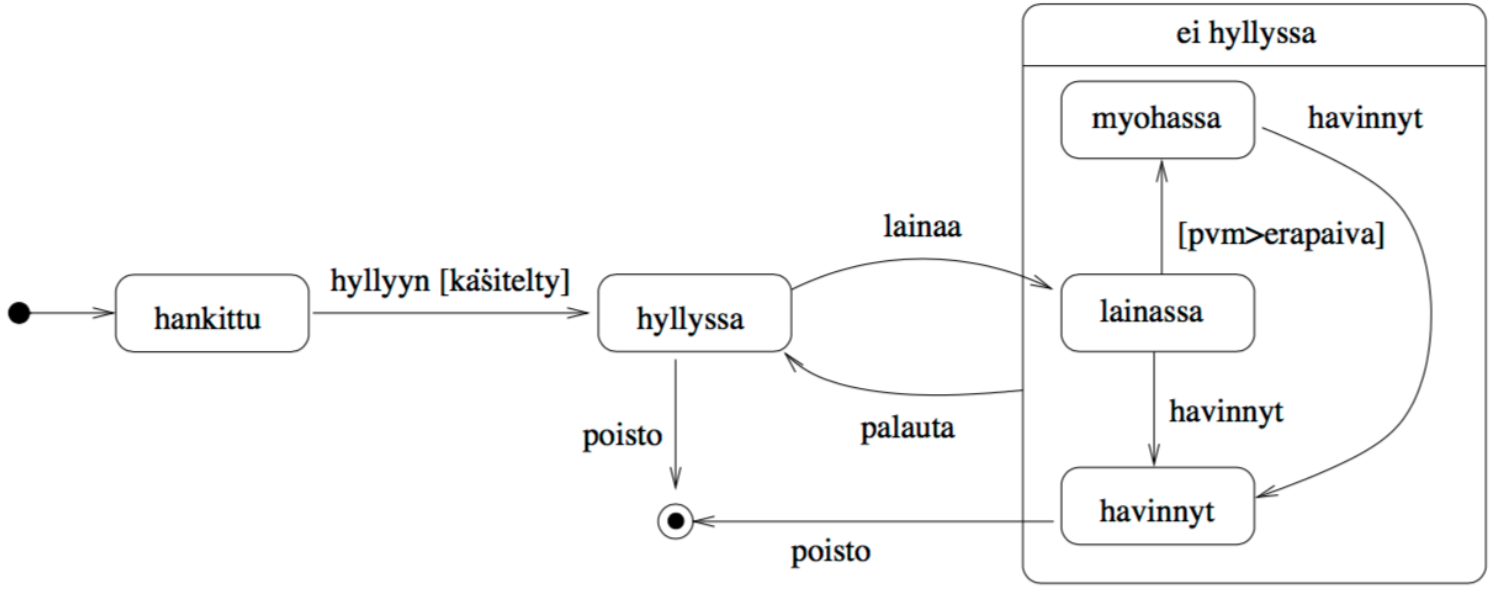
Tilakaavio

- Usein tila tarkoittaa tiettyjen oliomuuttujien arvojen sopivaa kombinaatiota tai yhteyksien olemassaoloa
- Kun kirja on lainassa, liittyy siihen Laina-olio. Hyllyssä olevaan kirjaan taas ei liity lainaa
- Kirjan tila selviää Laina-olioon yhteyden olemassaolon perusteella
- Alla yksityiskohtaisempi tilakaavo kirjasta
  - Muutamiin siirtymiin liittyy nyt *ehtoja*
  - Kirjalla on myös olion tuhoutumista kuvaava *lopputila*



Tilakaavio

- Edellisen sivun kuvan tilat *myohassa*, *lainassa* ja *havinnyt* sisältävät kaikki siirtymän tilaan *hyllyssä* tapahtumalla *palauta*
- Tämäntyyppisissä tilanteissa kaaviota on mahdollista yksinkertaistaa *sisäkkäisten tilojen* avulla
- Alla kirjan tilamalli, jossa on *ylitila* (engl. superstate) eihyllyssä, joka sisältää edellä mainitut kolme samankaltaista *alitilaa* (engl. substate).



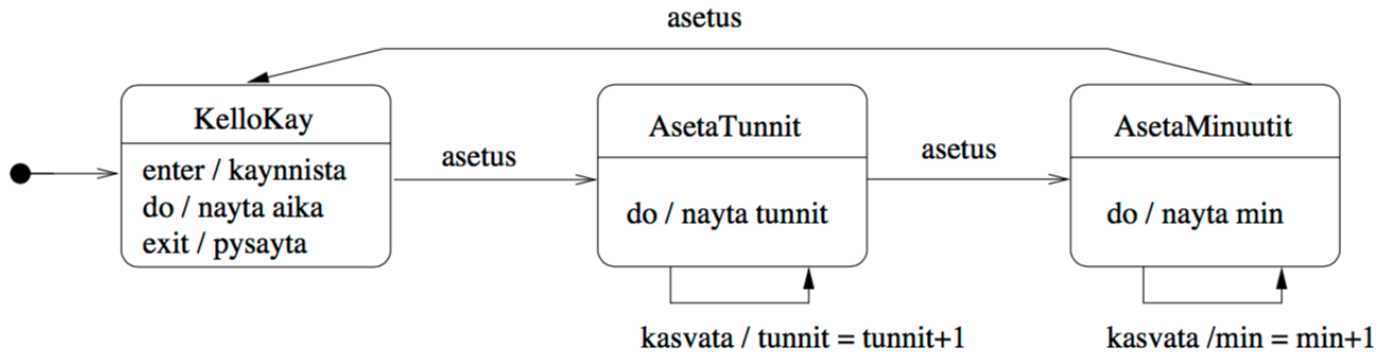
Tilakaavio

Toiminnot ja muuttujat

- Seuraavalla sivulla digitaalikellon toiminta tilakaaviona. Kaavio hyödyntää mahdollisuutta liittää tiloihin toimintoja
- Oltaessa tilassa *KelloKay* näytetään aikaa
  - tilaan liitetetty on toiminto *do / nayta aika*
- Tilaan tultaessa kello käynnistyy
  - tilaantulotoimintona *enter / kaynnista*
- Tilasta poistuttaessa kello pysähtyy
  - tilasta poistumistoimintona *exit / pysayta*
- Tapahtumalla *asetus* siirrytään tilaan, jossa voidaan asettaa tuntiviisarin aika toiminnolla kasvata
- Tapahtuman aikaansaama toiminto on merkitty muodossa *kasvata / tunnit = tunnit+1*
  - ensin on merkitty *tapahtuma* (metodikutsu kasvata), jonka perässä /-merkin jälkeen tapahtuman aikaansaama *toiminto* (muuttujan tunnit arvon kasvatus)
- Tilassa *AsetaTunnit* oltaessa näytetään tuntiviisarin aikaa.

Tilakaavio

Toiminnot ja muuttujat

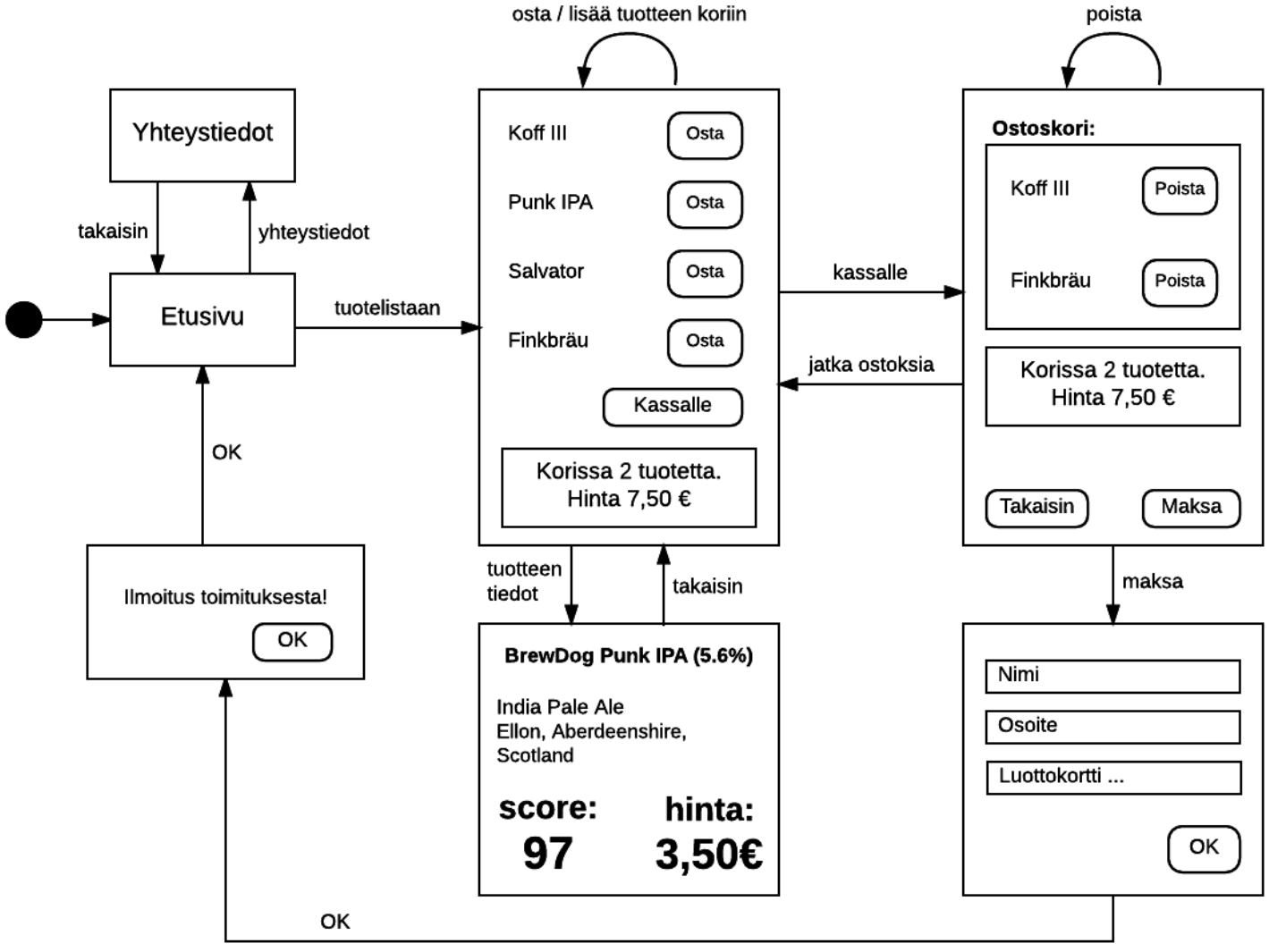


- Tilakaaviossa esiintyvät toiminnot voivat liittyä suoraan tilakaavion kuvaaman olion attribuutteihin, esim. tunnit = tunnit+1
- Tilakaavion siirtymiin liittyvät tapahtumat kuten *kasvata* taas ovat yleensä olion metodeja
- Toisaalta, joskus tilakaaviolla kuvataan ainoastaan sitä, miltä järjestelmän toiminta näyttää ulospäin
  - todellisuudessa järjestelmä muodostuu useista olioista
  - Tällöin tilakaaviossa esiintyvät "muuttujanimet" eivät välttämättä vastaa suoraan mitään toteutustason attribuuttia

Milloin ja mihin tilakaavioita kannattaa tehdä?

- Voidaan käyttää vaatimusmäärittelyssä tai suunnittelussa
  - Kirja ja Kello ovat esimerkkejä määrittelyvaiheen tilamalleista, joissa kuvaillaan sovelluksen toimintalogiikkaa ottamatta kantaa toteutukseen
- Tilakaavio on mielekästä tehdä ainoastaan asioista/olioista, joilla on selkeä elinkaari, joka sisältää erilaisia toimintatiloja, joissa olio/asia on ulkoiselta käyttäytymiseltään erilainen
- Eräs tilamallien sovelluskohde on sovelluksen käyttöliittymän navigaatiorakenteen kuvaaminen, esimerkki seuraavalla sivulla
  - tiloina ovat sovelluksen eri näkymät ja siirtymät tilojen välillä kuvaavat käyttäjän navigointia sovelluksen näkymien välillä
- Tilamallinnus on avainasemassa esim. tietoliikenneprotokollien tai reaaliaikajärjestelmien mallinnuksessa
  - Kurssilla tietoliikenteen perusteet tilakaaviota käytetään paljon
  - Kurssilla laskennan mallit tarkastellaan tilakaavioiden teoreettista puolta

Verkkokaupan käyttöliittymän navigaatiorakenteen kuvaaminen tilakaaviona



Aktiviteettikaavio

- Tilakaaviot kuvaavat lähinnä yksittäisen olion toimintaa poikkeuksena käyttöliittymän navigaatiorakenteen kuvaus
- *Aktiviteettikaavioilla* (engl. activity diagram) on mahdollisuus kuvata suurempaa toiminnallista kokonaisuutta, esimerkiksi:
  - Kokonaista liiketoimintaprosessia
  - Tiedon ja työn kulkua järjestelmässä monen toimijan kannalta
  - Käyttötapauksen etenemislogiikkaa
- Aktiviteettikaavioissa kuvataan sarja *toimintoja* ja niiden suoritusjärjestys
  - Toiminnot on kuvattu pyöreäreunaisina suorakulmioina
  - Toimintojen peräkkäisyys niitä yhdistävinä nuolina
  - Toimintojen rinnakkaisuus kuvataan haarautumisen avulla
- Samankaltaisuudesta huolimatta aktiviteettikaavioita ei pidä sekoittaa tilakaavioihin
- Tarkastellaan ensin aktiviteettikaavioiden käyttöä liiketoimintaprosessien kuvaamisessa
  - Esimerkkinä kaavio, joka kuvaa tilauksen vastaanottamiseen, toimittamiseen ja laskutukseen liittyviä toimintoja

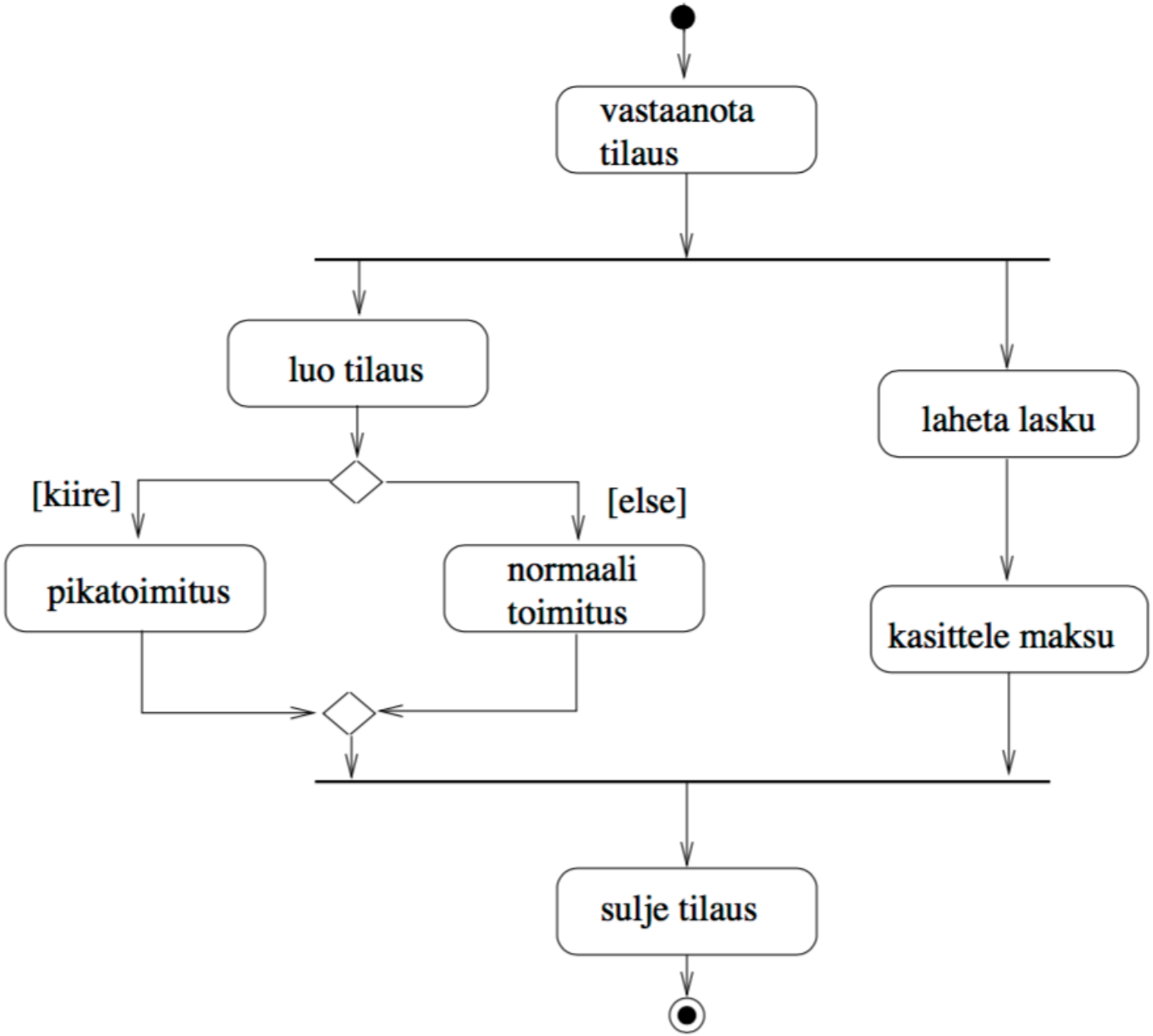
Aktiviteettikaavio

Tilauksen vastaanottaminen, toimitus ja laskutus

- *Aloitussymboli* ohjaa ensimmäiseen toimintoon (engl. action), eli tilauksen vastaanottoon
- Tämän jälkeen kontrolli *haarautuu* (engl. fork) kahteen rinnakkain etenevään toimintosarjaan
- Vasemman ja oikean haaran toiminnot siis etenevät rinnakkain toisistaan riippumattomina
- Oikea haara kuvaa laskutuksen (laskun lähetys ja maksun vastaanotto) ja vasen haara toimituksen
- Toimitus sisältää vielä haarautumisen pikatoimitukseen ja normaaliin toimitukseen, näistä siis valitaan ainoastaan toisen haaran toiminto.
- Laskutus- ja toimintohaara *yhdistyvät* (engl. join)
- Eli yhdistymissymbolin (viiva johon saapuu kaksi nuolta ja josta lähtee yksi nuoli) jälkeen kontrolli jatkaa ainoastaan yhdessä haarassa ja toiminnoissa ei ole enää rinnakkaisuutta
- Viimeisen toiminnon (sulje tilaus) jälkeen aktiviteetti loppuu

Aktiviteettikaavio

Tilauksen vastaanottaminen, toimitus ja laskutus



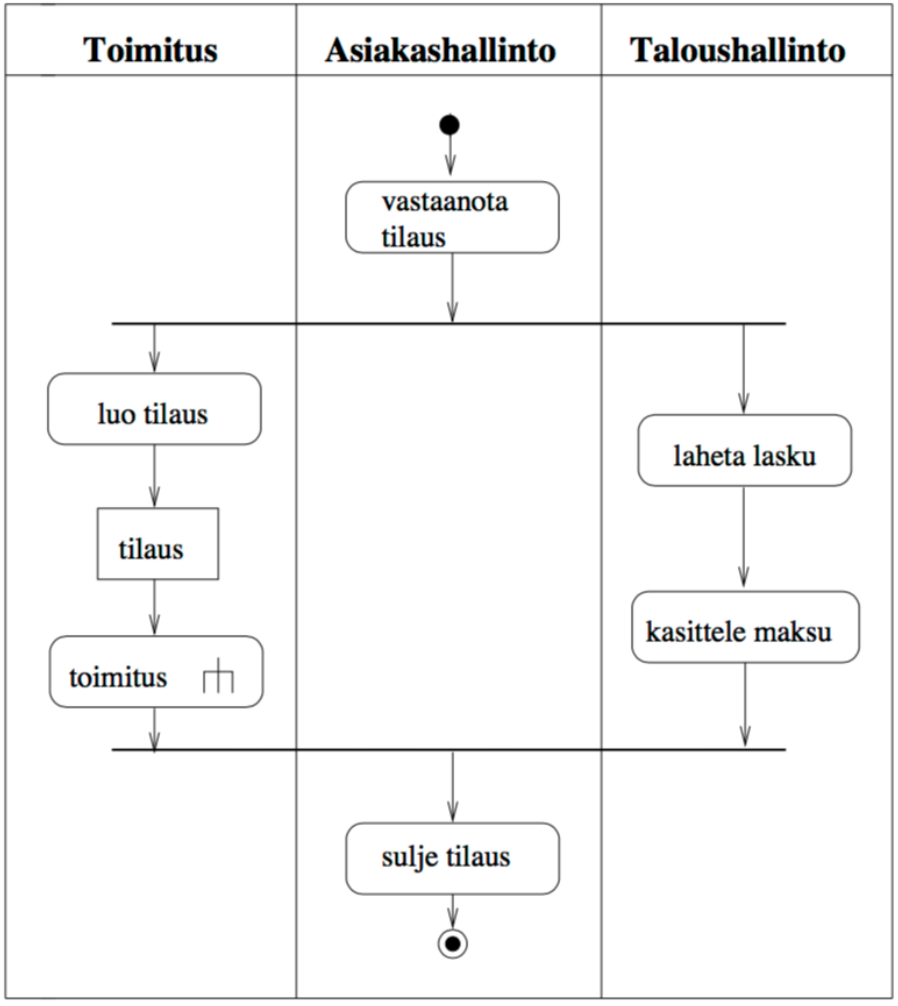
Aktiviteettikaavio

Kaistoihin jaettu aktiviteettikaavio

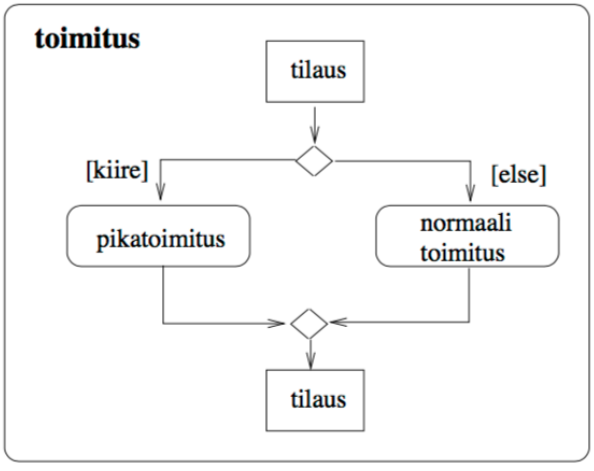
- Esimerkissämme aktiviteettikaavio siis kuvaa mitä yhdelle tilaukselle tapahtuu sen elinkaaren aikana
- Kaavion eri toiminnot ovat todennäköisesti organisaation eri toimijoiden suorittamia
- Tätä voidaan korostaa, jakamalla kaavio *kaistoihin* (engl. swimlane), eli erillisiin osiin, jotka jaottelevat sen kuka on vastuussa toiminnon suorittamisesta
- Tilauksen käsittely jaettuna toimituksen, asiakashallinnon ja taloushallinnon vastuisiin on esitetty seuraavan kalvon kuvassa
- Toiminnon luo tilaus seurauksena syntyvä Tilaus-olio on otettu malliin mukaan
  - Tilaus-olio siirtyy parametrina toimintoon toimitus
- Toimitus on nyt mallinnettu ainoastaan karkealla tasolla sisältäen viitteen (haarukkasymboli) tarkentavaan aktivitettikaavioon

Aktiviteettikaavio

Kaistoihin jaettu aktiviteettikaavio



- Kaista ilmaisee kenen vastuulla toiminnon suoritus on
- Toimitus tarkentavassa kaaviossa
- Toimintojen välillä liikkuva data (tilaus) merkitty malliin



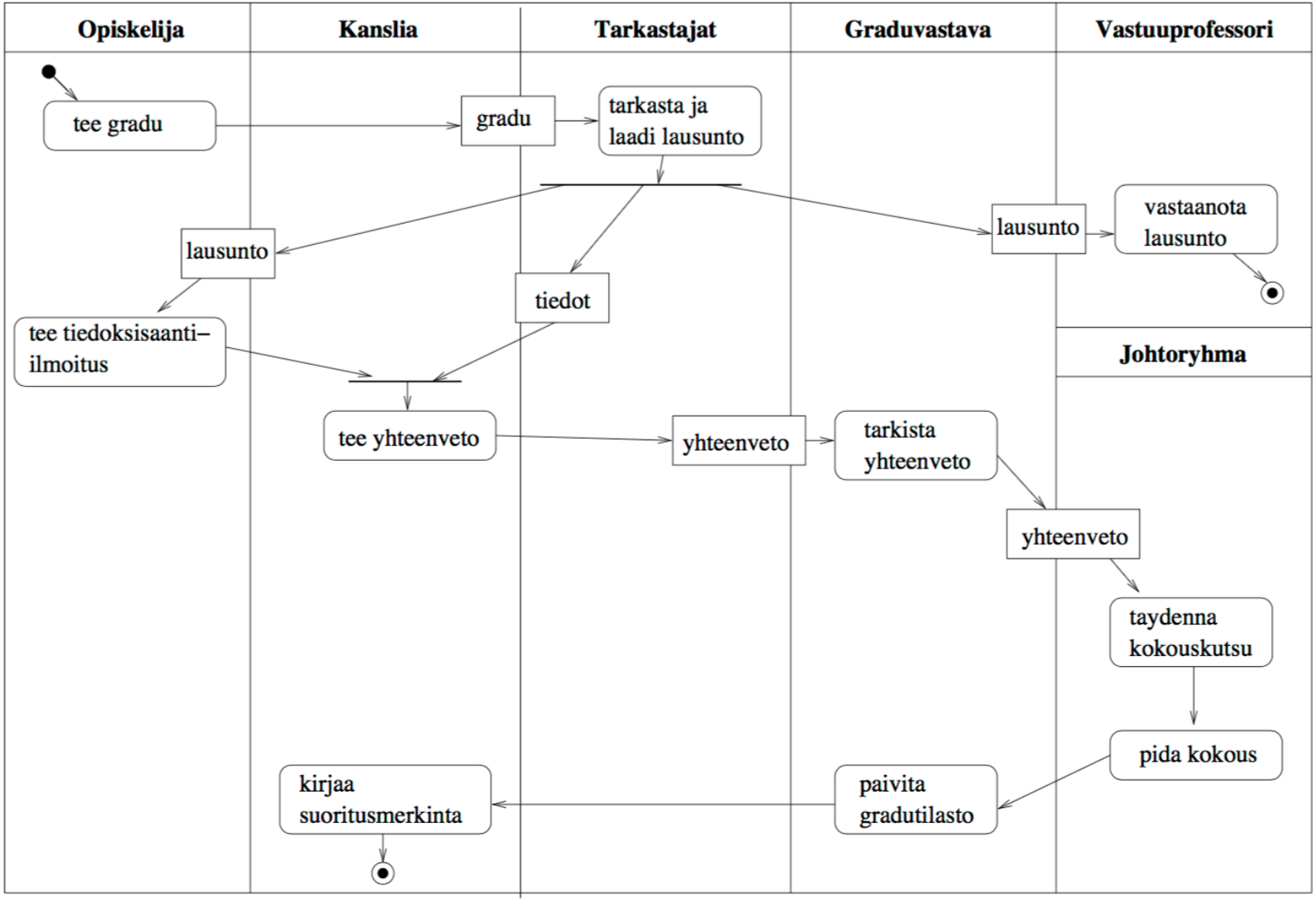
Aktiviteettikaavio

TKTL:n Pro Gradu -tutkielmien hyväksymiseen liittyvät toimet aktiviteettikaaviona

- Gradun valmistuttua opiskelija toimittaa sen tarkastajille
- Tarkastettuaan gradun tarkastajat laativat lausunnon, joka toimitetaan sekä opiskelijalle että linjan vastuuprofessorille
- Tässä kohdassa toiminta haarautuu ja samaan aikaan tarkastajat toimittavat gradun tiedot kansliaan
- Saatuaan opiskelijalta tiedoksisaanti-ilmoituksen ja tarkastajalta gradun tiedot, toimittaa kanslia gradusta yhteenvedon graduvastaavalle
- Tarkastettuaan yhteenvedon, toimittaa graduvastaava sen johtoryhmälle
- Johtoryhmä täydentää kokouskutsua gradun osalta ja pitää kokouksen
- Kokouksen jälkeen johtoryhmä pyytää graduvastaavaa päivittämään laitoksen gradutilastoa
- Graduvastaava toimittaa tiedon johtoryhmän hyväksynnästä kansliaan, jossa kirjataan opiskelijalle suoritusmerkintä

Aktiviteettikaavio

TKTL:n Pro Gradu -tutkielmien hyväksymiseen liittyvät toimet aktiviteettikaaviona



Aktiviteettikaavio

Prosessikuvauksesta aktiviteettikaavioksi

- Lähteenä edellisessä TKTL:n intranetistä löytyvä tekstuaalinen kuvaus https://www.cs.helsinki.fi/henkilokunta-intranet/hyvaksymismenettely.pdf
- Aktiviteettikaaviona tehty prosessin kuvaus on pakostakin hiukan ylimalkainen
- Esim. eri toimenpiteiden välillä välitettävä tieto on määriteltävä tarkemmin kaavion ulkopuolella, esim. luokkakaavioiden avulla
- Aktiviteettikaavio kuitenkin antaa joissain tilanteissa pelkkää tekstuaalista kuvausta paremman yleiskuvan prosessin kulusta

Aktiviteettikaavio

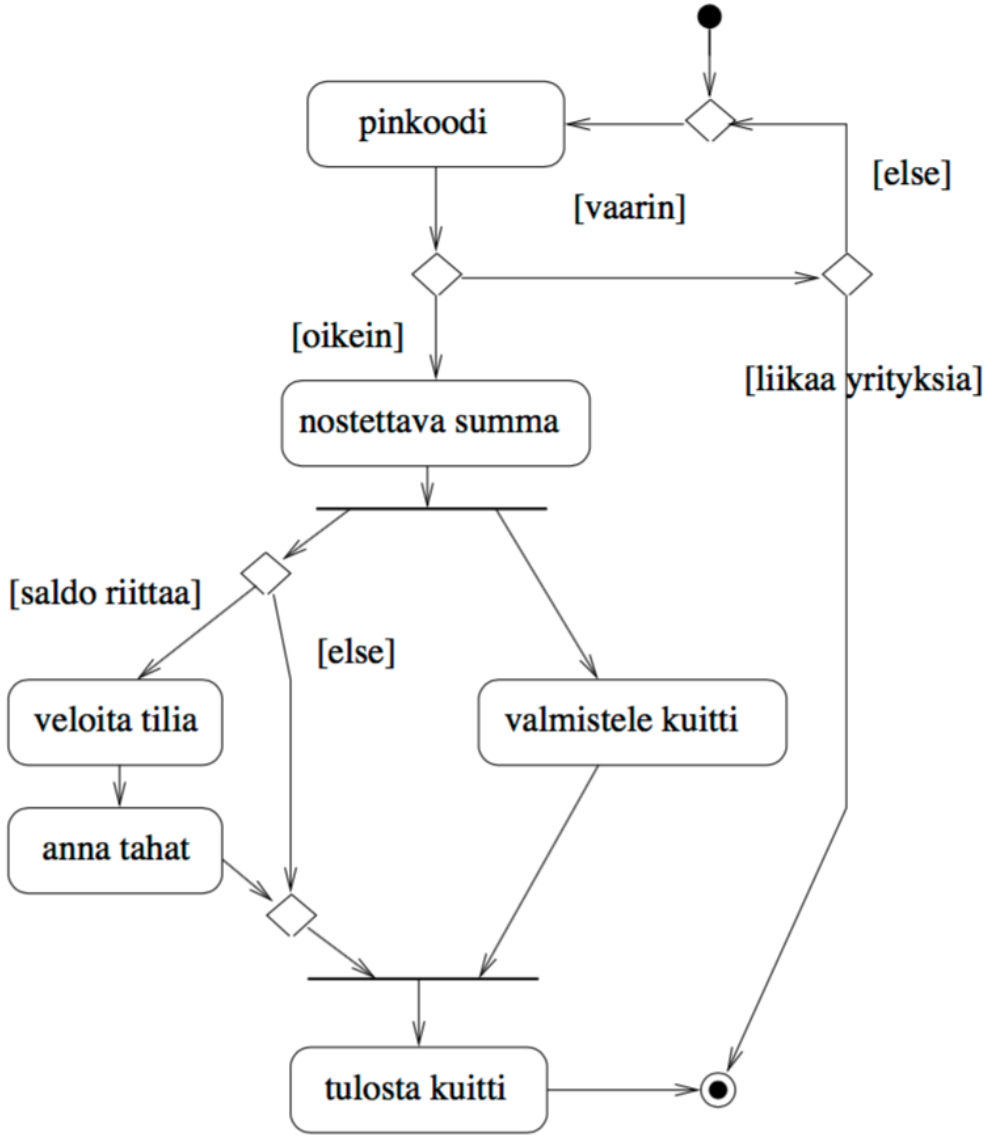
Käyttötapaus aktiviteettikaaviona

- Aktiviteettikaaviota voidaan hyödyntää myös yksittäisen käyttötapauksen toimintalogiikan kuvaamiseen
- Esimerkiksi käyttötapaus **nostotapahtuma pankkiautomaatista** voitaisiin mallintaa seuraavan kalvon aktiviteettikaaviolla
- Esimerkin aktiviteettikaavio kuvaa ainoastaan yhden käyttötapauksen aikaisia toimintoja, eikä esimerkiksi yritäkään kuvailla pankkiautomaatin toimintaa kokonaisuudessaan

- **Käyttötapaus nostotapahtuma pankkiautomaatista**
  1. Käyttäjä syöttää PIN-koodin
     - Liian monta väärää koodia ja automaatti nielaisee kortin
  2. Jos saldo riittää, annetaan rahat
  3. Lopuksi palautetaan kortti ja tulostetaan kuitti

Aktiviteettikaavio

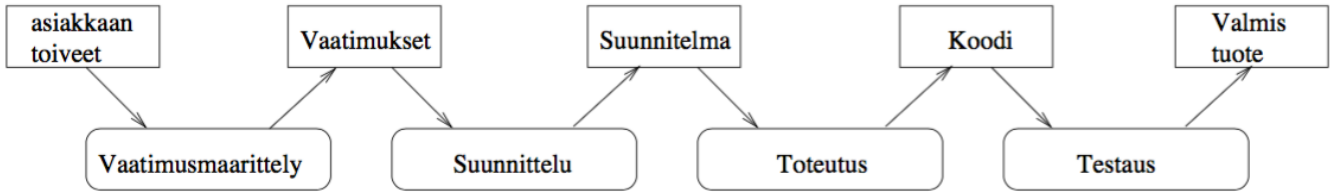
Käyttötapaus aktiviteettikaaviona



Aktiviteettikaavio

Aktiviteettikaavion käyttö ja hyödyt

- Aktiviteettikaaviot ovat tällä kurssilla esitellyistä vähiten ohjelmistokehityksessä käytetty kaaviotyyppi
- Samantapaisia kaavioita käytetään paljon muilla aloilla erilaisia toimintaprosesseja kuvailtaessa
- Esim. vesiputousmallin mukainen ohjelmistokehitysprosessi mallinnettiin toisella luennolla aktiviteettikaaviona:



- Aktiviteettikaavioita sovelletaan ohjelmistotuotannossa lähes yksinomaan vaatimusmäärittelyssä
  - Aktiviteettikaavioiden avulla voidaan ohjelmiston vaatimuksien kartoitusvaiheessa esim. kuvailla työprosesseja, joita halutaan automatisoida kehitettävän ohjelmiston avulla

Aktiviteettikaavio

Aktiviteettikaaviot ja vuokaaviot

- Voidaan ajatella, että aktivitettikaaviot ovat vanhan kunnon vuokaaviotekniikan (engl. flow chart diagram) modernisoitu ja UML-kieleen otettu versio
- Vuokaavioita on käytetty tietotekniikassa jo 60-luvulta asti toimintaprosessien ja jopa algoritmien abstraktina kuvaamismenetelmänä
- Ei tällä kurssilla vuokaavioista sen enempää
- Vaikuttaa siltä, että vuokaavioiden kulta-aika on ohi
- http://en.wikipedia.org/wiki/Flowchart