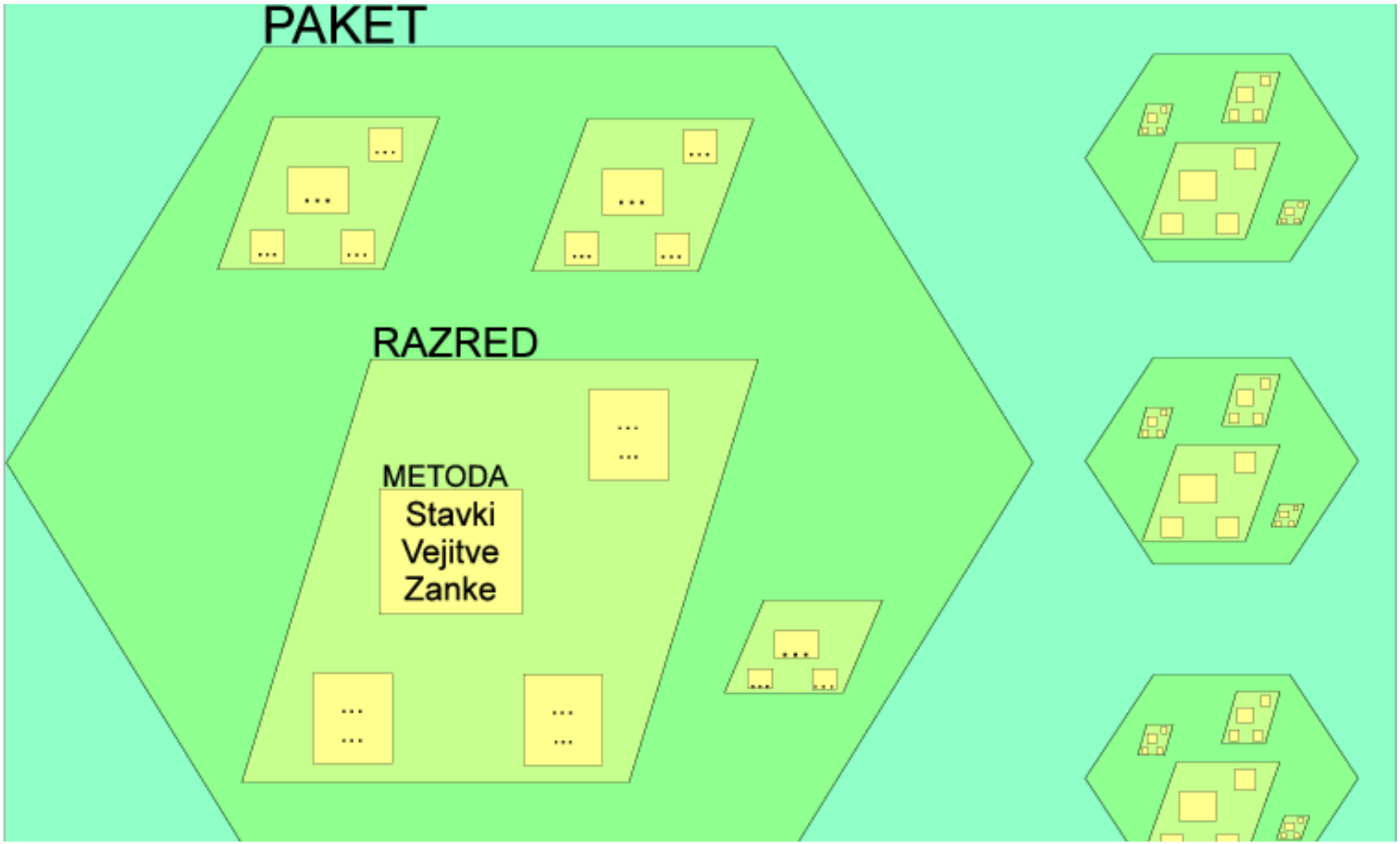
Strukturiranje kode

Programiranje 2, Tomaž Dobravec

Strukturiranje kode



Zanke `do`, `while` in `for`

- Java pozna tri zanke: `do`, `while`, `for`

```
int i=0;          | int i=0;          | int i;
do {              | while (i<=10) {   | for (i=1;i<=10;i++) {
  i=i+1;          |   i=i+1;          |   ....
  ....            |   ....            | }
} while (i<10)    | }                 |
```

- Če je izvajanje zanke nadzorovano s pomočjo nekega števca, običajno uporabimo `for`, če je pogoj zanke drugačne oblike, pa `do` ali `while` zanko

Vejitveni stavki

- Vejitveni stavek `if-else`

```
if (a==1)     |   if (x==5) {     |   if (q<5) {
  b=1;        |     c=1;d=3;      |     d=3;
              |   } else {        |   } else if (q<2) {
              |     c=4;          |     d=2;
              |   }               |   } else {
              |                   |     d=1;
              |                   |   }
```

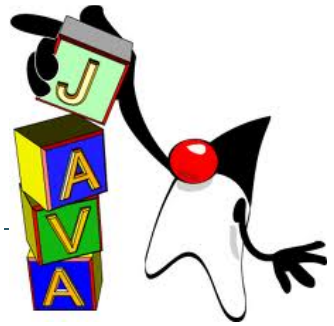
- Vejitveni stavek `switch`

```
switch (a) {
  case 1:  b=3;    break;
  case 2:  b=9;    break;
  case 3:  b=27    break;
  case 4:  b=81;   break;
  default: b=0;    break;
}
```

Pomembno: stavek `break` na koncu vsake veje.

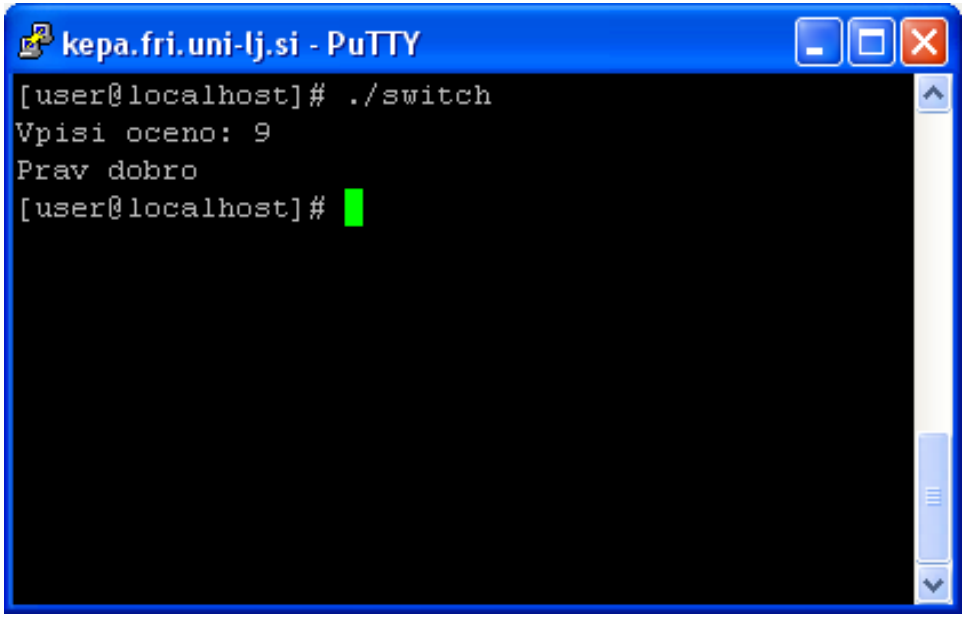
Naloga

Opisna ocena - stavek `switch`



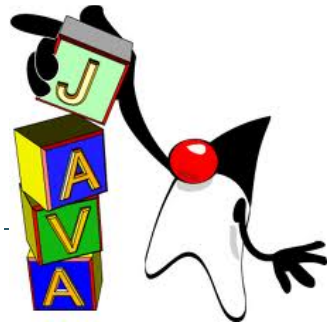
`strukt/SwitchOcena.java`

Napiši program, ki prebere oceno in izpiše ``Odlicno'' za 10, ``Prav dobro'' za 9, ...



Naloga

Stavek `switch` brez ukaza `break`



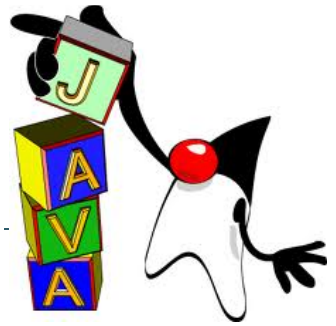
`strukt/SwitchSteviloDni.java`

Napiši metodo `steviloDni(int dan, int mesec)`, ki vrne število dni, ki so minili od začetka leta do dneva `dan.mesec`.

Opomba: za rešitev naloge uporabi ukaz `switch` in ne uporabi stavka `break` na koncu posamezne veje.

Naloga

Izpis abecede (`switch`)



`strukt/SwitchAbeceda.java`

Napiši program, ki izpiše črke slovenske abecedu (brez č, š in ž), kot je prikazano spodaj.

```
A! b* c* d*
E? f* g* h*
I& j* k* l*
m* n* O# p*
r* s* t* U/
v* z*
```

Metode

- Program v javi je sestavljen iz metod.
- Preprosti programi imajo samo eno metodo (`main`), zahtevnejši jih imajo več.
- Metode predstavljajo ločene celote programa; s pomočjo metod program postane preglednejši

Parametri metode in rezultat

- Metoda (lahko) prejme parametre in vrne rezultat

```
static double sredina(double x, double y) {
  return (x+y)/2
}
```

- Metodo *pokličemo* takole:

```
double a = sredina(5, 7); // a = 6
```

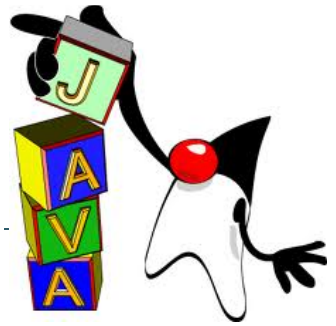
Preoblaganje metod

V javi lahko obstaja več metod z istim imenom a z različnim številom in/ali tipom parametrov (preoblaganje ali method overloading)

```
public static void izpisi(int x, int y, String napis) {
  // … metoda na položaj (x,y) izpiše napis
}
public static void izpisi(String napis) {
  // privzeto mesto izpisa je zgornji levi kot
  izpisi(0,0, napis);
}
public static void izpisi() {
  // privzeto besedilo izpisa je ”OPOZORILO”
  izpisi(”OPOZORILO!”);
}
```

Naloga

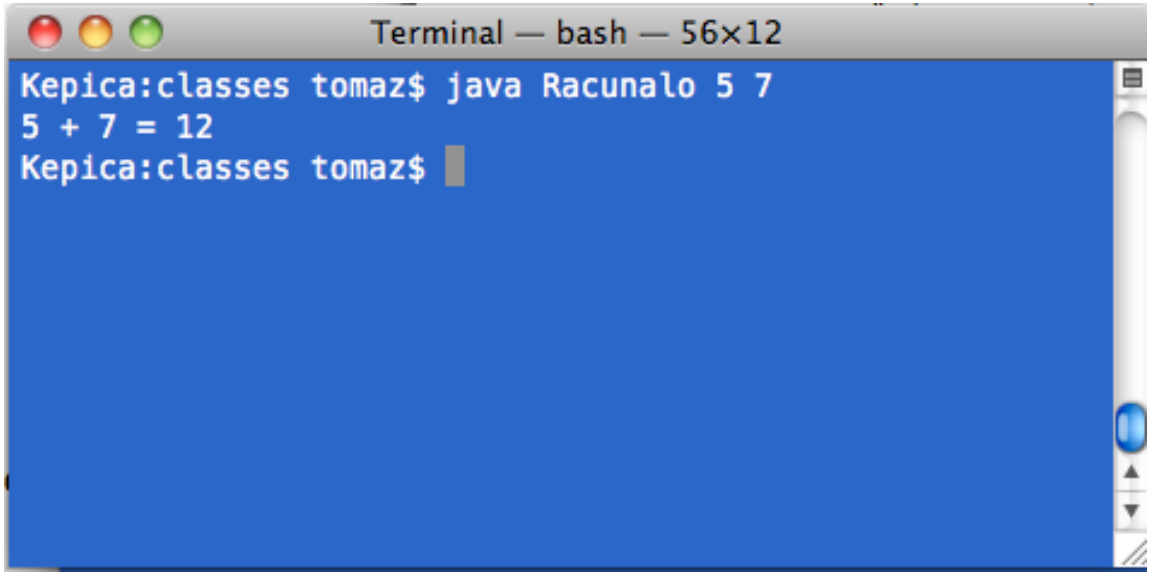
Pretvorba niza v celo število



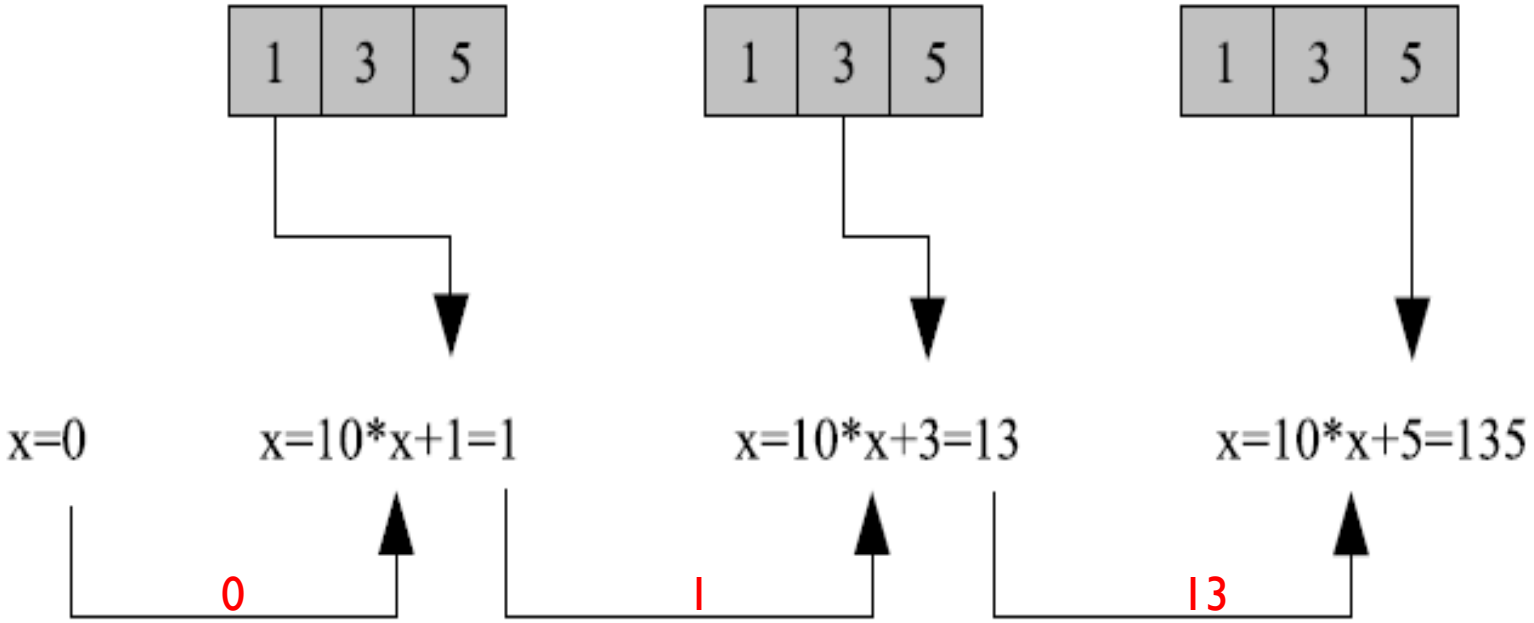
`strukt/Racunalo2.java, strukt/Racunalo3.java`

Napiši program, ki izpiše vsoto prvih dveh argumentov. Program naj NE uporablja nobene vgrajene metode za pretvorbo niza v število.

(algoritem za pretvorbo niza v število je opisan na naslednji prosojnici)



Pretvorba niza v celo število (postopek)

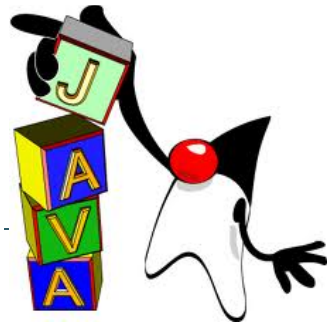


Program v več razredih

- Krajši programi so v celoti napisani v enem razredu, koda zahtevnejših programov pa je razdeljena v več razredov.
- V posameznem razredu je zbrana koda, ki smiselno sestavlja celoto.
- Razredi lahko uporabljajo drug drugega (atributi in metode iz enega razreda so vidni tudi v drugem razredu – če seveda niso skriti)

Naloga

Kodiranje besedila



`strukt/Kodiranje.java, strukt/Tajno.java`

Napiši program za preprosto kodiranje besedila.

Natančneje: napiši metodi

`String kodiraj(String niz)` in

`String odkodiraj(String niz)`,

ki zakodirata in odkodirata podano besedilo, ter program za preizkus delovanja teh metod.

Metodi naj bosta napisani v enem, glavni program pa v drugem razredu.

(preprost algoritem za kodiranje besedila je opisan na naslednji prosojnici)

Kodiranje besedila - postopek

```
// Preprosto kodiranje besedila

// Kodiranje:   vsakemu znaku pristejemo konstanto
// Odkodiranje: vsakemu znaku odstejemo konstanto

// Primer: zamik=3
// Original: A B C D E F G H ... V W X Y Z
// Kodirano: D E F G H I J K ... Y Z A B C

// Operaciji pristevanja in odstevanja delamo
// po modulu 26 (26 crk abecede)
```

```
// kodiranje (originalni znak -> kodiran znak)
kZnak = (char) ('A' + (oZnak -'A' + zamik) % 26);
```

```
// odkodiranje (kodiran znak -> originalni znak)
oZnak = (char) ('A' + (26 + kZnak - 'A' - zamik) % 26);
```

Paketi

Na nivoju operacijskega sistema je paket mapa (folder).

Namen in uporaba:

- smiselno združevanje razredov,
- preprečevanje konfliktov imen,
- lažja distribucija programov.

Obstoječi paketi

- Java s seboj prinaša mnogo paketov: `java.applet`, `java.awt`, `java.io`, `java.lang`, `...`
- Pakete lahko dobimo v jar datotekah, ki jih prenesemo s spleta.
- Pakete ustvarimo sami.

Imena paketov

Paket smiselno poimenujemo (glede na njegov namen in vsebino razredov).

- Primer: primerno ime za paket, ki vsebuje metode za delo z matematičnimi funkcijami, je, na primer, `matematika`.

Za preprečevanje konflikta imen paketu dodamo "predpono", ki enolično opisuje, kdo je avtor paketa. Običajno je predpona sestavljena in obrnjenega internetnega naslova.

Primer: podjetje *Kava d.o.o.* s spletno stranji

```
http://kava.si
```

bo svojim paketom dodalo predpono `si.kava`. Paket z matematičnimi funkcijami se bo pri njih imenoval `si.kava.matematika`.

Uporaba paketov

- Razred, ki ni v našem paketu, moramo pred uporabo uvoziti:

```
import java.util.Random;
Random r = new Random();
```

- Lahko uvozimo tudi celoten paket:

```
import java.util.*
```

- Če ne želimo uvažati, lahko pri uporabi podamo celotno ime:

```
java.util.Random r = new java.util.Random();
```

Avtomatski uvoz paketa `java.lang`

Razred Math se nahaja v paketu `java.lang`.

Zakaj lahko napišemo

```
double pi = Math.PI;
```

brez uvoza `import java.lang.Math`?

Ker java paket `java.lang` avtomatsko uvozi sama!

Opomba: paket `java.lang` je edini paket z avtomatskim nalaganjem!

Statičen uvoz

Statične metode in atribute lahko uporabljamo brez navedbe razreda, če jih prej *statično uvozimo*.

Namesto

```
double pi = Math.PI;
```

lahko napišemo

```
import static java.lang.Math.*;
…
double pi = PI;
```

Uporaba statičnega uvoza ni priporočljiva!

Pot do paketov

```
package si.fri.math.geometrija;

public class Trikotnik {
  // stranice trikotnika
  static int a=3;
  static int b=4;
  static int c=5;

  public static void izpis() {
    System.out.printf("Obseg trikotnika s stranicami "
            + "%d, %d in %d je %d\n", a, b, c, a+b+c);
  }

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

```
D:\> cd program\si\fri\math\geometrija
D:\program\si\fri\math\geometrija> dir
Trikotnik.java
D:\program\si\fri\math\geometrija>javac Trikotnik.java
D:\program\si\fri\math\geometrija>java Trikotnik.java
Exception in thread "main" java.lang.NoClassDefFoundError ...
D:\program\si\fri\math\geometrija>cd \program
D:\program>java Trikotnik
Exception in thread "main" java.lang.NoClassDefFoundError ...
D:\program>java si.fri.math.geometrija.Trikotnik
Obseg trikotnika s stranicami 3, 4 in 5 je 12

D:\program>cd ..
D:\>java si.fri.math.geom.Trikotnik
Exception in thread "main" java.lang.NoClassDefFoundError ...
D:\>java program.si.fri.math.geom.Trikotnik
Exception in thread "main" java.lang.SecurityException ...
```

Pot do paketov

```
D:\>java -cp d:\program si.fri.math.geometrija.Trikotnik
Obseg trikotnika s stranicami 3, 4 in 5 je 12


D:\>SET CLASSPATH=d:\program
D:\>java si.fri.math.geom.Trikotnik
Obseg trikotnika s stranicami 3, 4 in 5 je 12
D:\>c:
C:\>cd Windows
C:\WINDOWS>java si.fri.math.geom.Trikotnik
Obseg trikotnika s stranicami 3, 4 in 5 je 12
```

Dostopnostna določila

Java pozna 4 dostopnostna določila (anlg. access modifiers):

Določilo	Pomen
`public`	dostopno povsod in vsakomur
`protected`	dostopno v vseh razredih tega paketa ter v vseh podrazredih (tudi, če niso v tem paketu)
`default` (brez določila)	dostopno v vseh razredih tega paketa
`private`	dostopno samo v tem razredu

Čitljivost javanske kode

- Zelo pomembno je, da je programska koda ČITLJIVA!
- K čitljivosti pripomorejo:
  - uporaba dogovorjenih pravil o poimenovanju spremenljivk metod, razredov in paketov,
  - uporaba dogovorjenih pravil o oblikovanju programa (zamiki) in
  - razumno komentiranje kode.

Podrobneje o čitljivosti kode:

http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

Komentarji v *javi*

Java pozna tri tipe komentarjev:

// komentar	enovrstični komentarji; komentar je vse od znaka // do konca vrstice
/* komenar */	večvrstični komentarji; komentar je vse, kar je ujeto med znaka /* in */
/** komantar */	komentar za dokumentacijo (angl. *doc comments*); tak komentar se pojavi tik pred deklaracijo identifikatorja (spremenljivke, metode, razreda, ...) in se uporablja za avtomatsko generiranje dokumentacije;

Dokumentacijski komentarji

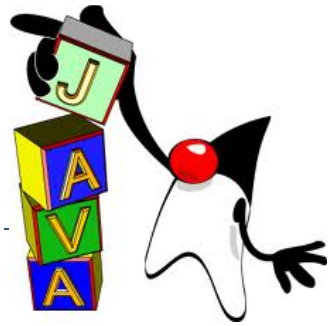
Na podlagi dokumentacijskih komentarjev Java avtomatsko zgradi HTML dokumentacijo.

- Del dokumentacijskih komentarjev so tudi značke:

Značka	primer uporabe
@see	Povezave v rubriki "Glej tudi"
{@link}	Hiperpovezave med opisnim besedilom
@param	Opis parametrov metode
@return	Opis rezultata metode
@throws	Opis izjem, ki jih lahko vrze metoda
@author	Opis avtorja metode
@version	Stevilka verzije

Naloga

Dokumentacijski komentar



`komentar/Pozdrav.java`

Napiši program, ki `n`-krat izpiše

```
Pozdravljen, ime
```

(`ime` in `n` sta prvi in drugi argument ob klicu programa).

V programu uporabi različne načine komentarjev (enovrstični, dvovrstični, dokumentacijski). Poženi ukaz `javadoc` in poglej rezultat (html kodo s komentarji).