

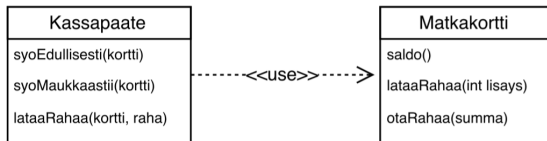
Ohjelmistotekniikan menetelmät

luento 5, 29.11.2016

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

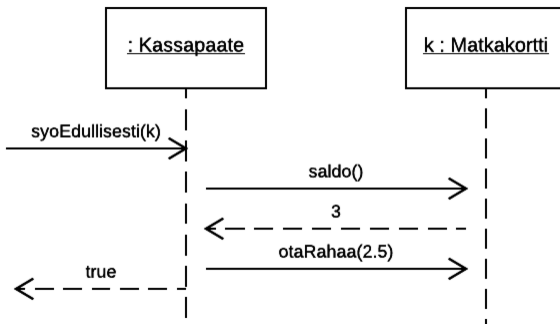
Koska järjestelmän toiminnan kulmakivenä on järjestelmän sisältämien olioiden yhteistyö, tarvitaan menetelmä yhteistyön kuvaamiseen

- ▶ 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

Olioiden yhteistyön mallintaminen

Sekvenssikaavio

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



Olioiden yhteistyön mallintaminen

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

Olioiden yhteistyön mallintaminen

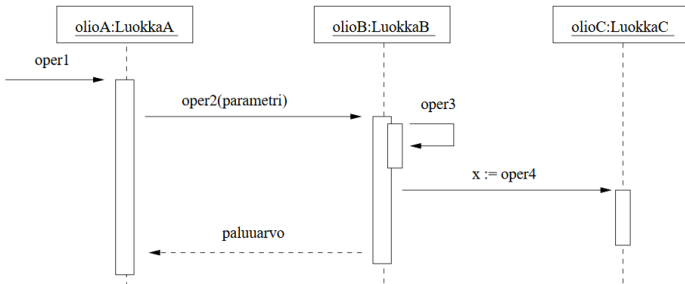
Sekvenssikaavio

- ▶ 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

Olioiden yhteistyön mallintaminen

Sekvenssikaavio

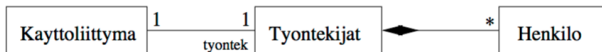
- ▶ Joskus on hyödyllistä piirtää **aktivaatiopalkki**, joka merkitsee ajan jolloin olio on aktiivisena, eli sen suoritus on kesken
- ▶ Aktivaatiopalkkia harvemmin jaksetaan piirtää paperille
- ▶ Merkinnällä $x := \text{oper4}$ tarkoitetaan, että metodia `oper4()` kutsutaan ja sen palautusarvo otetaan talteen muuttujaan `x`



Olioiden yhteistyön mallintaminen

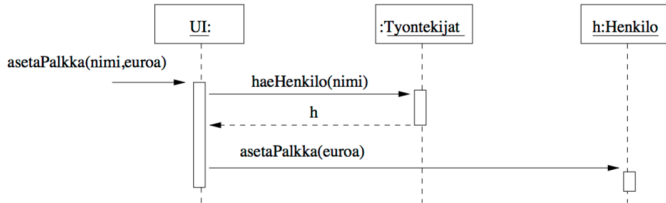
Henkilöstöhallinta ja palkanmaksu

- ▶ Tarkastellaan Henkilöstöhallintajärjestelmää
 - ▶ Työntekijät-olio pitää kirjaa työntekijöistä, jotka Henkilö-oliota
 - ▶ Sovelluslogiikka-olio hoitaa korkeamman tason komentojen käsittelyn
- ▶ Tarkastellaan operaatiota lisääPalkka(nimi, palkka)
 - ▶ Lisätään parametrina annetulle henkilölle uusi palkka
- ▶ Suunnitellaan, että operaatio toimii seuraavasti:
 - ▶ Ensin sovelluslogiikka hakee Tyontekijat-oliolta viitteen Henkilo-olion
 - ▶ Sitten sovelluslogiikka kutsuu Henkilo-olion palkanasetusmetodia
- ▶ Seuraavalla sivulla operaation suoritusta vastaava sekvenssikaavio
 - ▶ Havainnollistuksena myös osa luokan Sovelluslogiikka koodista



Olioiden yhteistyön mallintaminen

Henkilöstöhallinta ja palkanmaksu



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

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

Olioiden yhteistyön mallintaminen

Henkilöstöhallinta 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 huomaamalla parametrin h käyttö edellisen sivun kuvassa
- ▶ `haeHenkilo()`-metodikutsun paluuarvo on h
- ▶ Kyseessä on sama h , joka on sekvenssikaaviossa esiintyvän olion nimi!

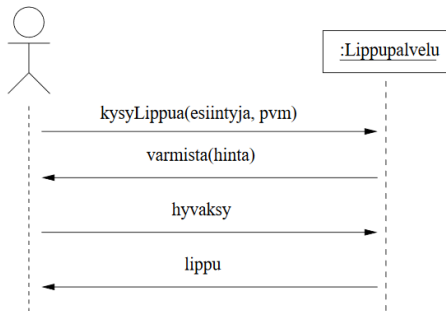
Olioiden yhteistyön mallintaminen

Käyttötapausten kulun kuvaaminen sekvenssikaaviona

- ▶ Tarkastellaan alkeellista lippupalvelun tietojärjestelmää ja sen käyttötapausta **Lipun varaus, tilanne missä lippuja löytyy**
- ▶ Käyttötapausten 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
- ▶ Käyttötapausten kulun voisi kuvata myös sekvenssikaavion avulla ajatellen koko järjestelmän yhtenä oliona
- ▶ tällöin kyseessä *Järjestelmätason sekvenssikaavio*

Olioiden yhteistyön mallintaminen

Käyttötapausten Lipun varaus kuvaava sekvenssikaavio

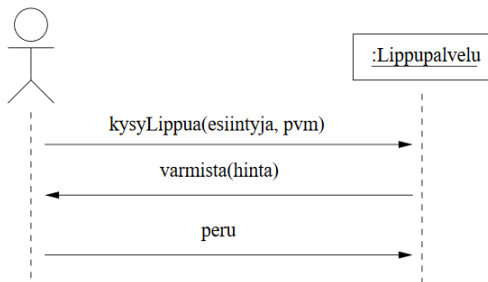


- Huom: Olioiden aktivaatiopalkit on jätetty kuvaamatta, sillä niille ei ole tarvetta esimerkissä

Olioiden yhteistyön mallintaminen

Vihtoehtoinen skenaario

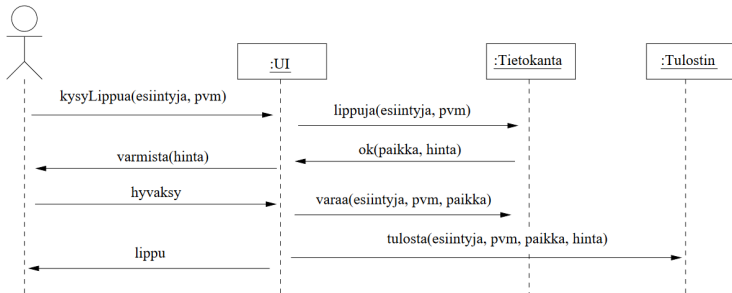
- ▶ Kuten kohta huomaamme, on myös yhteen sekvenssikaavioon mahdollista sisällyttää valinnaisuutta
- ▶ Toinen, usein selkeämpi vaihtoehto on kuvata vaihtoehtoiset skenaariot omina kaavioinaan
- ▶ Alla järjestelmätason sekvenssikaaviona tilanne, jossa asiakas hylkää tarjotun lipun



Olioiden yhteistyön mallintaminen

Järjestelmätasolta suunnittelutasolle

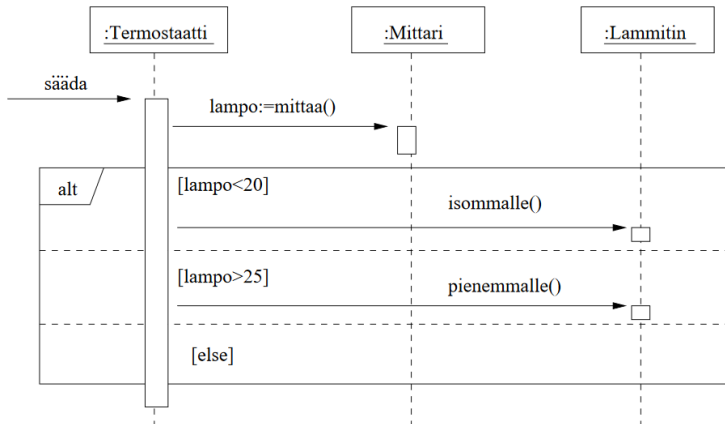
- ▶ Järjestelmätason sekvenssikaaviosta käy selkeästi ilmi käyttäjän ja järjestelmän interaktio
- ▶ Järjestelmän sisälle ei vielä katsota
- ▶ Seuraava askel on siirtyä suunnitteluun ja tarkentaa miten käyttötapauksen skenaario toteutetaan suunniteltujen olioiden yhteistyönä



Olioiden yhteistyön mallintaminen

Valinnaisuus

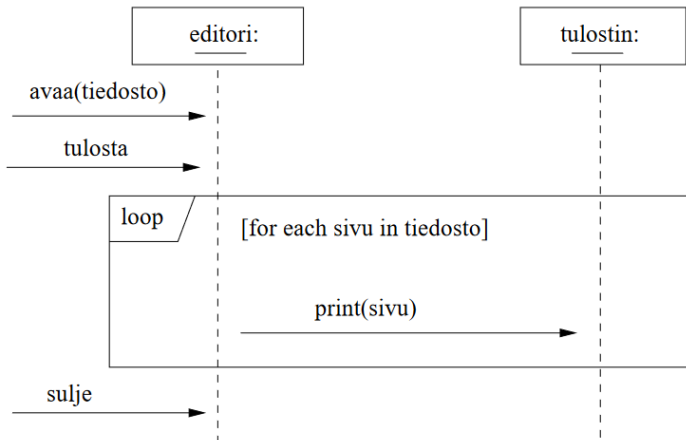
- ▶ Kaavioihin voidaan liittää lohko, jolla kuvataan valinnaisuutta
 - ▶ Vähän kuin if-else
 - ▶ Eli parametrina saadun arvon perusteella valitaan jokin kolmesta katkoviivan erottamasta alueesta



Olioiden yhteistyön mallintaminen

Toisto

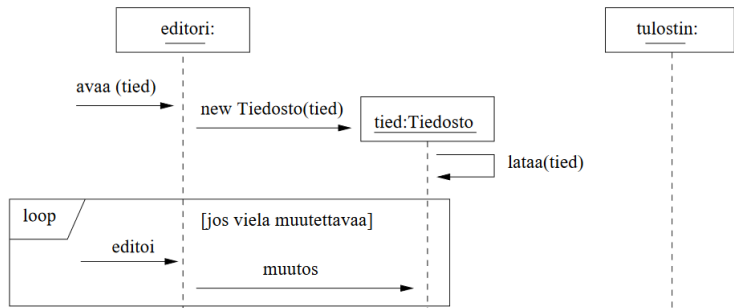
- ▶ Myös toistolohko mahdollinen (vrt. for tai while)
 - ▶ Huomaa miten toiston määrä on ilmaistu [ja] -merkkien sisällä
 - ▶ Voidaan käyttää myös vapaamuotoisempaa ilmausta, kuten "tulostetaan kaikki sivut erikseen"



Olioiden yhteistyön mallintaminen

Olioiden luominen

- ▶ Esimerkki luentomonisteesta ¹
- ▶ **Huom:** Uusi olio ei aloita ylhäältä vaan vasta siitä kohtaa milloin se luodaan

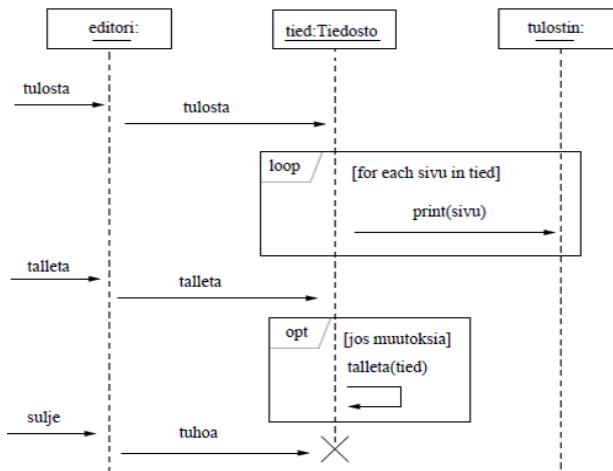


¹Katso koko esimerkki sivulta 65

Olioiden yhteistyön mallintaminen

Olioiden tuhoaminen

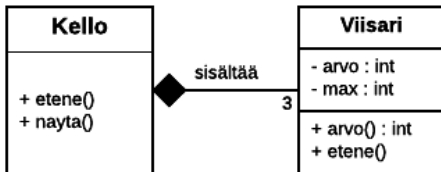
- ▶ Esimerkistä nähdään miten olion tuhoutuminen merkitään
- ▶ Mukana myös valinnainen (opt) lohko, joka suoritetaan jos ehto on tosi



Olioiden yhteistyön mallintaminen

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 →

Olioiden yhteistyön mallintaminen

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() );
    }
}
```

Olioiden yhteistyön mallintaminen

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; }  
}
```

Olioiden yhteistyön mallintaminen

Viisari

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

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

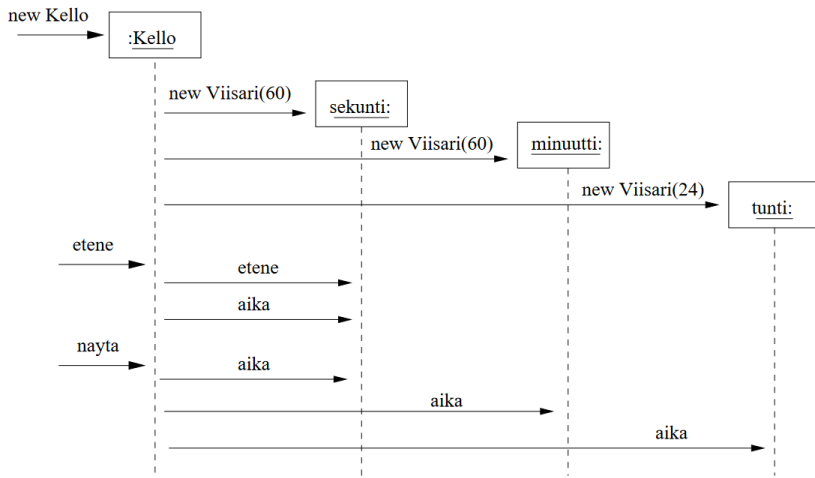
    public void etene(){
        arvo++;
        if ( arvo==max ) arvo = 0;
    }

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

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

Olioiden yhteistyön mallintaminen

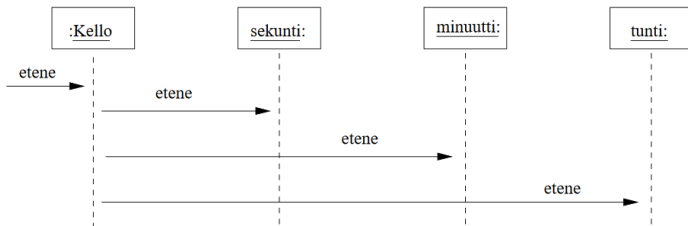
Kellon synty ja lähtee käymään



Olioiden yhteistyön mallintaminen

Kellon eteneminen tasatunnilla

- ▶ Tasatunnilla jokainen viisari etenee:



- ▶ Kaaviosta jätetty pois aika()-metodikutsut
- ▶ Eli jotta sekvenssikaavio ei kasvaisi liian suureksi, otetaan mukaan vain olennainen

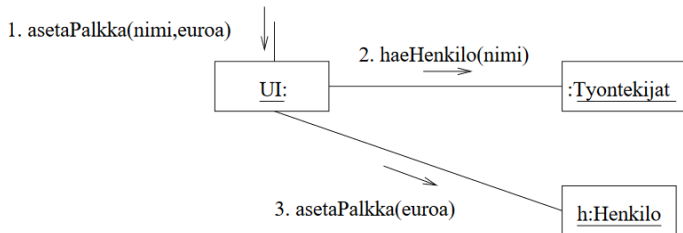
Kommunikaatiokaavio

Vaihtoehtoinen tapa kuvata yhteistoimintaa

Kommunikaatiokaavio

Vaihtoehtoinen tapa kuvata yhteistoimintaa

- ▶ Toinen tapa olioiden yhteistyön kuvaamiseen on kommunikaatiokaavio (engl. *communication diagram*)²
- ▶ Alla muutaman sivun takainen esimerkki, jossa henkilölle asetetaan palkka
- ▶ Metodien suoritusjärjestys ilmenee numeroinnista, olioiden sijoittelu on vapaa

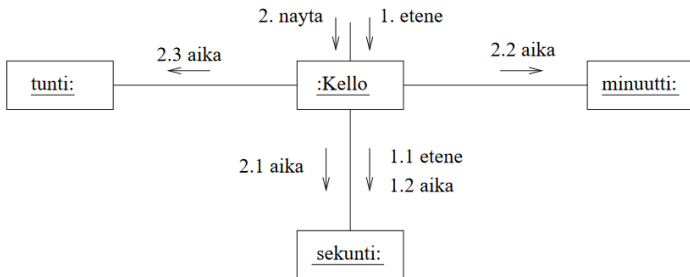


²Vanhalta nimitykseltä yhteistoimintakaavio (engl. *collaboration diagram*)

Kommunikaatiokaavio

Vaihtoehtoinen tapa kuvata yhteistoimintaa

- ▶ Viestien järjestyksen voi numeroida juoksevasti: 1, 2, 3, ...
- ▶ Tai allaolevan esimerkin³ tyyliin hierarkkisesti:
 - ▶ Kellolle kutsutaan metodia etene(), tällä numero 1
 - ▶ Eteneminen aiheuttaa sekuntiviisarille suoritettut metodikutsut etene() ja näytä(), nämä numeroitu 1.1 ja 1.2
 - ▶ Seuraavaksi kellolle kutsutaan metodia näytä(), numero 2
 - ▶ Sen aiheuttamat metodikutsut numeroitu 2.1, 2.2, 2.3, ...



³Kello ja viisarit esimerkki

Kommunikaatiokaavio

Yhteenveto olioiden yhteistoiminnan kuvaamisesta

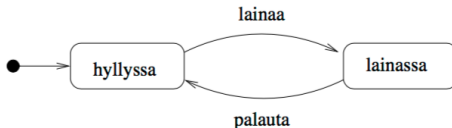
- ▶ Sekvenssikaavioita käytetään useammin kun kommunikaatiokaavioita
 - ▶ Sekvenssikavio lienee luokkakaavioiden jälkeen eniten käytetty UML-kaaviotyyppi
- ▶ Sekä sekvenssi- että kommunikaatiokaavioilla tärkeä asema oliosuunnittelussa
- ▶ Kaaviot kannattaa pitää melko pieninä ja niitä ei kannata tehdä 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 pakkauskaavioihin
- ▶ Kokonaan kurssin ulkopuolelle jää 5 eri kaaviotyyppiä

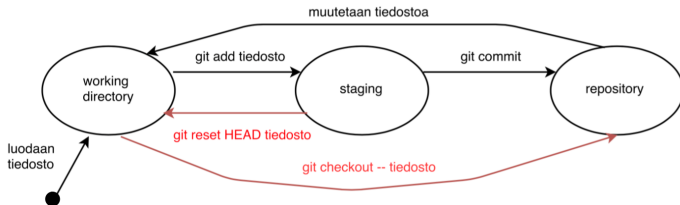
Tilakaavio

- ▶ Yksittäisten olioiden käyttäytymistapa voi olla erilainen eri tilanteissa
 - ▶ Viime luennon kirjastoiesimerkin 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 kaksi tilaa *hyllyssä* ja *lainassa* sekä *alkutila*
- ▶ 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 vastaavt siirtymät

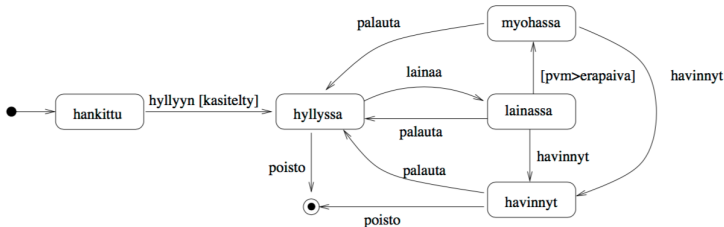


Tilakaavio

- ▶ Usein tila tarkoittaa tiettyjen olioimuuttujien 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

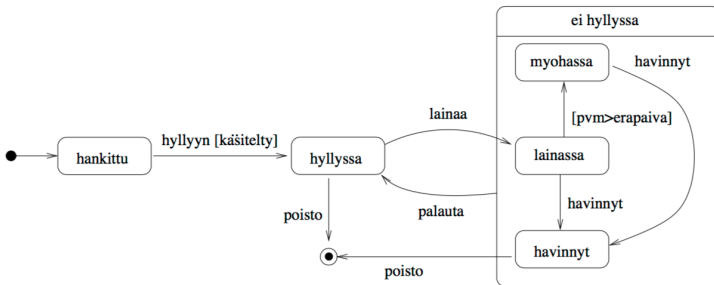
Tilakaavio

- ▶ Usein tila tarkoittaa tiettyjen olioimuuttujien 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
 - ▶ Muutamisiin siirtymiin liittyy nyt ehtoja
 - ▶ Kirjalla on myös olion tuhoutumista kuvaava lopputila



Tilakaavio

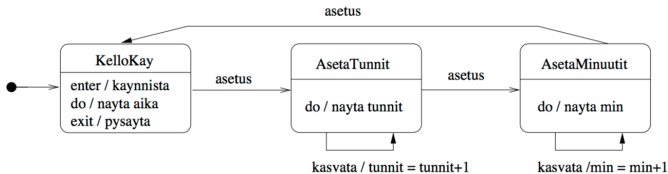
- ▶ Edellisen sivun kuvan tilat *myohassa*, *lainassa* ja *havinnyt* sisältävät kaikki siirtymän tilaan *hyllyssä* tapahtumalla *palauta*
- ▶ Tämäntyyppisissä tilanteissa kaaviota on mahdollista yksinkertaistaa sisäkkäisten tilojen avulla
- ▶ Alla kirjan tilamalli, jossa on *ylitila* (engl. superstate)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



Tilakaavio

- ▶ 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*
- ▶ Toiminnolla *asetus* siirrytään tilaan, jossa voidaan asettaa tuntiviisarin aika toiminnolla *kasvata*
- ▶ Tapahtuman aikaansaama toimenpide on merkitty muodossa *kasvata / tunnit = tunnit+1*
 - ▶ ensin on merkitty *tapahtuma* (metodikutsu *kasvata*), jonka perässä */*-merkin jälkeen tapahtuman aikaansaama *toimenpide* (muuttujan *tunnit* arvon kasvatus)
- ▶ Tilassa *AsetaTunnit* oltaessa näytetään tuntiviisarin aikaa.

Tilakaavio

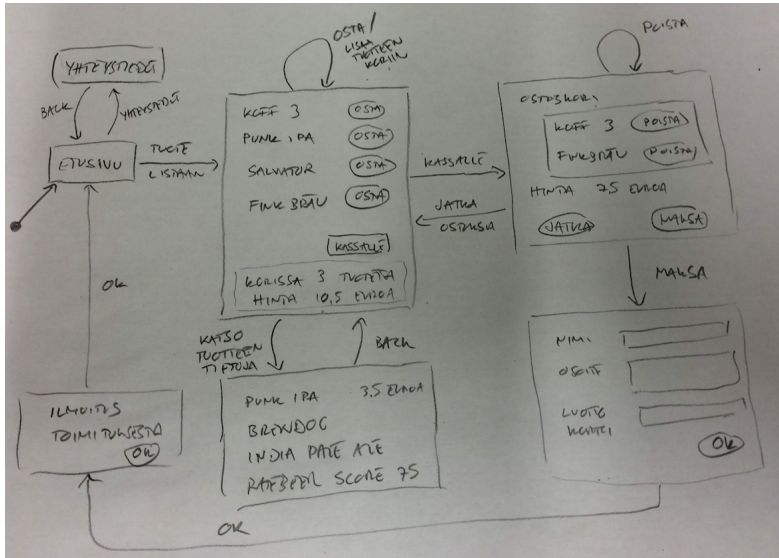


- ▶ Tilakaaviossa esiintyvät toiminnot voivat liittyä suoraan tilakaavion kuvaaman olion attribuutteihin, esim. $\text{tunnit} = \text{tunnit} + 1$
- ▶ Tilakaavion siirtymiin liittyvät tapahtumat kuten **kasvata** taas ovat yleensä olion metodeja
- ▶ Toisaalta, joskus tilakaaviolla kuvataan ainoastaan 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 navigaatorakenteen 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 navigaatorakenteen kuvaaminen tilakaaviona



Aktiviteettikaavio

- ▶ Tilakaaviot kuvaavat lähinnä yksittäisen olion toimintaa poikkeuksena käyttöliittymän navigaatorakenteen kuvaus
- ▶ *Aktiviteettikaavioilla* (engl. activity diagram) on mahdollisuus kuvata suurempaa toiminnallista kokonaisuutta, esimerkiksi:
 - ▶ Kokonaista liiketoimintaprosessia
 - ▶ Tiedon ja työn kulkua järjestelmässä monen toimijan kannalta
 - ▶ Käyttötapauksen etenemislogiikkaa

Aktiviteettikaavio

- ▶ Tilakaaviot kuvaavat lähinnä yksittäisen olion toimintaa poikkeuksena käyttöliittymän navigaatorakenteen kuvaus
- ▶ *Aktiviteettikaavioilla* (engl. activity diagram) on mahdollisuus kuvata suurempaa toiminnallista kokonaisuutta, esimerkiksi:
 - ▶ Kokonaista liiketoimintaprosessia
 - ▶ Tiedon ja työn kulkua järjestelmässä monen toimijan kannalta
 - ▶ Käyttötapausten 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 kuvaatilauksen vastaanottamiseen, toimittamiseen ja laskutukseen liittyviä toimintoja

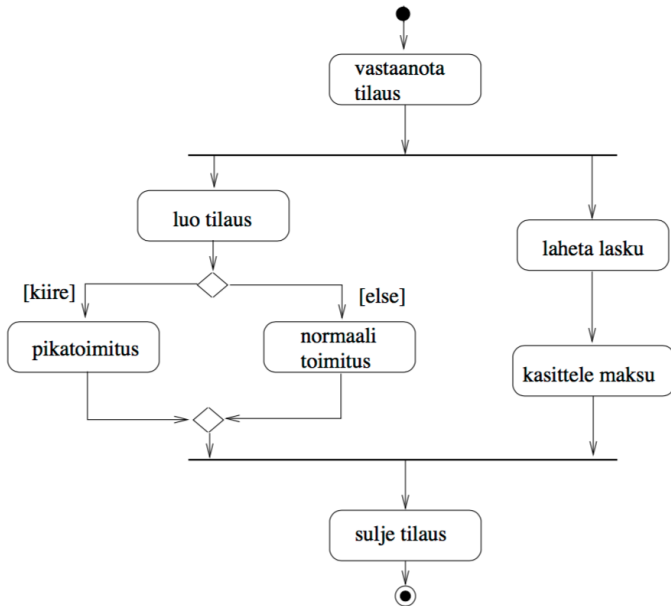
Aktiviteettikaavio

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



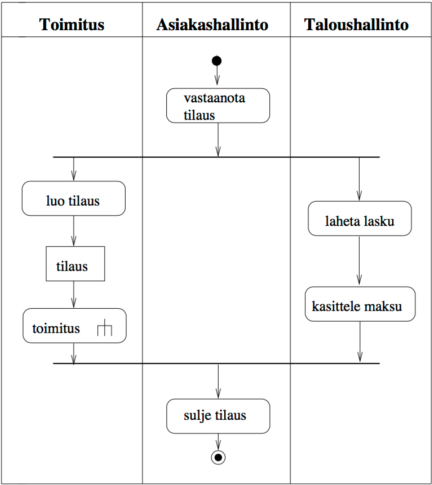
Aktiviteettikaavio

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 aktiviteettikaavioon

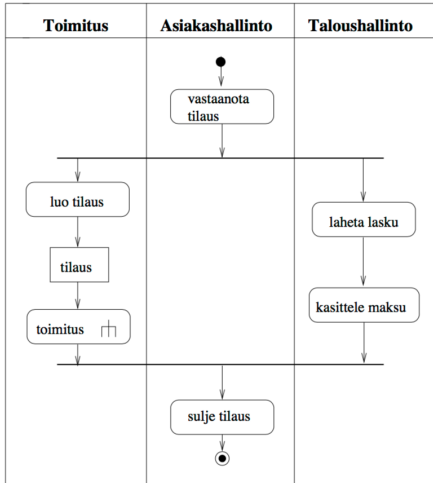
Aktiviteettikaavio

Kaistoihin jaettu aktiviteettikaavio



Aktiviteettikaavio

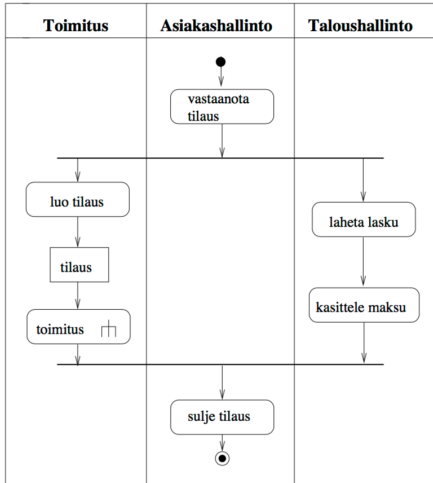
Kaistoihin jaettu aktiviteettikaavio



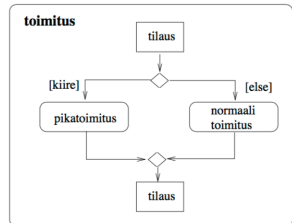
- ▶ Kaista ilmaisee kenen vastuulla toiminnon suoritus on
- ▶ Toimitus tarkentavassa kaaviossa

Aktiviteettikaavio

Kaistoihin jaettu aktiviteettikaavio



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



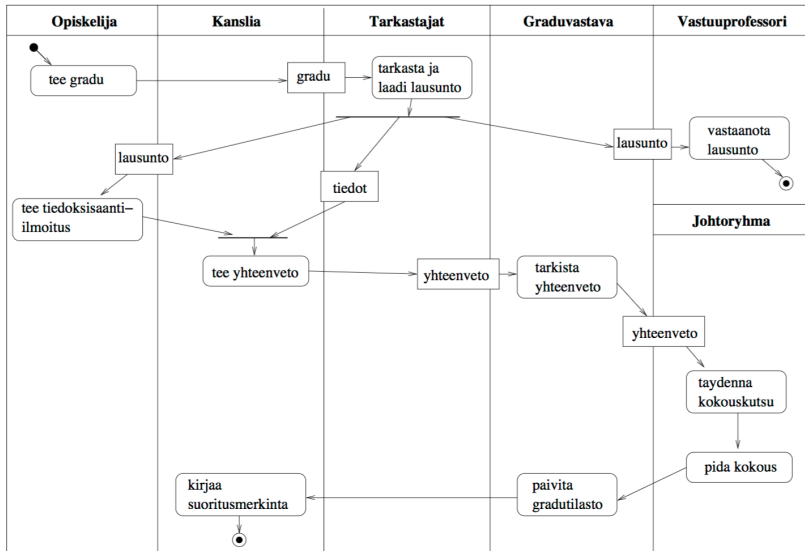
Aktiviteettikaavio

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 graduvaastaavalle
- ▶ Tarkastettuaan yhteenvedon, toimittaa graduvaastaava sen johtoryhmälle
- ▶ Johtoryhmä täydentää kokouskutsua gradun osalta ja pitää kokouksen
- ▶ Kokouksen jälkeen johtoryhmä pyytää graduvaastaavaa päivittämään laitoksen gradutilastoa
- ▶ Graduvastaava toimittaa tiedon johtoryhmän hyväksynnästä kansliaan, jossa kirjataan opiskelijalle suorituserkintä

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

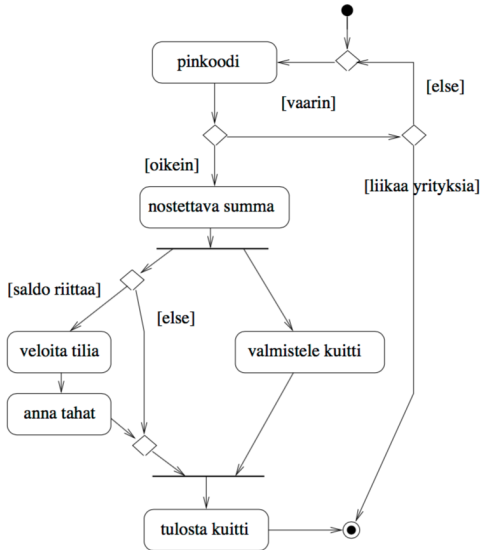
Aktiviteettikaavio

Käyttötapaus aktiviteettikaaviona

- ▶ Aktiviteettikaaviota voidaan hyödyntää myös yksittäisen käyttötapausten toimintalogiikan kuvaamiseen
- ▶ Esimerkiksi käyttötapaus **nostotapahtuma pankkiautomaatista** voitaisiin mallintaa seuraavan kalvon aktiviteettikaaviolla
- ▶ Esimerkin aktiviteettikaavio kuvaa ainoastaan yhden käyttötapausten 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

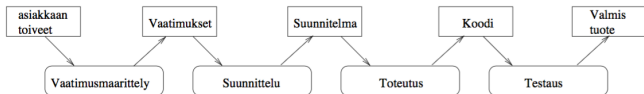
Aktiviteettikaavion käyttö ja hyödyt

- ▶ Aktiviteettikaaviot ovat tällä kurssilla esitellyistä vähiten ohjelmistokehityksessä käytetty kaaviotyyppi
- ▶ Samantapaisia kaavioita käytetään paljon muilla aloilla erilaisia toimintaprosesseja kuvailtaessa

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 aktiviteettikaaviot 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
- ▶ Yleisemminkin vaikuttaa siltä, että vuokaavioiden kulta-aika on ohi
- ▶ <http://en.wikipedia.org/wiki/Flowchart>