1. Olio-ohjelmointi

Sisällys

- Olio-ohjelmointi on eräs ohjelmointiparadigma.
- Ohjelmiston analyysi ja suunnittelu.
- Olioparadigman etuja ja kritiikkiä.

Ohjelmointiparadigmoja

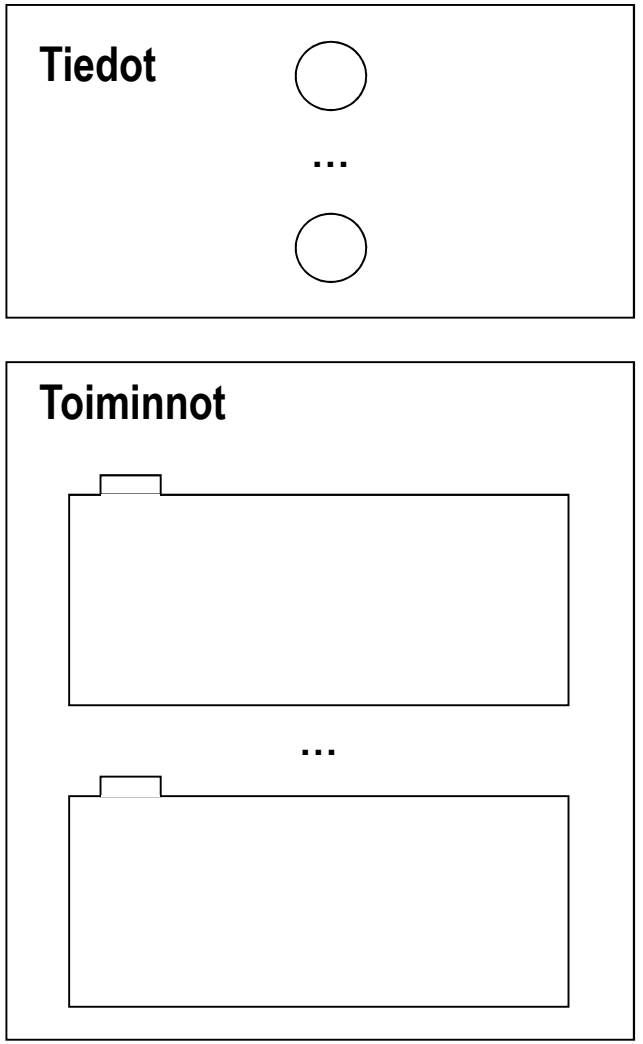
- Tietokoneohjelman toteuttamiseksi on tarjolla useita enemmän tai vähemmän toisistaan poikkeavia lähestymistapoja eli ohjelmointiparadigmoja.
- Paradigmoissa kiinnitetään vaihtelevasti huomiota tietoihin ja toimintoihin.
- Imperatiivinen paradigma on yleisin tapa ohjelmoida.
  - Ohjelmalla on tila, jota muutetaan vaiheittain käskyillä.
  - Proseduraalinen ja olio-ohjelmointi ovat imperatiivisen paradigman suuntauksia.
- Funktionaalinen ohjelmointi (esimerkiksi Lisp) ja logiikka-ohjelmointi (esimerkiksi Prolog) ovat muut pääparadigmat.

Proseduraalinen ohjelmointi

- Entinen valtaparadigma, jossa ohjelma jaetaan pienempiin osiin helposti ymmärrettäviksi ja hallittaviksi aliohjelmiksi.
- Rakenteinen ohjelmointi on eräs tämän paradigman muoto.
  - Kehitettiin ohjelmistokriisin ratkaisuksi 1970-luvulla.
  - GOTO-lause korvattiin ohjausrakenteilla (valinta ja toisto), jolloin päästiin eroon "spagettikoodista".
  - Ohjelman rakennetta selvennetään sisentämällä.
  - Tuloksena helpommin ymmärrettäviä ja ylläpidettäviä ohjelmia.
- Muun muassa Fortran, COBOL, Basic, Pascal, C, C++ ja Python.

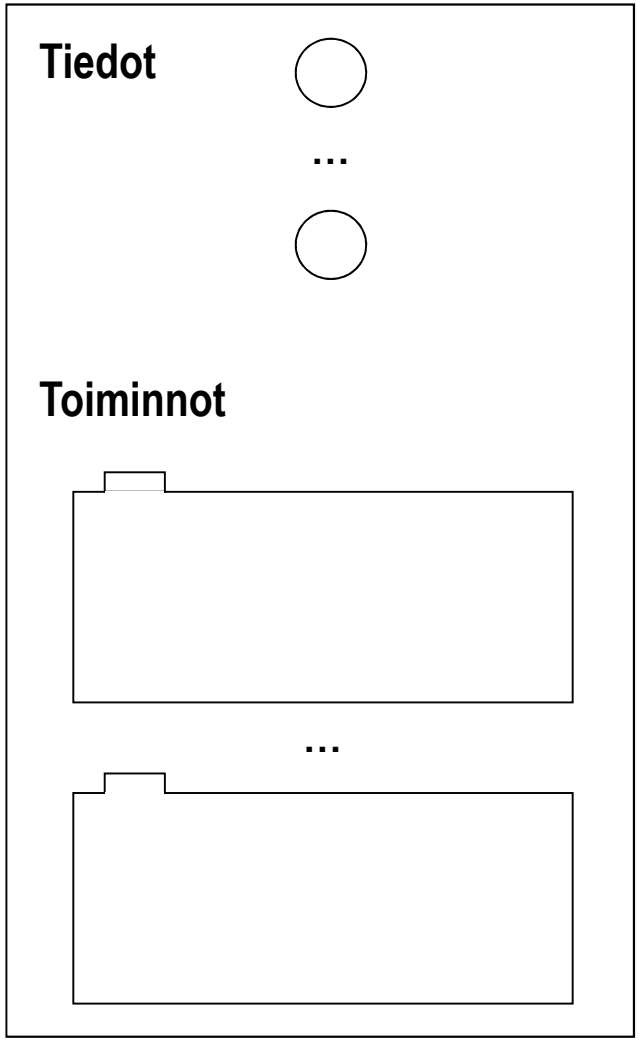
Proseduraalinen ohjelmointi

- Tiedot (globaalit muuttujat ja tietorakenteiden määrittelyt) ja niihin liittyvät toiminnot (aliohjelmat) ovat erilliset.
- Päätavoitteena on määritellä modulaarisuusperiaatteen mukaisesti mahdollisimman pitkälti toisistaan riippumattomia aliohjelmia, jotka kommunikoivat keskenään liittymiensä (paluuarvo ja paramerit) kautta.



Olio-ohjelmointi

- Nykyisin vallitseva ohjelmointi-paradigma, jossa ohjelman keskiössä ovat tiedot.
- Oliokielet tukevat olioparadigmaa vaihtelevasti.
- Muun muassa Java, C++, Python ja Smalltalk.
- Tiedot (attribuutit) ja niihin liittyvä toiminnallisuus (metodit) on yhdistetty luokiksi.



Ohjelmointiparadigmoja

- C++ on hybridikieli, jossa oliomekanismit liitetty perinteiseen C-ohjelmointikieleen.
  - C++:lla on helppo kirjoittaa huono olio-ohjelma.
  - Python on C++:n kaltainen siinä mielessä, että kieli ei ohjaa olio-ohjelmoinnin suuntaan, kun tavoitteena on ohjelmoida olioita.
- Monet ohjelmointikielet ovat itse asiassa moniparadigmakieliä, joissa kieli sisältää usean ohjelmointiparadigman käsitteitä.
- Javaankin sisältyy paljon rakenteisen ohjelmoinnin käsitteitä, vaikka Java luokitellaan lähes puhtaaksi olio-ohjelmointikieleksi.

Ohjelmiston analyysi ja suunnittelu

- Olioparadigma ei rajoitu pelkästään ohjelmointiin: Ohjelmistoja analysoidaan ja suunnitellaan olioperustaisesti (Object-Oriented Analysis/Design, OOA/D).
- Kurssilla käytetään UML:ää lähinnä OOA-tehtäviin.
- Luokkakaavioita tulee pian kalvoilla vastaan, mutta laajemmin UML opetetaan kurssin loppupuolella.
- Myöhemmillä kursseilla UML:ää enemmänkin sovelletaan kuin opetetaan.

Olioparadigman etuja

- Vastaa paremmin ihmisen tapaa hahmottaa maailmaa kuin esimerkiksi tiettyyn laskentamalliin perustuva ohjelmointiparadigma.
- Ohjelmisto on samanrakenteinen sovellusalueen käsitteiden kanssa, jolloin ylläpito on helpompaa ja ylläpitokustannukset pienempiä.
  - Ympäristössä tapahtuva muutos aiheuttaa olio-ohjelmaan karkeasti saman kokoisen muutoksen (jatkuvuus).
  - Rakenteisessa ohjelmoinnissa esimerkiksi yhden globaalin muuttujan poisto saattaa “rikkoa” koko ohjelman.
- Tukee ohjelmiston osien uudelleenkäyttöä.

Olioparadigman kritiikkiä

- Turhan raskas ja monimutkainen lähestymistapa pienten sovellusten toteuttamiseen.
- Olioparadigmaan kuuluva hierarkkinen käsitteiden välinen periytymismekanismi ei sovi jokaisen sovellusalueen kuvaamiseen.
- Olioperustaisuuden odotetut hyödyt jääneet osin toteutumatta.
- Olioperustaisuudesta voi olla enemmän haittaa kuin hyötyä ensimmäisellä ohjelmointikurssilla.