Ohjelmointi 1 / syksy 2007

7/20: Paketti kasassa ensimmäistä kertaa

Paavo Nieminen

nieminen@jyu.fi

Tietotekniikan laitos

Informaatioteknologian tiedekunta

Jyväskylän yliopisto

Tämän luennon rakenne

- Eilinen loppuun: moniulotteiset taulukot ja merkkijonot
- ”Paketti kasassa ensimmäistä kertaa”: nyt on nähty mistä ohjelmoinnissa on kyse.
  - Syntaksin käsite
  - Ohjelman rakentuminen syntaktisten rakenteiden hierarkiana
  - Suoritus tapahtuu lauseiden jatkumona
  - Ohjelman ”tarina”: Muuttujat, lauseet, kontrollirakenteet, aliohjelmat.
  - Olio-ohjelmoinnissa oliot ja viitteet
- Jatko on kertaamista ja syöksymistä syvemmälle kuhunkin aiheeseen

Taulukko 1/4: perusidea

- Määritelmä: Taulukko on indeksoitu, määrätynkokoinen, kokoelma tietyn tyyppisiä muuttujia (Javassa siis primitiivejä tai tietyn luokan olioiden viitteitä).
- Taulukko on tyypillinen tietorakenne ohjelmointikielissä, ja sen syntaksikin on usein lähes samanlainen; opettelemme asian Javassa, mikä helpottaa saman idean käyttöä missä tahansa kielessä myöhemmin.
- Vahvasti tyypitetyissä kielissä (joissa siis muuttujien tyyppien suhteen ollaan jatkuvasti ”niuhotarkkoja”) myös taulukon tyyppi on määriteltävä tarkasti eikä se voi muuttua esittelynsä jälkeen.

Taulukko 2/4: syntaksi javassa

- Taulukon eli indeksoidun, määrätynkokoisen, kokoelman tietyn tyyppisistä muuttujista voi Javassa esitellä seuraavalla syntaksilla:

```
Tyyppi[] taulukonNimi;
```

- Kuten primitiiveille, taulukolle voi antaa esittelyn yhteydessä sisällön seuraavalla syntaksilla:

```
Tyyppi[] taulukonNimi =
   {arvo1, arvo2, ..., arvoN};
```

(Ero primitiivin alustukseen: arvoja on monta ja ne ovat kiharasulkujen sisällä)

- Alkioiden käyttö tapahtuu hakasulkusyntaksilla:

```
esimMuuttuja = esimTaulu[indeksi];
```

- Jos taulukkoa ei alusteta, se luodaan new:llä:

```
double[] neljanLuvunTaulu = new double[4];
```

Taulukko 3/4: Javassa taulukko on olio

- Javassa on huomattava, että taulukko on olio.
- Se käyttäytyy kaikistellen, kuten oliot aina.
- Taulukon tyyppi on siis olioluokka (vaikka sen käyttöön on erityinen syntaksi).
- Syntaksissa (ed. kalvo) `Tyyppi` on mikä tahansa tyyppi (primitiivi tai olioviite) ja taulukosta tulee viite indeksoituun, määrätynkokoiseen, kokoelmaan juuri sen tyyppisiä alkioita kuin mikä `Tyyppi` on.

Taulukko 4/4: useampia ulottuvuuksia

Alustava havainto moniulotteisista taulukoista:

- Taulukko voi sisältää taulukoita, esim. voidaan ajatella rivivektoria taulukkona koordinaateista, ja matriisia taulukkona rivivektoreista. Syntaksi kaksiulotteiselle taulukolle:

```
Tyyppi[][] taulukonNimi;
```

- Toinen esimerkki, jossa myös alustus:

```
double[][][] isoTaulu =
   { { {1.0, 2.0},
       {3.0, 4.0} },
     { {5.0, 6.0},
       {7.0, 8.0} } };
```

- Dimensioiden määrää rajoittaa vain se, mikä on järkevää.

Merkkijonot 1/4: Perusidea

- Yksi merkki on yksi merkki (esim. ’A’ tai ’b’ tai ’\’).
- Teksti koostuu useista merkeistä (esim. ”Arton ABC-kirja”).
- Sanotaan merkkien jonoa merkkijonoksi.
- Merkkijono vaatii tietokoneessa muistia vähintään niin paljon kuin sen merkit yhteensä.
- Merkkijonoja on kätevää käsitellä kuin ne olisivat kokonaisuuksia.
- Merkkijonoja käytetään todella moneen asiaan, mm. konsolisyöttöön, tulostukseen ja tiedon tallentamiseen.

Merkkijonot 2/4: Javassa merkkijono on olio

- Javassa on huomattava, että merkkijono on olio.
- Se käyttäytyy kaikistellen, kuten oliot aina. Erityispiirteinä mahdollisuus käsitellä merkkejä yksittäin ja osajonoina.
- Merkkijonon tyyppi on siis olioluokka, String (tai StringBuilder tai StringBuffer; ero ilmenee myöh. kalvolla).

Merkkijonot 3/4: esimerkkinä argumenttilista

Argumenttilista on taulukko, joka sisältä merkkijonoviitteitä.
$\rightarrow$ Havaitaan tarkasti, mikä on String[] args, ja miten sitä käytetään: ohjelma, joka tulostaa argumenttinsa ja tutkii niistä ominaisuuksia kuten merkkijonon pituus, ensimmäinen ja viimeinen merkki ym. pikkuhassua. Samalla String-luokan APIa tutummaksi.

$\rightarrow$ samalla kertautuu olioyksilön ja -viitteen ero: Parametrina mainiin tuleva muuttuja `args` on viite taulukko-olioon, josta on viitteitä merkkijono-olioihin.

Merkkijonot 4/4: Muuttumaton ja muuttuvainen olio

Merkkijonot ovat ensimmäinen kohtaaminen muuttuvaisten (mutable) ja muuttumattomien (immutable) olioiden kanssa:

- Esim. merkkijonoliteraalit ilmenevät suorituksen aikana String-luokan olioina. Niiden sisäinen tila ei koskaan voi muuttua olion elon aikana! Niiden luokan rajapinta ei mahdollista tätä.
- String-luokan merkkijono-operaatioissa syntyy aina uusi olio, jos tulos on erilainen kuin aiempi/aiemmat String-merkkijonot. Esim. yhteenliittäminen plussalla.
- Muuttuvainen merkkijono on tehtävä erikseen StringBuilder -luokan oliona (tai StringBuffer).

esimerkki olioiden ja sisältöjen vertailusta

Merkkijonot ovat myös ensimmäinen kohtaaminen sellaisten olioiden kanssa, joiden sisältö voi olla sama, vaikka olioyksilöt ovat eri.

- Samuuden vertailu metodilla equals()
- Merkkien mukainen mukainen vertailu compareTo()
- HUOM: Yhtäsuuruusoperaattori ”==” vertaa olioviitteitä (viittaavatko samaan yksilöön) eikä osaa kertoa mitään olioista, jotka ovat ”identtisiä kaksosia” mutta eri yksilöitä. Esim merkkijonot ”Marja” ja ”Marja”.

Java-kielen muuttujat ja tietoraken-teet

	Tyyppi	koko	literaaliesim.
Totuusarvot	boolean	?	true, false
Merkit	char	16	’A’, ’\u0041’
Kokonaisluvut	byte	8	14, 123, -128, 0x7A
	short	16	-14, 0123, 32767
	int	32	14, -1234, 32767
	long	64	14L, 92233720368547758O7
Liukuluvut	float	32	14.0, 1.234, 3.4e38
	double	64	14.0, 1.234, 4.9e-324
Olioviitteet	LuokanNimi		

Javassa erikoisasemassa olevat olioluokat: taulukot ja String, joille on omaa syntaksia itse kielessä. Esim.:

	Tyyppi	alustusliteraali
Taulukko	int[]	{1, 2, -7}
Merkkijono	String	"Heippa"

Siinäkö oli kaikki tietorakenteet?

- Primitiivityypit, merkkijonot ja taulukot on toteutettu useimmissa ohjelmointikielissä valmiina (eli kieli tarjoaa keinot eikä tarvitse itse ohjelmoida tietorakenteen toteutusta). Tulemme näkemään, millä tavoin tämä on tehty Javassa.
- Usein käytetään monipuolisempia tietorakenteita (esim. pinot, jonot, listat, puut, hakemistot, verkot, . . .) mutta ne pitää yleensä toteuttaa käyttäen yksinkertaisempia, ohjelmointikielen tarjoamia rakenteita. Miten? Luomalla sisäinen toteutus ohjelmointikielen keinoin ja tarjoamalla aliohjelmien/metodien kautta käytettävä rajapinta.

Luetaan APIa!

Mieluummin ennemmin kuin myöhemmin on opittava lukemaan APIa jos toistakin.

- API = ”Application Programming Interface”
- Suomeksi ”sovellusohjelmointirajapinta” tai ihan vaan ”rajapinta”
- Javassa jonkun luokkakokoelman tarjoamat julkiset metodit ja attribuutit.

Mitäpä APIsta löytyy

$\longrightarrow$ Katsotaan esimerkiksi String, StringBuilder ja StringBuffer -APIen dokumentaatiota.

Pari uutta havaintoa: Konstruktorit, aliohjelmien kuormittaminen.

$\longrightarrow$ Katsotaan vertailun vuoksi satunnaisesti valitun Java-luokan APIa.

Paketti kasassa ensimmäistä kertaa

”Paketti kasassa ensimmäistä kertaa”: nyt on nähty mistä ohjelmoinnissa on kyse.

- Syntaksin käsite
- Ohjelman rakentuminen syntaktisten rakenteiden hierarkiana
- Suoritus tapahtuu lauseiden jatkumona
- Ohjelman ”tarina”: Muuttujat, lauseet, kontrollirakenteet, aliohjelmat.
- Olio-ohjelmoinnissa oliot ja viitteet

Jatko on kertaamista ja syöksymistä syvemmälle kuhunkin aiheeseen