Ohjelmointi 1 / syksy 2007

8/20: Luokat, oliot ja APIt

Paavo Nieminen

nieminen@jyu.fi

Tietotekniikan laitos

Informaatioteknologian tiedekunta

Jyväskylän yliopisto

Kohti ymmärrystä ohjelmoinnista

- Hierarkkinen ja rekursiivinen ajattelumalli: esim. ohjelman suunnittelu, syntaksi, tiedon järjestäminen (tietorakenteet, tiedostojärjestelmä).
- Koodauskäytänteiden merkitys; esimerkkejä selkeistä ja epäselkeistä koodeista.
- Tarkennusta olioiden käyttämiseen: luominen, konstruktori, rajapinta.
- Tarkennusta aliohjelmiin: kuormittaminen, ”method signature”.
- Jälleen API-dokumentaation lukeminen: esim. StringBuilder/StringBuffer -APIt.

Ajattelu ja kommunikointi

- Hierarkkinen ja rekursiivinen ajattelumalli toteutuu monessa paikassa tietojenkäsittelyssä, esim:
  - ohjelman suunnittelu,
  - syntaksi
  - tiedon järjestäminen (tietorakenteet, tiedostojärjestelmä).
- Koodauskäytänteet ovat elinehto ohjelmoijan kommunikoinnille:
  - rakenteen sisällä oleva rakenne sisennettävä jämptisti!
  - kaikki nimet on annettava merkityksen mukaan!
  - Javassa luokat isolla ja muuttujat pienellä alkukirjaimella!
  - rivien pituus pieni; jakaminen selkeästi
  - vain selkeitä ohjelmointiratkaisuja — ei hämäriä!

$\longrightarrow$ esimerkkejä selkeistä ja epäselkeistä koodeista.

Kommunikaation taito haltuun!

- Omia ja toisten koodeja sekä ohjelmoinnin rakenteita on helpompaa opetella kun kaikki koodit KIRJOITTAA SELKEÄSTI alusta lähtien ja oppii myös lukemaan toisten koodia olettaen että niissä on sisennykset ym. kohdallaan.
- Toistaiseksi demoissa ollut vielä huolimattomuutta muotoseikoissa . . .
- Osaavalta ohjelmoijalta sitä ei suvaita kuin selkeissä vahinkotapauksissa!
- Tähän kannattaa keskittyä tuleva viikko, koska kyseessä ei ole vain esitysmuoto, vaan ymmärtäminen ja kommunikointi!

Metodin nimen kuormittaminen ”method signature”

- Yleensä kaikkien nimettyjen asioiden on oltava yksikäsitteisiä (ei samaa nimeä uudelleen).
- Metoditkin ovat, mutta yksikäsitteisyydessä otetaan nimen lisäksi huomioon parametrilista.
- Jos on saman niminen mutta erityyppiset parametrit vastaanottava metodi kuin joku toinen, sanotaan, että kyseinen nimi on kuormitettu (overloaded).
- Kuormitus on tullut jo vastaan, jos on lukenut APIsta esim. erilaisten konstruktorien otsikoita ja toimintaa.

Katsotaan tarkemmin, mikä on luokka

- Oliolla on sisäinen, piilotettu tila.
- Julkinen rajapinta metodeina (== aliohjelmina, jotka tuntevat jonkun olioyksilön eli instanssin sisäisen tilan).
- Luokka kuvailee, millaisia siihen luokkaan kuuluvat oliot ovat.
- Konstruktori on erityismerkityksessä oleva metodi.
- Konstruktoria kutsutaan olioyksilön luonnin (eli instantoinnin) yhteydessä.

$\rightarrow$ Ensimmäinen esimerkki (ja ehkä myös syksyn viimeinen) itse tehdystä luokasta, johon kuuluvan yksilön eli instanssin voi luoda.

Mistä tyvestä puuhun

- Äsken nähtiin itse tehty olioluokka, jonka tyyppisen olion voi luoda.
- Periaatteessa luokan tekeminen on helppoa, mutta siihen syvennytään vasta jatkokursseilla.
- Olioperustainen ohjelmointi ja suunnittelu soveltuu ISOJEN ohjelmien tekemiseen, ja siihen sisältyy rutkasti monimutkaisuutta ja vaaranpaikkoja.
- Alusta loppuun ohjelmointi edellyttää vahvaa ymmärrystä tietokoneen sekä ohjelmoinnin perusrakenteiden toiminnasta, joihin keskitymme aluksi.

StringBuilderin API, ero Stringiin

$\longrightarrow$ Esimerkkiohjelma: alleviivauksen tulostaminen.

$\longrightarrow$ Toteutetaan Stringeillä ja plus-operaattorilla; mikä on suoritusaika

$\longrightarrow$ Toteutetaan StringBuilderilla ja append-metodilla; mikä on suoritusaika

Havaittaneen dramaattinen ero, joka johtuu ”konepellin alla” tapahtuvista olioiden luonneista.