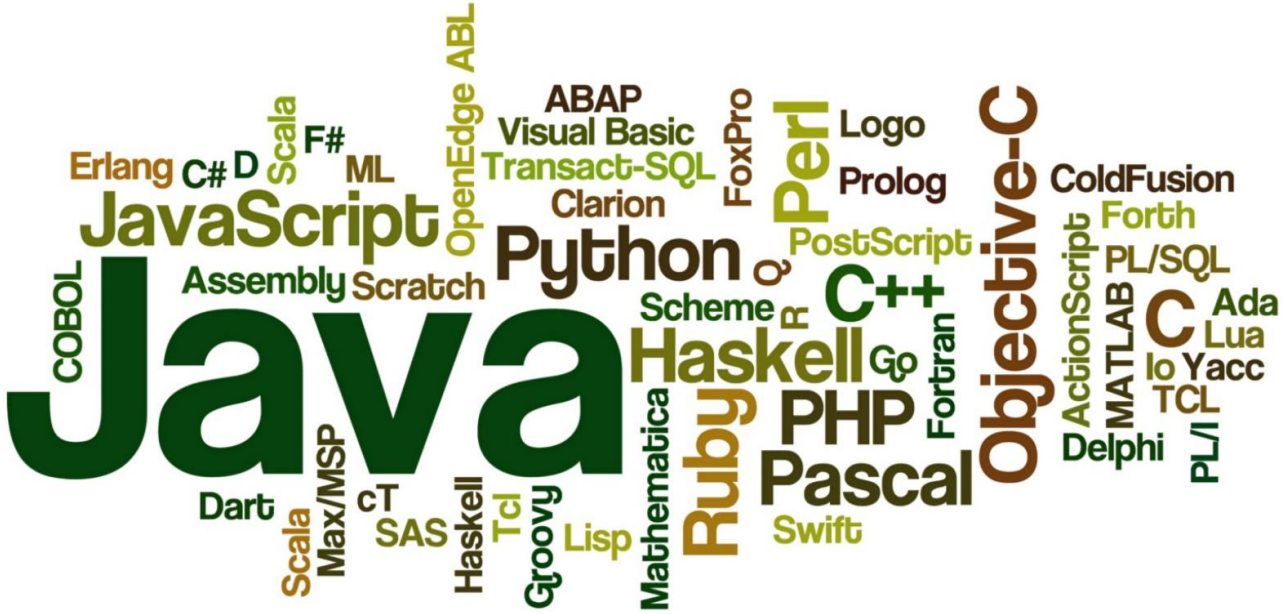
Osnove programskega jezika *java*

Programiranje 2, Tomaž Dobravec

Izbira programskega jezika

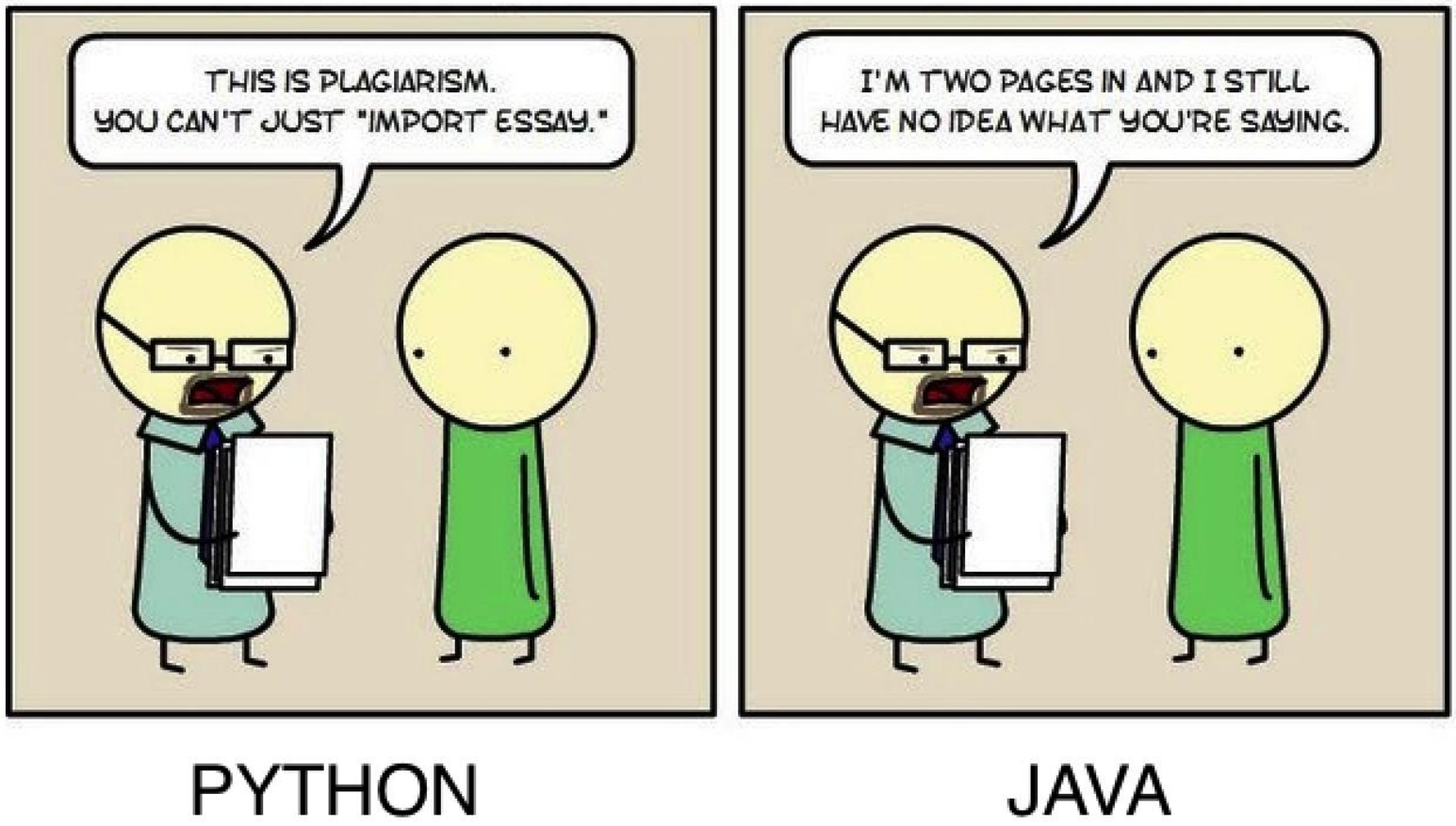


[seznam jezikov]

- Kateri je najboljši?
- Katerega naj izberem?

Python ali Java?

Če bi bilo pisanje programa kot pisanje eseja...

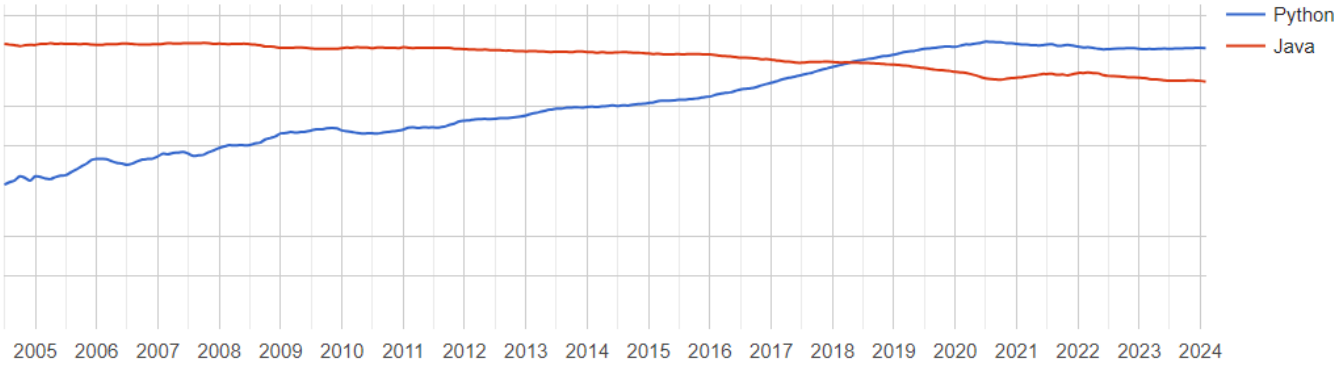


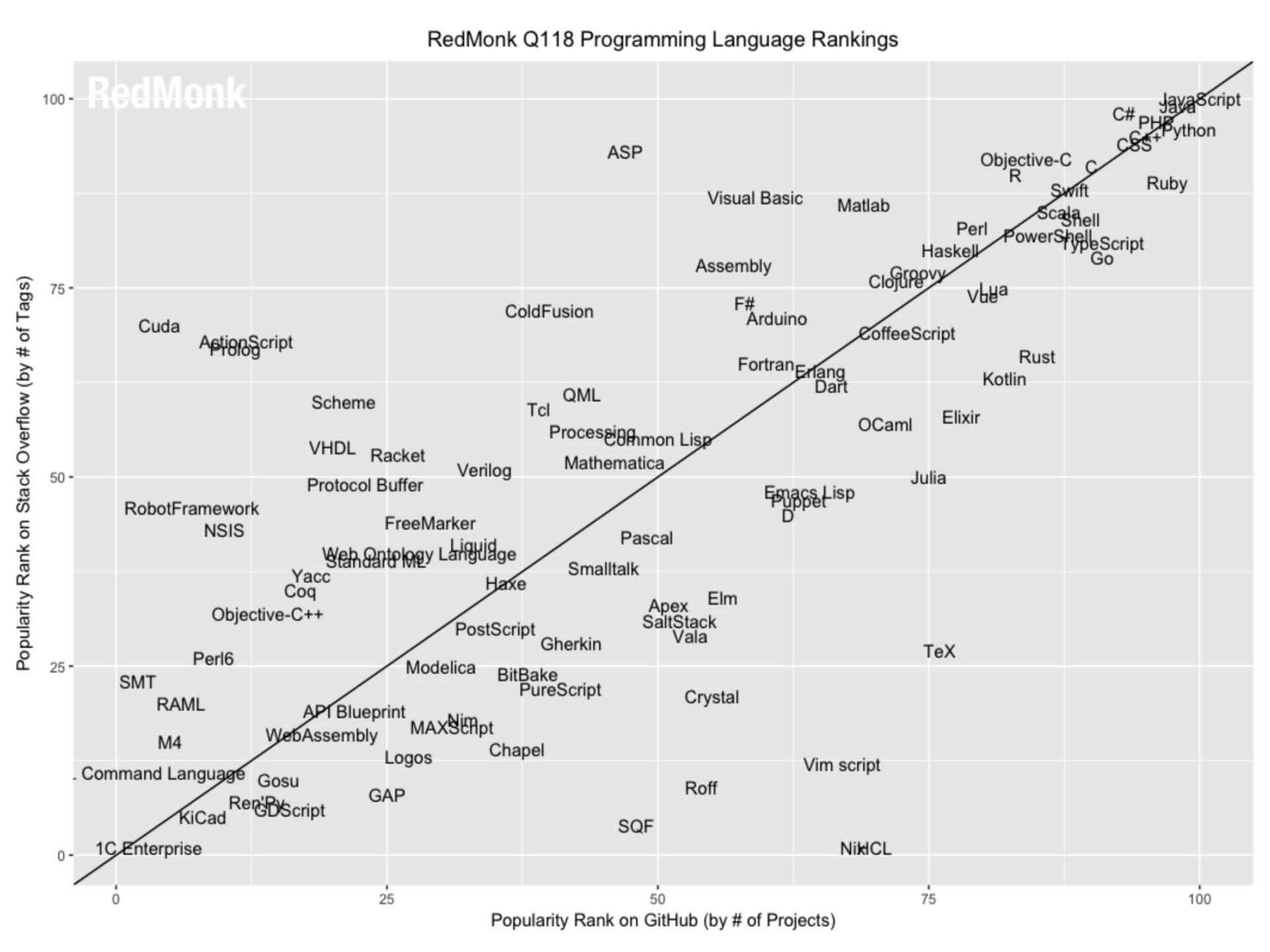
[vir]

PYPL (PopularitY of Programming Language)

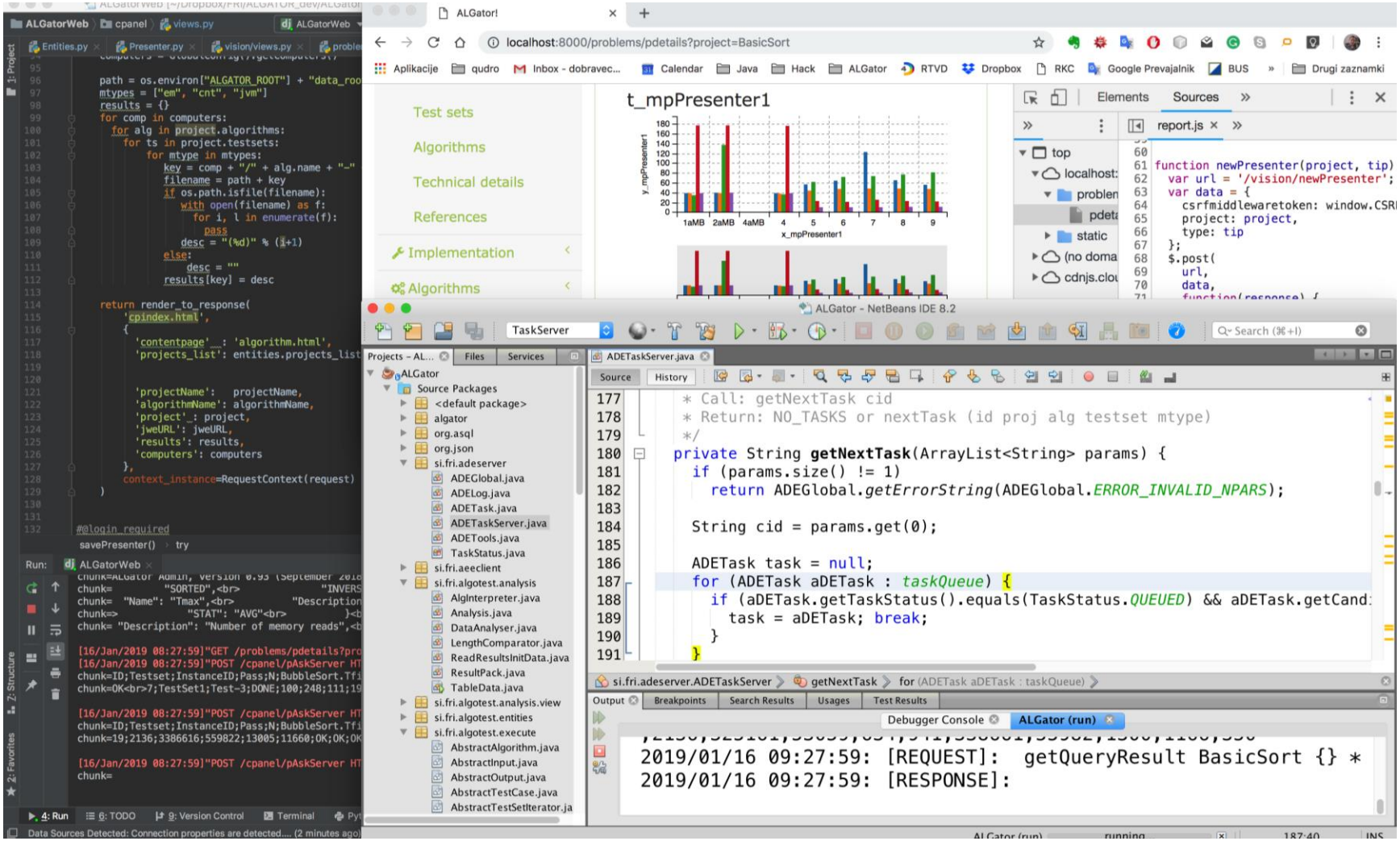
Worldwide, Feb 2024 :

Rank	Change	Language
1		Python
2		Java
3		JavaScript
4	↑	C/C++
5	↓	C#




RedMonk Q118 Programming Language Rankings
RedMonk
Popularity Rank on Stack Overflow (by # of Tags)
Popularity Rank on GitHub (by # of Projects)
0
25
50
75
100
JavaScript
Java
C#
PHP
Python
C++
CSS
Objective-C
R
C
Ruby
Swift
Visual Basic
Matlab
Scala
Shell
Perl
PowerShell
TypeScript
Haskell
Go
Assembly
Groovy
Clojure
Lua
Vue
F#
Arduino
ColdFusion
Cuda
CoffeeScript
ActionScript
Prolog
Rust
Fortran
Erlang
Kotlin
Dart
Scheme
QML
Tcl
Elixir
OCaml
Processing
Common Lisp
VHDL
Racket
Mathematica
Verilog
Julia
Protocol Buffer
Emacs Lisp
Puppet
RobotFramework
D
FreeMarker
NSIS
Pascal
Liquid
Web Ontology Language
Standard ML
Smalltalk
Yacc
Haxe
Coq
Apex
Elm
Objective-C++
SaltStack
PostScript
Vala
Gherkin
TeX
Perl6
Modelica
BitBake
SMT
PureScript
Crystal
RAML
API Blueprint
Nim
MAXScript
WebAssembly
M4
Chapel
Logos
Vim script
Command Language
Gosu
Roff
GAP
RenPy
GDScript
KiCad
SQF
1C Enterprise
NikiCL

Izbira programskega jezika



uradna maskota Duke

Kaj je to *java*

- *Java* je sodoben programski jezik.
  - Java 1.0 1996 (začetek razvoja: 1991)
  - C 1970
  - Python 1991
  - C# 2000

logotip (skodelica kave)

- Java je otok v Indoneziji, kjer pridelajo veliko kave.

Verzije jezika *java*

- Oak 1991
- Java 1.0 1996

- Java 5.0 2004 (resne spremembe jezika; Java Tiger)
- Java 8.0 2014 (podpora lambda izrazom)

- od 2017 naprej nove verzije na pol leta
  - razlike med verzijami majhne
  - Java 8.0, **Java 11.0**, Java 17.0 **…** LTS verzije

Lastnosti jezika *java*

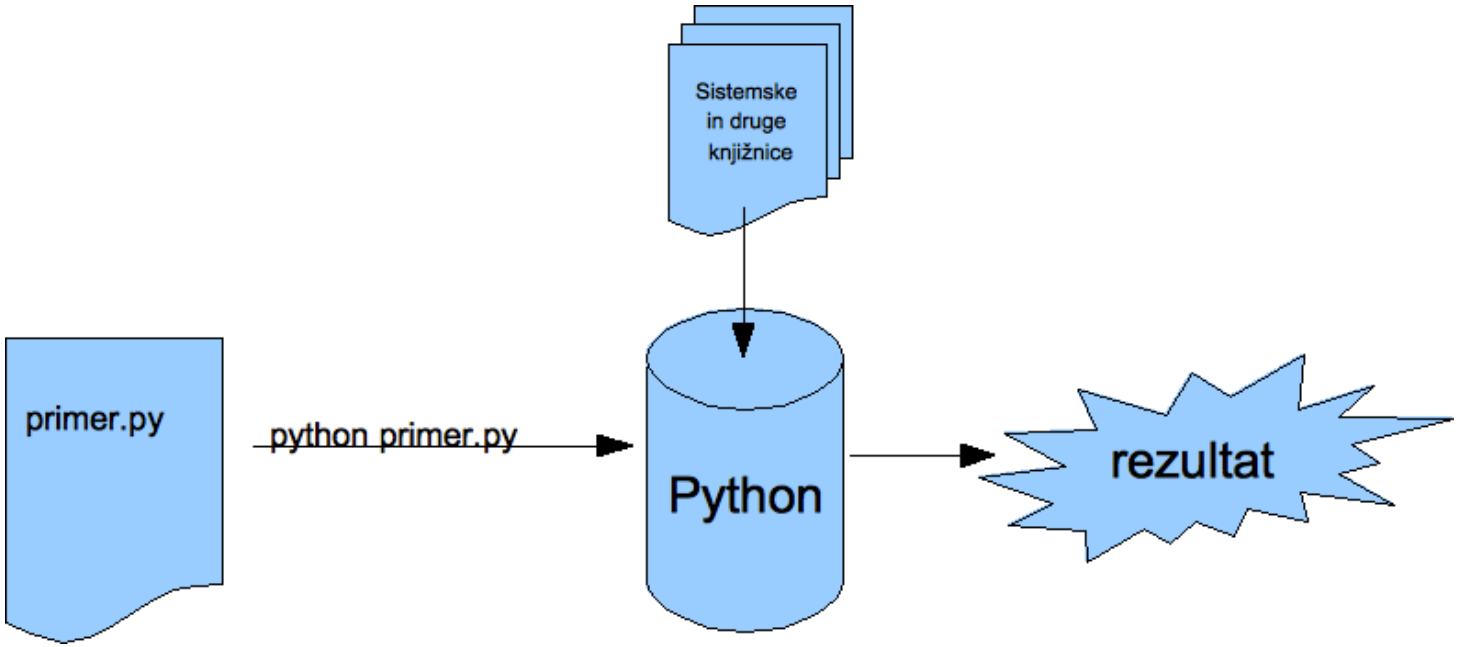
Java je

- preprost,
- predmetno usmerjen,
- robusten,
- neodvisen od sistema,
- varen,
- visoko zmogljiv

programski jezik.

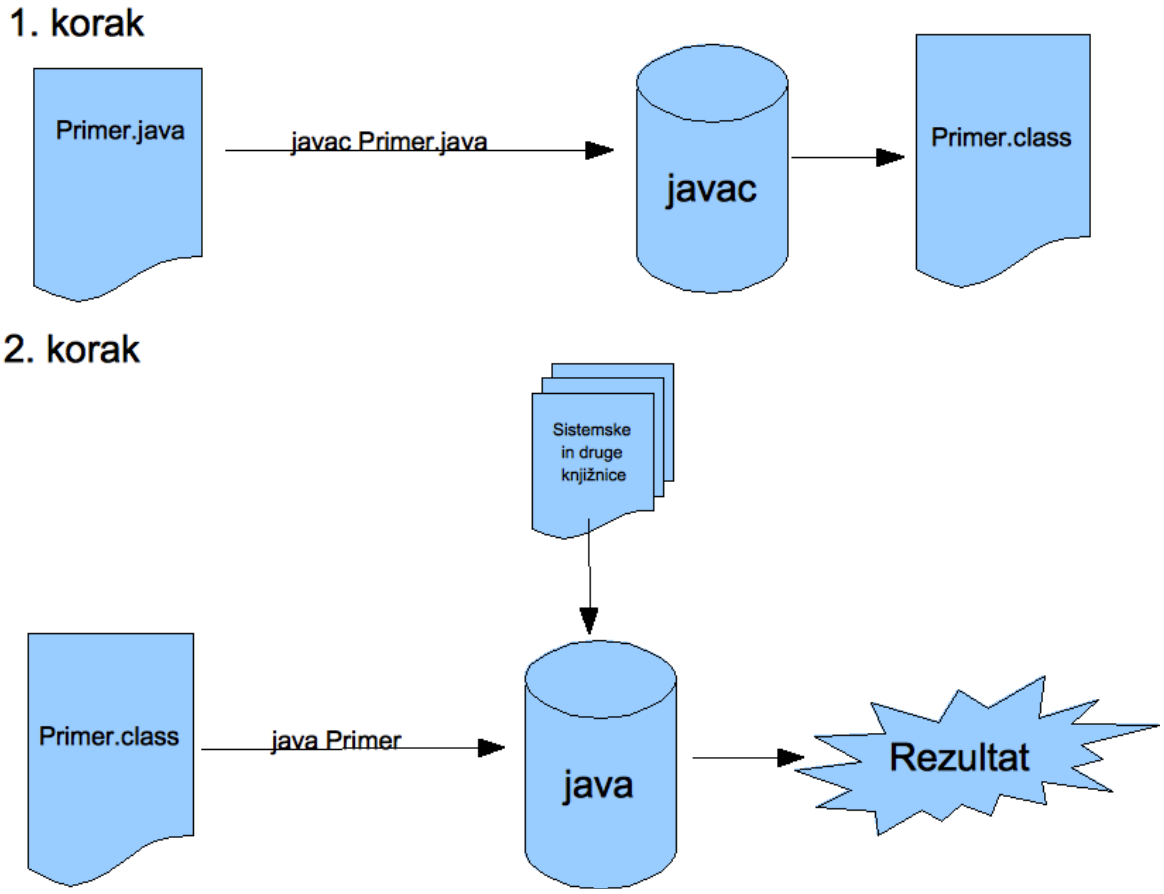
Izvajanje pythonskih programov

```
[user@localhost]# python primer.py
```



Izvajanje *java*nskih programov

```
[user@localhost]# javac Primer.java
[user@localhost]# java Primer
```



Prvi program – primerjava s pythonom

Naloga: Napiši program, ki na zaslon izpiše števila od 10 do 1

```
import sys

def odstevanje(n):
  for i in range(n,0,-1):
    sys.stdout.write("%i\n" % i)

odstevanje(10)
```

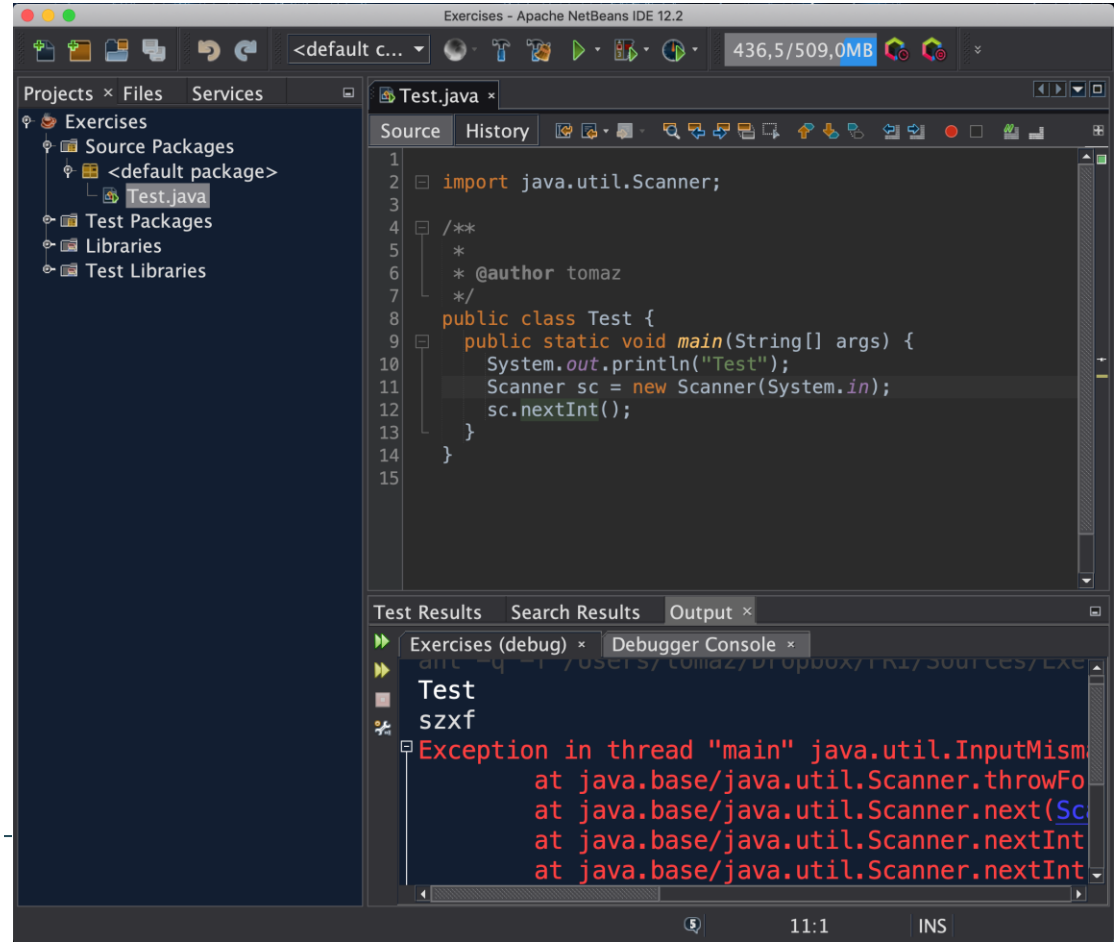
```
for i in range(10,0,-1):
  print i
```

```
6
5
4
3
2
1
[user@localhost]#
```

```
import java.lang.*;

public class Odstevanje {

  static void odstevanje(int n) {
    for (int i = n; i > 0; i--) {
      System.out.printf("%d \n", i);
    }
  }

  public static void main(String[] args) {
    odstevanje(10);
  }
}
```

Priporočeno razvojno okolje

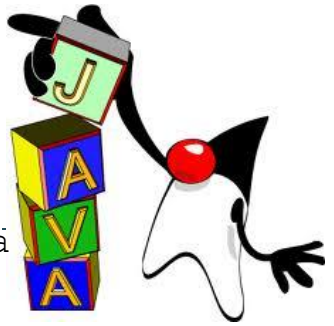
- Prvi koraki v javi
  - urejevalnik besedila + poganjanje in izvajanje v lupini
- Nadaljevanje
  - NetBeans
  - Eclipse
  - IntelliJ IDEA



Spoznavanje osnov jezika skozi primere

- Prvi program
- Ponavljanje ukazov (zanka)
- Spremenljivke in tipi
- Formatiran izpis
- Osnovno o nizih in tabelah
- Uporaba metod
- Argumenti programa
- Pretvorba tipov
- Naključna števila
- Branje iz tipkovnice
- Branje iz tekstovne datoteke
- Pisanje v tekstovno datoteko

Naloga Prvi program

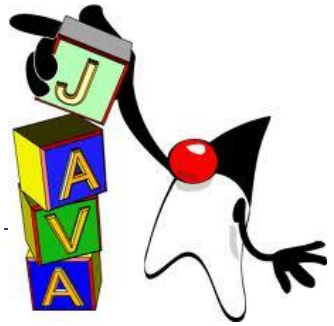


Obresti.java

Napiši program, ki izračuna in na zaslon izpiši stanje na bančnem računu po `n` letih, če vežemo `G` denarja po obrestni meri `p%`.

Primer izvajanja programa:

```
$ javac Obresti.java
$ java Obresti
Glavnica: 1000.0
Število let: 10
Obrestna mera: 5.0
Koncni znesek: 1628.894626777442
```

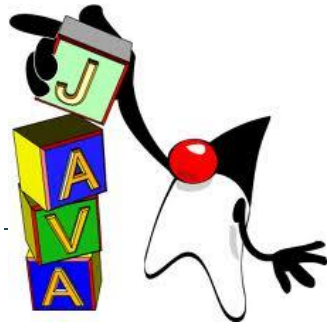


Za izdelavo programa bomo potrebovali:

- vrednosti za glavnico, obrestno mero in število let; → spremenljivke
- formulo za izračun; → $G_n = G * \left(1 + \frac{p}{100}\right)^n$
- metodo za izpis; → `System.out.println()`
- program bomo komentirali. → `//...` `/*... */`

Naloga

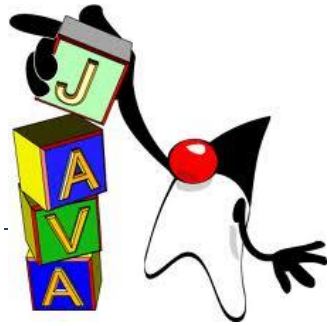
Prvi program – rešitev




```
/*
   Program Obresti izračuna in izpiše znesek na
   bančnem računu po n letih, pri dani osnovni
   glavnici G in obrestni meri p.
*/
public class Obresti {

  // Metoda main se izvrši ob klicu programa z ukazom
  //   java Obresti
  public static void main(String[] args) {

    // Najprej deklariramo štiri spremenljivke;
    // pri vsaki povemo njen tip in ime
    double p;  // obrestna mera
    int n;     // stevilo let
    double G;  // glavnica
    double Gn; // koncni znesek

    // Nato spremenljivke inicializiramo (jim nastavimo vrednost).
    p = 5;       // 5% obrestna mera
    n = 10;      // 10 let
    G = 1000;    // glavnica: 1000 EUR

    // Izračun končnega zneska; za izračun potence x^n
    // uporabimo metodo Math.pow(x,n).
    Gn = G * Math.pow(1 + p/100, n);

    // Izpis rezultata
    // Zanimivo: pri izpisu lahko "seštevamo" različne tipe
    // (v prvem izpisu, na primer, String in double);
    // java bo double vrednost najprej pretvorila v String,
    // nato bo seštela (združila) dva niza.
    System.out.println("Glavnica: " + G);
    System.out.println("Število let: " + n);
    System.out.println("Obrestna mera: " + p);
    System.out.println("Koncni znesek: " + Gn);
  }
}
```

Naloga Večkratni izpis

JavaZakon.java

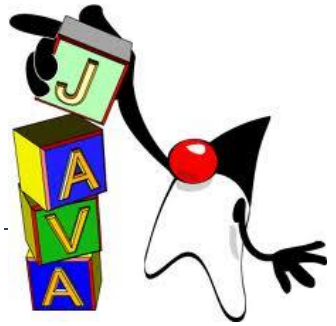


Napiši program, ki petkrat izpiše besedilo "Java je zakon!". Vrstice izpisa naj bodo oštevilčene.

Primer izvajanja programa:

```
$ javac JavaZakon.java
$ java JavaZakon
1. Java je ZAKON!
2. Java je ZAKON!
3. Java je ZAKON!
4. Java je ZAKON!
5. Java je ZAKON!
```

Naloga Večkratni izpis – možne rešitve 2/2



Program lahko napišemo z večkratno uporabo metode `println()`:

```
public class JavaZakon {
  public static void main(String [] args) {
    System.out.println("1. Java je ZAKON!");
    System.out.println("2. Java je ZAKON!");
    System.out.println("3. Java je ZAKON!");
    System.out.println("4. Java je ZAKON!");
    System.out.println("5. Java je ZAKON!");
  }
}
```

ali s pomočjo ZANKE:

```
public class JavaZakon {
  public static void main(String [] args) {
    for (int i=1; i<=5; i=i+1) {
      System.out.println(i + ". Java je ZAKON!");
    }
  }
}
```

Spremenljivke in tipi

- Java je strogo tipiziran jezik: vsaka spremenljivka ima točno določen tip.
- Osnovni (atomarni) *java*nski tipi: `char`, `byte`, `short`, `int`, `long`, `float`, `double`, `boolean`. Za delo z nizi uporabljamo razred `String`.

- Javanska spremenljivka potrebuje deklaracijo (podamo tip).

```
int i;
```

- Ob deklaraciji lahko spremenljivko tudi inicializiramo (podamo vrednost).

```
int i = 1;
```

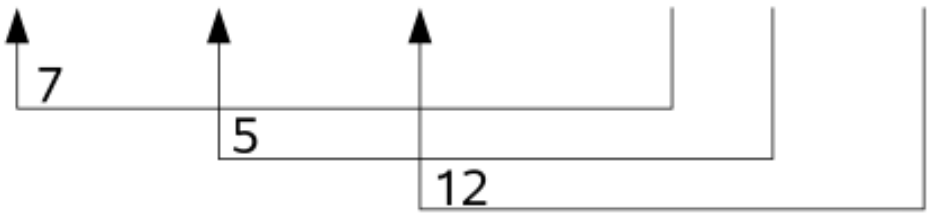
Formatiran izpis

- "Živo" besedilo izpisujemo z metodo `printf()`

Primer:

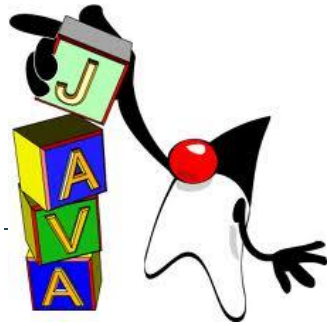
```
public class VsotaXY {
  public static void main(String args[]) {
    int x = 7;
    int y = 5;
    System.out.printf("%d + %d = %d\n", x, y, x+y);
  }
}
```

```
printf("%d + %d = %d \n", x, y, x+y);
```



Naloga

Cena srečke



`Srecke.java`

Napiši program `Srecke.java`, ki izpiše tabelo cen za prodajo srečk, če ena srečka stane `1,25 EUR`.

Primer izvajanja programa:

```
$ javac Srecke.java
$ java Srecke
Stevilo sreck | Cena (EUR)
-------------------------
    1         |  1,25
    2         |  2,50
    3         |  3,75
    4         |  5,00
    5         |  6,25
    6         |  7,50
    7         |  8,75
    8         | 10,00
    9         | 11,25
   10         | 12,50
```

Zahtevnejši formatiran izpis

- Formati izpisa:

decimalno število	`%d`
realno število	`%f`
znak	`%c`
niz	`%s`
celo število v osmiškem sistemu	`%o`
celo število v šesnajstiškem sistemu	`%x, %X`

- Prazen prostor (presledki)

```
printf("x=%5d", 10)
```

→ `x=␣␣␣10`

- Število decimalnih mest

`%.3f` … izpis realnega števila na tri decimalke

http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Formatter.html

Zahtevnejši formatiran izpis

```
public class Printf {
  public static void main(String args[]) {
  System.out.printf ("Znaki: %c %c \n", 'a', 65);
  System.out.printf ("Cela stevila: %d %d\n", 2012 ,123456789);
  System.out.printf ("Presledki pred stevilom: %10d \n", 2012);
  System.out.printf ("Znak '0' pred stevilom: %010d \n", 2012);
  System.out.printf ("Stev. sistemi: %d %x %o %#x %#o %x %X\n",
    100, 100, 100, 100, 100, 10, 10);
  System.out.printf ("Realna stevila: %4.2f %+.0e %E \n",
    3.1416, 3.1416, 3.1416);
  System.out.printf ("Niz: %s \n", "To je niz");
  }
}
```

```
Znaki:␣a␣A
Cela␣stevila:␣2012␣123456789
Presledki␣pred␣stevilom:␣␣␣␣␣␣␣␣2012
Znak␣'0'␣␣pred␣stevilom:␣0000002012
Stev.␣sistemi:␣100␣64␣144␣0x64␣0144␣a␣A
Realna␣stevila:␣3,14␣+3e+00␣3.141600E+00
Niz:␣To␣je␣niz
```

Osnovno o nizih

- Niz je objekt razreda `String`.
- Dolžino niza dobimo s klicem metode `length()`.
- `i`-to črko niza dobimo z metodo `charAt(i)`.
- Metode objekta razreda `String`: `substr()`, `indexOf()`, `replaceAll()`, `isEmpty()`, `split()`, `trim()`, …

(glej: seznam vseh metod)

- **pomembno**: za primerjavo nizov uporabljamo metodo `equals()`:

NAPAČNO!

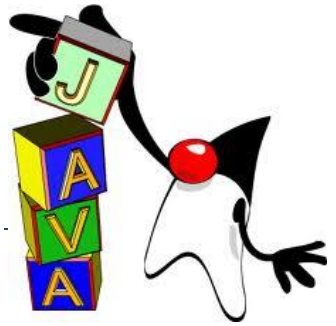
```
if (niz1 == niz2)
```

PRAVILNO

```
if (niz1.equals(niz2))
```

Naloga

Osnovno o nizih



StatistikaNiza.java

- Napiši program, ki izpiše kratko statistiko podanega `niza`. Statistika naj vsebuje: prvo in zadnji črko niza, število besed v nizu ter celotno dolžino niza. Poleg tega naj program `niz` izpiše še v obrnjenem vrstnem redu (od zadaj naprej).

```
Vpisi niz: Danes je lep dan!
Prva   crka:      D
Zadnja crka:      !
Brez presledkov: Danesjelepdan!
Stevilo besed:    4
Celotna dolzina: 17
Obrnjen niz:      !nad pel ej senaD
```

Osnovno o tabelah 1/2

- Tabela je osnovna javanska podatkovna struktura, v kateri lahko hranimo več podatkov ***istega*** tipa.

- **Deklaracija tabele:** `tip [] ime_tabele;`
  - tabela števil … `int [] stevila;`
  - tabela nizov … `String [] nizi;`

- Elementi tabele: `tabela[0], tabela[1], tabela[2], …`
- Število elementov v tabeli: `tabela.length`
- Zadnji element tabele: `tabela[tabela.length-1]`

Osnovno o tabelah 2/2

- Ko tabelo ustvarimo, ji določimo velikost; kasneje se velikosti tabele **ne da spreminjati.**

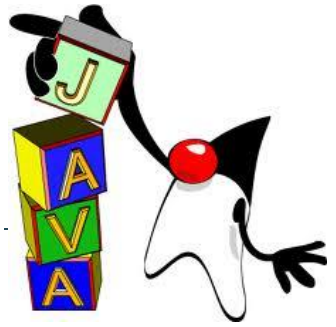
- Tabelo **ustvarimo** z ukazom `new`:

```
        int [] tabela  = new int[10];      … tabela za 10
števil
        String [] nizi = new String[5];  … tabela za 5 nizov
```

Pozor: ukaz `length` nam pove, koliko je tabela velika, ne pa tudi, koliko elementov smo v njo dejansko shranili.

Naloga

Število besed na dano črko



Stetje.java

Naloga: Napiši program, ki prešteje, koliko besed v tabeli `besede`

```
String [] besede = new String [] {
    ”pomlad", ”jabolko", ”jesen", "zima", ”lopar", ”bor”
};
```

se začne na posamezno črko angleške abecede. Izpis programa naj bo tak:

Začetek izpisa programa:

```
Število besed na črko 'a': 0
Število besed na črko 'b': 1
Število besed na črko 'c': 0
Število besed na črko 'd': 0
Število besed na črko 'e': 0
Število besed na črko 'f': 0
Število besed na črko 'g': 0
Število besed na črko 'h': 0
Število besed na črko 'i': 0
Število besed na črko 'j': 2
Število besed na črko 'k': 0
Število besed na črko 'l': 1
Število besed na črko 'm': 0
```

Argumenti programa

```
$ ls *.txt
```
*.txt je argument programa ls

```
$ mv besedilo.txt b.txt
```
besedilo.txt in b.txt sta argumenta programa mv

```
$ javac Racunalo.java
```
Racunalo.java je argument programa javac

```
$ java Racunalo 3 5
```
Racunalo, 3 in 5 so argumenti programa java

```
$ java Racunalo 3 5
```
3 in 5 sta argumenta programa Racunalo

Argumenti programa

- V programu, katerega metoda `main` je deklarirana z

```
public static void main(String[] args)
```

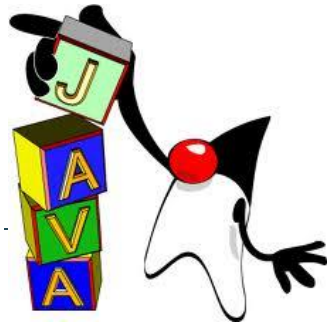
so argumenti, ki so podani ob klicu, shranjeni v tabeli `args`.

Argumenti so torej:

```
args[0], args[1], ..., args[args.length-1]
```

Naloga

Argumenti programa



osnove/Args.java

- Napiši program, ki izpiše vse svoje argumente.

Primer izpisa programa:

```
$ javac Args.java
$ java Args
Stevilo argumentov: 0
$ java Args prvi drugi tretji
Stevilo argumentov: 3
Argument 1: prvi
Argument 2: drugi
Argument 3: tretji
$ java Args beseda "dve besedi"
Stevilo argumentov: 2
Argument 1: beseda
Argument 2: dve besedi
```

Netbeans in argumenti

- Za uporabo argumentov v programu NetBeans storimo naslednje:
  - preko dialoga za nastavitve (desni klik na ime projekta v oknu `Projects`, izberemo `Properties`) v razdelku `Run` nastavimo `Main Class` (`Args`) in `Arguments` (vpišemo argumente),
  - program poženemo z F6 (in ne Shitf-F6), saj v tem primeru poganjamo celoten projekt in ne posamezne datoteke.

IntelliJ in argumenti

- Za uporabo argumentov v programu IntelliJ IDEA storimo naslednje:
  - V glavnem oknu programa izberemo `Add configuration…`
  - V dialogu, ki se odpre:
    - klik na `+` (Add new configuration), izberemo `Application`,
    - nastavimo `Main class` in `Program arguments`,
    - izberemo `Apply`.
  - Program poženemo z `Run` (`Ctrl+R`)

Uporaba statičnih metod 1/2

Metoda združuje zaporedje ukazov; *klic* metode sproži izvajanje teh ukazov.

Primer 1:

a) deklaracija metode:

void … metoda ne vrača rezultata

```
static void izpisiStevilo(int i) {
  System.out.println(i);
}
```

b) klic metode:

```
public static void main(String [] args) {
  izpisiStevilo(4);
}
```

Uporaba statičnih metod 2/2

Primer 2:

int… metoda vrača rezultata tipa **int**

a) deklaracija metode, ki vrača rezultat:

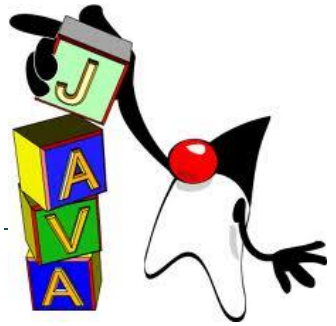
```
static int sestej(int a, int b) {
  return a + b;
}
```

b) klic metode (rezultat "ujamemo" v spremenljivko `vsota`):

```
public static void main(String [] args) {
  int vsota = sestej(4, 7);
  System.out.println(vsota);
}
```

Naloga

Uporaba metod



`StatistikaNizaZMetodo.java`

- Naloga: Program `StatistikaNiza.java` popravi tako, da bo vsa logika obdelave niza združena v metodi

```
void statistikaNiza(String niz) {
   …
}
```

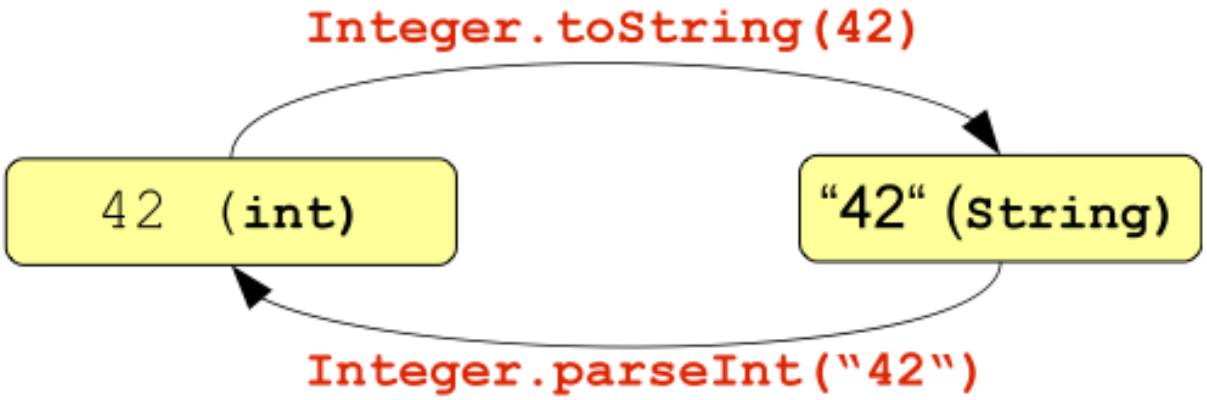
V metodi `main()` metodo `statistikaNiza()` kliči dvakrat (z dvema različnima nizoma).

Pretvorba tipov

- Če seštejemo števili, ki sta podani v obliki niza, dobimo …?

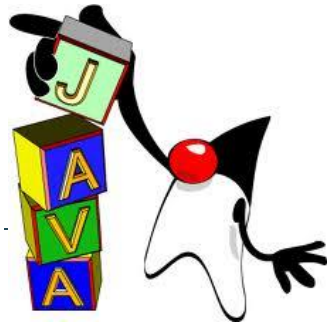
```
"13" + "7" = "137"
```

- Za pretvorbo med številom (`int`) in nizom (`String`) uporabimo:



Naloga

Računalo (pretvorba tipov)



osnove/Racunalo.java

- Napiši program `Racunalo.java`, ki izračuna in izpiše vsoto prvih dveh argumentov.

Primer izvajanja programa:

```
$ javac Racunalo.java
$ java Racunalo
Napačno število argumentov.
$ java Racunalo 5 7
5 + 7 = 12
```

Naključna števila

- Za generiranje psevdo-naključnih števil v javi uporabljamo razred `Random`.

```
Random rnd = new Random();
int nakljucnoStevilo = rnd.nextInt();
```

Metoda **`nextInt()`** vrne naključno celo število iz območja $[-2^{31}...2^{31}-1]$.

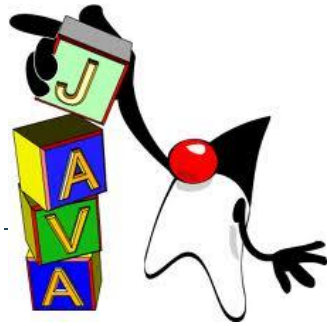
- Celo naključno število iz intervala `[0...n-1]`:

```
int r = rnd.nextInt(n);
```

- Realno naključno število iz intervala `[0, 1)`:

```
double r = rnd.nextDouble();
```

Naloga Loto listek (naključna števila)



Loto.java

- Napiši program, ki izpiše 7 naključnih števil (pomoč pri izpolnjevanju osnovnega loto listka).

Primer izvajanja programa:

```
$ javac Loto.java
$ java Loto
22 5 5 30 28 11 5
$ java Loto
3 6 13 15 37 38 33
```

Naključna števila izbranega intervala

- Kako ustvarim naključno celo število iz intervala `[a...b]` če sta `a` in `b` celi števili in `a < b` ?

```
int r = a + rnd.nextInt(b-a+1);
```

- Kako ustvarim naključno realno število iz intervala `[x,y)` če sta `x` in `y` realni števili in `x < y` ?

```
double r = x+(y-x)*rnd.nextDouble();
```

Branje iz tipkovnice

- Za branje iz tipkovnice bomo uporabili razred `Scanner` in standardno ročico `System.in`.

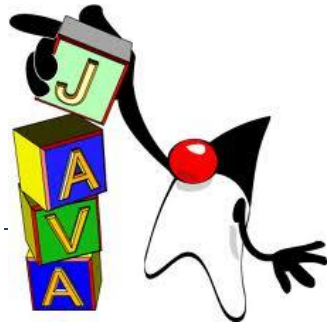
```
Scanner sc = new Scanner(System.in);
```

Na objektu razreda `Scanner` kličemo metode

`nextInt()`	...	branje celega števila
`nextLine()`	...	branje celotne vrstice
`next()`	...	branje ene besede
`nextDouble()`	...	branje realnega števila

Naloga

Povprečje ocen (branje iz tipkovnice)



Ocene.java

- Napiši program za računanje povprečja vpisanih ocen.

Podrobneje: program naj bere ocene, ki jih vpisuje uporabnik, dokler ta ne vpiše 0. Takrat naj program izračuna povprečje vpisanih ocen in ga izpiše.

Primer izvajanja programa:

```
$ javac Povprecje.java
$ java Povprecje
Vpisi oceno: 7
Vpisi oceno: 6
Vpisi oceno: 9
Vpisi oceno: 10
Vpisi oceno: 10
Vpisi oceno: 8
Vpisi oceno: 0
Povprecje 6 prebranih ocen je 8,33
```

Branje iz tekstovne datoteke

Branje tekstovne datoteke je podobno branju iz tipkovnice:

- Uporabimo razred `Scanner`,
- namesto `System.in` uporabimo `new File(ime_datoteke)`

```
Scanner sc = new Scanner(new File("CHF2011.txt"));
```

- za branje uporabimo metode `nextInt(),next(),...`
- konec datoteke preverim z metodo `hasNext()`

```
while (sc.hasNext()) {
  datum = sc.next();
  ...
}
```

Branje iz tekstovne datoteke

- Obstaja več metod tipa `hasNext()`:
  - `hasNext()`
  - `hasNextInt()`
  - `hasNextLine(), …`

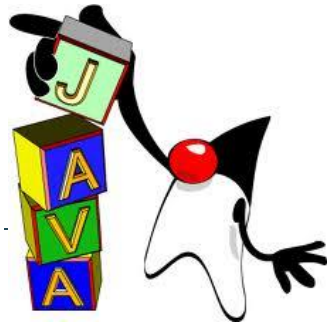
- Nasvet: za preverjanje in branje vedno uporabim metodo istega tipa
  - `hasNext()` + `next()` ali
  - `hasNextInt()` + `nextInt(),…`

- Ko datoteke ne potrebujem več, jo moram zapreti

```
sc.close();
```

Naloga

Tečajna lista
Branje iz tekstovne datoteke



Tecaj.java

- Napiši program, ki prebere tekstovno datoteko `bitcoin.txt`, v kateri so zbrani podatki o vrednosti valute bitcoin in izpiše datum najvišjega in najnižjega tečaja.

bitcoin.txt

```
30.10.2017 6098.99
31.10.2017 6379.87
01.11.2017 6666.27
02.11.2017 7031.29
03.11.2017 7192.26
04.11.2017 7413.46
05.11.2017 7343.86
06.11.2017 6968.08
07.11.2017 7080.00
08.11.2017 7400.00
09.11.2017 7126.41
10.11.2017 6715.99
11.11.2017 6338.41
12.11.2017 5699.48
13.11.2017 6450.51
```

Primer izvajanja programa:

```
$ javac Tecaj.java
$ java Tecaj bitcoin.txt
MIN tečaj:     4,3800 (datum: 18.02.2012)
MAX tečaj: 19208,0600 (datum: 17.12.2017)
```

Pisanje v tekstovno datoteko

- Za pisanje v tekstovno datoteko uporabimo razred `PrintWritter`

```
PrintWriter pw = new PrintWriter("veckratniki.txt");
```

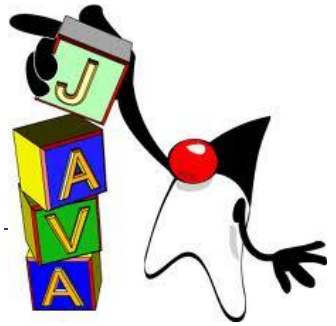
- Objekt razreda `PrintWritter` se uporablja enako kot objekt `System.out`:
  - `pw.printf("%d", i),`
  - `pw.prinln("To je test"), …`

- Ko datoteke ne potrebujemo več, jo moramo zapreti

```
pw.close();
```

Naloga

Večkratniki
Pisanje v tekstovno datoteko



Veckratniki.java

- Napiši program, ki v datoteko `veckratniki.txt` zapiše večkratnike števila `n` (od `a*n` do `b*n`). Števila `a`, `b` in `n` preberi iz tipkovnice.

Primer izvajanja programa:

```
$ javac Veckratniki.java
$ java Veckratniki
Vpisi n: 7
Vpisi a: 3
Vpisi b: 6
$ cat veckratniki.txt
3 * 7 = 21
4 * 7 = 28
5 * 7 = 35
6 * 7 = 42
```

Uporaba paketov

- Paketi v *javi* omogočajo boljšo organizacijo kode in preprečujejo konflikte imen
- Paket napovemo z rezervirano besedo `package`
- Osnovno razumevanje:

paket === mapa (folder)

- *Java*nske razrede bomo razporejali v pakete smiselno glede na namen kode (razredi s podobno vsebino bodo v istem paketu)

Več o paketih v poglavju Strukturiranje kode