Vilniaus technologijų ir verslo profesinio mokymo centras
Energetikos ir mechatronikos skyrius

AKADEMIJA.IT
INFOBALT IR TECH CITY

Nesudėtingos programinės įrangos kūrimas

Vaclav Zelenkevič

2018 m. spalio 28 d.

1. Informaciją apie dalyką
2. Java programavimo aplinka
3. Java kalbos sintaksė
4. Algoritmų ir duomenų struktūrų pagrindai
5. Java darbas su duomenimis

1. Informaciją apie dalyką

1.1. Pagrindinės temos
1.2. Literatūra, šaltiniai

1. Java programavimo aplinka
2. Java kalbos sintaksė
3. Algoritmų ir duomenų struktūrų pagrindai
4. Objektinis programavimas
5. Programų testavimas ir derinimas
6. Objektinio programavimo principų taikymas
7. Darbas su duomenimis
8. Java klasių biblioteka
9. Įvadas į funkcinį programavimą
10. Kodavimo standartai

1. *https://docs.oracle.com/javase/tutorial*
2. *www.oracle.com/technetwork/java/codeconvtoc-136057.html*
3. *https://docs.oracle.com/javase/specs*
4. *https://www.tutorialspoint.com/java*
5. *https://www.learnjavaonline.org*
6. *https://codingbat.com/java*
7. Bruce Eckel, Thinking in Java
8. Joshua Bloch, Effective Java Second Edition
9. Mark Grand, Java Enterprise Design Patterns
10. Patrick Niemeyer, Jonathan Knudsen, Learning Java
11. John Cheesman, John Daniels, UML Components

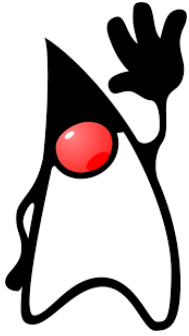
2. Java programavimo aplinka

2.1. Padiskutuokime ...
2.2. A long time ago in a galaxy far, far away ...
2.3. Padiskutuokime ...
2.4. Yra daug kalbų, bet kuri populiariausia?
2.5. Šiek tiek Java istorijos ...
2.6. Java savybės
2.7. Java redakcijos
2.8. Java JRE/JDK
2.9. JDK tools and utilities
2.10. Kaip dirba Java kodas?
2.11. Java - JVM architektūra
2.12. Programų kūrimo procesas
2.13. Integruotos kūrimo aplinkos
2.14. Netbeans
2.15. IntelliJ IDEA
2.16. Eclipse

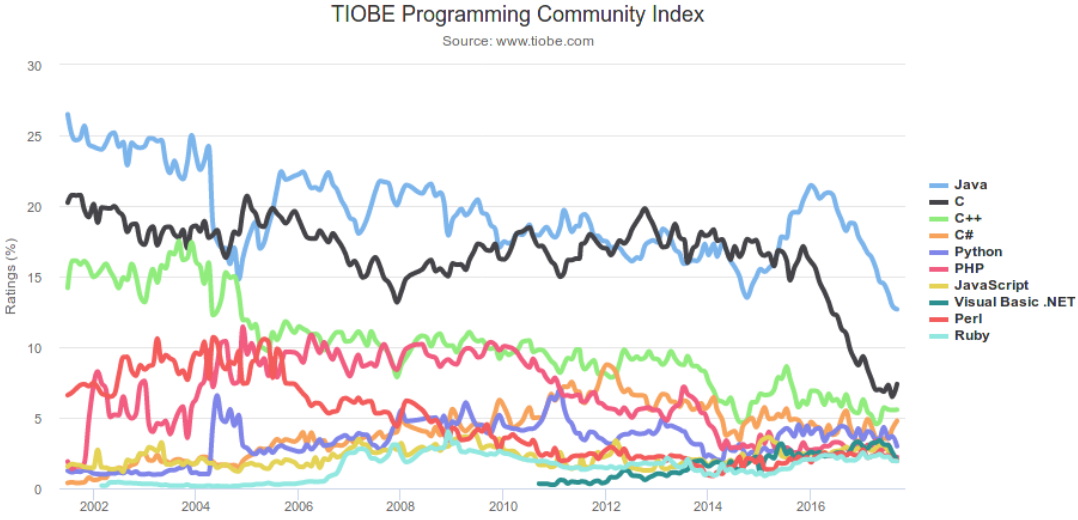
Java - kas tai?


Java


Java
Sun
microsystems
CERTIFIED
Programmer

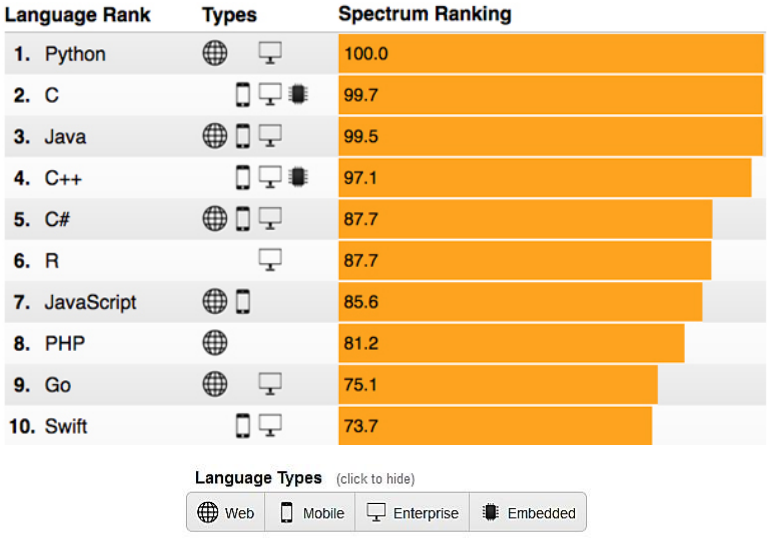


Kodėl Java?

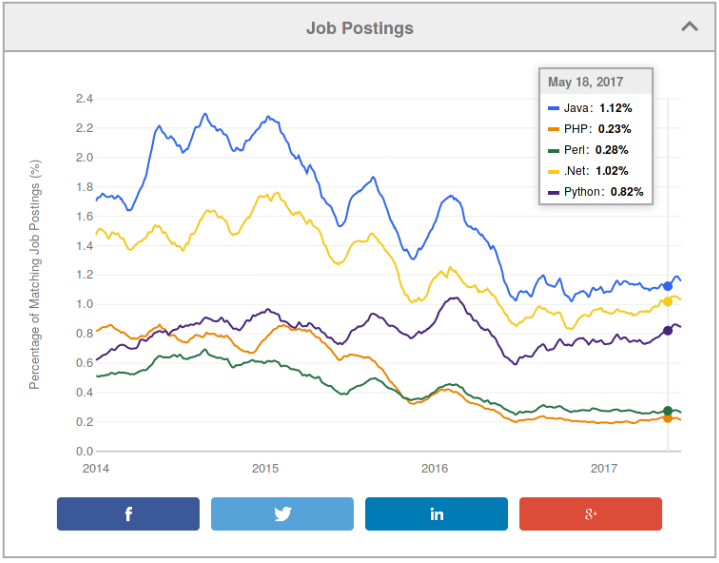

TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
Java
C
C++
C#
Python
PHP
JavaScript
Visual Basic .NET
Perl
Ruby

Sep 2017	Sep 2016	Change	Programming Language	Ratings	Change
1	1		Java	12.687%	-5.55%
2	2		C	7.382%	-3.57%
3	3		C++	5.565%	-1.09%
4	4		C#	4.779%	-0.71%
5	5		Python	2.983%	-1.32%
6	7	^	PHP	2.210%	-0.64%
7	6	v	JavaScript	2.017%	-0.91%
8	9	^	Visual Basic .NET	1.982%	-0.36%
9	10	^	Perl	1.952%	-0.38%
10	12	^	Ruby	1.933%	-0.03%
11	18	^^	R	1.816%	+0.13%
12	11	v	Delphi/Object Pascal	1.782%	-0.39%
13	13		Swift	1.765%	-0.17%
14	17	^	Visual Basic	1.751%	-0.01%
15	8	vv	Assembly language	1.639%	-0.78%

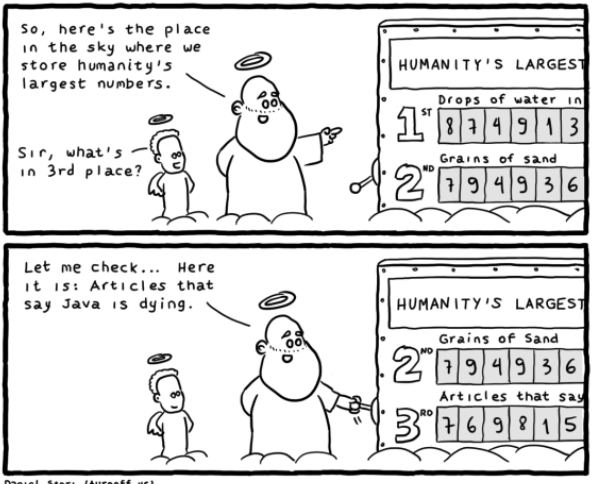
The 2018 Top Programming Languages

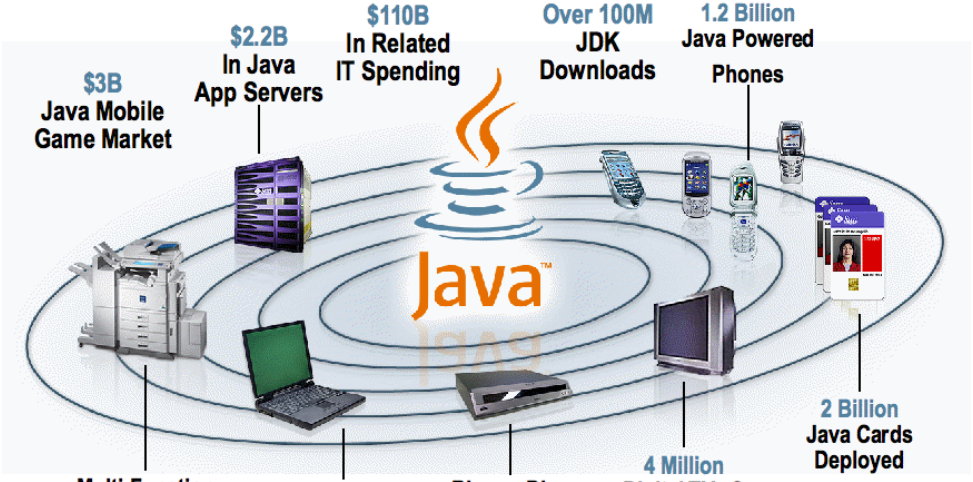


Java, PHP, Perl, .Net, and Python Job Trends



1. **Bendruomenė:**
   - Programuotojai > 9 mln.
2. **Palaiko:**
   - Oracle
   - Intel
   - HP
   - Fujitsu
   - IBM
   - RedHat
   - ARM
3. **Naudoja:**
   - Google
   - CERN
   - NASA




$3B
Java Mobile
Game Market
$2.2B
In Java
App Servers
$110B
In Related
IT Spending
Over 100M
JDK
Downloads
1.2 Billion
Java Powered
Phones
Java
2 Billion
Java Cards
Deployed
Multi-Function
Printers
700M
PCs with
Java
Blu-ray Disc
Devices
4 Million
Digital TVs &
Set-Top Boxes

Socialiniai tinklai

Korporatyviniai įrankiai

Žaidimai

IDE įrankiai

Android (žaidimai ir aplikacijos)

Write once, run anywhere!

James Arthur Gosling

Oak

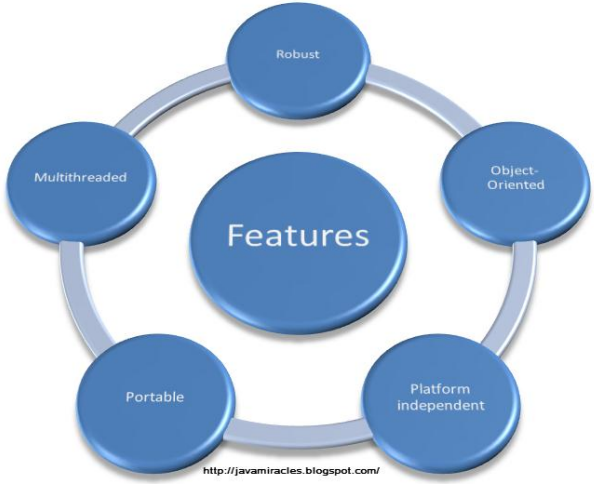
Įdomus faktas: *https://youtu.be/Ahg8OBYixL0*

Java Version/Code Name	Release Date	Important Features/Code Name
JDK Alpha and Beta	1995	Initial release
JDK1.0(OAK)	23rd Jab 1996	Initial release
JDK1.1	19th Feb 1997	Reflection ,JDBC, Inner Classes, MI
J2SE1.2(Playground)	8th Dec 1998	Collection,JIT,String memory map
J2SE 1.3(Kestrel)	8th May 2000	Java Sound,Java Indexing,JNDI
J2SE 1.4(Merlin)	6th Feb 2002	Assert,regex,exception chaining
J2SE 5.0(Tiger)	30th Sept 2004	Generic,autoboxing,enums,varargs,for each,static import
Java SE 6.0(Mustang)	11th Dec 2006	JDBC4.0,Java compiler API Annotations
Java SE7.0(Dolphin)	28th July 2011	String in switch case, resource management in exception,catching multiple exception
Java SE8.0	18th March 2014	Lambad expression,Annotation on Java type,Data and Time API

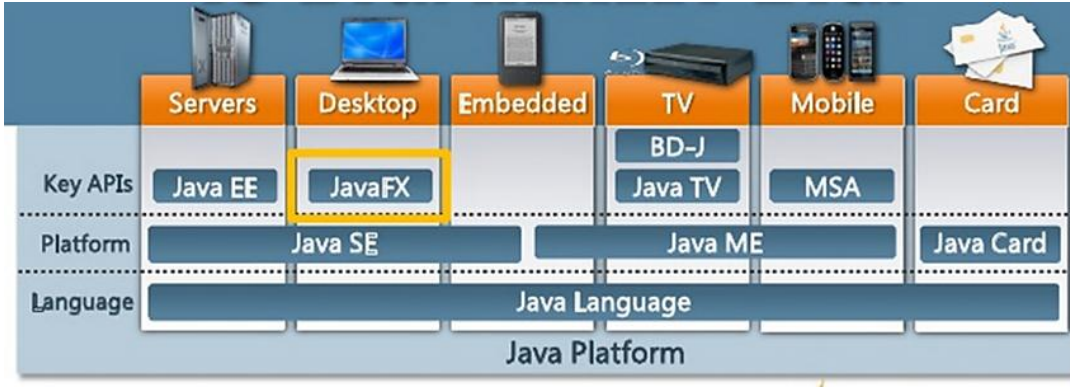
Papildoma informacija:

http://oracle.com.edgesuite.net/timeline/java/

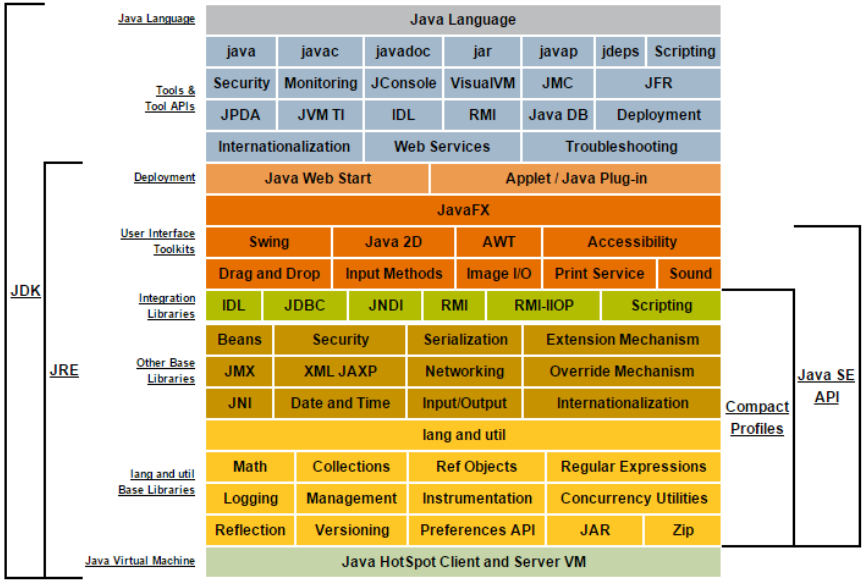
Write once, run anywhere!

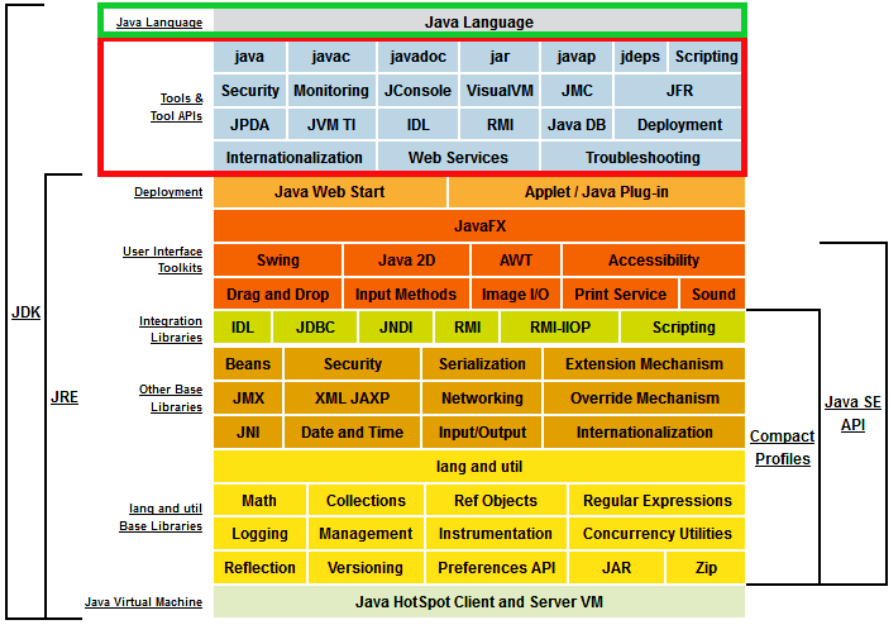


- Java SE arba kitaip - Standart Edition
- Java EE - Enterprise Edition
- Java ME - Micro Edition



Description of Java Conceptual Diagram




Java Language
Java Language
java
javac
javadoc
jar
javap
jdeps
Scripting
Security
Monitoring
JConsole
VisualVM
JMC
JFR
JPDA
JVM TI
IDL
RMI
Java DB
Deployment
Internationalization
Web Services
Troubleshooting
Tools & Tool APIs
Deployment
Java Web Start
Applet / Java Plug-in
JavaFX
User Interface Toolkits
Swing
Java 2D
AWT
Accessibility
Drag and Drop
Input Methods
Image I/O
Print Service
Sound
Integration Libraries
IDL
JDBC
JNDI
RMI
RMI-IIOP
Scripting
Other Base Libraries
Beans
Security
Serialization
Extension Mechanism
JMX
XML JAXP
Networking
Override Mechanism
JNI
Date and Time
Input/Output
Internationalization
lang and util
lang and util Base Libraries
Math
Collections
Ref Objects
Regular Expressions
Logging
Management
Instrumentation
Concurrency Utilities
Reflection
Versioning
Preferences API
JAR
Zip
Java Virtual Machine
Java HotSpot Client and Server VM
JDK
JRE
Compact Profiles
Java SE API

Java komandinės eilutės įrankių aprašymai:

- **JDK 8:**
  http://docs.oracle.com/javase/8/docs/technotes/tools/
- **JDK 9:**
  http://docs.oracle.com/javase/9/docs/technotes/tools/
- **JDK 10:**
  https://docs.oracle.com/javase/10/tools/

Pagrindiniai Java komandinės eilutės įrankiai:

- **java** - the launcher for Java applications
- **javac** - the compiler for the Java programming language
- **javadoc** - API documentation generator
- **jar** - create and manage Java Archive (JAR) files
- **javap** - Class file disassembler
- **jdb** - the Java Debugger

Java įrankis skirtas paleisti Java programas

Komandos naudojimo formatas:
$ java [options] mainclass [args...]
$ java [options] -jar filename [args]
options – komandinės eilutės parametrai
classname – paleidžiamos klasės vardas
filename – paleidžiamo Java archyvo vardas (JAR). Veikia tik su **-jar** parametru
args – argumentai pateikiami main() metodui. Komandinė eilutė.

Papildoma informacija:

https://docs.oracle.com/javase/10/tools/java.htm

Java įrankis skirtas surinkti Java programas iš išeities kodo

Komandos naudojimo formatas:
$ javac [options] [sourcefiles]
options – komandinės eilutės parametrai
sourcefiles – išeities (source) Java failai (pvz.: MyClass.java)
classes – class failai, kurie apdorojami ieškant annotacijų (pvz.: MyPackage.MyClass)
@argsfiles – options ir sourcefiles rinkiniai surašyti į failus

Papildoma informacija:

https://docs.oracle.com/javase/10/tools/javac.htm

Java įrankis skirtas generuoti HTML puslapius su Java API dokumentacija iš Java išeities (source) failų

Komandos naudojimo formatas:
$ javadoc [options] [packagenames] [sourcefiles] [@files]
packages – paketų vardai iš kurių norime generuoti dokumentaciją (pvz.: java.lang)
source-files – išeities (source) Java failai kuriuos norite dokumentuoti (pvz.: MyClass.java)
options – komandinės eilutės parametrai
@**argsfiles** – options ir sourcefiles rinkiniai surašyti į failus

Papildoma informacija:

https://docs.oracle.com/javase/10/tools/javadoc.htm

Java įrankis skirtas dirbti su Java Archive (JAR) failais

Komandos naudojimo formatas:

Sukurti JAR failą:

$ jar [OPTION...] [[–release VERSION] [-C dir] files] ...

Programa paleidžiama iš komandinės eilutės ir supakuota į JAR failą paleidžiama taip: java -jar jar-file. META-INF/MANIFEST.MF faile turi būti nurodyta Main klasė įrašant tokią eilutę: Main-Class: classname. JAR failai gali būti pasirašomi elektroniniu būdu, tvarkomi kodo versijavimo sistemose, gali užtikrinti paketo vientisumą. Šioms funkcijoms reikalinga informacija saugoma manifest faile.

Papildoma informacija:

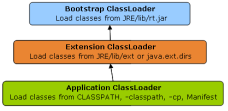
https://docs.oracle.com/javase/10/tools/jar.htm

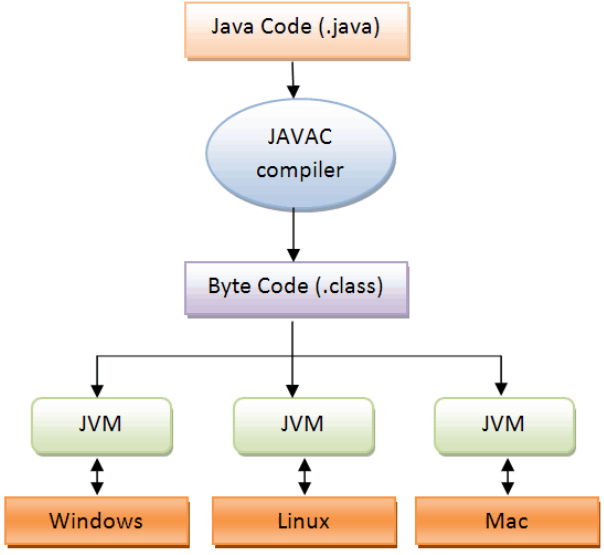
Classpath - tai stringas iš nuorodų atskirtų ";" į vartotojo sukurtas klases. Šios nuorodos reikalingos įvairiems Java įrankiams ir programoms (pvz.: javac, java ir t.t.)
Classpath nuorodos rodo į jar ir zip failus arba class failų direktoriją. Jei kalba eina apie class failus direktorijoje - užtenka nurodyti kelią iki aukščiausio lygio paketo direktorijos.
Classpath galima nustatyti kaip sistemos kintamąjį 'CLASSPATH' arba kiekvienu atveju individualiai kviečiant java įrankius nurodžius parametrą -classpath.
"*" wildcard naudojimas reiškia, kad visi direktorijoje esantys jar failai bus įtraukti į classpath, bet .class failai - ne.
Visi ***.class** failai iš foo bus įtraukti į classpath įrašius tiesiog "foo".
Java paketų, nepaisant to, kad jie atvaizduojami kaip katalogai failų sistemoje, nereikia rašyti į classpath.

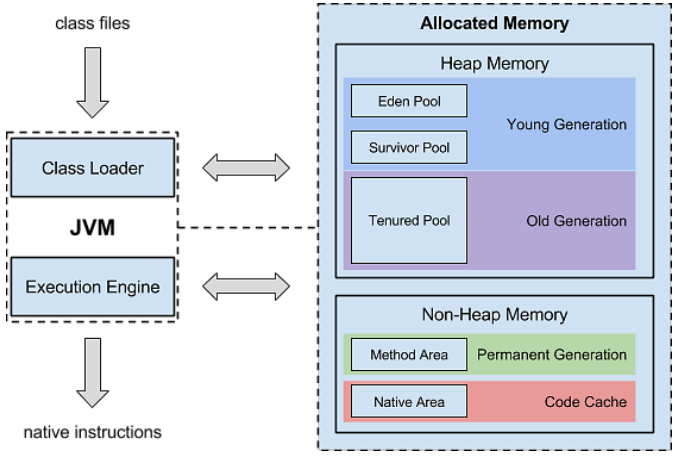
Classloading - tai mechanizmas, kuris ieško reikalingų klasių Java programos veikimui. Absoliučia dauguma atvejų naudojamas standartinis classloading mechanizmas, kurį pateikia JRE.

Klasių ieškoma iš eilės šiose vietose:

- Bootstrap klasės: veikimui būtinos runtime klasės iš rt.jar, internationalization klasės iš i18n.jar ir pan.
- Instaliuoti plėtiniai specialiuose kataloguose: JRE instaliaciniame kataloge lib/ext; Specializuotuose OS plėtinių saugyklose (nuo Java6), pvz.: /usr/jdk/packages/lib/ext naudojant Solaris.
- Classpath nurodytuose jar ir zip failuose, direktorijose (tokia tvarka, kaip jie išvardinti classpath).



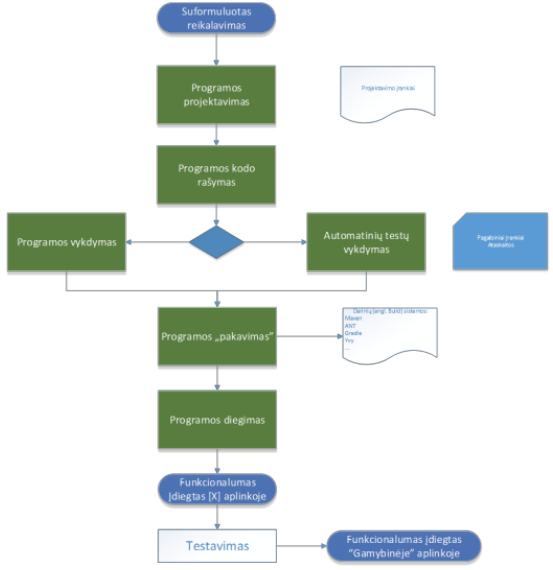

Java Code (.java)
JAVAC compiler
Byte Code (.class)
JVM
JVM
JVM
Windows
Linux
Mac



Papildoma informacija:

https://docs.oracle.com/javase/specs/jvms/se10/html/

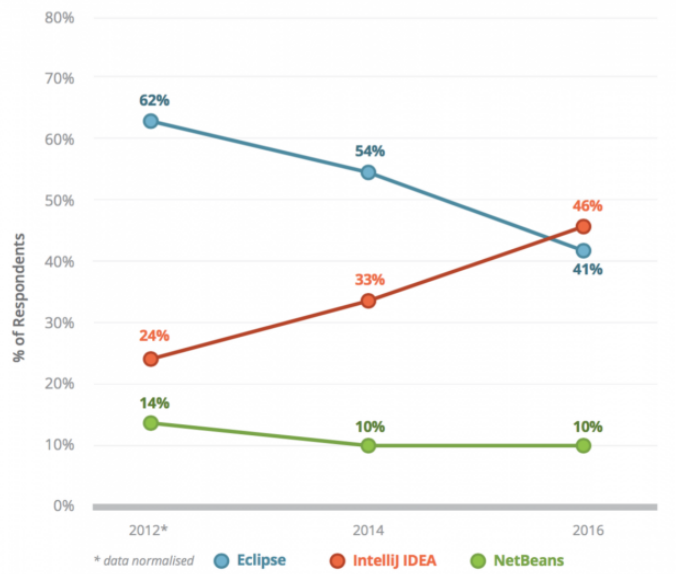
https://anturis.com/blog/java-virtual-machine-the-essential-guide/


Suformuluotas reikalavimas
Programos projektavimas
Projektavimo įrankiai
Programos kodo rašymas
Programos vykdymas
Automatinių testų vykdymas
Pagalbiniai įrankiai
Programos „pakavimas"
Programos diegimas
Funkcionalumas įdiegtas [X] aplinkoje
Testavimas
Funkcionalumas įdiegtas "Gamybineje" aplinkoje

IDE: INTEGRATED DEVELOPMENT ENVIRONMENT

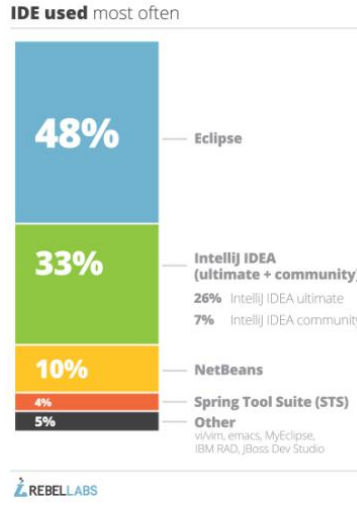
1. Padeda rašyti kodą: užuominos, šablonai, informacija apie klaidas, patarimai
2. Padeda suprasti kodą: programinių bibliotekų dokumentacija, vizualizuojami sąryšiai tarp programos elementų
3. Programos derinimas (debug, tuning)
4. Programos vykdymas
5. Automatizavimas: pakavimas, kodo kokybės kontrolė, diegimas į serverius, serverių valdymas
6. Naudotojo sąsajos projektavimas
7. Bendradarbiavimas: versijavimas, užduočių valdymas
8. Išorinių technologijų integravimas: duomenų bazių peržiūra

IDE tendencijos



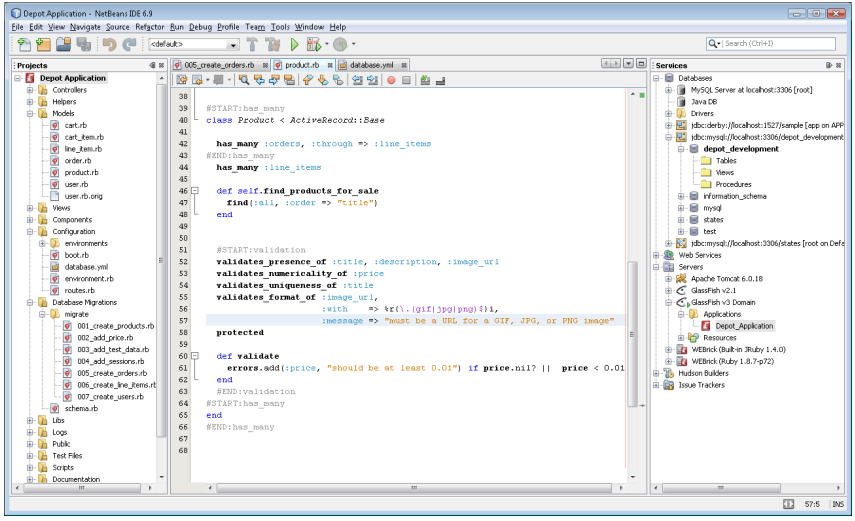
- Eclipse
- Netbeans
- Intelli Jidea
- Jdeveloper

  ...
- Kaip pasirinkti?

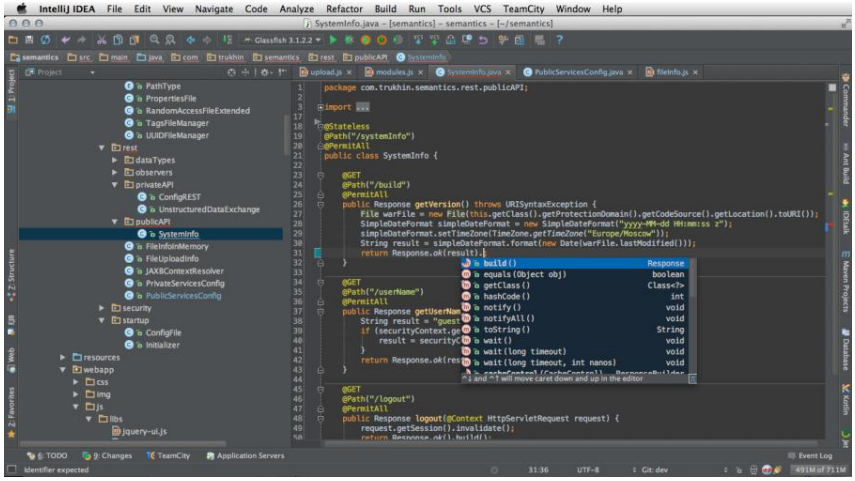


Papildoma informacija:

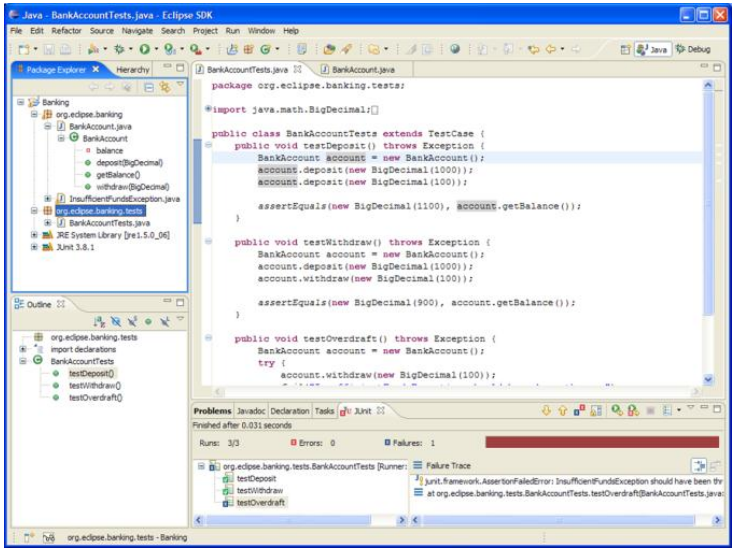
https://en.wikibooks.org/wiki/Java_Programming/Java_IDEs



Papildoma informacija:
https://netbeans.org/



Papildoma informacija:
https://www.jetbrains.com/idea/



Papildoma informacija:

https://eclipse.org

3. Java kalbos sintaksė

3.1. Paprasčiausios Java programos struktūra
3.2. Java kalbos identifikatoriai (Java Identifiers: naming)
3.3. Java kalbos raktiniai žodžiai (keywords)
3.4. Java kintamųjų tipai
3.5. Java reiškiniai: expressions
3.6. Java sakiniai: statements
3.7. Java blokai: blocks
3.8. Java duomenų tipai
3.9. Java primityvūs duomenų tipai
3.10. Java boolean tipas
3.11. Java Integer tipas
3.12. Java char tipas
3.13. Java Float-Point tipas
3.14. Java wrapper class
3.15. Java operatoriai
3.16. Java atmintis
3.17. Java duomenų tipų konvertavimas
3.18. Java String duomenų tipas
3.19. Java klasė Math
3.20. Java masyvai: arrays

```
1   /**
2   * The HelloWorld class implements an application that
3   * simply prints "Hello World!" to standard output.
4   */
5
6   class HelloWorld {
7       public static void main(String[] args) {
8           // Prints "Hello, World" to the terminal window.
9           System.out.println("Hello World!");
10      }
11  }
```

3.1.1. Programos kodas. Java HelloWorld.java

Java programą sudaro:

- Klasės apibrėžimas
- Metodo apibrėžimas
- Komentarai (taip pat JavaDoc)

Java identifikatoriai (Java Identifiers) tai Java kalbos elementai turintys vardus (pavadinimus). Vardai naudojami klasių, interfeisų, metodų, kintamųjų, laukų ir parametrų įvardinimui. **Identifikatoriai privalo tenkinti šias taisykles** (**reikia prisiminti**):

- Identifikatoriai yra jautrus raidžių registrui (**Case sensitivity**).
- Identifikatoriai turi prasidėti lotyniška raide, valiutos ($) arba pabraukimo (_) simboliu. **Pirmasis simbolis negali būti skaitmuo! $ simbolio reikia vengti**.
- Po pirmo simbolio identifikatoriai gali turėti bet kokį simbolių derinį.
- Pagrindiniai Java raktiniai žodžiai (**Java keywords**) ir **literalai true**, **false** ir **null** negali būti naudojami kaip identifikatoriai.

Pavyzdžiai:

Geri identifikatorių pvz.: *age, _book, new_look, _Table_.*

Blogi identifikatorių pvz.: *123abc, do, 1show, abstract, :e, f#.*

Java Language Keywords

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

*not used

**added in 1.2

***added in 1.4

****added in 5.0

Naming Convention:-

Do not use the Java Keywords as you variable. Normally, The variable should start with lowercase and follow by uppercase for the next word, eg. custArray, printManager

Kintamieji būna:

- *objekto* (**Instance Variables / Non-static Variables**)
- *klasės* (**Class Variables (Static Fields)**)
- *lokalūs (vietiniai)* (**Local Variables**)

Svarbu!

Visi kintamieji privalo turėti pradinę reikšmę!

Papildoma informacija:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html

Objekto kintamieji (Instance variables / Non-static Variables):

- Deklaruojami klasėje, bet ne metode, konstruktoriuje ar bloke.
- Yra sukuriami kai objektas yra sukurtas su new ir naikinami, kai objektas yra sunaikinamas.
- Gali būti inicializuojami iki arba po panaudojimo.
- Turi prieigos modifikatorius.
- Matomi visiems klasės metodams, konstruktoriams ir blokams.
- Turi numatytąją reikšmę (skaičiams = 0, loginiams = false, objekto nuorodoms = null).
- Gali būti prieinamos tiesiogiai pagal kintamojo pavadinimą, o jei metodas statinis - panaudojant pilną vardą: ObjectReference.VariableName.

Klasės kintamieji (Class Variables (Static Fields)):

- Deklaruojami klasėje, bet ne metode, konstruktoriuje ar bloke, su static modifikatoriumi.
- Klasėje gali būti tik viena kintamojo kopija, neatsižvelgiant į tai, kiek objektų buvo sukurta iš jo.
- Statiniai kintamieji naudojami retai, nebent inicializuojant konstantas. Konstantos tai kintamieji deklaruoti su public/private, final ir static modifikatoriais. Konstantos nekeičia savo pradinės reikšmės.
- Sukuriami, kai programa paleidžiama ir sunaikinami, kai programa sustabdoma.
- Matomumas yra toks pat kaip ir objekto kintamųjų.
- Turi numatytąją reikšmę (skaičiams = 0, loginiams = false, objekto nuorodoms = null).
- Kintamieji gali būti pasiekiami nurodant klasės ir kintamojo pavadinimą: ClassName.VariableName.

Lokalūs (vietiniai) kintamieji (Local Variables)

- Deklaruojami metoduose, konstruktoriuose ar blokuose.
- Sukuriami, kai kuriamas metodas, konstruktorius ar blokas ir naikinami kai išeiname iš metodo, konstruktoriaus ar bloko.
- Negali turėti prieigos modifikatorių.
- Matomi tik deklaruoto metodo, konstruktoriaus ar bloko viduje.
- Realizuojami (sukuriami) tik steko viduje.
- Neturi numatytosios reikšmės, todėl turi būti deklaruojami ir inicializuojami prieš panaudojimą.

```
1   class Variables {
2   // Lokali konstanta
3   private final String NAME = "DEMO APPLICATION";
4   // Objekto kintamasis (instance variable)
5   private String objectDescription;
6   // Statinis klases kintamasis (class variable)
7   public static int counter = 0;
8       public static void main(String[] args) {
9           // Vietinis metodo kintamasis
10          String defaultName = "World";
11          // Vietinis ciklo kintamasis
12          for (int i = 0; i < args.length; i++)
13              System.out.println("Hello, " + args[i]);
14          if (args.length == 0)
15              System.out.println("Hello, " + defaultName);
16      }
17  }
```

Reiškiniai yra sudaryti iš kintamųjų, operatorių ir paprogramių iškvietimų. Pavyzdžiai paryškinti:

1. int **cadence = 0**
2. **anArray[0] = 100**
3. System.out.println(**"Element 1 at index 0: " + anArray[0]**)
4. int **result = 1 + 2** // result is now 3
5. if (**value1 == value2**)
6. System.out.println(**"value1 == value2"**)

Reiškiniai visada grąžina reikšmę, kurios tipas priklauso nuo panaudotų kintamųjų ir operatorių.

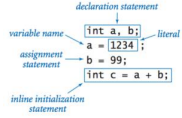
Papildoma informacija:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/expressions.html

Sakiniai Java kalboje daugmaž atitinka įprastos kalbos sakinius, kuriais kalbame. Sakiniai visada užbaigiami ženklu ; Kai kurie reiškiniai gali būti sakiniais. **Sakinių pavyzdžiai:**

1. aValue = 8933.234; // Priskyrimo sakinys
2. aValue++; // Reikšmės padidinimo sakinys
3. System.out.println(”Hello World!”); // paprogramės iškvietimo sakinys
4. Bicycle myBike = new Bicycle(); // objekto sukūrimo sakinys

Reiškiniai visada grąžina reikšmę, kurios tipas priklauso nuo panaudotų kintamųjų ir operatorių.



Papildoma informacija:

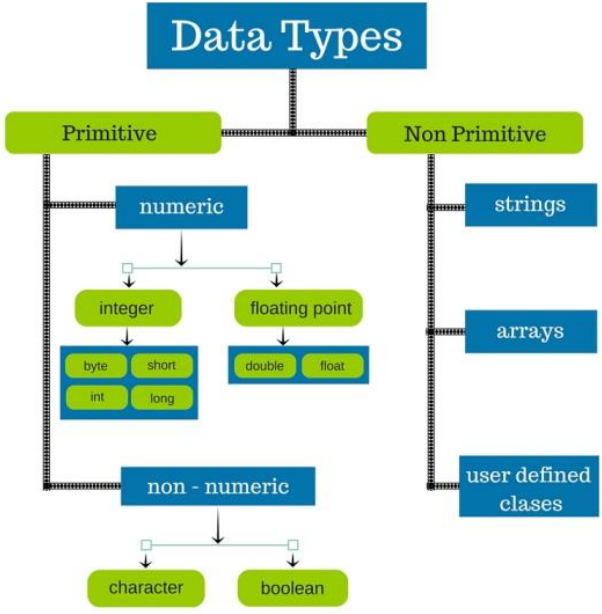
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/expressions.html

Blokas yra grupė reiškinių/sakinių parašytų tarp skliaustelių. Bloko pavyzdys: **Bloko pavyzdys:**

```
1   class Blocks {
2       public static void main(String[] args) {
3           boolean condition = true;
4           if (condition) { // begin block 1
5               System.out.println("Condition is true.");
6           } // end block one
7           else { // begin block 2
8               System.out.println("Condition is false.");
9           } // end block 2
10      }
11  }
```

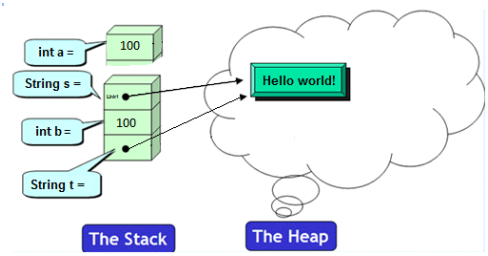
Papildoma informacija:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/expressions.html


Data Types
Primitive
Non Primitive
numeric
strings
integer
floating point
arrays
byte
short
int
long
double
float
non - numeric
user defined clases
character
boolean

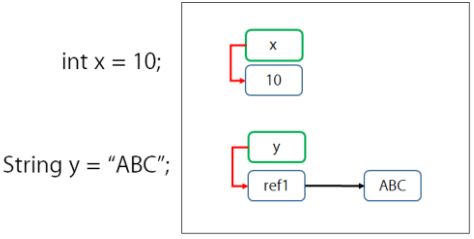
Primityvūs:

- int a = 100;
- int b = a;



Rodykliniai:

- String s = ”Hello world!”;
- String t = s;
- String u = null;



Java kalboje yra apibrėžti du skirtingi duomenų tipai:

- Primityvūs
- Rodykliniai (nuorodos tipo)

Svarbu: Java yra griežtai tipizuota kalba!

Laukai, kintamieji, metodų parametrai ir gražinamos reikšmės privalo turėti tipą!

Java primityvūs duomenų tipai. Skliaustuose nurodyta klasė kevalas (wrapper class):

Loginiai

- *boolean (Boolean)*

Simboliniai

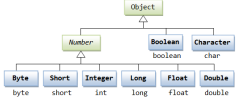
- *char (Character)*

Aritmetiniai (sveikieji)

- *byte (Byte), short (Short), int (Integer), long (Long)*

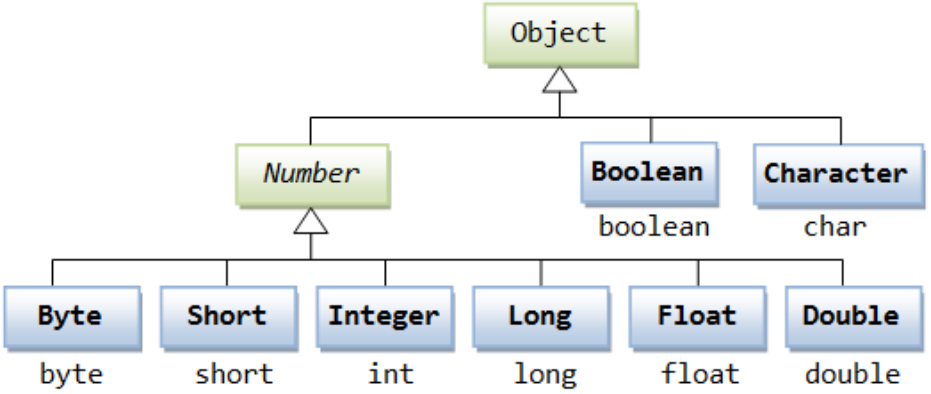
Aritmetiniai (realieji)

- *float (Float), double (Double)*



Papildoma informacija:

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html


Object
Number
Boolean
boolean
Character
char
Byte
byte
Short
short
Integer
int
Long
long
Float
float
Double
double

Primitive Types					
Type Name	Wrapper class	Value	Range	Size	Default Value
`byte`	`java.lang.Byte`	integer	−128 through +127	8-bit (1-byte)	`0`
`short`	`java.lang.Short`	integer	−32,768 through +32,767	16-bit (2-byte)	`0`
`int`	`java.lang.Integer`	integer	−2,147,483,648 through +2,147,483,647	32-bit (4-byte)	`0`
`long`	`java.lang.Long`	integer	−9,223,372,036,854,775,808 through +9,223,372,036,854,775,807	64-bit (8-byte)	`0`
`float`	`java.lang.Float`	floating point number	±1.401298E−45 through ±3.402823E+38	32-bit (4-byte)	`0.0`
`double`	`java.lang.Double`	floating point number	±4.94065645841246E−324 through ±1.79769313486232E+308	64-bit (8-byte)	`0.0`
`boolean`	`java.lang.Boolean`	Boolean	`true` or `false`	8-bit (1-byte)	`false`
`char`	`java.lang.Character`	UTF-16 code unit (BMP character or a part of a surrogate pair)	`'\u0000'` through `'\uFFFF'`	16-bit (2-byte)	`'\u0000'`

Papildoma informacija:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Pagrindinės tipo reikšmės:

- boolean brushedTeethToday = true
- boolean haveDog = false

Pagrindinės loginės operacijos:

- boolean haveSpareTime = !isBusy
- boolean canGoToPark = haveSpareTime & weatherIsGood
- boolean hadGoodTime = learnedJavaOnStepic | wentToPark
- boolean tastesGood = addedKetchup ^addedHoney

Sutrumpinta užrašymo forma:

- value &= expression; // value = value & expression
- value |= expression; // value = value | expression
- value ^= expression; // value = value ^ expression


BOOLE ORDERS LUNCH
NO, NO, YES, NO, NO, YES
YES, NO, NO, NO, YES...
Menu

Boolean tipas dažniausiai naudojamas valdymo sakiniuose.
Susideda iš dviejų loginių literalų:

- **true**
- **false**

Loginės operacijos:

- **!** - neigimas
- **&** - ir
- **|** - arba
- **^** - neigiantis arba
- **&&** - "tingus" ir
- **||** - "tingus" arba

Boolean naudojimo pavyzdžiai:

- !true // false
- true & true // true
- true | false // true
- false ^ true // true
- true ^ false // true
- false ^ false // false
- true ^ true // false
- false && true // false, second operand does not evaluate
- true || false // true, second operand does not evaluate

Integer

Type	Size	Minimum Value	Maximum Value
byte	One byte	-128	127
short	Two bytes	-32,768	32,767
int	Four bytes	-2,147,483,648	2,147,483,647
long	Eight bytes	-9,223,372,036,854,775,808	9,223,372,036,854,775,807

Integer tipo duomenys gali būti užrašomi taip:

- int decimal = 99
- int octal = 0755
- int hex = 0xFF
- int binary = 0b101
- int tenMillion = 10_000_000
- long tenBillion = 10_000_000_000L

Su Integer tipo duomenimis gali būti atliekamos aritmetinės operacijos:

- int sum = a + b;
- int diff = a - b;
- int mult = a * b;
- int div = a / b;
- int rem = a % b;
- int inc = a++ + ++b;
- int dec = --a - b--;

Perpildymas! (byte b = 127; b++;)

Su Integer tipo duomenimis gali būti atliekamos bitinės operacijos:

- int neg = a
- int and = a & b
- int or = a | b
- int xor = a ^ b
- int arithmeticShiftRight = a >> b
- int logicalShiftRight = a >>> b
- int shiftLeft = a << b

Logical Operators (Bit Level)

& | ^ ~

```
int a = 10; // 00001010 = 10
int b = 12; // 00001100 = 12
```

& AND	a	00000000000000000000000000001010	10
	b	00000000000000000000000000001100	12
	a & b	00000000000000000000000000001000	8
| OR	a	00000000000000000000000000001010	10
	b	00000000000000000000000000001100	12
	a | b	00000000000000000000000000001110	14
^ XOR	a	00000000000000000000000000001010	10
	b	00000000000000000000000000001100	12
	a ^ b	00000000000000000000000000000110	6
~ NOT	a	00000000000000000000000000001010	10
	~a	11111111111111111111111111110101	-11

Char tipo duomenys gali būti užrašomi taip:

- char literal = 'a'
- char tab = '\t'
- char lineFeed = '\n'
- char carriageReturn = '\r'
- char singleQuote = '\''
- char backslash = '\\'
- char hex = '\u03A9'

Visi char tipo duomenys saugomi UTF koduotėje.

PDF · fr · en v · d · m	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
002	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
003	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
004	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
005	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
006	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
007	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Range of Floating-Point Values in Java			
Data Type	**Width (bits)**	**Minimum Positive Value MIN_VALUE**	**Maximum Positive Value MAX_VALUE**
float	32	1.401298464324817E-45f	3.402823476638528860e+38f
double	64	4.94065645841246544e-324	1.79769313486231570e+308

Float tipo duomenys gali būti užrašomi taip:

- double simple = -1.234
- double exponential = -123.4e -2
- double hex = 0x1.Fp10
- float floatWithSuffix = 36.6f
- double doubleWithSuffix = 4d

Su Float tipo duomenimis gali būti atliekamos aritmetinės operacijos:

- double sum = a + b
- double diff = a - b
- double mult = a * b
- double div = a / b
- double rem = a % b
- double inc = a++ + ++b
- double dec = --a - b--

Realiųjų duomenų specifika:

- double positiveInfinify = 1.0 / 0.0;
- double negativeInfinify = -1.0 / 0.0;
- double nan = 0.0 / 0.0;
- boolean notEqualsItself = nan != nan;
- 0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1 != 1

Didžioji aritmetika:

- BigInteger two = BigInteger.valueOf(2)
- BigInteger powerOfTwo = two.pow(100)
- BigDecimal one = BigDecimal.valueOf(1)
- BigDecimal divisionResult = one.divide(new BigDecimal(powerOfTwo))

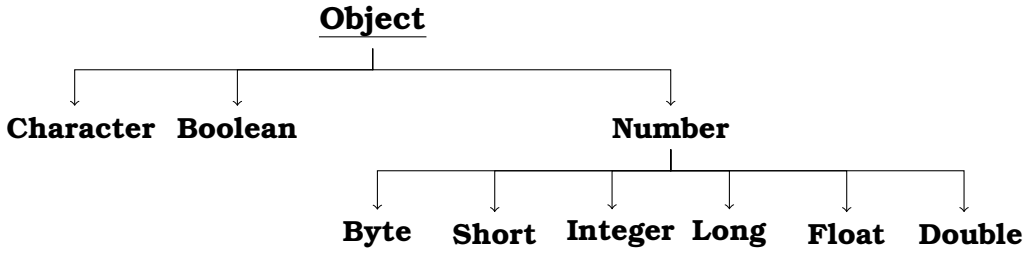
Papildoma informacija:

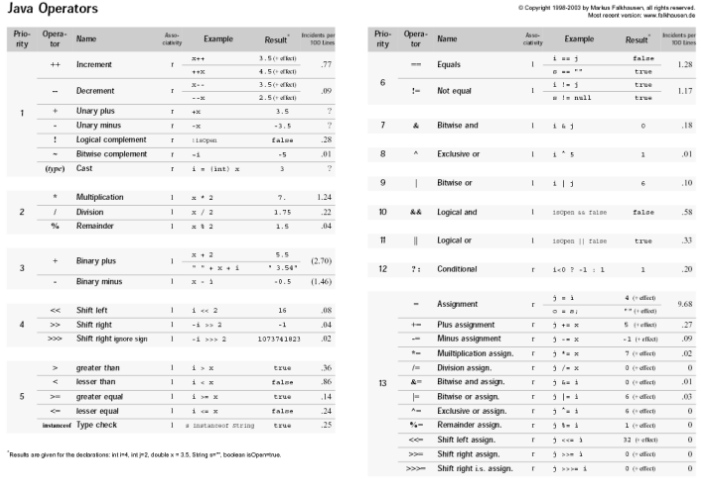
https://docs.oracle.com/javase/10/docs/api/java/math/BigDecimal.html

Klasių kevalų galimybių naudojimas:

- short maxShortValue = Short.MAX_VALUE
- int bitCount = Integer.bitCount(123)
- boolean isLetter = Character.isLetter('a')
- float floatInfinity = Float.POSITIVE_INFINITY
- double doubleNaN = Double.NaN
- boolean isNaN = Double.isNaN(doubleNaN)
- long fromString = Long.parseLong("12345")
- String fromLong = Long.toString(12345)

Data Type	Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean


Object
Character
Boolean
Number
Byte
Short
Integer
Long
Float
Double



Java Operators

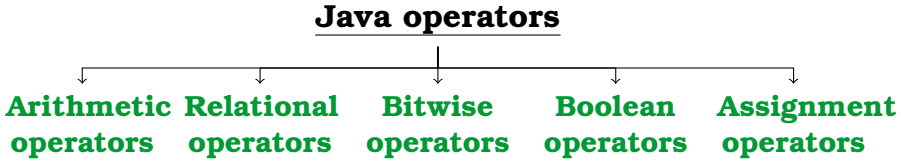
Priority	Operator	Name	Associativity	Example	Result*	Incidents per 100 Lines
1	++	Increment	r	x++	3.5 (+ effect)	.77
				++x	4.5 (+ effect)	
	--	Decrement	r	x--	3.5 (+ effect)	.09
				--x	2.5 (+ effect)	
	+	Unary plus	r	+x	3.5	?
	-	Unary minus	r	-x	-3.5	?
	!	Logical complement	r	!isOpen	false	.28
	~	Bitwise complement	r	~i	-5	.01
	(type)	Cast	r	i = (int) x	3	?
2	*	Multiplication	l	x * 2	7.	1.24
	/	Division	l	x / 2	1.75	.22
	%	Remainder	l	x % 2	1.5	.04
3	+	Binary plus	l	x + 2	5.5	(2.70)
				" " + x + i	" 3.54"	
	-	Binary minus	l	x - i	-0.5	(1.46)
4	<<	Shift left	l	i << 2	16	.08
	>>	Shift right	l	-i >> 2	-1	.04
	>>>	Shift right ignore sign	l	-i >>> 2	1073741823	.02
5	>	greater than	l	i > x	true	.36
	<	lesser than	l	i < x	false	.86
	>=	greater equal	l	i >= x	true	.14
	<=	lesser equal	l	i <= x	false	.24
	instanceof	Type check	l	x instanceof String	true	.25

*Results are given for the declarations: int i=4, int j=2, double x = 3.5, String s="", boolean isOpen=true.

Priority	Operator	Name	Associativity	Example	Result*	Incidents per 100 Lines
6	==	Equals	l	i == j	false	1.28
				s == ""	true	
	!=	Not equal	l	i != j	true	1.17
				s != null	true	
7	&	Bitwise and	l	i & j	0	.18
8	^	Exclusive or	l	i ^ 5	1	.01
9	|	Bitwise or	l	i | j	6	.10
10	&&	Logical and	l	isOpen && false	false	.58
11	||	Logical or	l	isOpen || false	true	.33
12	? :	Conditional	r	i<0 ? -1 : 1	1	.20
13	=	Assignment	r	j = i	4 (+ effect)	9.68
				o = s;	"" (+ effect)	
	+=	Plus assignment	r	j += x	5 (+ effect)	.27
	-=	Minus assignment	r	j -= x	-2 (+ effect)	.09
	*=	Multiplication assign.	r	j *= x	7 (+ effect)	.02
	/=	Division assign.	r	j /= x	0 (+ effect)	0
	&=	Bitwise and assign.	r	j &= i	0 (+ effect)	.01
	|=	Bitwise or assign.	r	j |= i	6 (+ effect)	.03
	^=	Exclusive or assign.	r	j ^= i	6 (+ effect)	0
	%=	Remainder assign.	r	j %= i	1 (+ effect)	0
	<<=	Shift left assign.	r	j <<= i	32 (+ effect)	0
	>>=	Shift right assign.	r	j >>= i	0 (+ effect)	0
	>>>=	Shift right i.s. assign.	r	j >>>= i	0 (+ effect)	0

Papildoma informacija:

http://www.falkhausen.de/de/table/operator.pdf



- Arithmetic operators: +, -, *, /, %, ++, --
- Relational operators: ==, !=, >, <, >=, <=
- Bitwise operators: &, |, ^, ~, <<, >>, >>>
- Boolean operators: &&, ||, !
- Assignment operators: =, +=, -=, *=, /=, %=, <<=, >>=, &=, ^=, |=

Arithmetic operators

Operator	Description
+	Adds values on either side of the operator
-	Subtracts right-hand operand from left-hand operand
*	Multiplies values on either side of the operator
/	Divides left-hand operand by right-hand operand
%	Divides left-hand operand by right-hand operand and returns remainder
++	Increases the value of operand by 1
--	Decreases the value of operand by 1

```
1   public class JavaOperatorsAritmetic {
2       public static void main(String[] args) {
3           int a = 10;
4           int b = 20;
5           int c = 25;
6           int d = 25;
7           System.out.println("a + b = " + (a + b));
8           System.out.println("a - b = " + (a - b));
9           System.out.println("a * b = " + (a * b));
10          System.out.println("b / a = " + (b / a));
11          System.out.println("b % a = " + (b % a));
12          System.out.println("c % a = " + (c % a));
13          System.out.println("a++ = " + (a++));
14          System.out.println("b-- = " + (a--));
15          System.out.println("d++ = " + (d++));
16          System.out.println("++d = " + (++d));
17      }
18  }
```

Relational operators

Operator	Description
==	Checks if the values of two operands are equal or not, if yes then condition becomes true
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true

```
1   public class JavaOperatorsRelational {
2       public static void main(String[] args) {
3           int a = 10;
4           int b = 20;
5           System.out.println("a == b = " + (a == b) );
6           System.out.println("a != b = " + (a != b) );
7           System.out.println("a > b = " + (a > b) );
8           System.out.println("a < b = " + (a < b) );
9           System.out.println("b >= a = " + (b >= a) );
10          System.out.println("b <= a = " + (b <= a) );
11      }
12  }
```

Bitwise logical operators

Operator	Description
&	Binary AND Operator copies a bit to the result if it exists in both operands
|	Binary OR Operator copies a bit if it exists in either operand
^	Binary XOR Operator copies the bit if it is set in one operand but not both
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand
>>>	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros

```
1   public class JavaOperatorsBitwise {
2       public static void main(String[] args) {
3           int a = 60; /*60 = 0011 1100*/
4           int b = 13; /*13 = 0000 1101*/
5           int c = 0;
6           c = a & b; /*12 = 0000 1100*/
7           System.out.println("a & b = " + c );
8           c = a | b; /*61 = 0011 1101*/
9           System.out.println("a | b = " + c );
10          c = a ^ b; /*49 = 0011 0001*/
11          System.out.println("a ^ b = " + c );
12          c = ~a; /*-61 = 1100 0011*/
13          System.out.println("~a = " + c );
14          c = a << 2; /*240 = 1111 0000*/
15          System.out.println("a << 2 = " + c );
16          c = a >> 2; /*15 = 1111*/
17          System.out.println("a >> 2 = " + c );
18          c = a >>> 2; /*15 = 0000 1111*/
19          System.out.println("a >>> 2 = " + c );
20      }
21  }
```

Boolean logical operators

Operator	Description
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true
||	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false

```
1  public class JavaOperatorsLogical {
2      public static void main(String[] args) {
3          boolean a = true;
4          boolean b = false;
5          System.out.println("a && b = " + (a&&b));
6          System.out.println("a || b = " + (a||b) );
7          System.out.println("!(a && b) = " + !(a && b));
8      }
9  }
```

Papildomai dėl & ir &&, | ir ||:

- funca () & funcb () – bus patikrinta funca () ir funcb ()
- funca () | funcb () – bus patikrinta funca () ir funcb ()
- funca () && funcb () – jei funca ()=false, funcb nebus tikrinama
- funca () || funcb () – jei funca ()=true, funcb nebus tikrinama

Assignment operators

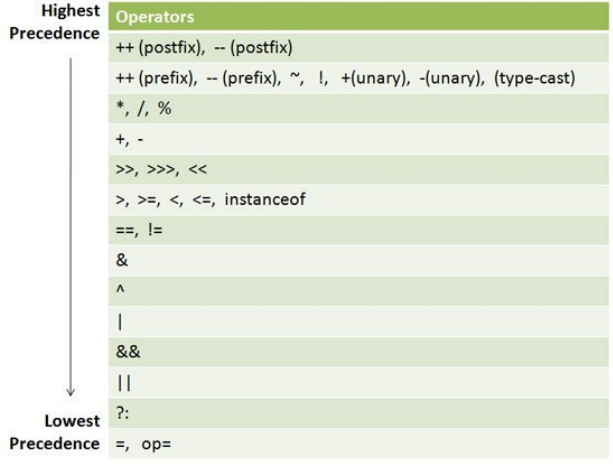
Operator	Description
=	Simple assignment operator. Assigns values from right side operands to left side operand
+=	Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand
-=	Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand
*=	Multiply AND assignment operator. It multiplies right operand with the left operand and assign the result to left operand
/=	Divide AND assignment operator. It divides left operand with the right operand and assign the result to left operand
%=	Modulus AND assignment operator. It takes modulus using two operands and assign the result to left operand

Assignment operators

Operator	Description
<<=	Left shift AND assignment operator
>>=	Bitwise AND assignment operator
&=	Right shift AND assignment operator
^=	Bitwise exclusive OR and assignment operator
|=	Bitwise inclusive OR and assignment operator

```
1   public class JavaOperatorsAssignment {
2       public static void main(String[] args) {
3           int a = 10, b = 20, c = 0;
4           c = a + b;
5           System.out.println("c = a + b = " + c );
6           c += a ;
7           System.out.println("c += a = " + c );
8           c -= a ;
9           System.out.println("c -= a = " + c );
10          c *= a ;
11          System.out.println("c *= a = " + c );
12          a = 10;
13          c = 15;
14          c /= a;
15          System.out.println("c /= a = " + c );
16          a = 10;
17          c = 15;
18          c %= a;
19          System.out.println("c %= a = " + c );
20      }
21  }
```

```
1   public class JavaOperatorsAssignment1 {
2       public static void main(String[] args) {
3           int a = 10, b = 20, c = 0;
4           c <<= 2;
5           System.out.println("c <<= 2 = " + c );
6           c >>= 2;
7           System.out.println("c >>= 2 = " + c );
8           c >>= 2;
9           System.out.println("c >>= 2 = " + c );
10          c &= a;
11          System.out.println("c &= a = " + c );
12          c ^= a;
13          System.out.println("c ^= a = " + c );
14          c |= a;
15          System.out.println("c |= a = " + c );
16      }
17  }
```



	Operators
Highest Precedence	++ (postfix), -- (postfix)
	++ (prefix), -- (prefix), ~, !, +(unary), -(unary), (type-cast)
	*, /, %
	+, -
	>>, >>>, <<
	>, >=, <, <=, instanceof
	==, !=
	&
	^
	|
	&&
	||
	?:
Lowest Precedence	=, op=

Startertutorials.com

Papildoma informacija:

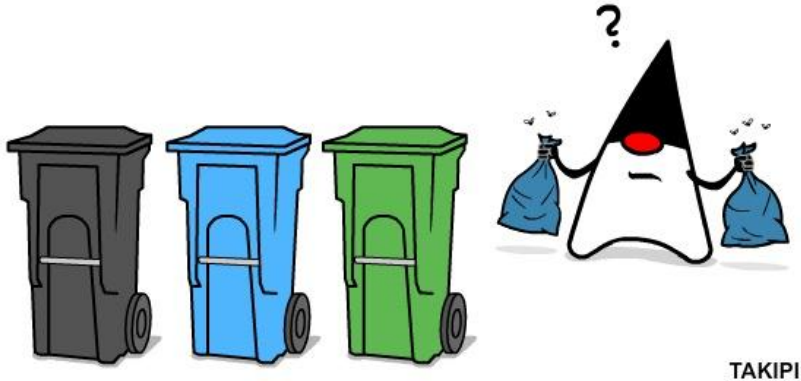
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

Deklaravimas:

- BigInteger number

Atminties išskyrimas:

- number = new BigInteger (”12345”)



Saugus duomenų tipų konvertavimas:

- byte byteValue = 123
- short shortValue = byteValue
- int intValue = shortValue
- long longValue = intValue
- char charValue = '@'
- int intFromChar = charValue
- long longFromChar = charValue
- double doubleFromFloat = 1f
- float floatFromLong = longValue
- double doubleFromInt = intValue

Nesaugus duomenų tipų konvertavimas:

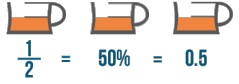
- int intValue = 1024
- byte byteValue = (byte) intValue
- double pi = 3.14
- int intFromDouble = (int) pi
- float largeFloat = 1e20f
- int intFromLargeFloat = (int) largeFloat
- double largeDouble = 1e100
- float floatFromLargeDouble = (float) largeDouble

Automatinis duomenų tipų konvertavimas:

- double doubleValue = 1d + 1f
- float floatValue = 1f * 1
- long longValue = 1L - '0'
- byte a = 1
  byte b = 2
  byte c = (byte) (a + b)

Įdomūs pavyzdžiai:

- byte a = 1
  a += 3 // a = (byte) (a + 3)
- byte b = -1
  b >>>= 7 // b = (byte) (b >>> 7)



1. String duomenų tipas sudarytas iš char masyvo ir skirtas simbolių eilutei saugoti.
2. String yra objektas, todėl gali turėti null reikšmę.
3. String nėra primityvus tipas, bet speciali klasė (objektas), su kuria Java kompiliatorius leidžia naudoti platesnį spektrą operacijų nei su kitomis klasėmis.
4. Specialūs char simboliai: \t (tab), \r (carriage return), \n (line feed), \" (double quote), \' (single quote), \\(backslash).

Pavyzdžiai:

- String hello = ”Hello World!”
- String specialChars = ”\r \n \t \” \’ \\”
- String empty = ” ”
- char [] charArray = {’a’, ’b’, ’c’}
- String string = new String (charArray)
- char [] charsFromString = string.toCharArray()
- String zeros = ”\u0000 \u0000”

1. Eilutės ilgį galima sužinoti panaudojant length() paprogramę.
2. String yra nekeičiamas (immutable), bet kokios operacijos su String realiai sukūria naują String, ir jį iš naujo priskiria.
3. Bet kur kode sutikęs užrašą tarp dvigubų kabučių pvz („Some text“) kompiliatorius sukuria String objektą, kuris saugo savyje šią simbolių eilutę.
4. String palyginimams netinka paprasti operatoriai. Tam yra yra skirtos specialios String operacijos (equals(), startsWith(), isEmpty() ir t.t.).
5. String klasė turi dviejų tipų operacijas – objekto lygio (pvz.: replace, endWith) ir klasės lygio (pvz.: String.valueOf(intVariable))

Pavyzdžiai:

- String s = ”stringIsImmutable”
- int length = s.length()
- char firstChar = s.charAt(0)
- boolean endsWithTable = s.endsWith(”table”)
- boolean containsIs = s.contains(”Is”)
- boolean referenceEquals = s1 == s2
- boolean contentEquals = s1.equals(s2)
- boolean contentEqualsIgnoreCase = s1.equalsIgnoreCase(s2)

Strings concatenation (apjungimas):

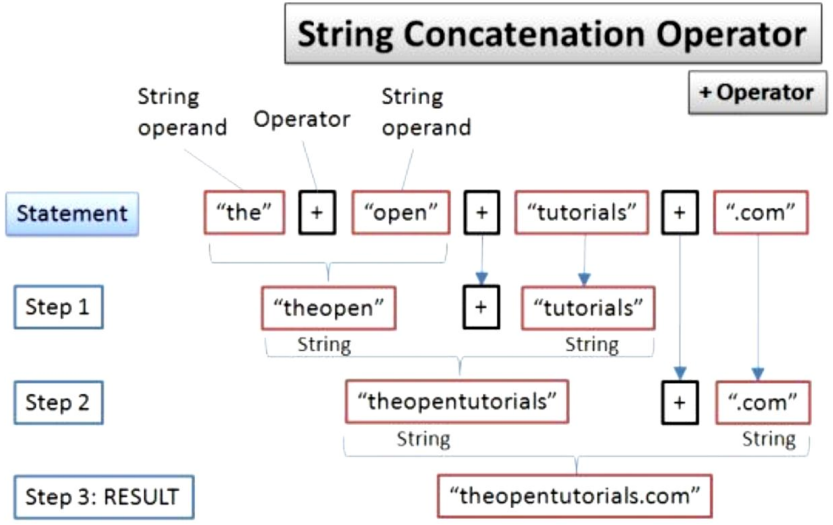
1. String str1 = „Rock”
2. String str2 = „Star”
3. String str3 = str1 + str2

Pavyzdžiai:

1. StringBuilder sb = new StringBuilder()
2. sb.append (str1)
3. sb.append (str2)
4. String str3 = sb.toString()

Papildoma informacija:

https://docs.oracle.com/javase/tutorial/java/data/strings.html


String Concatenation Operator
+ Operator
String operand
Operator
String operand
Statement
"the"
+
"open"
+
"tutorials"
+
".com"
Step 1
"theopen"
+
"tutorials"
String
String
Step 2
"theopentutorials"
+
".com"
String
String
Step 3: RESULT
"theopentutorials.com"

Klasėje **java.lang.Math** galima rasti įvairių matematinių metodų:

- double s = Math.sin(Math.PI);
- double q = Math.sqrt(16);
- double r = Math.ceil(1.01);
- int a = Math.abs(-13);
- int m = Math.max(10, 20);
- float rd = Math.round(10.9);

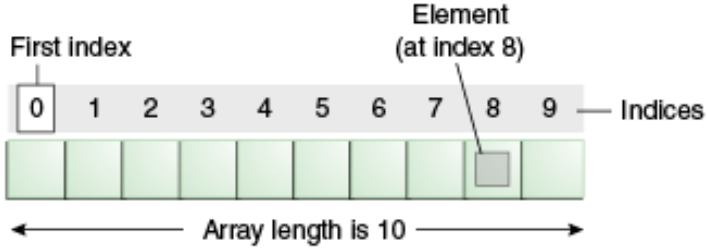
Papildoma informacija:

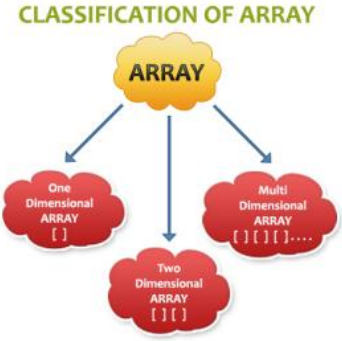
JDK 10 – *https://docs.oracle.com/javase/10/docs/api/java/lang/Math.html*

```
1   class Blocks {
2       public static void main(String[] args) {
3           double pow = Math.pow(2,2);
4           System.out.println("pow = " + pow);
5           double sqrt = Math.sqrt(4);
6           System.out.println("sqrt = " + sqrt);
7           System.out.println(Math.random());
8           System.out.println(Math.random()+3);
9           System.out.println(Math.random()*5);
10          System.out.println( (int)(Math.random()*5));
11          System.out.println(Math.random()*5+3);
12          System.out.println( (int)(Math.random()*5+3));
13          System.out.println( (int)(Math.random()*11) - 5);
14      }
15  }
```

Masyvas – vienodo tipo objektų seka vienu vardu. Masyvai gali būti vienmačiai, dvimačiai ir t.t.. Vienmatis masyvas gali būti apibrėžtas taip: **int [] numbers** arba **int numbers []**. Masyvai be elementų kiekio masyve (masyvas turi tik nuorodą, atmintis dar nepaskirta):

- int [] numbers
- String [] args
- Boolean bits []





Masyvo sukūrimas su masyvo nustatytu dydžiu:

- int [] numbers = new int [100]
- String [] args = new String [1]
- boolean [] bits = new boolean [0]

Papildoma informacija:

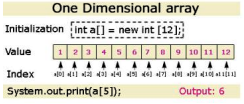
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

Masyvo užpildymas:

- int [] numbers = new int [] {1, 2, 3, 4, 5}
- boolean [] bits = new boolean [] true, false
- int [] primes = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

Darbas su masyvais:

- int [] numbers = {1, 2, 3, 4, 5}
- int arrayLength = numbers.length
- int firstNumber = numbers [0]
- int lastNumber = numbers [arrayLength - 1]
- int indexOutOfBounds = numbers [5] // error

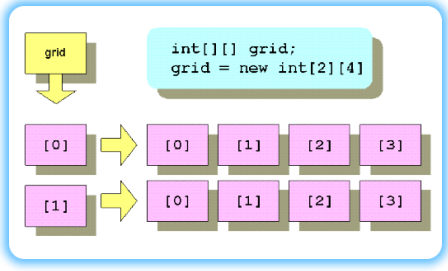


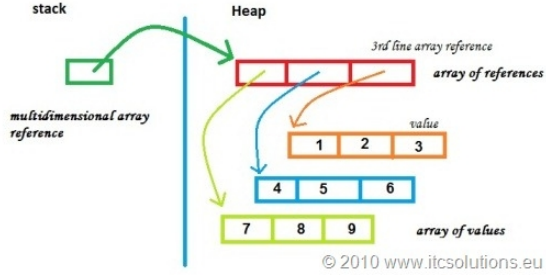
Papildoma informacija:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

Dvimačiai masyvai:

- int [][] matrix1 = new int [2][2]
- int [][] matrix2 = {{1,2}, {3,4}}
- int [] firstRow = matrix2 [0]
- int someElement = matrix2 [1][1]





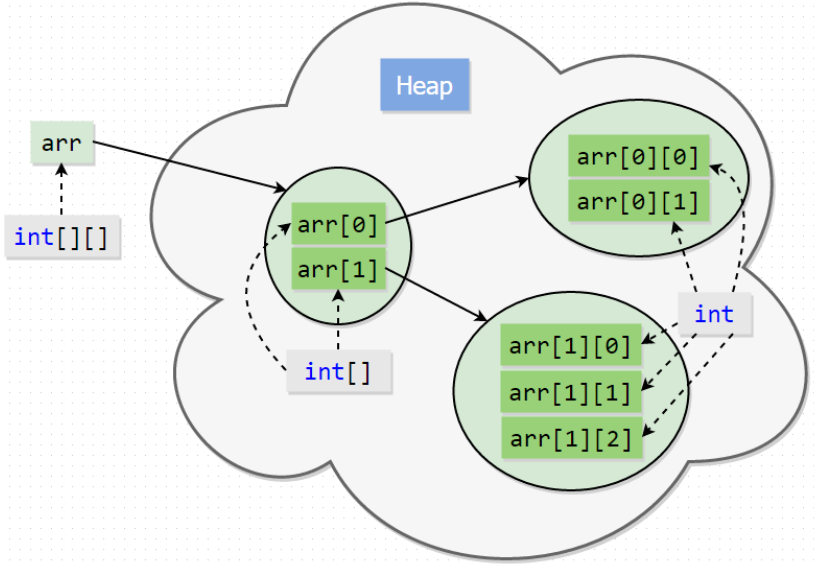
Papildoma informacija:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

Dvimačiai masyvai:

```
1   public class Arrays {
2       public static void main(String[] args) {
3           int [] OneDimensionalArray1 = { 1, 2, 3 };
4           int []  oneDimensionalArray2 = { 4, 5, 6, 7 };
5           int [] oneDimensionalArray3 = { 8, 9, 10, 11, 12 };
6           int [][] twoDimensionalArray = { OneDimensionalArray1,
7           oneDimensionalArray2, oneDimensionalArray3 };
8               for (int i = 0; i < twoDimensionalArray.length; i++) {
9                   for (int j = 0; j < twoDimensionalArray[i].length; j++) {
10                      System.out.print(twoDimensionalArray[i][j] + "\t");
11                  }
12              System.out.println();
13              }
14      }
15  }
```

Dvimačiai masyvai:



Masyvų palyginimas:

- int [] a = {1, 2, 3}
- int [] b = {4, 5, 6}
- boolean equals1 = a == b
- boolean equals2 = a.equals (b)
- boolean equals3 = Arrays.equals (a, b)
- boolean equals4 = Arrays.deepEquals (a, b)

Masyvų spausdinimas:

- int [] a = {1, 2, 3}
- System.out.println (a)
- System.out.println(Arrays.toString(a))
- System.out.println(Arrays.deepToString(a))

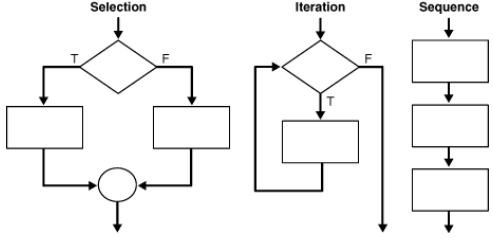
Papildoma informacija:

http://www.javabrahman.com/quick-tips/
how-to-print-array-in-java-using-arrays-tostring-deeptostring-stream-methods/

4. Algoritmų ir duomenų struktūrų pagrindai

4.1. Java sudėtingi sakiniai: control flow statements
4.2. Java: if-then
4.3. Java: if-then-else
4.4. Java: if-the, if-then-else demo
4.5. Java: switch
4.6. Java: while
4.7. Java: do while
4.8. Java: while, do while demo
4.9. Java: for
4.10. Java: break, continue, return
4.11. Java metodai: methods
4.12. Java modifikatoriai: modifiers
4.13. Kas yra algoritmas?
4.14. Algoritmai: tiesinė paieška
4.15. Algoritmai: n-min paieška
4.16. Algoritmai: n-max paieška
4.17. Algoritmai: rūšiavimas
4.18. Java sarašai: java.util.List

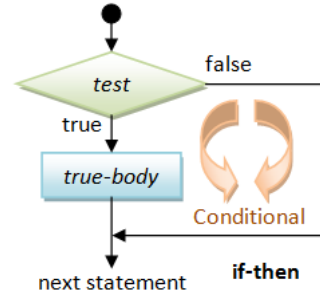
Java naudoja sprendimo priėmimo sakinius: **decision-making statements (if-then, if-then-else, switch)**, kartojimo sakinius: **looping statements (for, while, do-while)** ir šakojimo sakinius: **branching statements (break, continue, return)**. Sudėtingi sakiniai reikalingi kodo vykdymo sąlygoms, išsišakojimams ir pasikartojimams valdyti. **Java kalboje naudojami:**



Papildoma informacija:

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html

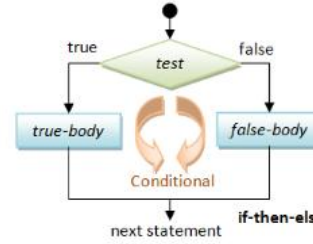
```
1   if-then
2   void applyBrakes() {
3       // the "if" clause: bicycle must be moving
4       if (isMoving){
5       // the "then" clause: decrease current speed
6           currentSpeed--;
7       }
8   }
```



Papildoma informacija:

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html

```
1   if-then-else
2   void applyBrakes() {
3         if (isMoving) {
4           currentSpeed--;
5         } else {
6           System.err.println("The bicycle has stopped!");
7         }
8   }
```

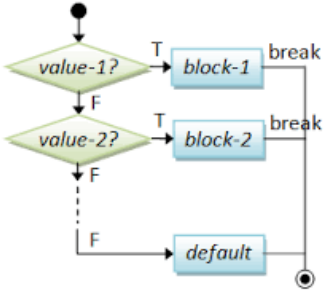


Papildoma informacija:

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html

```
1   class IfThenElseDemo {
2       public static void main(String[] args) {
3       int testscore = 76;
4       char grade;
5           if (testscore >= 90) {
6               grade = 'A';
7           } else if (testscore >= 80) {
8               grade = 'B';
9           } else if (testscore >= 70) {
10              grade = 'C';
11          } else if (testscore >= 60) {
12              grade = 'D';
13          } else {
14          grade = 'F';
15      }
16      System.out.println("Grade = " + grade);
17  }
```

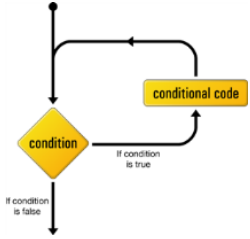
```
1   public class Switch {
2       public static void main(String[] args) {
3       int choice = 3;
4       String chosenString;
5           switch (choice) {
6               case 1: chosenString= "You choose 1!";
7               break;
8               case 2: chosenString= "You choose 2!";
9               break;
10              case 3: chosenString= "You choose 3!";
11              break;
12              case 4: chosenString= "You choose 4!";
13              break;
14              default: chosenString= "Invalid choice";
15              break;
16          }
17      System.out.println(chosenString);
18      }
19  }
```



Papildoma informacija:

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html

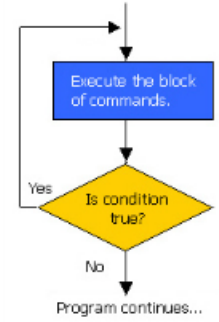
```
1   while
2   class While {
3       public static void main(String[] args){
4       int count = 1;
5           while (count < 11) {
6               System.out.println("Count is: " + count);
7               count++;
8           }
9       }
10  }
```



Papildoma informacija:

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html

```
1   do-while
2   class DoWhile {
3       public static void main(String[] args){
4       int count = 1;
5           do {
6               System.out.println("Count is: " + count);
7               count++;
8           } while (count < 11);
9       }
10  }
```



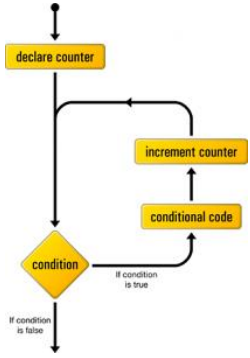
Papildoma informacija:

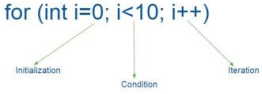
http://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html

```
class WhileDemo {
    public static void main(String[] args){
    int count = 1;
        while (count < 11) {
            System.out.println("Count is: " + count);
            count++;
        }
    }
}
```

```
class DoWhile {
    public static void main(String args[]) {
    int n = 5;
        do {
            System.out.println("Sample : " + n);
            n--;
        } while(n > 0);
    }
}
```

```
1   class For {
2       public static void main(String[] args){
3           for(int i=1; i<11; i++){
4               System.out.println("Count is: " + i);
5           }
6       }
7   }
```





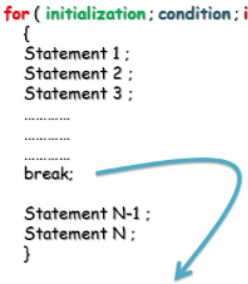
Papildoma informacija:

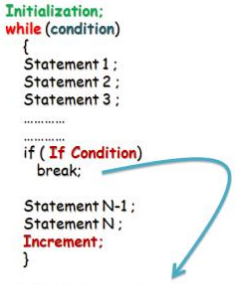
http://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html

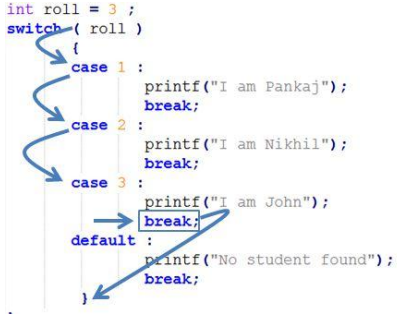
break – išeina iš paskutinio ciklo ar switch sakinio. „Labeled break“ – išeina iš pažymėto ciklo ar switch sakinio.
continue – nutraukia paskutinio ciklo tolimesnių sakinių vykdymas ir pereina prie kitos iteracijos. „Labeled continue“ – nutraukia pažymėto ciklo tolimesnių sakinių vykdymą
return – nutraukia paprogramės vykdymą, jį atiduoda ten iš kur buvo iškviesta ir gali grąžinti rezultatą

A mind is like a parachute. It doesn't work if it is not open.

Papildoma informacija:
http://docs.oracle.com/javase/tutorial/java/nutsandbolts/branch.html


for (initialization ; condition ; increment)
{
Statement 1 ;
Statement 2 ;
Statement 3 ;
……
……
……
break;
Statement N-1 ;
Statement N ;
}
OutsideStatement 1;


Initialization;
while (condition)
{
Statement 1 ;
Statement 2 ;
Statement 3 ;
……
……
if (If Condition)
break;
Statement N-1 ;
Statement N ;
Increment;
}
OutsideStatement 1;


int roll = 3 ;
switch (roll)
{
case 1 :
printf("I am Pankaj");
break;
case 2 :
printf("I am Nikhil");
break;
case 3 :
printf("I am John");
break;
default :
printf("No student found");
break;
}

```
for (int i = 0; i < arrayOfInts.length; i++) {
    if (arrayOfInts[i] == searchfor) {
        foundIt = true;
    break;
    }
}
```

```
search:
for (int i = 0; i < arrayOfInts.length; i++) {
    for (j = 0; j < arrayOfInts[i].length; j++) {
        if (arrayOfInts[i][j] == searchfor) {
        foundIt = true;
        break search;
        }
    }
}
```

```
for (int i = 0; i < max; i++) { //interested only in p's
    if (searchMe.charAt(i) !='p') {
        continue; //process p's
    numPs++;
    }
}
```

```
String searchMe = "Look for a substring in me";
String substring = "sub";
test:
for (int i = 0; i < max; i++) { //interested only in p's
    int n = substring.length();
    int j = i;
    int k = 0;
    while (n-- ! = 0) {
        if (searchMe.charAt(j++) !=substring.charAt(k++)) {
            continue test;
        }
    }
    foundIt = true;
    break test;
}
```

Return grąžina kodo vykdymą į tą vietą, kurioje buvo iškviesta paprogramė. Return gali negrąžinti jokios reikšmės arba grąžina po return sekančio reiškinio reikšmę.

```
1   public int calculateSum(int []arr) throws InvalidNumberException {
2   int sum = 0;
3   for (int value : arr ) {
4       if (value <= 0 )
5           throw new InvalidNumberException("I accept only positive numbers");
6           sum += value;
7       }
8   return sum;
9   }
```

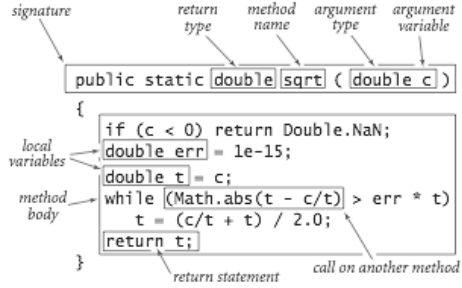
Pagrindiniai metodo elementai:

- Metodo priklausomybė – static. Neprivalomas.
- Metodo tipas - public, private, protected. Neprivalomas.
- Grąžinamos reikšmės tipas arba raktinis žodis void jei metodas negrąžina jokios reikšmės. Privalomas.
- Paprogramės pavadinimas. Privalomas.
- Parametrų sąrašas skliausteliuose () - parametrai atskiriami kableliu, kiekvienam parametrui nurodomas tipas ir pavadinimas. Jei nėra parametrų - tušti skliausteliai.
- Galimų klaidų (exception) sąrašas. Neprivalomas.
- Paprogramės kodas tarp skliaustelių. Privaloma dalis (gali būti tuščia).

```
1       private static String getGreeting(String name) {
2           if (name == null) {
3               return "Hello anonymous!";
4           } else {
5               return "Hello " + name + "!";
6           }
7       }
```

Metodo pavadinimus rekomenduojama sieti su pavadinimu veiksmo, kurį atliksime. Pavyzdžiui “nuskaityti”, “bėgti”, “pažaliuoti”. Metodai atpažįstami pagal savo “parašą” (signature and overloading), kurį sudaro pavadinimas ir parametrai.

```
1   public class DataArtist {
2       public void draw(String s) {
3           ...
4       }
5       public void draw(int i) {
6           ...
7       }
8       public void draw(double f) {
9           ...
10      }
11  }
```



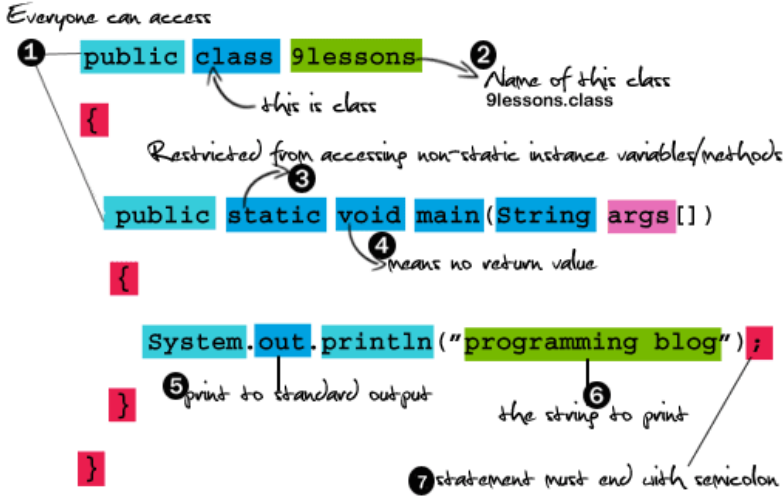
Anatomy of a static method

Metodų pavyzdžiai:

```
1   class Storage {
2       public void recordValue(long value) {};
3       public void recordValue(double value) {};
4       public void recordValue(String value) {};
5       private long convertValue(String value, int base) {};
6       public long getValue() {};
7       static public long getCounter() {};
8   }
```

Papildoma informacija:

http://docs.oracle.com/javase/tutorial/java/javaOO/methods.html



Papildoma informacija:

http://www.9lessons.info/2008/09/analysis-of-java-class.html

Modifiers, which are Java keywords, may be applied to classes, interfaces, constructors, methods, and data members.

1. Access modifiers alter visibility:
   - no modifier
   - private
   - public
   - protected
2. Non-access modifiers alter other functionalities:
   - abstract
   - final
   - native
   - strictfp
   - static
   - synchronized
   - transient
   - volatile

Access Modifiers

Modifier	Visibility
package-private	The default package-private limits access from within the package.
private	The private method is accessible from within its class. The private data member is accessible from within its class. It can be indirectly accessed through methods (i.e., getter and setter methods).
protected	The protected method is accessible from within its package, and also from outside its package by subclasses of the class containing the method. The protected data member is accessible within its package, and also from outside its package by subclasses of the class containing the data member.
public	The public modifier allows access from anywhere, even outside of the package in which it was declared. Note that interfaces are public by default.

Access Modifiers

Most Restrictive ← → Least Restrictive

Access Modifiers ->	private	Default/no-access	protected	public
Inside class	Y	Y	Y	Y
Same Package Class	N	Y	Y	Y
Same Package Sub-Class	N	Y	Y	Y
Other Package Class	N	N	N	Y
Other Package Sub-Class	N	N	Y	Y

Same rules apply for inner classes too, they are also treated as outer class properties

Non-access Modifiers

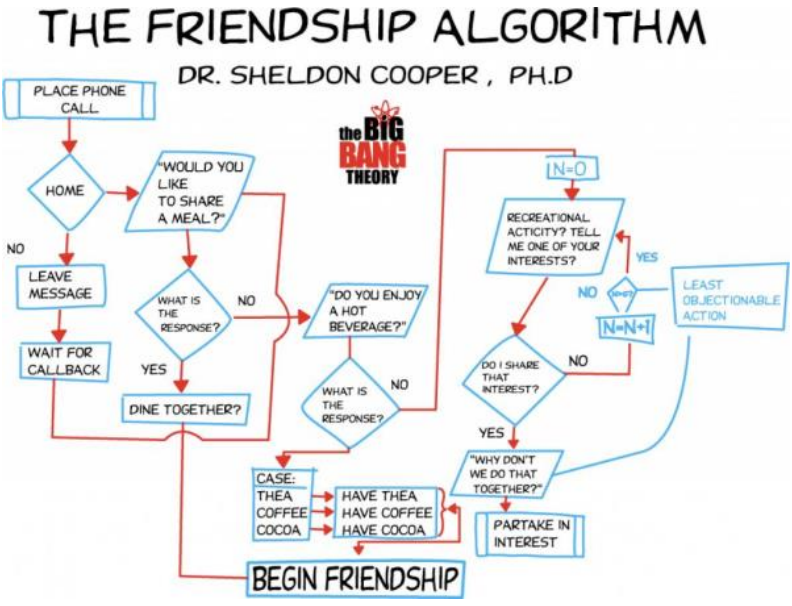
Modifier	Usage
abstract	An abstract class is a class that is declared with the keyword abstract. It cannot be simultaneously declared with final. Interfaces are abstract by default and do not have to be declared abstract. An abstract method is a method that contains only a signature and no body. If at least one method in a class is abstract, then the enclosing class is abstract. It cannot be declared final, native, private, static, or synchronized.
default	A default method, a.k.a. defender method, allows for the creation of a default method implementation in an interface.
final	A final class cannot be extended. A final method cannot be overridden. A final data member is initialized only once and cannot be changed. A data member that is declared static final is set at compile time and cannot be changed.
native	A native method is used to merge other programming languages such as C and C++ code into a Java program. It contains only a signature and no body. It cannot be used simultaneously with strictfp.

Non-access Modifiers

Modifier	Usage
static	Both static methods and static variables are accessed through the class name. They are used for the whole class and all instantiations from that class. A static data member is accessed through the class name. Only one static data member exists no matter how many instances of the class exist.
strictfp	A strictfp class will follow the IEEE 754-1985 floating-point specification for all of its floating-point operations. A strictfp method has all expressions in the method as FP-strict. Methods within interfaces cannot be declared strictfp . It cannot be used simultaneously with the native modifier.
synchronized	A synchronized method allows only one thread to execute the method block at a time, making it thread safe. Statements can also be synchronized.

Non-access Modifiers

Modifier	Usage
transient	A transient data member is not serialized when the class is serialized. It is not part of the persistent state of an object.
volatile	A volatile data member informs a thread both to get the latest value for the variable (instead of using a cached copy) and to write all updates to the variable as they occur.


THE FRIENDSHIP ALGORITHM
DR. SHELDON COOPER, PH.D
the BIG BANG THEORY
PLACE PHONE CALL
HOME
NO
LEAVE MESSAGE
WAIT FOR CALLBACK
"WOULD YOU LIKE TO SHARE A MEAL?"
WHAT IS THE RESPONSE?
NO
YES
DINE TOGETHER?
"DO YOU ENJOY A HOT BEVERAGE?"
WHAT IS THE RESPONSE?
NO
CASE:
THEA
COFFEE
COCOA
HAVE THEA
HAVE COFFEE
HAVE COCOA
BEGIN FRIENDSHIP
N=0
RECREATIONAL ACTICITY? TELL ME ONE OF YOUR INTERESTS?
YES
NO
WHAT?
LEAST OBJECTIONABLE ACTION
N=N+1
DO I SHARE THAT INTEREST?
NO
YES
"WHY DON'T WE DO THAT TOGETHER?"
PARTAKE IN INTEREST

Algoritmas (lot. algorismus > Algorithmi > arab. Al Chorezmi) – tai tam tikra veiksmų seka, kurią reikia atlikti norint pasiekti tam tikrą rezultatą. Algoritmo koncepciją iliustruoja paprasčiausias kiaušinių išvirimo receptas, kuris galėtų būti toks:

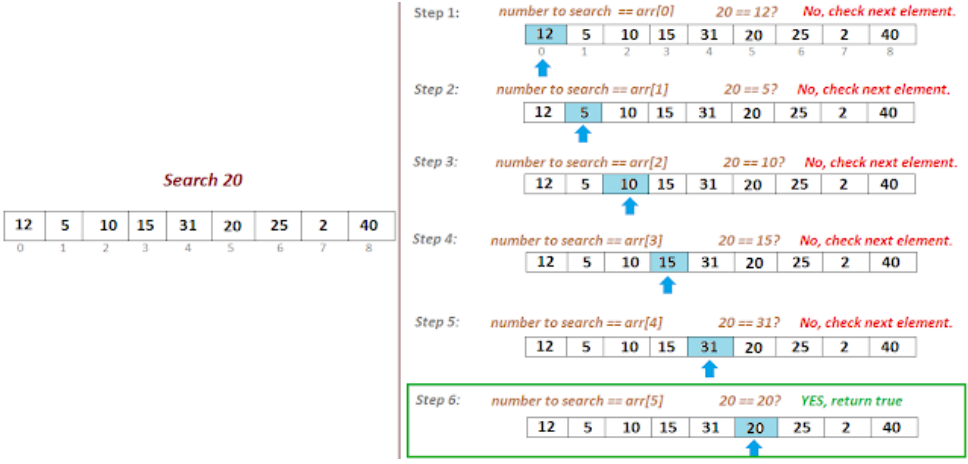
1. *paimti puodą;*
2. *į puodą įdėti n kiaušinių;*
3. *pripilti vandens;*
4. *uždėti puodą ant viryklės;*
5. *įjungti viryklę;*
6. *virti;*
7. *išjungti viryklę;*
8. *nuimti puodą;*
9. *išpilti karštą vandenį;*
10. *įpilti šaltą vandenį;*
11. *išimti kiaušinius.*

Algoritmas turi patenkinti šias sąlygas:

1. jis turi atlikti darbą;
2. jis turi būti aiškus ir nedviprasmiškas;
3. jis turi apibrėžti žingsnių seką, reikalingą darbui atlikti;
4. atliekamų žingsnių skaičius turi būti baigtinis;
5. jam atlikti turi pakakti baigtinio laiko ir baigtinių resursų.

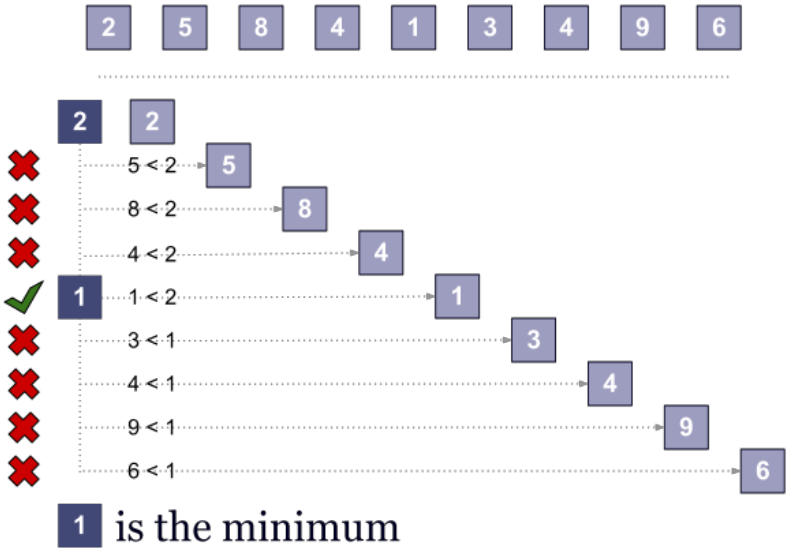
Reikalavimai **4-5** garantuoja, kad algoritmas bus baigtas baigtiniu laiku ir su baigtiniais resursais. Algoritmai, tenkinantys tik sąlygas **1-3**, vadinami daliniais (angl. partial) algoritmais, o tenkinantys visas penkias sąlygas – pilnais (angl. total) algoritmais. Algoritmams būdingos tokios bendrosios savybės:

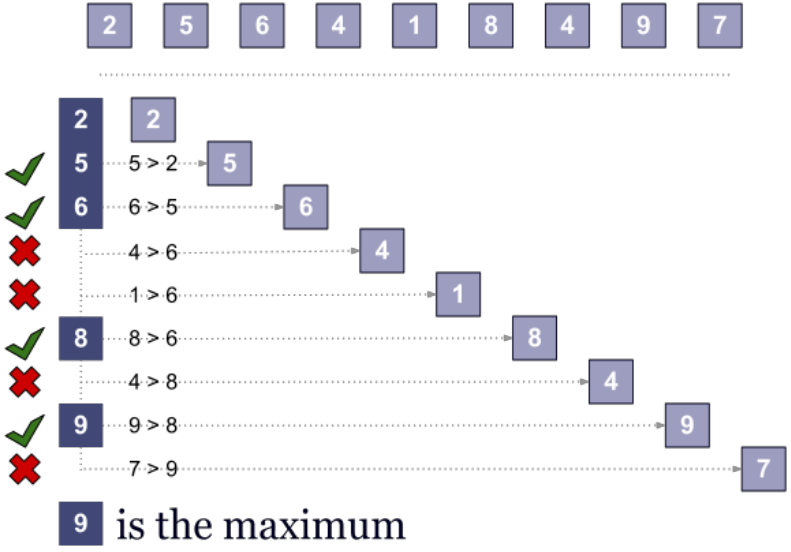
- **Diskretumas**
- **Baigtumas**
- **Rezultatyvumas**



Papildoma informacija:

https://www.youtube.com/watch?v=ltav5tp43RY


2 5 8 4 1 3 4 9 6
2
2
5 < 2
5
8 < 2
8
4 < 2
4
1
1 < 2
1
3 < 1
3
4 < 1
4
9 < 1
9
6 < 1
6
1 is the minimum


2 5 6 4 1 8 4 9 7
2
5
6
8
9
5 > 2
6 > 5
4 > 6
1 > 6
8 > 6
4 > 8
9 > 8
7 > 9
9 is the maximum

Pagrindinia rūšiavimo algoritmai:

- Bubble sort (burbuliukų metodas)
- Selection sort (atrankos metodas)
- Insertion sort (įterpimo metodas)
- Quick sort (greito rūšiavimo metodas)
- Merge sort (apjungimo metodas)

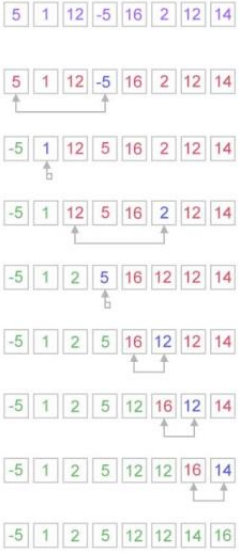
Papildoma informacija:

https://www.toptal.com/developers/sorting-algorithms/

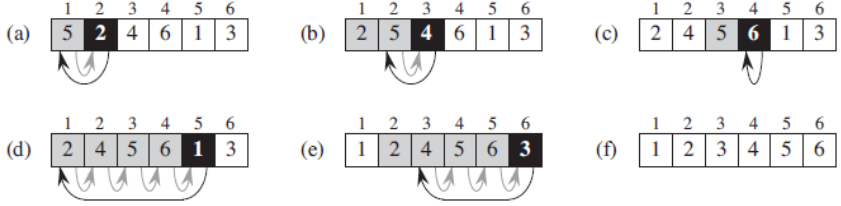
Bubble sort


5 1 12 -5 16
unsorted
5 1 12 -5 16
5 > 1, swap
1 5 12 -5 16
5 < 12, ok
1 5 12 -5 16
12 > -5, swap
1 5 -5 12 16
12 < 16, ok
1 5 -5 12 16
1 < 5, ok
1 5 -5 12 16
5 > -5, swap
1 -5 5 12 16
5 < 12, ok
1 -5 5 12 16
1 > -5, swap
-5 1 5 12 16
1 < 5, ok
-5 1 5 12 16
-5 < 1, ok
-5 1 5 12 16
sorted

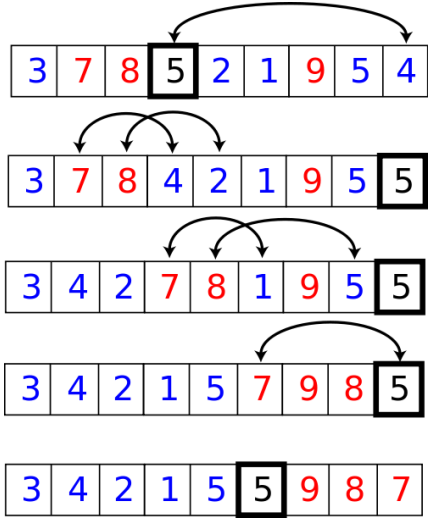
Selection sort



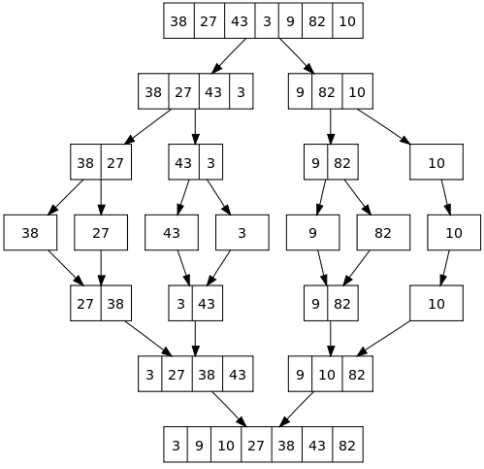
Insertion sort



Quick sort



Merge sort


38 27 43 3 9 82 10
38 27 43 3
9 82 10
38 27
43 3
9 82
10
38
27
43
3
9
82
10
27 38
3 43
9 82
10
3 27 38 43
9 10 82
3 9 10 27 38 43 82

Išbandome:

- SortingBubble.java (Bubble rūšiavimas)
- SortingSelection.java (Selection rūšiavimas)
- SortingInsertion.java (Insertion rūšiavimas)
- SortingQuick.java (Quick rūšiavimas)
- SortingMerge.java (Merge rūšiavimas)
- SortingString.java (String rūšiavimas)
- PrimeNumberOne.java (pirminio skaičiaus nustatymas)
- GreatestCommonDivisor.java (bendras didžiausias daliklis)
- LeastCommonMultiple.java (bendras mažiausias kartotinis)
- GcmLcm.java (mažiausias kartotinis/didžiausias daliklis)

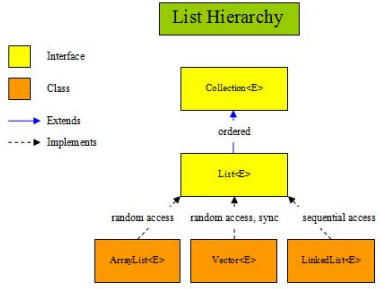
Papildoma informacija:

https://www.youtube.com/user/AlgoRythmics/videos
https://visualgo.net/en/sorting
https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html

Java kalboje yra daug klasių, skirtų darbui su sąrašais ir kitomis panašaus pobūdžio struktūromis. **Sąrašas** - tvarkingas sąrašas (*ordered List*), kuriame objektai yra saugomi tokia tvarka kokia jie buvo įtraukti į sąrašą. Prieiga prie sąrašo elementų yra galima pagal sąrašo indeksą. Aptarsime tik List tipo klases: **ArrayList**, **LinkedList**.

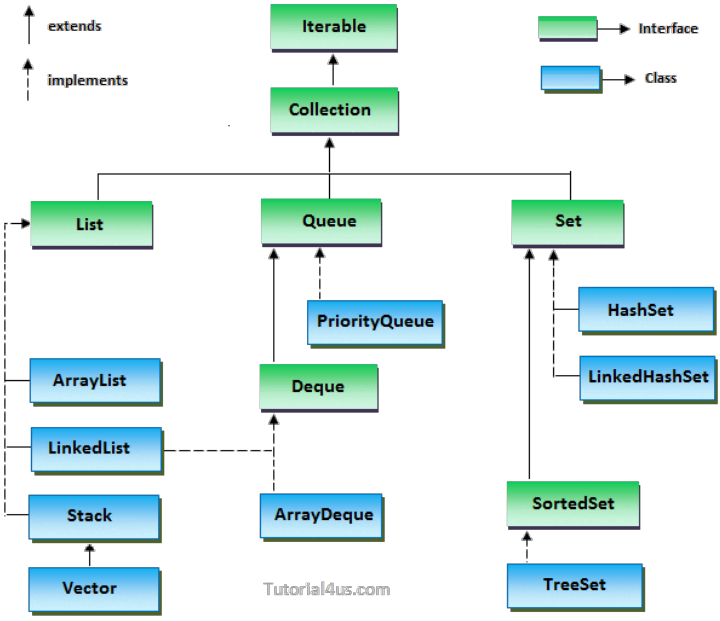
Dinaminių sąrašų (List) **skirtumai** nuo masyvų (Array):

- sąrašams nereikia iš anksto išskirti vietos, jų narių kiekis gali nuolat kisti;
- sąrašai gali būti sudaryti tik iš objektų (t.y. negali būti sudaryti iš primityvių tipų, tokių kaip int, double, boolean ir kt.)

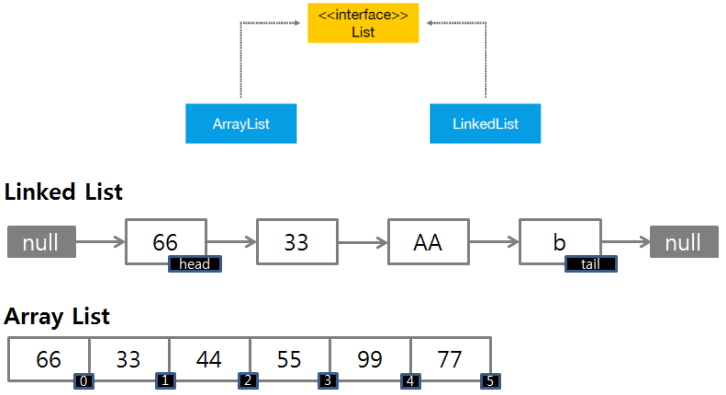


Papildoma informacija:

https://docs.oracle.com/javase/tutorial/collections/interfaces/list.html


extends
implements
Iterable
Collection
Interface
Class
List
Queue
Set
PriorityQueue
HashSet
LinkedHashSet
ArrayList
Deque
LinkedList
Stack
ArrayDeque
SortedSet
Vector
Tutorial4us.com
TreeSet

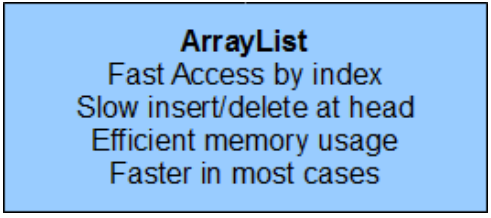
Java sąrašų (klasių) **ArrayList** ir **LinkedList** skirtumai:



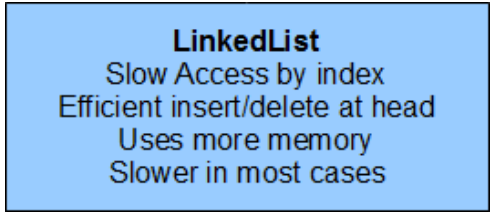
Papildoma informacija:

https://docs.oracle.com/javase/10/docs/api/java/util/ArrayList.html
https://docs.oracle.com/javase/10/docs/api/java/util/LinkedList.html

ArrayList: An implementation that stores elements in a backing array. The array's size will be automatically expanded if there isn't enough room when adding new elements into the list. It's possible to set the default size by specifying an initial capacity when creating a new ArrayList. Basically, an ArrayList offers constant time for the following operations: *size*, *isEmpty*, *get*, *set*, *iterator*, and *listIterator* amortized constant time for the *add* operation and linear time for other operations. Therefore, this implementation can be considered if we want fast, random access of the elements.



LinkedList: An implementation that stores elements in a doubly linked list data structure. It offers constant time for adding and removing elements at the end of the list; and linear time for operations at other positions in the list. Therefore, we can consider using a *LinkedList* if fast adding and removing elements at the end of the list is required.



List tipo objektų kūrimas:

- ArrayList al = new ArrayList();
- List al = new ArrayList();
- LinkedList ll = new LinkedList();
- List ll = new LinkedList();

Pagrindiniai metodai:

- add() – įterpti elementą;
- contains() – patikrinti, ar sąraše yra elementas;
- get() – paimti elementą iš sąrašo;
- indexOf() – gauti elemento indeksą sąraše;
- remove() – pašalinti elementą;
- size() – sąrašo dydis;
- toArray() – surašyti elementus į masyvą.

ArrayList sukūrimas, užpildymas, išvedimas (gavimas pagal indeksą), paieška, papildymas.

```
1   ArrayList<String> al = new ArrayList<String>();
2   al.add("JAVA");
3   al.add("C++");
4   al.add("PERL");
5   al.add("PHP");
6   System.out.println(al);
7   System.out.println("Element at index 1: "+al.get(1));
8   System.out.println("Does list contains JAVA?" + al.contains("JAVA"));
9   al.add(2,"PLAY");
10  System.out.println(al);
11  System.out.println("Is arraylist empty? "+al.isEmpty());
12  System.out.println("Index of PERL is "+al.indexOf("PERL"));
13  System.out.println("Size of the arraylist is: "+al.size());
```

ArrayList skaitymas naudojant **Iterator**.

```
1  ArrayList<String> arrl = new ArrayList<String>();
2  arrl.add("First");
3  arrl.add("Second");
4  arrl.add("Third");
5  arrl.add("Random");
6  Iterator<String> itr = arrl.iterator();
7  while (itr.hasNext()) {
8  System.out.println(itr.next());
9  }
```

ArrayList kopijavimas / klonavimas.

```
1  ArrayList<String> arrl = new ArrayList<String>();
2  arrl.add("First");
3  arrl.add("Second");
4  arrl.add("Third");
5  arrl.add("Random");
6  System.out.println("Actual ArrayList:" + arrl);
7  ArrayList<String> copy = (ArrayList<String>) arrl.clone();
8  System.out.println("Cloned ArrayList:" + copy);
```

Vieno **ArrayList** sąrašo įtraukimas (kopijavimas) į kitą.

```
1   ArrayList<String> arrl = new ArrayList<String>();
2   arrl.add("First");
3   arrl.add("Second");
4   arrl.add("Third");
5   arrl.add("Random");
6   System.out.println("Actual ArrayList:"+arrl);
7   List<String> list = new ArrayList<String>();
8   list.add("one");
9   list.add("two");
10  arrl.addAll(list);
11  System.out.println("After Copy: "+arrl);
```

ArrayList sąrašo išvalymas.

```
1  ArrayList<String> arrl = new ArrayList<String>();
2  arrl.add("First");
3  arrl.add("Second");
4  arrl.add("Third");
5  arrl.add("Random");
6  System.out.println("Actual ArrayList:"+arrl);
7  arrl.clear();
8  System.out.println("After clear ArrayList:"+arrl);
```

ArrayList perkėlimas į array.

```
1   ArrayList<String> arrl = new ArrayList<String>();
2   arrl.add("First");
3   arrl.add("Second");
4   arrl.add("Third");
5   arrl.add("Random");
6   System.out.println("Actual ArrayList:"+arrl);
7   String[] strArr = new String[arrl.size()];
8   arrl.toArray(strArr);
9   System.out.println("Created Array content:");
10  for(String str:strArr){
11      System.out.println(str);
12  }
```

Reikiamo sąrašo formavimas iš esamo **ArrayList**.

```
1  ArrayList<String> arrl = new ArrayList<String>();
2  arrl.add("First");
3  arrl.add("Second");
4  arrl.add("Third");
5  arrl.add("Random");
6  arrl.add("Click");
7  System.out.println("Actual ArrayList:"+arrl);
8  List<String> list = arrl.subList(2, 4);
9  System.out.println("Sub List: "+list);
```

ArrayList elementų sąrašo apvertimas (reversed ArrayList).

```
1   ArrayList<String> list = new ArrayList<String>();
2   list.add("Java");
3   list.add("Cric");
4   list.add("Play");
5   list.add("Watch");
6   list.add("Glass");
7   Collections.reverse(list);
8   System.out.println("Results after reverse operation:");
9   for(String str: list){
10      System.out.println(str);
11  }
```

ArrayList elementų sąrašo maišymas (shuffle).

```
1   ArrayList<String> list = new ArrayList<String>();
2   list.add("Java");
3   list.add("Cric");
4   list.add("Play");
5   list.add("Watch");
6   list.add("Glass");
7   list.add("Movie");
8   list.add("Girl");
9   Collections.shuffle(list);
10  System.out.println("Results after shuffle operation:");
11  for(String str: list){
12      System.out.println(str);
13  }
14  Collections.shuffle(list);
15  System.out.println("Results after shuffle operation:");
16  for(String str: list){
17      System.out.println(str);
18  }
```

ArrayList dviejų sąrašo elementų vietų pakeitimas (swap).

```
1   ArrayList<String> list = new ArrayList<String>();
2   list.add("Java");
3   list.add("Cric");
4   list.add("Play");
5   list.add("Watch");
6   list.add("Glass");
7   list.add("Movie");
8   list.add("Girl");
9   System.out.println("Results before swap operation:");
10  for(String str: list){
11      System.out.println(str);
12  }
13  Collections.swap(list, 2, 5);
14  System.out.println("Results after swap operation:");
15  for(String str: list){
16      System.out.println(str);
17  }
```

ArrayList sąrašo elementų rūšiavimas.

```
1   ArrayList<String> listofcountries = new ArrayList<String>();
2   listofcountries.add("India");
3   listofcountries.add("US");
4   listofcountries.add("China");
5   listofcountries.add("Denmark");
6   System.out.println("Before Sorting:");
7   for(String counter: listofcountries){
8       System.out.println(counter);
9   }
10  Collections.sort(listofcountries);
11  System.out.println("After Sorting:");
12  for(String counter: listofcountries){
13      System.out.println(counter);
14  }
```

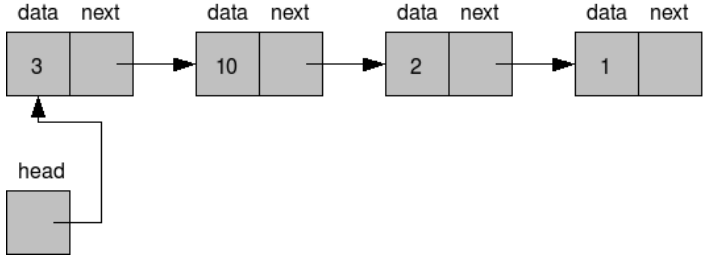
Išbandome:

- ArrayListClone.java
- ArrayListGetSubList.java
- ArrayListIterator.java
- ArrayListSort.java
- ArrayListSwapTwoElements.java
- ArrayListToArray.java
- BasicArrayList.java
- ClearArrayList.java
- CollectionListCopyToArrayList.java
- ElementCheck.java
- ReverseArrayList.java
- ShuffleArrayList.java

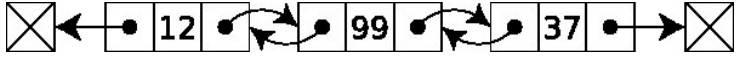
Išbandome:

- BasicLinkedList.java
- ClearLinkedList.java
- CollectionsListCopyToLinkedList.java
- LinkedListAddFirstElement.java
- LinkedListAddLastElement.java
- LinkedListClone.java
- LinkedListElementCheck.java
- LinkedListGetSubList.java
- LinkedListIterator.java
- LinkedListRemoveElements.java
- LinkedListReverse.java
- LinkedListShuffle.java
- LinkedListSort.java
- LinkedListSwapTwoElements.java
- LinkedListToArray.java

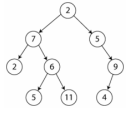
Paprastas sąrašas: tik į vieną pusę, galima įterpti elementus tik į vieną pusę, sunku pašalinti elementą – reikia perbėgti iš naujo. LinkedList single link implementation:



Dvipusis sąrašas: galima pereiti nuo bet kurio galo bet kuria kryptimi, galima įterpti elementą į bet kurią vietą, užima daugiau atminties, bet labai paprastas pašalinimas, nereikalingo elemento. LinkedList double link implementation:

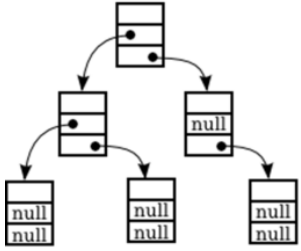


Medžiai yra hierarchinės duomenų struktūros, juose tarp medžio elementų egzistuoja „**tėvų - vaikų**“ santykiai. Kiekvienas elementas yra susietas su vienu ar daugiau elementų. Medžio elementai yra vadinami medžio **viršūnėmis**. Elementas, kuris neturi tėvo, vadinamas **šaknimi** ar **šakniniu elementu**. Elementai neturintys vaikų vadinami **lapais**. **Viršūnės**, kurios nėra lapai, dar vadinamos vidinėmis viršūnėmis. Medžio aukščiu vadinamas atstumas nuo šaknies iki toliausiai esančio lapo. Medžio viršūnės lygis nusako jos eilės tvarką skaičiuojant nuo šaknies (šaknis yra 0 lygio, jos vaikas 1...). Dvejetainis medis (**binary tree**) – tai toks medis, kurio kiekviena viršūnė turi ne daugiau kaip 2 vaikus, kurie vadinami dešiniuoju ir kairiuoju medžio pomedžiu. Dvejetainis paieškos medis (**binary search tree**) – tai dvejetainis medis, kuris surūšiuotas pagal reikšmes jo viršūnėse.



Dvejetainis medis:

- Visada yra išrūšiuotas pagal kokį nors požymį
- Labai greita paieška
- Labai sudėtingas įterpimas, gali tekti perdaryti medį



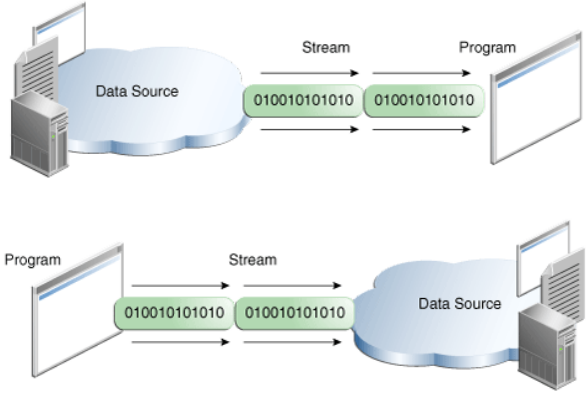
Papildoma informacija:

https://youtu.be/qH6yxkw0u78?list=PL2_aWCzGMAwI3W_JlcBbtYTwiQSsOTa6P
https://youtu.be/H5JubkIy_p8?list=PL2_aWCzGMAwI3W_JlcBbtYTwiQSsOTa6P
https://youtu.be/pYT9F8_LFTM?list=PL2_aWCzGMAwI3W_JlcBbtYTwiQSsOTa6P

5. Java darbas su duomenimis

5.1. Java I/O

An I/O Stream represents an input source or an output destination. A stream can represent many different kinds of sources and destinations, including disk files, devices, other programs, and memory arrays. Some streams simply pass on data; others manipulate and transform the data in useful ways.



Most of the classes covered in the I/O Streams section are in the java.io package. Most of the classes covered in the File I/O section are in the java.nio.file package.

java.io provides for system input and output through data streams, serialization and the file system.

java.nio defines buffers, which are containers for data, and provides an overview of the other NIO packages.

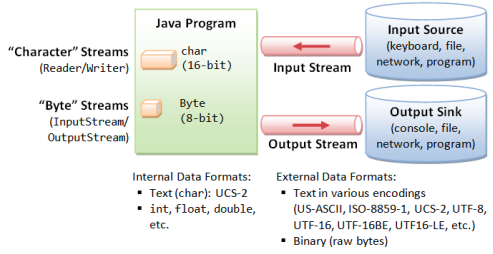
Additional information:

https://docs.oracle.com/javase/8/docs/api/java/io/package-summary.html

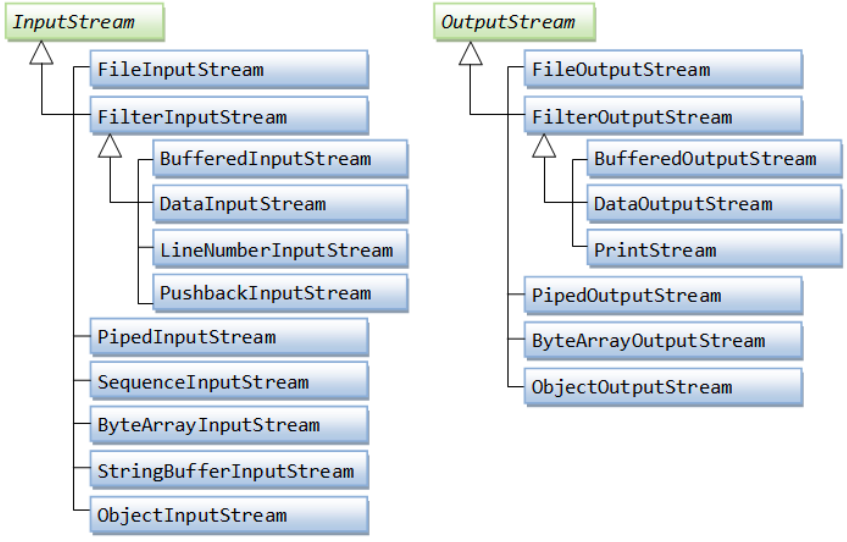
https://docs.oracle.com/javase/8/docs/api/java/nio/package-summary.html

Byte Streams

Programs use byte streams to perform input and output of 8-bit bytes. All byte stream classes are descended from **InputStream** and **OutputStream**. There are many byte stream classes. To demonstrate how byte streams work, we'll focus on the file I/O byte streams, **FileInputStream** and **FileOutputStream**. Other kinds of byte streams are used in much the same way; they differ mainly in the way they are constructed.

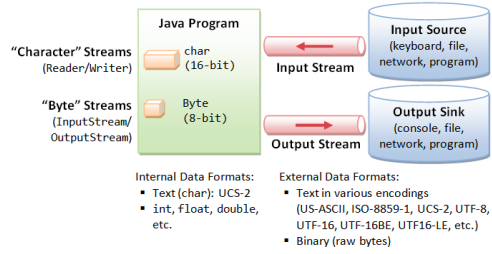


Byte Streams

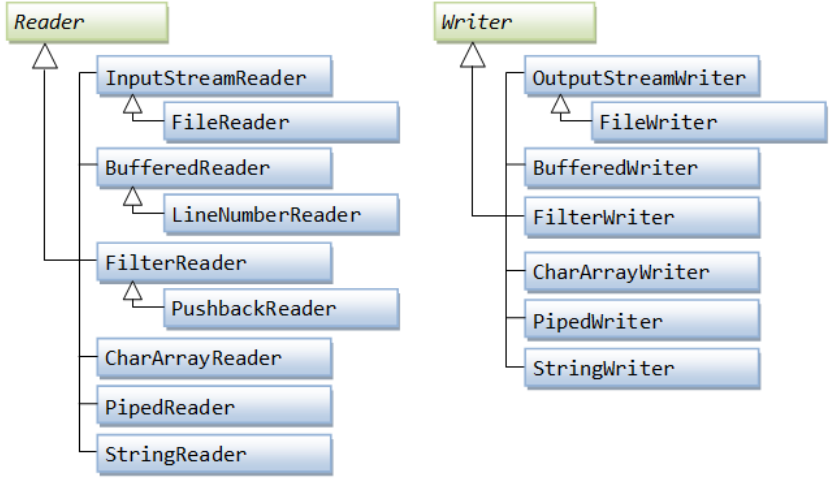


Character Streams

The Java platform stores character values using Unicode conventions. Character stream I/O automatically translates this internal format to and from the local character set. In Western locales, the local character set is usually an 8-bit superset of ASCII. All character stream classes are descended from **Reader** and **Writer**. As with byte streams, there are character stream classes that specialize in file I/O: **FileReader** and **FileWriter**.

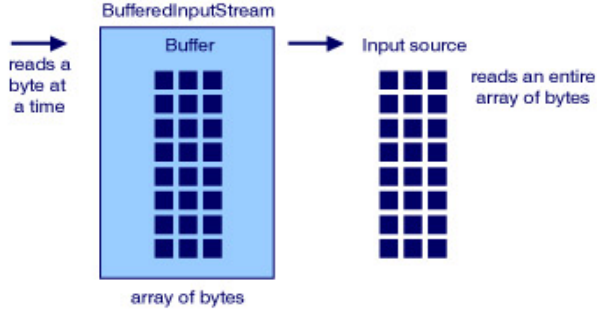


Character Streams



Buffered Streams

There are four buffered stream classes used to wrap unbuffered streams: **BufferedInputStream** and **BufferedOutputStream** create buffered byte streams, while **BufferedReader** and **BufferedWriter** create buffered character streams.



Data Streams

Data streams support binary I/O of primitive data type values (boolean, char, byte, short, int, long, float, and double) as well as String values. All data streams implement either the **DataInput** interface or the **DataOutput** interface. This section focuses on the most widely-used implementations of these interfaces, **DataInputStream** and **DataOutputStream**.