Tartu Ülikool

Arvutiteaduse Instituut

J. Kiho

VÄIKE JAVA LEKSIKON

Tartu 1997

Saateks

Käesolevasse leksikoni on koondatud programmeerimiskeele Java põhimõisted. Süsteemsetest klassidest on puudutatud ainult üksikuid, eeskätt paketis `java.lang` kirjeldatud vahendeid. Põgusamalt ja vaid osaliselt käsitletakse keelest C pärinevaid mõisteid.

Teatavaks aluseks, eriti terminite valimise ja neile eestikeelsete vastete leidmise osas oli *Tarkvarasüsteemide seminari* töö 1996/97 õppeaasta kevadsemestril (`http://www.cs.ut.ee/~kiho/softsys/` fail:`leksikon.ps`). Selles käsitletud terminid leiduvad lisatud inglise-eesti sõnastikus; leksikonis kajastub üksnes osa neist.

Kuna eestikeelne terminoloogia on veel välja kujunemata, siis on rakendatud paindlikku koostamissüsteemi, mille kohaselt TEX-põhitekstis eestikeelsed terminid esituvad (inglisekeelsete) makrodena. Mingi termini asendamisel uuega tuleb lihtsalt teha muudatus makrode tabelis ja kompileerida uuesti põhitekst.

Vormistuslikud vahendid on kokkuvõtlikult esitatud järgmises "artiklis":

märksõna, *inglisekeelne vaste* – antud märksõnale (mis võib olla ka sõnaühend) vastavat mõistet selgitav artikkel; *kaldkirjas* on esitatud siin muuhulgas määratletud mingi teine mõiste (kui viimase kohta ongi omaette artikkel, siis sisaldab see vaid viidet antud märksõnale); kujul →mingi teine märksõna viidatakse käesoleva artikliga lähedalt seotud mingile teisele märksõnale; Java keelekonstruktsioonide konkreetsed elemendid esitatakse `kirjutusmasinakirjas`, meta-elemendid aga *kursiivis*, mittekohustuslikud elemendid võetakse nurksulgudesse, meta-termin kujul *ühikute loetelu* tähendab komadega eraldatud ühikuid, mitmuse vormis meta-termin kujul *ühikud* aga tühikutega eraldatud ühikuid; vajadusel esitatakse lõpus märksõnu, mille artikleid on soovitatav lisaks vaadata, ning antud märksõna sünonüüme. Vt. ka *täiendavad märksõnad*. Sünon. *sünonüümid*.

Leksikoni elektronvariant asub aadressil:
`http://www.cs.ut.ee/~kiho/softsys/` (fail: `J_leksikon.ps`)

J. Kiho
`kiho@cs.ut.ee`

Kasutatud materjale

1. J.Gosling, B.Joy, G.Steele. The Java Language Specification.
   Addison-Wesley, 1996
   `http://java.sun.com/docs/books/jls/html/index.html`
2. K.Arnold, J.Gosling. The Java Programming Language.
   Addison-Wesley, 1996 (Appendix D)
   `http://java.sun.com/docs/books/jls/html/1.1Update.html`
3. P. Niemeyer, J.Peck. Exploring Java. O'Reilly, 1996
4. J.Lewis, W.Loftus. Java Software Solutions. Addison-Wesley, 1997

abstraktne klass, *abstract class* – klass, mis kas sisaldab →abstraktse meetodi kirjeldust või saab selle pärimise teel ülemklassist (või liideselt), kuid jätab selle realiseerimata; peab olema piiritlejaga `abstract`; mõeldud kasutamiseks ülemklassina (nt. klassidele, mis realiseerivad iga tema abstraktse meetodi).

abstraktne meetod, *abstract method* – ilma kehata (keha asemel on semikoolon) ehk realiseerimata →meetod; klassikirjelduses peab olema piiritlejaga `abstract`, ei saa olla liiki `private`, `final` ega `static`. Vt. ka *ülekate*.

algataja, *initializer* – klassikirjelduse element kujul

`static` *blokk*

või kujul

blokk

esimene neist kirjeldab klassi laadimisel sooritatavaid lisategevusi, olles mõeldud eeskätt klassiväljade algatamiseks; teine neist kirjeldab isendiloomel (enne klassi konstruktori keha täitmist) sooritatavaid lisategevusi, olles mõeldud eeskätt isendiväljade algatamiseks; ei tohi sisaldada naasmisdirektiivi.

algtüüp, *primitive type* – →tüüp.

avaldisdirektiiv, *expression-statement* – direktiiv kujul

avaldis `;`

sooritatakse avaldises näidatud tehted, kuid avaldise väärtust edaspidi kasutada pole võimalik; lubatud vormid:

muutuja`++;` *muutuja*`--;` `++`*muutuja*`;` `--`*muutuja*`;` // ühikmuutmised
muutuja omistamisoperaator avaldis`;` // omistamine
meetodi või konstruktori nimi `(`[*argumentide loetelu*]`);` // väljakutse.

blokk, *block* – algoritmi esitus kujul

{ [*laused*] }

kus lauseks on kas →lokaalmuutuja kirjeldus või →direktiiv või →lokaalklass. Sünon. *plokk*.

direktiiv, *statement* – otseselt algoritmilisi tegevusi kirjeldav lause; erijuhul omakorda blokk või tühidirektiiv (ainult semikoolon); ülejäänud direktiiviliikideks on →avaldisdirektiiv, →lülitidirektiiv, →tingimusdirektiiv, →eelkontrolliga tsüklidirektiiv, →järelkontrolliga tsüklidirektiiv, →üldtsüklidirektiiv, →katkestusdirektiiv, →jätkamisdirektiiv, →naasmisdirektiiv, →erindiseade direktiiv, →sünkroondirektiiv, →katsendidirektiiv; iga direktiivi ees on null või enam *märgendit* kujul *identifikaator* : .

eelkontrolliga tsüklidirektiiv, *while-statement* – direktiiv kujul

`while` (*loogiline avaldis*) *sisu*

kus sisuks on →direktiiv; analoogne keele C vastava konstruktsiooniga.

epiloog, *finally-clause* – →katsendidirektiiv.

erind, *exception* – programmi töö käigus tekkida võiv eriolukord, mis ei pruugi tingimata olla saatuslik programmi edasisele täitmisele; tüüpilistele erinditele vastavad süsteemsed erindiklassid (paketis `java.lang`); erindi tekkimisel luuakse tavaliselt vastava klassi isend, mis on abiks eriolukorda analüüsivas programmiosas.

erindiklass, *exception class* – klass, mille (vahetuks või kaugemaks) ülemklassiks on süsteemklass `Exception` paketist `java.lang`; viimases leidub rohkesti mitmesugustele erinditele vastavaid süsteemseid erindiklasse; lisaks nendele saab moodustada ka muid vajalikke alamklasse, nt.

```
class nimi extends Exception {
   public nimi( ) { super( ); }
   public nimi(String s) { super(s); }
}
```

erindiseade direktiiv, *throw-statement* – direktiiv kujul

`throw` *avaldis*;

kus (tavaliselt →isendiloome) avaldise väärtuseks on osuti →erindiklassi isendile; täitmisel toimub vastav →erindiseade.

erindiseade, *throw* – mingi erindiklassi isendi loomine, millega kaasneb hõlmava bloki järsk sooritatus; toimub automaatselt teatavate tüüpiliste →erindite korral; võimalik ka programselt esile kutsuda →erindiseade direktiivi kasutades.

ilmutatud teisendus, *explicit conversion* – →teisendus.

isend, *instance* – programmi töö ajal eksisteeriv klasstüüpi objekt. Vt. ka *isendiloome*. Sünon. *eksemplar*.

isendiloome, *instantiation* – klasstüüpi objekti (→isendi) loomine; toimub avaldise

`new` *klassinimi* `(` *argumentide loetelu konstruktorile* `)`

arvutamisel, mille käigus, arvestades nii klassi liikmete kui ka päritud liikmete kirjeldusi, hõivatakse objektile vajalik mälu ja rakendatakse argumentide poolest sobivat →konstruktorit; avaldise väärtuseks on osuti loodud objektile; mälupuuduse korral tekib erind `OutOfMemoryError`. Vt. ka *vaikekonstruktor, algataja.*

isendimeetod, *instance method* – →meetod.

isendiväli, *instance field* – →väli. Sünon. *isendimuutuja.*

juurdepääsetavus, *accessibility* – kirjeldusosise kasutatavuse e. nähtavuse skoop; täpsustatakse piiritlejate `public private protected` abil; klassi ja liidese nähtavust laiendab piiritleja `public`: vaikimisi on need osised nähtavad ainult oma paketis, `public` korral aga kõikjal; ka liikme vaikenähtavus piirdub paketiga, kuhu kuulub seda liiget sisaldav klassi- või liidesekirjeldus, piiritlejaga `public` liige on aga nähtav kõikjal, kus on nähtav teda kirjeldav klass või liides; klassi liikme (mis ei ole `public`) kasutatavust saab täpsustada võtmesõnadega `private` ja `protected`:

`private` liige on nähtav ainult teda kirjeldavas klassis (ega kuulu ka pärimisele);

`protected` liige on nähtav nii oma paketis kui ka teda kirjeldava klassi alamklassides (mis võivad kuuluda ka teistesse pakettidesse);

`private protected` liige on nähtav teda kirjeldavas klassis ja selle alamklassides (mis võivad kuuluda ka teistesse pakettidesse);

piiritleja `protected` lubab pärimist, keelab aga isendi vastava välja kasutamise alamklassis: `protected` ja `private protected` liikmed lülitatakse pärimisel küll alamklasside kirjeldustesse, kuid antud klassi isendi `protected` ja `private protected` väljad ei ole alamklassides nähtavad; nimetatud kolme piiritleja abil reguleeritakse ka →konstruktorite nähtavust, seega avardades või kitsendades isendiloome võimalusi.

järelkontrolliga tsüklidirektiiv, *do-statement* – direktiiv kujul

`do` *sisu* `while (`*loogiline avaldis*`);`

kus sisuks on →direktiiv; analoogne keele C vastava konstruktsiooniga.

järsk sooritatus, *abrupt completion* – →sooritatus.

jätkamisdirektiiv, *continue-statement* – direktiiv kujul

`continue` [*märgend*]`;`

peab olema hõlmatud tsüklidirektiiviga; lõpetab vastava märgendatud tsükli (märgendi puudumisel – lähima hõlmava tsükli) sisu järjekordse täitmise.

katkestusdirektiiv, *break-statement* – direktiiv kujul

`break` [*märgend*]`;`

lõpetab vastavalt märgendatud hõlmava direktiivi (märgendi puudumisel – lähima hõlmava tsükli- või lülitidirektiivi) täitmise; märgendita katkestusdirektiiv peab olema hõlmatud tsüklidirektiiviga või lülitidirektiiviga, märgendiga katkestusdirektiiv aga vastavalt märgendatud direktiiviga.

katsendidirektiiv, *try-statement* – direktiiv kujul

`try` *blokk* [*püünised*] [*epiloog*]

kus *püüniseks* on lause

`catch` (*formaalne parameeter*) *blokk*

ja *epiloogiks*

`finally` *blokk*

katsendidirektiivi täitmist alustatakse võtmesõna `try` järel olevast blokist (*põhiblokist*); selle täitmise käigus võib toimuda suunamine mõnesse püünisesse: kui põhibloki täitmine lõpeb erindiseadega, siis (võimaluse korral) antakse juhtimine veel (esimese) sellise püünise blokki, mille formaalse parameetri tüübiks on seatud erindi osutitüüp, seejuures parameetrile omistatakse osuti seatud erindile; kui epiloog puudub, siis katsendidirektiivi täitmine lõpeb kas põhiblokist väljumisega (kui suunamist püünisesse ei toimunud) või püüniseblokist väljumisega; kui epiloog on antud, siis pärast põhibloki (kui suunamist püünisesse ei toimunud) või püünisebloki täitmist läheb täitmisjärg epiloogi blokile, millest väljumisega lõpeb ka katsendidirektiivi täitmine; epiloog sooritatakse alati, sõltumata põhi- või püüniseblokist väljumise põhjusest. Sünon. *katsend.*

kitsendav osutiteisendus, *narrowing reference conversion* – →teisendused (vasakul lähtetüüp, paremal sihttüüp; K, L ja M tähistavad vastavalt klassi, liidese ja massiivi osutitüüpi):

$K \Rightarrow K'$, kus K on K' ülemklass;

$K \Rightarrow L$, kus K ei ole liiki `final` ja ei L ei ole K liideseks;

`Object` $\Rightarrow L$, M;

$L \Rightarrow K$, kus K ei ole liiki `final`;

$L \Rightarrow K$, kus K on `final` ja L on K liideseks;

$L \Rightarrow L'$, kus L' ei ole L ülemliides ning L ja L' ei sisalda kokkulangeva signatuuriga, kuid erineva tagastustüübiga meetodit;

$M \Rightarrow M'$, kus M elemenditüüp on teisendatav M' elemenditüübiks mingi kitsendava teisendusega;

on peamiselt informatsiooniks kompilaatorile, kuid programmi täitmise ajal siiski kontrollitakse sihttüübi nõuetekohasust.

kitsendav primitiiviteisendus, *narrowing primitive conversion* – →teisendused

`double` ⇒ `byte`, `short`, `char`, `int`, `long`, `float` ;

`float` ⇒ `byte`, `short`, `char`, `int`, `long`;

`long` ⇒ `byte`, `short`, `char`, `int`;

`int` ⇒ `byte`, `short`, `char`;

`char` ⇒ `byte`, `short`;

`short` ⇒ `byte`, `char`;

`byte` ⇒ `char`;

informatsioon ei pruugi säilida; mitte-ujupunktarvu teisendamisel lõigatakse vajadusel vasakpoolsed bitid, ujupunktarvu teisendamisel mitte-ujupunktarvuks esitatakse see eelnevalt (ümardatult, võimalikult lähedase) täisarvuna.

klass, *class* – klasstüübi kirjeldus, mis määrab klassi nime ja (vahetu) ülemklassi ning sellesse tüüpi kuuluvate objektide struktuuri, funktsionaalsuse ja muud omadused; erandina puudub ülemklass süsteemklassil nimega `Object`, viimane on ka (vahetuks või kaugemaks) ülemklassiks kõigile teistele klassidele; võib deklareerida ka klassis realiseeritavate liideste loetelu; programmeerimine keeles Java seisneb klasside koostamises. Vt. ka *klassikirjeldus*. Sünon. *klassikirjeldus*.

klassikirjeldus, *class declaration* – klassi spetsifikatsioon kujul

[*klassi piiritlejad*] `class` *klassinimi*

[`extends` *ülemklassinimi*] [`implements` *liidesed*]

klassi keha

klassi piiritlejatena saab kasutada võtmesõnu `public final abstract` (tühikutega eraldatult); vaikimisi on klassi nime skoobiks pakett, kuhu see klassikirjeldus kuulub, piiritleja `public` korral aga saab sellele klassile viidata ka väljastpoolt paketti; piiritlejaga `final` keelatakse selle klassi kasutamine ülemklassina; piiritleja `abstract` viitab sellele, et antud klassi keha on (võib olla) vaid osaliselt kirjeldatud ja seega isendiloome välistatud ning klass on mõeldud kasutamiseks eeskätt ülemklassina; kirjeldatavale klassile ülemklassiks valitud klassi nimi näidatakse võtmesõna `extends` järel (vaikimisi on ülemklassiks klass nimega `Object`); võtmesõna `implements` järel loetletakse (komaga eraldatult) nende →liideste nimed, mida antud klass realiseerib; iga näidatud liidese iga, sh. ka päritud (abstraktse) meetodi jaoks peab antud klassi kehas leiduma →ülekate vastava mitteabstraktse meetodiga; klassi keha koosneb looksulgudesse võetud null või enamast liikme-, konstruktori- ja/või algatajakirjeldusest. Vt. ka *konstruktor, liige, algataja*. Sünon. *klass*.

klassimeetod, *class method* – →meetod. Sünon. *staatiline meetod*.

klassiväli, *class field* – →väli. Sünon. *staatiline väli, klassimuutuja*.

klasstüüp, *class type* – abstraktne andmetüüp: ühtede ja samade omadustega, st. ühetaolise struktuuri ja funktsionaalsusega objektide hulk, mis on määratud mingi →klassikirjeldusega; viimase nimi on ka selle klasstüübi nimeks; lisaks klassikirjelduses otseselt näidatud omadustele on klasstüüpi objektil veel vastava klassi ülemklassilt →pärimise teel saadud omadused. Vt. ka *tüüp*.

kommentaar, *comment* – kompileerimisel ignoreeritav (1) programmi teksti osa, mis algab järjekordse (sõnevälise) sümbolipaariga `/*` ja lõpeb sellele esimesena järgneva (sõnevälise) sümbolipaariga `*/` või (2) programmi rea lõpuosa alates reas leiduvast esimesest (sõne- ja kommentaarivälisest) sümbolipaarist `//` ; kommentaarid kujul `/** ... */` on sisendiks automaatsele dokumentaatorile (nt. programmile `javadoc`), mis nende põhjal koostab vastava HTML-dokumendi.

kompileerimisüksus, *compilation unit* – klasside ja liideste komplekt, enamasti kujul

[*paketikirjeldus*] [*impordikirjeldused*] *klassid ja liidesed*

kus paketikirjeldus koosneb võtmesõnast `package` ja sellele järgnevast paketi nimest ning semikoolonist; impordikirjelduseks on kas

`import` *liitnimi* . *lihtnimi* ;

või

`import` *paketinimi* . * ;

esimest tüüpi lause võimaldab antud kompileerimisüksusevälise klassi- või liidese täisnime *liitnimi.lihtnimi* asemel selle kompileerimisüksuse piires kasutada lihtnime, teine aga näidatud nimega paketi kõikide `public`-liiki klasside ja liideste puhul piirduda nende lihtnimedega; vaikimisi sisaldab iga kompileerimisüksus impordikirjeldust `import java.lang.*;` . Vt. ka *pakett.*

konstruktor, *constructor* – klassi kehas kirjeldatud eriline protseduur, mida rakendatakse →isendiloome käigus (nt. vastloodud isendi väljade algväärtustamiseks); klassi C iga konstruktor kirjeldatakse kujul

[*konstruktori piiritlejad*] C([*formaalsete parameetrite loetelu*])
[`throws` *erindiklasside loetelu*]
konstruktori keha

mitme konstruktori puhul on tegemist →üledefineerimisega, st. ühe ja sama klassi konstruktorid peavad üksteisest erinema vähemalt formaalsete parameetrite arvu (see võib olla ka 0) ja/või nende tüüpide poolest; konstruktori piiritlejatena saab kasutada võtmesõnu `public protected private` (tühikutega eraldatult), mis iseloomustavad →juurdepääsetavust sarnaselt →meetodi vastavate liikidega; ka konstruktori `throws`-konstruktsioon ja keha on analoogsed meetodi vastavate osistega, kuid tagastada ei saa väärtust (keelatud on avaldisega naasmisdirektiiv) ja konstruktori kehaks peab olema blokk ning selle esimeseks elemendiks võib veel olla antud klassi või selle (vahetu) ülemklassi konstruktori väljakutse:

`this` (*argumendid*)`;` // sobiva C(*argumendid*) väljakutse

või

`super` (*argumendid*)`;` // sobiva konstruktori väljakutse ülemklassist;

konstruktori kui meetodi väljakutse ei ole lubatud.
Vt. ka *vaikekonstruktor.*

laiendav osutiteisendus, *widening reference conversion* – →teisendused (vasakul lähtetüüp, paremal sihttüüp; K, L ja M tähistavad vastavalt klassi, liidese ja massiivi osutitüüpi):

$K \Rightarrow K'$, kus K' on K ülemklass;
$K \Rightarrow L$, kus L on K liides;
`void` $\Rightarrow K$, L, M;
$L \Rightarrow L'$, kus L' on L ülemliides;
L, $M \Rightarrow$ `Object`;
$M \Rightarrow$ `Cloneable`; //`Cloneable` on süsteemklass paketist `java.lang`

$M \Rightarrow M'$, kus M elemenditüüp on teisendatav M' elemenditüübiks mingi laiendava teisendusega;
on informatsiooniks kompilaatorile, ei too kaasa tegevusi programmi täitmise ajal.

laiendav primitiiviteisendus, *widening primitive conversion* – →teisendused

`byte` ⇒ `short`, `int`, `long`, `float`, `double`;
`short` ⇒ `int`, `long`, `float`, `double`;
`char` ⇒ `int`, `long`, `float`, `double`;
`int` ⇒ `long`, `float`, `double`;
`long` ⇒ `float`, `double`;
`float` ⇒ `double`;

informatsioon säilib, kuid teisendustel `int` ⇒ `float`, `long` ⇒ `float` ja `long` ⇒ `double` võib esineda täpsuse kadu.

lihtmuutuja, *simple variable* – →muutuja.

lihtnimi, *simple name* – identifikaator.

liides, *interface* – eeskätt abstraktsete meetodite komplekt, mille realisatsioon(id) täpsustatakse selle liidesega seostatud klassi(de)s; kirjeldatakse kujul

[`public`] `interface` *liidesenimi*
[`extends` *liidesenimede loetelu*]
liidese keha

võtmesõna `public` laiendab →juurdepääsetavust, võtmesõna `extends` järel loetletakse ülemliideste nimed; liidese keha koosneb looksulgudesse võetud null või enamast liikmekirjeldusest; liikmeks on kas siseklass, siseliides, abstraktne meetod

[`public`] *tagastustüüp meetodinimi* ([*formaalsete parameetrite loetelu*])
[`throws` *erindiklasside loetelu*] ;

või konstantne väli (konstantsed väljad)

[`public`] [`static`] [`final`] *tüüp muutujakirjeldite loetelu*;

kus muutujakirjeldid peavad olema algväärtustajaga; meetodipiiritleja `public` ja kõik väljapiiritlejad on soovitav ära jätta, kuna need lisatakse vaikimisi; nii liidese kui ka abstraktse meetodi kirjeldusele võib lisada veel piiritleja `abstract`, kuid antud kontekstis on see ülearune (ja ka ebasoovitav). Vt. ka *meetod, muutujakirjeldi.* Sünon. *liidesekirjeldus.*

liidesekirjeldus, *interface declaration* – →liides.

liige, *member* – →väli, →meetod, →siseklass või →siseliides klassi- või liidesekirjelduses.

liitnimi, *path name* – identifikaator või punktiga eraldatud identifikaatorid.

lokaalklass, *local class* – konkreetses blokis kirjeldatud, ilma piiritlejateta klass; nähtav ainult selles blokis.

lokaalmuutuja, *local variable* – blokis kasutatav lihtmuutuja, mis on selles blokis kirjeldatud vastava →muutujakirjeldusega; nõutud on ilmutatud algväärtustamine omistamisoperaatori või muutuja algväärtustaja abil; võib olla liiki `final`, mis kuulutab programmis lokaalmuutujale esimesena omistatud väärtuse konstantseks, keelates selle edaspidise muutmise; algväärtustamine →muutujakirjeldis on soovitatav. Sünon. *muutuja*.

lõim, *thread* – erilise meetodina (`run( )`) kirjeldatud alamülesande sooritamise suhteliselt iseseisvalt toimiv protsess. Vt. ka *lõimeklass*.

lõimeklass, *thread class* – süsteemklass `Thread` (paketis `java.lang`) või selle alamklass; klassi isend vastab ühele lõimele ja sisaldab klassist `Thread` pärinevaid meetodeid (`start( )`, `stop( )`, `sleep( )`) lõime töö juhtimiseks ning osutit nn. *sihtobjektile*; viimases asub lõime poolt (meetodiga `start( )`) käivitatav ja sooritatav meetod `run( )`; sihtobjekt peab olema klassist liidesega `Runnable`, erijuhul võib selleks olla ka lõimeklassi enda isend; süsteemse lõimeklassi isendi loomisel saab sihtobjekti osuti ette anda konstruktori argumendina.

lülitidirektiiv, *switch-statement* – direktiiv kujul

`switch (`*avaldis*`)` { *variandid* }

kus variant algab ühe või mitme variandimärgendiga, millele järgneb blokk (blokki ümbritsevad looksulud võib ka ära jätta); variandimärgendiks on üldiselt

`case` *avaldis* `:`

üheks variandimärgendiks võib olla ka

`default :`

analoogne keele C vastava konstruktsiooniga.

massiiv, *array* – ühte ja sama tüüpi (elemenditüüpi) →muutujate (massiivi elementide) hulk, millele juurdepääs on organiseeritud tasemeti; ühemõõtmelise massiivi osutiks on muutuja tüüpi *elemenditüüp* `[ ]` , nt.

```
int [ ] a;
```

sellise massiivi elementide arvuks on konstant `a.length` (määratakse massiiviloomel), elementideks aga (antud näite korral `int`-tüüpi) muutujad `a[0], a[1], ... , a[a.length - 1]`; m-mõõtmeline massiiv (kus $m > 1$) koosneb $(m-1)$-mõõtmelistest massiividest ja lisaks veel nende osutitest koostatud ühemõõtmelisest massiivist; osuti viimasele ongi m-mõõtmelise massiivi osutiks; m-mõõtmelise massiivi osuti tüüp kirjeldatakse m tühja nurksulupaari abil, nt. kolmemõõtmelisel juhul

```
int [ ][ ][ ] b;
```

kui muutujale `b` on omistatud osuti konkreetsele kolmemõõtmelisele massiivile, siis `b[0], b[1], ... , b[b.length - 1]` on osutid vastavatele kahemõõtmelistele massiividele (täisarvulistele maatriksitele), seega nt. `b[2][5]` on osuti neist kolmanda maatriksi kuuendale reale ja `b[2][5][0]` tähistab selle rea esimest elementi (kui muutujat); rea (üldiselt mõõtme) elementide arv ei pruugi olla konstantne, seega kahedimensionaalne massiiv ei ole tingimata ristkülikmaatriks; toodud näite korral on maatriksite arvuks `b.length`, konstandid `b[2].length` ja `b[2][5].length` aga näitavad vastavalt kolmanda maatriksi ridade arvu ja kuuenda rea pikkust selles maatriksis. Vt. ka massiiviloome.

massiivi element, *array element* – →muutuja. Vt. ka massiiv.

massiivialgati, *array initializer* – kindla mahu ja struktuuriga, elemenditüüpi avaldistest koosnev massiivi "toorik"; esitatakse nurksulgude ja komade abil, ühemõõtmelise massiivi korral:

{ [*avaldiste loetelu*] }

m-mõõtmelise massiivi korral:

{ [$(m-1)$-*mõõtmeliste massiivialgatite loetelu*] }

kasutatakse →massiiviloomel.

massiiviloome, *array creation* – massiivtüüpi objekti loomine; toimub enamasti järgmiselt:

1) ühemõõtmelise massiivi puhul avaldise

`new` *elemenditüüp* `[` *pikkus* `]`

või avaldise

[`new` *elemenditüüp* `[ ]`] *massiivialgati*
arvutamisel, mille käigus hõivatakse täisarvulise avaldisega *pikkus* näidatud arv (selle puudumisel aga massiivialgatis loetletud avaldiste arv) mälukohti elemenditüüpi väärtuste jaoks ning salvestatakse nendesse kas massiivialgatis loetletud avaldiste väärtused või (massiivialgati puudumisel) elemenditüübi vaike-algväärtused; avaldise väärtuseks on osuti loodud massiivile (osutitüübiks on *elemenditüüp* `[ ]`);
2) kahemõõtmelise massiivi puhul avaldise
(a) `new` *elemenditüüp* `[` *pikkus1* `]` `[` [*pikkus2*] `]`
või avaldise
(b) [`new` *elemenditüüp* `[ ][ ]`] *massiivialgati*
arvutamisel; juhul (a), kus massiivialgati puudub, kui *pikkus2* on antud, luuakse *pikkus1* ühemõõtmelist massiivi, igaühes *pikkus2* elemenditüübi vaike-algväärtust; kui *pikkus2* ei ole antud, siis neid massiive ei looda (tegemist on nn. *osalise massiiviloomega*); loodud massiivide osutitest (tüüpi *elemenditüüp*`[ ]`) (või tühiosutitest, kui *pikkus2* puudub) moodustatakse omakorda ühemõõtmeline *pikkus1*-elemendiline massiiv ning selle osuti (tüüpi *elemenditüüp* `[ ] [ ]`) saab avaldise (a) väärtuseks;
massiivialgati olemasolu korral (antud juhul sisaldab see $n \geq 0$ ühemõõtmelist massiivialgatit) luuakse vastavalt n ühemõõtmelist massiivi, mille elementidele omistatakse vastavate avaldiste väärtused; loodud massiivide osutitest (tüüpi *elemenditüüp*`[ ]`) moodustatakse omakorda ühemõõtmeline n-elemendiline massiiv ning selle osuti (tüüpi *elemenditüüp* `[ ] [ ]`) saab avaldise (b) väärtuseks;
3) suurema mõõtmega massiivi loome toimub analoogselt; ka siin võib osaliseks massiiviloomeks ära jätta viimaste mõõdete pikkusi;
mälupuuduse korral tekib erind `OutOfMemoryError`.

meetod, *method* – klassi kehas (erikujulisena ka →liidese kehas) kirjeldatud funktsioon, esitatakse kujul
[*meetodi piiritlejad*]
tagastustüüp meetodinimi([*formaalsete parameetrite loetelu*])
[`throws` *erindiklasside loetelu*]
meetodi keha
meetodi piiritlejatena saab kasutada võtmesõnu `public protected private abstract static final synchronized native` (tühikutega eraldatult), millest

`public protected private` iseloomustavad →juurdepääsetavust;

`abstract` – realiseerimata meetod, meetodi keha asemel on vaid semikoolon; tohib esineda ainult kas liideses või siis sellises klassis, mille liigiks on samuti `abstract`;

`static` – nn. *klassimeetod*, mille väljakutse on võimalik kõikjalt, kus klass (mis sisaldab selle meetodi kirjeldust) on nähtav, ja seda sõltumata isendite olemasolust; osuteid `this` ja `super` ei saa klassimeetodi kehas kasutada; meetod ei või olla samaaegselt `static` ja `abstract`; meetodit, mis ei ole `static`, nimetatakse *isendimeetodiks*, sellise meetodit väljakutse on võimalik ainult mingi olemasoleva isendi kaudu (meetodi nimele lisandub veel isendi *osuti.* , klassisisesel kasutamisel on selleks vaikimisi `this`.);

`final` – keelab alamklassides selle meetodi ülekatte; ülearune, kui klass ise on `final` või antud meetod on liiki `private`;

`synchronized` – nõuab enne meetodi rakendamist isendi lukustust (isendimeetodi korral) või klassi lukustust (klassimeetodi korral);

`native` – mõnes muus keeles realiseeritud meetod, meetodi keha asemel on vaid semikoolon; ei saa samaaegselt olla liiki `abstract`;

enne meetodi keha tuleb võtmesõna `throws` järel loetleda need meetodi töö käigus tekkida võivad erindid (väljaarvatud klassid *Error* ja *RuntimeException*), mida selles meetodis ei töödelda; meetodi kehaks on kas semikoolon või blokk.

muutuja algväärtustaja, *variable initializer* – →muutujakirjeldi.

muutuja, *variable* – teatavat tüüpi väärtuse jaoks ettenähtud mäluväli: lihtmuutuja või massiivi element; muutujale saab omistada väärtust ning seda väärtust kasutada avaldise arvutamisel; *lihtmuutujale* viidatakse tema nime abil, m-mõõtmelise *massiivi elemendile* massiivi osutit sisaldava lihtmuutujaga, millele järgneb m nurksulgudesse võetud indeksavaldist; kasutatakse ka mõistete "lokaalmuutuja", "formaalne parameeter" ja "väli" üldnimetusena.

muutujakirjeldi, *variable declarator* – lokaalmuutuja, formaalse parameetri või välja kirjelduse osa kujul:

nimi dimensioon [`=` *muutuja algväärtustaja*]

kus dimensiooniks on null või enam nurksulupaari `[ ]` ja *muutuja algväärtustajaks* sobivat tüüpi avaldis või massiivialgati. Vt. ka *muutujakirjeldus, väli*.

muutujakirjeldus, *variable declaration* – lokaalmuutuja, formaalse parameetri või välja nime, tüüpi, liiki ja erijuhul ka algväärtust määrav konstruktsioon; lihtsamaid muutujakirjeldusi:

[`final`] *tüüp lokaalmuutuja nimi*;

– *lihtne muutujakirjeldus*, reserveerib koha antud tüüpi ja antud nimega lokaalmuutujale (ja salvestab sellele kohale antud tüübi vaike-algväärtuse);

[`final`] *tüüp formaalse parameetri nimi*

– *formaalse parameetri kirjeldus*, reserveerib koha antud tüüpi ja antud nimega muutujale vastava argumendi väärtuse salvestamiseks; esineb formaalsete parameetrite loetelu elemendina;

[`final`] *tüüp lokaalmuutuja nimi* `=` *muutuja algväärtustaja*;

– *algväärtustamisega muutujakirjeldus*, reserveerib koha antud tüüpi ja antud nimega lokaalmuutujale, arvutab muutuja algväärtustaja ja omistab saadud väärtuse vastloodud muutujale;

lihtsat ja algväärtustamisega muutujakirjeldust kasutatakse veel →välja kirjeldamisel, kus aga võimalike piiritlejate valik on märksa suurem; muutujakirjeldusi saab anda ka mõnevõrra teisendatult ning ühendatult:

(1) lihtsas ja ka algväärtustamisega muutujakirjelduses ning samuti formaalse parameetri kirjelduses võib tüübist suvalise arvu nurksulupaare viia dimensiooniks muutuja või formaalse parameetri nime järele, nt. on samaväärsed

```
int [ ][ ][ ] x;
int [ ][ ] x[ ];
int [ ] x[ ][ ];
int x[ ][ ][ ];
```

(2) ühte ja sama tüüpi ning liiki muutujate (kuid mitte formaalsete parameetrite) kirjeldused võib ühendada, nt.

`final int` *muutujakirjeldi1* `;`

`final int` *muutujakirjeldi2* `;`

`final int` *muutujakirjeldi3* `;`

saab asendada konstruktsiooniga

`final int` *muutujakirjeldi1*`,` *muutujakirjeldi2*`,` *muutujakirjeldi3* `;` .

mähisklass, *wrapper class* – klass, mille peamiseks ülesandeks on seostada mingi (tavaliselt konstruktori argumendina antud) objekti või väärtusega täiendavaid meetodeid; tühitüübi ja iga algtüübi jaoks leidub süsteemne mähisklass (paketis `java.lang`), vastavalt `Void`, `Byte`, `Short`, `Integer`, `Long`, `Float`, `Double`, `Boolean`, `Character`.

naasmisdirektiiv, *return-statement* – meetodis või konstruktoris esinev direktiiv kujul

`return` [*avaldis*]`;`

avaldiseta naasmisdirektiiv võib esineda ainult konstruktoris ja sellises meetodis, mille tagastustüübiks on tühitüüp `void`; sel juhul seisneb direktiivi täitmine antud meetodist või konstruktorist väljumises vastavasse väljakutse kohta; *avaldisega naasmisdirektiiv* võib esineda ainult sellises meetodis, mille tagastustüübiks on antud avaldise tüüp (täpsemalt, avaldise väärtus peab olema omistatav meetodi tagastustüüpi muutujale); sel juhul direktiivi täitmisel arvutatakse avaldise väärtus, loetakse see antud meetodi kui funktsiooni väärtuseks ja väljutakse meetodist vastavasse väljakutse kohta; kui naasmisdirektiiv sisaldub katsendidirektiivis, siis enne väljumist täidetakse veel vastav epiloog; viimase järsk sooritatus võib erijuhul saada takistuseks meetodist või konstruktorist väljumisele (selle naasmisdirektiivi kaudu).

operaator, *operator* – avaldises arvutuslikku tehet tähistav märk (märgid); alanevate prioriteeditasemete kaupa:

`[ ]` – massiivi elemendi indekseerimine;
`.` – objekti liikme (liitnime) täpsusutus;
`(`[*argumentide loetelu*]`)` – meetodi või konstruktori väljakutse;
`++` – järel-ühiksuurendamine;
`--` – järel-ühikvähendamine;

`++` – eel-ühiksuurendamine;
`--` – eel-ühikvähendamine;
`+` – unaarne pluss;
`-` – unaarne miinus;
$\sim$ – unaarne bitieitus;
`!` – unaarne loogiline eitus;

`new` – isendiloome või massiiviloome;
`(`*tüüp*`)` – tüübiteisendus;
`*` – korrutamine;
`/` – jagamine;
`%` – jäägi leidmine;

`+` – liitmine või sõnesidurdus;
`-` – lahutamine;

`<<` – nihutamine vasakule;
`>>` – aritmeetiline nihutamine paremale;
`>>>` – loogiline nihutamine paremale;

`<` – on väiksem kui;
`<=` – on väiksem võrdne kui;
`>` – on suurem kui;
`>=` – on suurem võrdne kui;
`instanceof` – (isend) on klassist;

`==` – võrdub;
`!=` – ei võrdu;

`&` – bitikaupa või loogiline korrutamine;

`^` – bitikaupa või loogiline mittesamaväärsus;

`|` – bitikaupa või loogiline liitmine;

`&&` – loogiline “ja”;

`||` – loogiline “või”;

`? :` – ternaarne tingimuslik tehe;

---

omistamisoperaatorid:
`=` – omistamine; mõnede binaarsete tehetega kombineeritult:
`+=  -=  *=  /=  %=  <<=  >>=  >>>=  &=  ^=  |=` .

osaline massiiviloome, *incomplete array creation* – →massiiviloome.

osutitüüp, *reference type* – →tüüp.

pakett, *package* – kõigi nende →kompileerimisüksuste hulk, mis algavad ühe ja sama paketikirjeldusega; üks (nimetu) pakett moodustub nendest kompileerimisüksustest, milles paketikirjeldus puudub.

peameetod, *main method* – kohustuslik klassimeetod iseseisvat rakendust kirjeldavas klassis; peab olema liiki `public static`, tagastustüübiga `void` ja signatuuriga `main(String[ ])`; väljakutse toimub selle klassi käivitamisel rakendava keskkonna poolt, nt. käsureaga kujul

`java` [*võtmed*] *klassi nimi* [*argumendid*]

seejuures peameetodile edastatavaks argumendiks on osuti sõnedest (käsurea argumentidest) koosnevale ühemõõtmelisele massiivile.

peitmine, *hiding* – →varje.

piiritleja, *modifier* – klassi, liidese, meetodi, konstruktori, välja ja lokaalmuutuja kasutusliiki täpsustav (vastava kirjelduse alguses esinev) võtmesõna. Sünon. *modifikaator.*

pärimine, *inheritance* – automaatne mehhanism, mis lisab igale klassikirjeldusele need vahetu ülemklassi ja liidese liikmekirjeldused (võimalik, et omakorda päritud), mis ei ole liiki `private`; selliselt lisatud liikmekirjeldusi nimetatakse *päritud liikmekirjeldusteks*; sama mehhanism toimib ka liidesekirjelduste puhul. Vt. ka *ülekate, varje.* Sünon. *pärilus.*

päritud liikmekirjeldus, *inherited member declaration* – →pärimine.

püünis, *catch-clause* – →katsendidirektiiv.

rakend, *applet* – Java rakendus www-lehekülje tarvis; esitatakse klassina, mille ülemklassiks on süsteemklass `Applet` (paketist `java.applet`); ei sisalda →peameetodit, kuna klassi isend luuakse ja käitatakse (tema teatavate meetodite väljakutsega) erilise →rakendikäituri poolt. Sünon. *aplett, rake.*

rakendikäitur, *applet viewer* – HTML-dokumendis rakendiviidaga antud rakendi(te) käitusprogramm; kuulub ka Web-lehitsejate koosseisu. Sünon. *rakendi-kuvaja, rakendinäitur.*

rakendiviit, *applet tag* – rakendi käitamist kirjeldav sektsioon HTML-dokumendis, Java-toetusega Web-lehitsejate jaoks kujul

`<APPLET` *atribuudid* `>`

[parameetrid]
`</APPLET>`
kus kohustuslikeks on atribuudid
`CODE` = *rakendiklassi nimi*
`WIDTH` = *rakendiakna laius*
`HEIGHT` = *rakendiakna kõrgus*
mittekohustuslike atribuutidega saab anda rakendiklassi sisaldava kataloogi URL-i (`CODEBASE`), ajutise nime jooksva rakendi isendile (`NAME`), ekraanipildi täpsustuse (`ALIGN`, `VSAPCE`, `HSPACE`); parameetriga
`<PARAM NAME =` *leppenimi* `VALUE =` *"tekst"*`>`
esitatakse rakendile HTML-dokumendist edastatav tekst (rakendis loob vastava sõne ja tagastab osuti sellele süsteemklassist `Applet` pärineva isendimeetodi väljakutse `getParameter(`*"leppenimi"*`)`).

realisatsioon, *implementation* – →ülekate. Sünon. *teostus*.

samasusteisendus, *identity conversion* – →teisendus kokkulangeva lähte- ja sihttüübi korral; tühitegevus.

signatuur, *signature* – meetodi või konstruktori iseloomustus, mis koosneb (meetodi või konstruktori) nimest ning formaalsete parameetrite tüüpide loetelust.

siseklass, *nested class* – klassi või liidese liikmeks olev klass; skoobilt analoogne →väljaga, kuid lubatud on vaid piiritlejad `protected`, `private`, `static`. Vt. ka *klassikirjeldus*.

siseliides, *nested interface* – klassi või liidese liikmeks olev liides; skoobilt analoogne →väljaga, kuid lubatud on vaid piiritlejad `protected`, `private`, `static`. Vt. ka *liides*.

sooritatus, *completion* – on *normaalne*, kui avaldise arvutamine või direktiivi sooritamine ei riku programmi loomulikku täitmisjärjekorda; vastasel korral on tegemist *järsu sooritatusega*.

standardpakett, *standard package* – Java põhikomplekti kuuluv pakett; nt.
`java.applet` – rakendite programmeerimise vahendid;
`java.awt` – graafikatöö ja -liideste vahendid;
`java.io` – sisestus- ja väljastusvahendid;

`java.lang` – standardsed keelevahendid;
`java.net` – võrgurakenduste programmeerimise vahendid;
`java.util` – muud abivahendid;

jt. Sünon. *Java pakett.*

sõneksteisendus, *string conversion* – antud avaldise väärtuse teisendamine sõneks; algtüübi korral meetodiga `valueOf( )` klassist `String`, osutitüübi korral vastava isendi meetodiga `toString( )`; need meetodid loovad uue sõnetüüpi objekti, mis kujutab vastavalt kas algtüüpi argumendi väärtuse või antud isendi tekstilist esitust ja tagastavad osuti sellele sõnele; standardne meetod `toString( )` leidub klassis `Object` ja kandub pärimise teel kõikidesse teistesse klassidesse (kus tavaliselt esitatakse selle ülekate); standardne `toString( )` loob sõne, mis sisaldab vaid antud isendi klassi nime ja tema paiskkoodi 16-süsteemis, tühiosuti korral aga loob sõne "null".

sünkroondirektiiv, *synchronized-statement* – direktiiv kujul

`synchronized (`*avaldis*`)` *blokk*

mille sooritamisel arvutatakse avaldise väärtus (mis peab olema osutitüüpi ja mitte `null`), seatakse lukustus viimase poolt osutatud isendile või massiivile, täidetakse blokk ja avatakse lukustus; tagab selle, et bloki töö ajal ükski teine lõim ei muuda osutatud isendit või massiivi.

süsteemklass, *system class* – →standardpaketti kuuluv klass.

tagastustüüp, *result type* – →naasmisdirektiiv. Vt. ka *meetod.*

teisendus, *conversion* – väärtuse tüübi teisendus, võib kaasa tuua väärtuse enda teisendamise; teisenduse liikideks on →samasusteisendus, →laiendav primitiiviteisendus, →kitsendav primitiiviteisendus, →laiendav osutiteisendus, →kitsendav osutiteisendus ja →sõneksteisendus; teisendust rakendatakse antud avaldise arvutamisel saadud (*lähtetüüpi*) väärtuse teisendamiseks mingit teist tüüpi (*sihttüüpi*) väärtuseks, see toimub kas vaikimisi või ilmutatult; *vaiketeisendus* leiab aset:
1) omistamisel omistamisoperaatoriga – lähtetüübiks on omistatava väärtuse tüüp ja sihttüübiks selle muutuja tüüp, millele väärtust omistatakse; siin on lubatud samasusteisendus ja mõlemad laiendavad teisendused ning erandina ka kitsendav primitiiviteisendus, juhul kui sihttüübiks on `byte`, `short` või `char` ning avaldiseks on konstantne avaldis (lähte)tüüpi `int`;
2) väljakutses argumendi väärtuse omistamisel formaalsele parameetrile;

erinevalt omistamisoperaatorist pole siin erand lubatud;
3) meetodist naasmisel avaldisega →naasmisdirektiivi korral; lähtetüübiks on avaldise väärtuse tüüp ja sihttüübiks selle meetodi tagastustüüp; lubatud on samad teisendused, mis omistamiselgi;
4) sõnesidurduses – kui binaarse operaatori `+` üks operandidest on tüüpi `String` ja teine mingit muud tüüpi, siis teisele operandile rakendatakse eelnevalt sõneksteisendust;
5) avaldise arvutamisel – operandi väärtus teisendatakse operaatori sooritamiseks vajalikku tüüpi (nt. enamiku binaarsete operaatorite operandid peavad olema ühte ja sama tüüpi väärtused); lubatud on samasusteisendus ja laiendav primitiiviteisendus;
ilmutatud teisendus esitatakse kujul

`(`*sihttüüp*`)` avaldis

kus lähtetüübiks on avaldise arvutamisel saadud väärtuse tüüp; lubatud on kõik teisenduse liigid, väljaarvatud sõneksteisendus;
erinevate teisendusliikide määratlustest tuleneb, et samasusteisendus ja sõneksteisendus on alati sooritatavad ning (mingil muul moel) ei ole võimalikud järgmised teisendused (vasakul lähtetüüp, paremal sihttüüp; K, L ja M tähistavad vastavalt klassi, liidese ja massiivi osutitüüpi):

osutitüüp ⇒ algtüüp;
algtüüp ⇒ osutitüüp;
`void` ⇒ algtüüp;
tüüp ⇒ `void`;
tüüp ⇒ `boolean`;
`boolean` ⇒ *tüüp*;
$K \Rightarrow K'$, kui K ei ole K' ülemklass ega alamklass;
$K \Rightarrow L$, kui klass K on liiki `final` ja L ei ole K liideseks;
$L \Rightarrow K$, kui klass K on liiki `final` ja L ei ole K liideseks;
$L \Rightarrow L'$, kui L ja L' sisaldavad kokkulangeva signatuuriga, kuid erineva tagastustüübiga meetodit;
$K \Rightarrow M$, kui K ei ole `Object`;
$M \Rightarrow K$, kui K ei ole `Object`;
$M \Rightarrow L$, kui L ei ole `Clonable`;
$M \Rightarrow M'$, kui M elemendi väärtuse teisendamine massiivi M' elemenditüüpi ei ole võimalik (st. muul moel, kui vaid samasusteisenduse ja sõneksteisenduse abil).

Sünon. *konversioon, tüübiteisendus.*

tingimusdirektiiv, *if-statement* – direktiiv kujul

`if (`*loogiline avaldis*`)` *direktiiv1* [`else` *direktiiv2*]

analoogne keele C vastava konstruktsiooniga.

tsüklidirektiiv, *loop statement* – direktiivide →eelkontrolliga tsüklidirektiiv, →järelkontrolliga tsüklidirektiiv ja →üldtsüklidirektiiv üldnimetus. Sünon. *tsükkel*.

tüübiteisendus, *casting* – →teisendus.

tüüp, *type* – teatud mõttes ühetaoliste objektide (väärtuste) hulk;

`void` – *tühitüüp*, kasutatakse ainult tagastustüübina niisuguste meetodite puhul, mis tegelikult väärtust ei tagasta;

ülejäänud tüübid sisaldavad väärtusi, mida saab omistada muutujatele ja väljadele ning tagastada meetoditest;

`byte` – 8-bitilised täisarvud, $-128\ldots127$;

`short` – 16-bitilised täisarvud, $-32768\ldots32767$;

`int` – 32-bitilised täisarvud, $-2147483648\ldots2147483647$;

`long` – 64-bitilised täisarvud, $-9223372036854775808\ldots9223372036854775807$;

`float` – 32-bitilised ujupunktarvud, $-3,4\times10^{38}\ldots3,4\times10^{38}$, 7 tüvekohta;

`double` – 64-bitilised ujupunktarvud, $-1,7\times10^{308}\ldots1,7\times10^{308}$, 15 tüvekohta;

`char` – 16-bitiliste *Unicode*-sümbolite hulk;

`boolean` = {`true`, `false`};

klassinimi – nimetatud klassi isendite osutite hulk;

elemenditüüp `[ ]` – nimetatud tüüpi elementidest koosnevate konkreetsete ühemõõtmeliste massiivide osutite hulk; elemenditüübiks võib olla suvaline mittetühi tüüp (ka käesolev);

viimast kahte tüüpi nimetatakse *osutitüüpideks*, kõiki eelmisi, väljaarvatud tühitüüp, aga *algtüüpideks*; lisaks konkreetsetele osutitele kuulub mõlemasse osutitüüpi ka eriline *tühiosuti* `null`; vahetu klasstüüp ja massiivtüüp Java-keeles puuduvad: klasstüüpi objekti (isendit) ja konkreetset massiivi ei saa omistada muutujale ega tagastada meetodist; puudub ka sõnetüüp, Javas käsitletavad sõned on süsteemse klassi `String` isenditeks.

vaike-algväärtus, *default value* – antud →tüüpi kuuluv selline väärtus, mis vaikimisi omistatakse seda tüüpi muutujale või väljale vajaliku mälukoha reserveerimisel; osutitüüpide korral – `null`, tüüpide `float` ja `double` korral – vastava ujupunktarvuna kujutatud arv 0, loogilise tüübi korral – `false`, ülejäänud (mittetühjade) tüüpide korral – vastava pikkusega 0-biti-järjend. Sünon. *vaikimisi-algväärtus.*

vaikekonstruktor, *default constructor* – parameetriteta konstruktor kujul

[`public`] C `( )` { `super( );` }

rakendatakse →isendiloomel, kui antud klass C ei sisalda ühtegi konstruktorit; piiritlejaks on `public` siis ja ainult siis, kui antud klass on liiki `public`; eeldab, et ülemklass (kui see pole `Object`) sisaldab parameetriteta konstruktorit; tühitegevus, kui ülemklassiks on `Object`. Sünon. *vaikimisi-konstruktor.*

vaiketeisendus, *implicit conversion* – →teisendus. Sünon. *vaikimisi-teisendus, ilmutamata teisendus.*

varje, *shadowing* – (reeglina taunitav) olukord, kus klassis kirjeldatud välja nimi langeb kokku ülemklassilt pärimise teel saadud välja nimega; vastavad väljad ei pruugi olla ühte ja sama tüüpi; vaikimisi (välja nime järgi) kasutatakse antud klassis kirjeldatud välja, ülemklassi vastavale väljale (mis võib olla ka isendiväli) viitamiseks lisatakse nimele prefiks `super.` . Sünon. *peitmine.*

võtmesõna, *keyword* – üks järgmistest sõnakujulistest terminaalsetest sümbolitest (tärn märgib senini veel mittekasutatavat):

`abstract`, `boolean`, `break`, `byte`, `byvalue`*, `case`, `cast`*, `catch`, `char`, `class`, `const`*, `continue`, `default`, `do`, `double`, `else`, `extends`, `false`, `final`, `finally`, `float`, `for`, `future`*, `generic`*, `goto`*, `if`, `implements`, `import`, `inner`*, `instanceof`, `int`, `interface`, `long`, `native`, `new`, `null`, `operator`*, `outer`*, `package`, `private`, `protected`, `public`, `rest`*, `return`, `short`, `static`, `super`, `switch`, `synchronized`, `this`, `throw`, `throws`, `transient`, `true`, `try`, `var`*, `void`, `volatile`, `while`;

neid ei tohi valida nimedeks (identifikaatoriteks). Sünon. *reserveeritud sõna.*

väli, *field* – klassi kehas (erijuhul →liidese kehas) esinev muutuja, kirjeldatakse kujul

[*välja piiritlejad*] *tüüp muutujakirjeldite loetelu* `;`

kus iga →muutujakirjeldi kirjeldab ühte välja; välja piiritlejatena saab kasutada võtmesõnu `public protected private static final transient volatile` (tühikutega eraldatult), millest

`public protected private` iseloomustavad →juurdepääsetavust;

`static` – nn. *klassiväli*, millele mäluväli eraldatakse klassi laadimisel ja mille kasutamine on võimalik kõikjalt, kus klass (mis sisaldab selle välja kirjeldust) on nähtav; klassiväli ei ole isendi osaks; välja, mis ei ole `static` nimetatakse *isendiväljaks*, selline väli kuulub iga isendi struktuuri ja selle kasutamine on võimalik ainult vastava isendi kaudu (välja nimele lisandub veel isendi *osuti.* , klassisisesel kasutamisel on selleks vaikimisi `this.`);

`final` – kuulutab väljale esimesena omistatud väärtuse konstantseks, keelates selle edaspidise muutmise; algväärtustamine muutujakirjeldis on soovitatav; kasutatav nii klassiväljade kui ka isendiväljade korral;

`transient` – ajutise iseloomuga isendiväli, selle väärtust ei kaasata isendi ümbersalvestamisel;

`volatile` – lõimes sünkroniseeritult kasutatav väli;

Vt. ka *muutujakirjeldi.* Sünon. *muutuja*

üldtsüklidirektiiv, *for-statement* – direktiiv kujul

`for(`[*eeltegevuste loetelu*]`;` [*loogiline avaldis*] `;` [*sammu järeltegevuste loetelu*]`)` *sisu*

kus sisuks on →direktiiv; analoogne keele C vastava konstruktsiooniga.

üledefineerimine, *overloading* – olukord, kus klassi kuulub mitu sama nimega, kuid erineva signatuuriga meetodit (ka pärimise teel saadut) või mitu konstruktorit; väljakutse puhul rakendatakse neist väljakutses antud argumentide tüüpide poolest sobivat.

ülekate, *overriding* – olukord, kus klassis kirjeldatud ja pärimise teel saadud meetodil on üks ja sama signatuur; väljakutse puhul rakendatakse klassis kirjeldatud meetodit; päritud meetodi väljakutseks taolisel juhul lisatakse aga meetodi nimele prefiks `super.` ; ülemklassi meetodi ülekate on lubatud ainult sellise meetodiga, millel on samasugune tagastustüüp ja erindiklasside (`throws`) loetelu ning vähemalt sama avar juurdepääsetavus; viimane nõue tähendab nt. seda, et `public`-meetodi ülekatteks saab kasutada ainult `public`-meetodit, `protected`-meetodi ülekatteks – ainult kas `protected`- või `public`-meetodit, piiritlejateta meetodi ülekate `private`-meetodiga pole võimalik; abstraktse meetodi ülekatet mitteabstraktse meetodiga nimetatakse selle abstraktse meetodi *realisatsiooniks.*

Inglise-eesti sõnastik

Poolpaksus kirjas märksõnale vastab artikkel leksikonis.

abrupt completion – **järsk sooritatus**
absolute position – absoluutne positsioon
abstract class – **abstraktne klass**
abstract method – **abstraktne meetod**
abstraction – abstraktsioon
acceptable – vastuvõetav
access modifier – juurdepääsu piiritleja
accessibility – **juurdepääsetavus**
action – toiming
allocation – paigutus
applet tag – **rakendiviit**
applet viewer – **rakendikäitur**
applet – **rakend**
application – rakendus
arc – sektor
area – ala
argument – argument
arithmetic exception – aritmeetikaerind
array copy – massiivi kopeerimine
array creation – **massiiviloome**
array element – **massiivi element**
array index out of bound exception – massiivi rajaerind
array initializer – **massiivialgati**
array store exception – massiivi salvestuserind
array – **massiiv**
attribute – atribuut
audio – audio
Abstract Windowing Toolkit – abstraktsed aknavahendid
base class – baasklass
base type – baastüüp
basic access modifier – juurdepääsu baaspiiritleja
binding – sidumine
block – **blokk**

body – keha
borderlayout – äärekujundus
break-statement – **katkestusdirektiiv**
browser – lehitseja
button – nupp
byte code verifier – baitkoodi verifikaator
byte-code – baitkood
callback – tagasiväljakutse
canvas – lõuend
cardlayout – kaartkujundus
casting – **tüübiteisendus**
catch-clause – **püünis**
checkbox group – märkeruudugrupp
checkbox – märkeruut
choice – valik
circle – ringjoon
class body – klassi keha
class declaration – **klassikirjeldus**
class field – **klassiväli**
class file – klassifail
class format error – klassi formaadiviga
class importing – klassi importimine
class instance – klassi isend
class loader – klassilaadur
class method – **klassimeetod**
class path – klassitee
class qualifier – klassi otsimistee
class structure – klassi struktuur
class type – **klasstüüp**
class – **klass**
client – klient
clipping – kärpimine
color model – värvimudel
color – värv
command line argument – käsurea argument
comment – **kommentaar**
compilation unit – **kompileerimisüksus**

compiler – kompilaator
completion – **sooritatus**
component – komponent
composition – kompositsioon
connection oriented protocol – ühendusega protokoll
connection – ühendus
connectionless protocol – ühenduseta protokoll
constant value – muutumatu väärtus
constructor – **konstruktor**
container – konteiner
content handler – sisukäsiti
content – sisu
context – kontekst
continue-statement – **jätkamisdirektiiv**
controller – kontroller
conversion – **teisendus**
coordinates – koordinaadid
copy – koopia
core – tuum
creation – loome
current instance – jooksev isend
cyclic reference – tsükliline osutus
daemon – deemon
data item – andmeelement
datagramm – datagramm
debugging – silumine
declaration – kirjeldus
default constructor – **vaikekonstruktor**
default level of visibility – vaikimisi-nähtavuspiirkond
default value – **vaike-algväärtus**
default – vaike-
derivation – tuletamine
derived type – tuletatud tüüp
destruction – hävitamine
development environment – arenduskeskkond
development kit – arenduskomplekt
development – arendus

dialog – dialoog
do-statement – **järelkontrolliga tsüklidirektiiv**
double buffering – topeltpuhverdus
drawing method – joonistusmeetod
dynamic binding – dünaamiline sidumine
embedded – sard-
empty stack exception – tühimagasini erind
empty statement – tühidirektiiv
empty type – tühitüüp
encapsulation – kapseldus
enumeration – loetelu
environment variable – keskkonnamuutuja
environment – keskkond
error class – veaklass
error handler – veakäsiti
error – viga
evaluation – arvutamine
event handling – sündmusekäsitlus
event – sündmus
exception class – **erindiklass**
exception handling – veakäsitlus
exception – **erind**
execution – sooritamine
explicit call – ilmutatud väljakutse
explicit conversion – **ilmutatud teisendus**
expression-statement – **avaldisdirektiiv**
expression – avaldis
extention – avardamine
field – **väli**
file not found exception – faili puudumise erind
file stream – failivoog
file – fail
final – lõplik
finalization – lõplikustamine
finally-clause – **epiloog**
flag – lipp
flowlayout – voogkujundus

focus – fookus
font metrics – fondimeetrika
font – font
for-statement – **üldtsüklidirektiiv**
formal parameter – formaalne parameeter
fragile base class – õrn baasklass
frame – raam
framework – raamistik
friendly class – sõbralik klass
fully qualified class name – täielik klassinimi
functional – funktsionaalne
FTP (File Transfer Protocol) – FTP
garbage collection – mälukoristus
general exception – üldRerind
graphics context – graafikakontekst
graphics object – graafikaobjekt
gridlayout – võrkkujundus
GUI (Graphics User Interface) – graafiline kasutajaliides
handler – käsiti
handling – käsitlemine
hashcode – paiskkood
hashtable – paisktabel
header – päis
heap – kuhi
hiding – **peitmine**
host name – hostinimi
host – host
HTML (HyperText Mark-up Language) – HTML
HTTP (HyperText Transfer Protocol) – HTTP
identity conversion – **samasusteisendus**
if-statement – **tingimusdirektiiv**
illegal argument exception – vale argumendi erind
image data – pildiandmed
image observer – pildivaatleja
image processing – pilditöötlus
image – pilt
implementation – **realisatsioon**

implicit conversion – **vaiketeisendus**
incomplete array creation – **osaline massiiviloome**
index out of bound exception – indeksi rajaerind
inheritance hierarchy – pärimishierarhia
inheritance tree – pärimispuu
inheritance – **pärimine**
inherited member declaration – **päritud liikmekirjeldus**
initializer – **algataja**
instance field – **isendiväli**
instance method – **isendimeetod**
instance – **isend**
instantiation – **isendiloome**
interface declaration – **liidesekirjeldus**
interface – **liides**
interpreter – interpretaator
invokation – väljakutse
irregularity – korrapäratus
J-code – J-kood
Java script – Javaskript
Java – Java
keyword – **võtmesõna**
kit – komplekt
label – märgend
layout manager – kujundusjuht
layout – kujundus
layoutscheme – kujundus-skeem
length – pikkus
library – teek
line – joon
list – nimekiri
literal – literaal
local class – **lokaalklass**
local variable – **lokaalmuutuja**
location – asukoht
locking – lukustus
loop statement – **tsüklidirektiiv**
low-level – madaltase

main method – **peameetod**
manager – juht
member – **liige**
menu item – menüüvalik
menu – menüü
menubar – menüüriba
method resolution system – meetodi otsimise süsteem
method signature – meetodi signatuur
method – **meetod**
model – mudel
modifier – **piiritleja**
monitor object – monitor-objekt
multidimensional array – mitmemõõtmeline massiiv
multithreaded – mitmelõimeline
MIME (Multipurpose Internet Mail Extensions) – MIME
MIME type – MIME tüüp
narrowing primitive conversion – **kitsendav primitiiviteisendus**
narrowing reference conversion – **kitsendav osutiteisendus**
native code – omakood
native – oma
negative array size exception – massiivi negatiivse suuruse erind
nested class – **siseklass**
nested interface – **siseliides**
networked application – võrgurakendus
new – uus
no such element exception – elemendi puudumise erind
null pointer exception – nullviida erind
null reference – tühiosuti
null – null
object creation – objekti loomine
object destruction – objekti hävitamine
object – objekt
off-screen buffer – eeljoonistuspuhver
off-screen drawing – eeljoonistamine
operator – **operaator**
origin – nullpunkt
oval – ovaal

overloading – **üledefineerimine**
overriding – **ülekate**
package – **pakett**
painting – ülejoonistamine
panel – paneel
parent class – ülemklass
path name – **liitnimi**
path – otsimistee
pipe – toru
polygon – hulknurk
port – port
portability – teisaldatavus
predefined – eeldefineeritud
primitive type – **algtüüp**
priority – prioriteet
private type – privaattüüp
private – privaatne
protected – kaitstud
protocol handler – protokollikäsiti
protocol – protokoll
protocoll exception – protokollierind
public variable – avalik muutuja
public – avalik
query – päring
random access file – otsepöördusfail
random – juhuslik
reclaim – taastumine
rectangle – ristkülik
reference type – **osutitüüp**
reference – osuti
relationship – seos
repainting – taasjoonistamine
resource – ressurss
restrict access – piiratud juurdepääs
result type – **tagastustüüp**
return-statement – **naasmisdirektiiv**
robust – töökindel

run-time environment – käituskeskkond
run-time exception – käitusaegne erind
run-time system – käitusaegne süsteem
run-time – käitusaegne
runnable interface – käitusliides
safe – ohutu
scalability – laiendatavus
scope – skoop
script language – skriptkeel
script – skript
scrollbar – kerimisriba
secure – turvaline
security exception – turbeerind
security manager – turbehaldur
security – turve
server – server
set – hulk
shadowing – **varje**
shape – kujund
shared string pool – sõnede ühisala
signature – **signatuur**
simple name – **lihtnimi**
simple variable – **lihtmuutuja**
singlethreaded – ühelõimeline
socket – pistikuliides
standard package – **standardpakett**
statement – **direktiiv**
static method – staatiline meetod
static variable – staatiline muutuja
static – staatiline
stream wrapper – voomähkur
stream – voog
string conversion – **sõneksteisendus**
string method – sõnemeetod
string – sõne
subclass – alamklass
subinterface – alamliides

submenu – alammenüü
subtype – alamtüüp
superclass – ülemklass
supertype – ülemtüüp
switch-statement – **lülitidirektiiv**
synchronized-statement – **sünkroondirektiiv**
system class – **süsteemklass**
terminal I/O – terminalikäitus
terminating – hävitamine
text area – tekstiala
text component – tekstikomponent
text encoding – teksti kodeering
text field – tekstiväli
thread class – **lõimeklass**
thread control – lõime juhtimine
thread group – lõimerühm
thread priority – lõime prioriteet
thread syncronization – lõime sünkroniseerimine
thread – **lõim**
throw-statement – **erindiseade direktiiv**
throw – **erindiseade**
token – lekseem
tool – töövahend
toolkit – töövahendite komplekt
transgression – korrapäratus
try-statement – **katsendidirektiiv**
try – katse
type system – tüübisüsteem
type – **tüüp**
unknown error – tundmatu viga
unknown host exception – tundmatu hosti erind
unknown server exception – tundmatu serveri erind
updating – värskendamine
URL (Uniform Resourse Locator) – URL
URL connection – URL-ühendus
valid – kehtiv
variable declaration – **muutujakirjeldus**

variable declarator – **muutujakirjeldi**
variable initializer – **muutuja algväärtustaja**
variable – **muutuja**
vector – vektor
verifier – verifikaator
verify error – verifitseerimisviga
view – vaade
virtual machine – virtuaalarvuti
virtual method – virtuaalmeetod
virtual – virtuaalne
visibility modifier – nähtavuse piiritleja
visibility – nähtavus
visible – nähtav
void type – tühitüüp
while-statement – **eelkontrolliga tsüklidirektiiv**
widening primitive conversion – **laiendav primitiiviteisendus**
widening reference conversion – **laiendav osutiteisendus**
window – aken
wrapper class – **mähisklass**