

Programiranje 1

Poglavje 3: Krmilni konstrukti

Luka Fürst

Zaporedje stavkov

- zaporedni stavki se enostavno izvršijo po vrsti, drug za drugim

```
stavek1  
stavek2  
...
```

- najprej *stavek1*, nato *stavek2* ...

Pogojni stavek

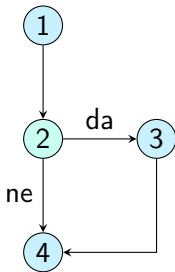
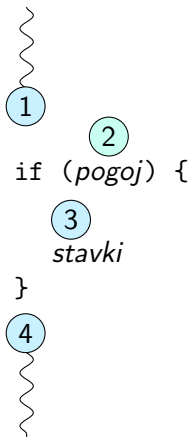
- omogoča, da se določen kos kode izvede samo v primeru, če je izpolnjen določen pogoj

```
if (pogoj) {  
    stavki  
}
```

- če je *pogoj* izpolnjen, se *stavki* izvedejo, sicer pa se ne
- če je stavek en sam, lahko zavite oklepaje izpustimo

```
if (pogoj)  
    stavek
```

Pogojni stavek

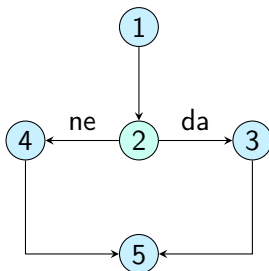
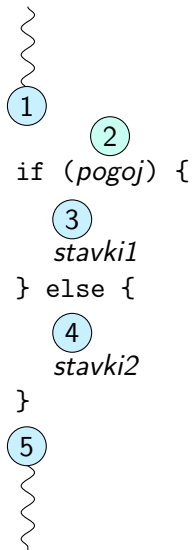


else

```
if (pogoj) {  
    stavki1  
} else {  
    stavki2  
}
```

- če je *pogoj* izpolnjen, se izvedejo *stavki1*, sicer pa *stavki2*

Pogojni stavek z else



Primer

- izpišimo, katero izmed dveh prebranih števil je večje

```
int prvo = sc.nextInt();
int drugo = sc.nextInt();
if (prvo > drugo) {
    System.out.println("Prvo število je večje");
} else {
    System.out.println("Drugo število je večje");
}
```

- kaj se zgodi, če sta števili enaki?
- slabo zastavljena naloga!

Upoštevanje možne enakosti

```
int prvo = sc.nextInt();
int drugo = sc.nextInt();
if (prvo > drugo) {
    System.out.println("Prvo število je večje");
} else {
    if (prvo < drugo) {
        System.out.println("Drugo število je večje");
    } else {
        System.out.println("Števili sta enaki");
    }
}
```


else if

```
if (pogoj1) {  
    stavki1  
} else {  
    if (pogoj2) {  
        stavki2  
    } else {  
        stavki3  
    }  
}
```



```
if (pogoj1) {  
    stavki1  
} else if (pogoj2) {  
    stavki2  
} else {  
    stavki3  
}
```

en stavek, zato lahko { in } izpustimo

else if

```
int prvo = sc.nextInt();
int drugo = sc.nextInt();
if (prvo > drugo) {
    System.out.println("Prvo število je večje");
} else if (prvo < drugo) {
    System.out.println("Drugo število je večje");
} else {
    System.out.println("Števili sta enaki");
}
```

else if

```
if (pogoj1) {  
    stavki1  
}  
else if (pogoj2) {  
    stavki2  
}  
else if (pogoj3) {  
    stavki3  
}  
...  
else {  
    stavkiN  
}
```

- $pogoj1 \implies stavki1$
- $\neg pogoj1 \wedge pogoj2 \implies stavki2$
- $\neg pogoj1 \wedge \neg pogoj2 \wedge pogoj3 \implies stavki3$
- ...
- $\neg pogoj1 \wedge \neg pogoj2 \wedge \dots \wedge \neg pogojN-1 \implies stavkiN$
- v vsakem primeru se izvede natanko eno zaporedje stavkov

Primer

- preberimo število točk (0–100) in izpišimo pripadajočo oceno

```
int tocke = sc.nextInt();
int ocena;    // bolje: int ocena = 0;
if (tocke >= 90) {
    ocena = 10;
} else if (tocke >= 80) {
    ocena = 9;
} else if (tocke >= 70) {
    ocena = 8;
} else if (tocke >= 60) {
    ocena = 7;
} else if (tocke >= 50) {
    ocena = 6;
} else {
    ocena = 5;
}
System.out.println(ocena);
```

Logični izraz in tip boolean

- *pogoj* v pogojnem stavku je **logični izraz**
 - izraz z vrednostjo tipa **boolean**
- spremenljivka tipa **boolean** ima dve možni vrednosti
 - **true** (izraz je resničen)
 - **false** (izraz je neresničen)
- primeri
 - `5 > 3 → true`
 - `6 < 1 → false`
 - `2 <= 2 → true`
 - `false → false`

Primerjalni operatorji

Operator	Pomen	Primer izraza z vrednostjo true
>	večji od	5 > 3
>=	večji ali enak kot	8 >= 2
<	manjši od	4 < 9
<=	manjši ali enak kot	7 <= 7
==	enak kot	3 == 3
!=	različen od	2 != 5

Logični operatorji

- $p \ \&\& \ q$
 - resničen natanko tedaj, ko sta izraza p in q **oba** resnična
- $p \ || \ q$
 - resničen natanko tedaj, ko je **vsaj eden** izmed izrazov p in q resničen
- $!p$
 - resničen natanko tedaj, ko je izraz p neresničen
- **kratkostično izvajanje**
 - $p \ \&\& \ q$: če je p neresničen, se q sploh ne preverja
 - $p \ || \ q$: če je p resničen, se q sploh ne preverja
- $!$ ima **prednost** pred $\&\&$, ta pa pred $||$
 - $op1$ ima prednost pred $op2 \implies op1$ veže močnejše kot $op2$

Primeri izrazov

Izraz	Vrednost
<code>3 > 2 && 6 < 5</code>	false
<code>0 < 8 6 == 5</code>	true
<code>false 2 == 8</code>	false
<code>1 > 6 4 == 3 6 < 9</code>	true
<code>4 != 4 && a > b</code>	false / napaka pri prev.
<code>(5 == 6) == (7 == 8)</code>	true
<code>!(2 < 3) !(5 == 4) && !true</code>	false

Primer

- program, ki izpiše, ali je prebrano leto prestopno
- izpolnjen mora biti eden od sledečih pogojev:
 - leto je deljivo s 400
 - leto je deljivo s 4, a ni deljivo s 100

```
int leto = sc.nextInt();  
if (leto % 400 == 0 || (leto % 4 == 0 && leto % 100 != 0)) {  
    System.out.println("da");  
} else {  
    System.out.println("ne");  
}
```

Primer

- Janezek mora za kazen 100-krat napisati *Ne bom več klepetal med poukom.*
- Janezkova rešitev:

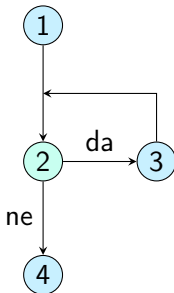
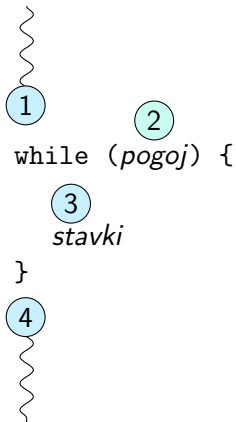
```
int stevec = 1;
while (stevec <= 100) {
    System.out.println("Ne bom več klepetal med poukom.");
    stevec = stevec + 1;
}
```

Zanka while

```
while (pogoj) {  
    stavki  
}
```

- če je *pogoj* izpolnjen, se izvedejo *stavki* in se postopek ponovi (*pogoj* se še enkrat preveri itd.)
- sicer se izvede stavek, ki sledi zanki

Zanka while



Kaj izpiše program?

```
int i = 1;
while (i <= 3) {
    System.out.println("a");
    i = i + 1;
}
System.out.println("b");
```

```
a
a
a
b
```

Zaporedje

- program, ki izpiše zaporedje števil od a do vključno b

```
int a = sc.nextInt();
int b = sc.nextInt();
int stevilo = a;
while (stevilo <= b) {
    System.out.println(stevilo);
    stevilo = stevilo + 1;
}
```

Sprotno seštevanje 1

- program, ki bere števila in jih sproti sešteva, dokler je vsota manjša ali enaka 42

```
Vnesite število: 6  
6  
Vnesite število: 7  
13  
Vnesite število: 4  
17  
Vnesite število: 10  
27  
Vnesite število: 9  
36  
Vnesite število: 10  
46
```

Sprotno seštevanje 1

```
int vsota = 0;
while (vsota <= 42) {
    System.out.print("Vnesite število: ");
    int vnos = sc.nextInt();
    vsota = vsota + vnos;
    System.out.println(vsota);
}
```


Sprotno seštevanje 2

- program, ki prebere 5 števil in sprti izpisuje njihovo vsoto

```
Vnesite število: 10  
10  
Vnesite število: 15  
25  
Vnesite število: 7  
32  
Vnesite število: 20  
52  
Vnesite število: 5  
57
```

Sprotno seštevanje 2

```
int stVnosov = 0;
int vsota = 0;

while (stVnosov < 5) {
    System.out.print("Vnesite število: ");
    int vnos = sc.nextInt();
    stVnosov = stVnosov + 1;
    vsota = vsota + vnos;
    System.out.println(vsota);
}
```

Sprotno seštevanje 3

- program, ki bere števila in sproti izpisuje njihovo vsoto, zaključi pa se, ko uporabnik vnese 5 števil ali ko vsota preseže 42

```
int stVnosov = 0;
int vsota = 0;

while (stVnosov < 5 && vsota <= 42) {
    System.out.print("Vnesite število: ");
    int vnos = sc.nextInt();
    stVnosov = stVnosov + 1;
    vsota = vsota + vnos;
    System.out.println(vsota);
}
```

Vsota vseh vhodnih števil

- program, ki izpiše vsoto vseh števil na vhodu
- če je `sc` spremenljivka tipa `Scanner`, potem ima izraz `sc.hasNextInt()` vrednost `true` natanko v primeru, če je na vhodu še kakšno število

```
int vsota = 0;
while (sc.hasNextInt()) {
    int stevilo = sc.nextInt();
    vsota = vsota + stevilo;
}
System.out.println(vsota);
```

Operatorji *op=*

- *sprem = sprem + izraz* \longrightarrow *sprem += izraz*
- podobno velja za druge aritmetične operatorje

daljša oblika	krajša oblika
<code>a = a + 3</code>	<code>a += 3</code>
<code>hitrost = hitrost - sprememba</code>	<code>hitrost -= sprememba</code>
<code>x = x * (y - 3)</code>	<code>x *= y - 3</code>
<code>p = p / q</code>	<code>p /= q</code>
<code>p = p % q</code>	<code>p %= q</code>

Operatorja ++ in --

- namesto `a += 1` lahko pišemo tudi `++a` ali `a++`
- namesto `a -= 1` lahko pišemo tudi `--a` ali `a--`
- prefiksna oblika
 - `++a`, `--a`
 - učinek: poveča/zmanjša spremenljivko
 - rezultat: nova vrednost
 - »najprej povečaj/zmanjšaj, potem uporabi«
- postfiksna oblika
 - `a++`, `a--`
 - učinek: poveča/zmanjša spremenljivko
 - rezultat: stara vrednost
 - »najprej uporabi, potem povečaj/zmanjšaj«

Operatorja ++ in --

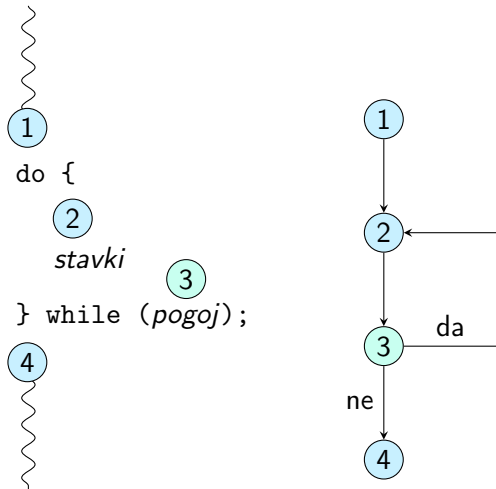
```
int a = 5;  
int b = 5;  
int c = a++;  
int d = ++b;  
System.out.println(a); // 6  
System.out.println(b); // 6  
System.out.println(c); // 5  
System.out.println(d); // 6
```

Zanka do

```
do {  
    stavki  
} while (pogoj);
```

- izvršijo se *stavki*
- če je *pogoj* izpolnjen, se postopek ponovi (*stavki* se še enkrat izvedejo in *pogoj* se še enkrat preveri)
- sicer se izvede stavek, ki sledi zanki
- *stavki* se v vsakem primeru izvršijo vsaj enkrat!

Zanka do



Primer

- program, ki nadleguje uporabnika, dokler ta ne vnese števila 42

```
do {  
    System.out.print("Odgovor na vprašanje o ...: ");  
    int odgovor = sc.nextInt();  
} while (odgovor != 42);
```

- napaka pri prevajanju
- spremenljivka odgovor izven svojega bloka ni vidna

```
int odgovor = 0;  
do {  
    System.out.print("Odgovor na vprašanje o ...: ");  
    odgovor = sc.nextInt();  
} while (odgovor != 42);
```

Ugibanje števila

- uporabnik vnese zgornjo mejo intervala (M)
- program izbere naključno število z intervala $[1, M]$
- uporabnik ugiba število, dokler mu ne uspe
- program po vsakem neuspešnem poskusu izpiše, ali je izbrano število večje ali manjše od pravkar vnesenega

Ugibanje števila

Vnesite zgornjo mejo: 50

Vaš poskus: 25

Izbrano število je večje od vnesenega.

Vaš poskus: 37

Izbrano število je manjše od vnesenega.

Vaš poskus: 31

Izbrano število je manjše od vnesenega.

Vaš poskus: 28

Izbrano število je večje od vnesenega.

Vaš poskus: 30

Izbrano število je manjše od vnesenega.

Vaš poskus: 29

Čestitke!

Naključna števila

- pomagamo si z razredom `Random` iz paketa `java.util`

```
import java.util.Random;    // razred uvozimo

public class ... {
    public static void main(String[] args) {
        ...
        // izdelamo objekt razreda
        Random random = new Random();
        ...
        // naključno število med 0 in  $n-1$ 
        int stevilo = random.nextInt(n);
        ...
    }
}
```

Ugibanje števila

```
Random random = new Random();
System.out.print("Vnesite zgornjo mejo: ");
int meja = sc.nextInt();
int izbrano = random.nextInt(meja) + 1;
int poskus = 0;

do {
    System.out.print("Vaš poskus: ");
    poskus = sc.nextInt();
    if (izbrano > poskus) {
        System.out.println("Izbrano število je večje.");
    } else if (izbrano < poskus) {
        System.out.println("Izbrano število je manjše.");
    }
} while (poskus != izbrano);
```

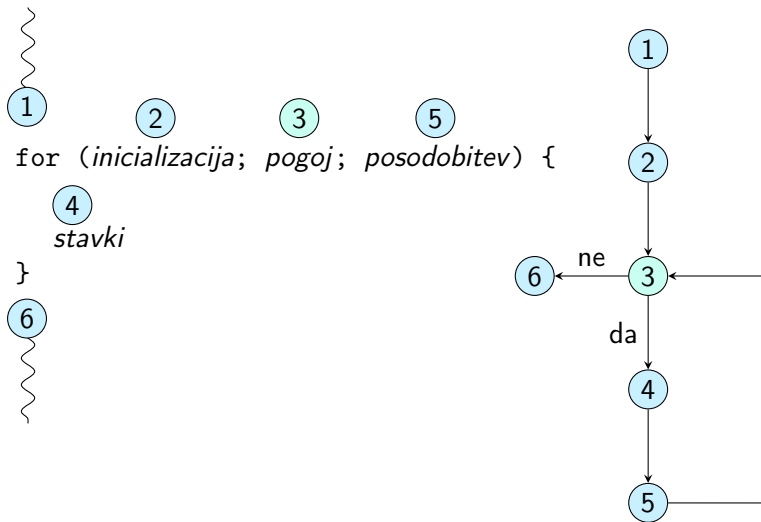
Zanka for

```
for (inicializacija; pogoj; posodobitev) {  
    stavki  
}
```

je okrajšava za

```
inicializacija;  
while (pogoj) {  
    stavki  
    posodobitev;  
}
```

Zanka for



Zaporedje števil od a do vključno b

- z zanko while

```
int stevilo = a;
while (stevilo <= b) {
    System.out.println(stevilo);
    stevilo++;
}
```

- z zanko for

```
for (int stevilo = a; stevilo <= b; stevilo++) {
    System.out.println(stevilo);
}
```

- spremenljivka stevilo obstaja samo do konca telesa zanke
- **izjema!** (sicer spremenljivka obstaja do konca bloka)

for vs. while/do

- `for`
 - sprehod po intervalu
 - število korakov je znano vnaprej
- `while/do`
 - ponavlja, dokler pogoj ni izpolnjen
 - število korakov ni znano vnaprej

Izpis velikih črk angleške abecede

- znaki od 'A' do 'Z' so predstavljeni z zaporednimi ASCII-kodami
- 'A' == 65, 'B' == 66 itd.

```
for (char crka = 'A'; crka <= 'Z'; crka++) {  
    System.out.println(crka);  
}
```

Kdo je najboljši?

- program, ki prebere število n in n rezultatov atletov ter izpiše zaporedno številko najboljšega in njegov rezultat

```
Koliko atletov tekmuje? 5  
Vnesite dolžino skoka za 1. atleta: 635  
Vnesite dolžino skoka za 2. atleta: 680  
Vnesite dolžino skoka za 3. atleta: 671  
Vnesite dolžino skoka za 4. atleta: 695  
Vnesite dolžino skoka za 5. atleta: 667  
Najboljši je 4. atlet (695).
```

Kdo je najboljši?

- spremenljivke
 - `dolzina`: dolžina skoka trenutnega atleta
 - `st`: zaporedna številka trenutnega atleta
 - `najDolzina`: dosedanji rekord
 - `najSt`: zaporedna številka dosedanjega rekorderja
- če je trenutna dolžina večja od doslej največje, posodobimo spremenljivki `najDolzina` in `najSt`

```
if (dolzina > najDolzina) {  
    najDolzina = dolzina;  
    najSt = st;  
}
```

Kdo je najboljši?

st	dolzina	najDolzina	najSt
		0	0
1	635	635	1
2	680	680	2
3	671	680	2
4	695	695	4
5	667	695	4

Kdo je najboljši?

```
System.out.print("Koliko atletov tekmuje? ");
int stAtletov = sc.nextInt();
int najDolzina = 0;
int najSt = 0;

for (int st = 1; st <= stAtletov; st++) {
    System.out.print("Vnesite dolžino skoka za "
                     + st + ". atleta: ");
    int dolzina = sc.nextInt();
    if (dolzina > najDolzina) {
        najDolzina = dolzina;
        najSt = st;
    }
}
System.out.println("Najboljši je " + najSt +
                   ". atlet (" + najDolzina + ").");
```

Praštevila

- program, ki prebere število n in izpiše vsa praštevila med 2 in vključno n
- primer za $n = 20$

```
2  
3  
5  
7  
11  
13  
17  
19
```


Praštevila

- osnovna ideja

```
for (int kandidat = 2; kandidat <= n; kandidat++) {  
    if (kandidat je praštevilo) {  
        System.out.println(kandidat);  
    }  
}
```

- število je praštevilo, če ima natanko dva delitelja

```
for (int kandidat = 2; kandidat <= n; kandidat++) {  
    izračunaj število deliteljev kandidata  
    if (število deliteljev == 2) {  
        System.out.println(kandidat);  
    }  
}
```

Praštevila

```
for (int kandidat = 2; kandidat <= n; kandidat++) {  
  
    // izračunaj število deliteljev kandidata  
    int stDeliteljev = 0;  
    for (int d = 1; d <= kandidat; d++) {  
        if (kandidat % d == 0) {  
            stDeliteljev++;  
        }  
    }  
  
    if (stDeliteljev == 2) {  
        System.out.println(kandidat);  
    }  
}
```

Poštevanka

- program, ki prebere število n in izpiše tabelo zmnožkov števil od 1 do n
- primer za $n = 6$

1	2	3	4	5	6
2	4	6	8	10	12
3	6	9	12	15	18
4	8	12	16	20	24
5	10	15	20	25	30
6	12	18	24	30	36

Poštevanka

- osnovna ideja

```
for (int i = 1; i <= n; i++) {  
    izpiši i-to vrstico zmnožkov  
}
```

- izpis i-te vrstice zmnožkov

```
for (int j = 1; j <= n; j++) {  
    int zmnozek = i * j;  
    System.out.print(zmnozek + " ");  
}  
System.out.println();
```

Poštevanka

```
for (int i = 1; i <= n; i++) {  
    // izpiši i-to vrstico zmnožkov  
    for (int j = 1; j <= n; j++) {  
        int zmnozek = i * j;  
        System.out.print(zmnozek + " ");  
    }  
    System.out.println();  
}
```

- manjka še desna poravnava

System.out.printf

- `System.out.printf(niz, arg_1, arg_2, ...);`
- izpiše se *niz*, vendar pa se *i*-ti podniz oblike `%*` zamenja z vrednostjo *arg_i*
 - izjema sta podniza `%n` (prelom vrstice) in `%%` (znak %)
- podnizi oblike `%*` podajajo tip in morebitna oblikovna določila
 - `%d`: celo število (byte, short, int, long)
 - `%f`: realno število (float, double)
 - `%c`: znak (char)
 - `%s`: niz (String)
 - `%b`: logična vrednost (boolean)

System.out.printf

```
int odgovor = 42;
double pi = 3.1415;
char crka = 'j';
String beseda = "java";
boolean danesJeSreda = true;

System.out.printf("Odgovor se glasi %d.%n", odgovor);
// Odgovor se glasi 42.

System.out.printf("Obseg kroga je %f-kratnik premera.%n", pi);
// Obseg kroga je 3.141500-kratnik premera.

System.out.printf("Beseda \"%s\" se začne s črko %c.%n",
                  beseda, crka);
// Beseda "java" se začne s črko j.

System.out.printf("danes je sreda: %b%n", danesJeSreda);
// danes je sreda: true
```

Oblikovna določila

- `%wx`
 - širina izpisa w , desna poravnava
- `%-wx`
 - širina izpisa w , leva poravnava
- `%0wd`
 - širina izpisa w , polnjenje z vodilnimi ničlami
- `%.df`
 - realno število, zapisano na d decimalk
- `%w.df`
 - širina izpisa w , d decimalk

Oblikovna določila

```
int a = 42;
double b = 3.1415;
String niz = "java";

System.out.printf("[%5d]%n", a);    // [   42]
System.out.printf("[%10s]%n", niz); // [          java]
System.out.printf("[%5d]%n", a);    // [42    ]
System.out.printf("[%05d]%n", a);   // [00042]
System.out.printf("[%0.3f]%n", b);  // [3.142]
System.out.printf("[%10.3f]%n", b); // [          3.142]
```

Poštevanka s pravilno obliko izpisa

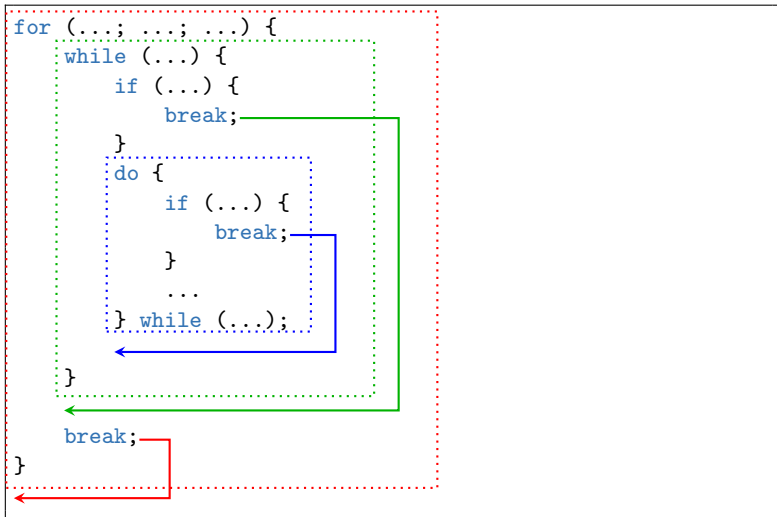
```
1 2 3 4 5 6
2 4 6 8 10 12
3 6 9 12 15 18
4 8 12 16 20 24
5 10 15 20 25 30
6 12 18 24 30 36
```

- števila so desno poravnana
- izpis na 4 mesta

```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        int zmnozek = i * j;
        System.out.printf("%4d", zmnozek);
    }
    System.out.println();
}
```

Stavek break

- prekine zanko, v kateri se **neposredno** nahaja



Izpis prvih 5 deliteljev števila n

rešitev z break

```
int stevec = 0;
for (int d = 1; d <= n; d++) {
    if (n % d == 0) {
        System.out.println(d);
        stevec++;
        if (stevec == 5) {
            break;
        }
    }
}
```

rešitev brez break

```
int stevec = 0;
for (int d = 1;
      d <= n && stevec < 5;
      d++) {
    if (n % d == 0) {
        System.out.println(d);
        stevec++;
    }
}
```

Stavek continue

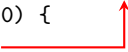
- v zanki, v kateri se neposredno nahaja, skoči na preverjanje pogoja (while, do) oz. posodobitev (for)

Stavek continue

```
for (...; ...; ...) {  
    while (...) {  
        if (...) {  
            continue;  
        }  
        do {  
            if (...) {  
                continue;  
            }  
            ...  
        } while (...);  
    }  
    if (...) {  
        continue;  
    }  
    ...  
}
```

Stavek continue

```
for (int i = 1; i <= 10; i++) {  
    if (i % 3 == 0) {  
        continue;  
    }  
    for (int j = 1; j <= i; j++) {  
        System.out.print("*");  
    }  
    System.out.println();  
}
```

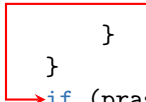


```
*  
**  
***  
****  
*****  
*****  
*****  
*****  
*****
```

Praštevila (učinkovitejši pristop)

- vsakega kandidata k delimo s števili od 2 do $k - 1$
- če se katerokoli deljenje izide, potem
 - vemo, da k ni praštevilo
 - prekinemo zanko in nadaljujemo s kandidatom $k + 1$

```
for (int kandidat = 2; kandidat <= n; kandidat++) {  
    boolean prastevalo = true;  
    for (int d = 2; d < kandidat; d++) {  
        if (kandidat % d == 0) {  
            prastevalo = false;  
            break;  
        }  
    }  
    if (prastevalo) {  
        System.out.println(kandidat);  
    }  
}
```



Praštevila še učinkoviteje

- z izjemo števila 2 so vsa praštevila liha
- pri vsakem (lih) kandidatu je smiselno preverjati samo deljenje z lihimi števili

```
System.out.println(2);
for (int kandidat = 3; kandidat <= n; kandidat += 2) {
    boolean prastevilo = true;
    for (int d = 3; d < kandidat; d += 2) {
        if (kandidat % d == 0) {
            prastevilo = false;
            break;
        }
    }
    if (prastevilo) {
        System.out.println(kandidat);
    }
}
```

Še malo bolje ...

- zadošča, da preverjamo deljivost do korena kandidata

```
System.out.println(2);
for (int kandidat = 3; kandidat <= n; kandidat += 2) {
    boolean prastevilo = true;
    int meja = (int) Math.round(Math.sqrt(kandidat));
    for (int d = 3; d <= meja; d += 2) {
        if (kandidat % d == 0) {
            prastevilo = false;
            break;
        }
    }
    if (prastevilo) {
        System.out.println(kandidat);
    }
}
```

Stavek switch

```
switch (izraz) {  
    case konstanta_1:  
        stavki_1  
        break;  
  
    case konstanta_2:  
        stavki_2  
        break;  
    ...  
  
    case konstanta_n:  
        stavki_n  
        break;  
  
    default:  
        stavki_{n + 1}  
        break;  
}
```

- *izraz* mora biti tipa byte, short, int, char, String ali enum
- če velja *izraz* == *konstanta_1*, se izvršijo *stavki_1*
- če velja *izraz* == *konstanta_2*, se izvršijo *stavki_2*
- ...
- če velja *izraz* == *konstanta_n*, se izvršijo *stavki_n*
- sicer se izvršijo *stavki_{n + 1}*
- odsek default lahko izpustimo

Primer

- program, ki izpiše opis podane šolske ocene

```
switch (ocena) {  
    case 5:  
        System.out.println("odlično");  
        break;  
    case 4:  
        System.out.println("prav dobro");  
        break;  
    case 3:  
        System.out.println("dobro");  
        break;  
    case 2:  
        System.out.println("zadostno");  
        break;  
    case 1:  
        System.out.println("nezadostno");  
        break;  
}
```

break znotraj switch

- break prekine najbolj tesno oklepajoči stavek for, while, do ali switch, v katerem se nahaja
- če break izpustimo, se izvajanje nadaljuje pri naslednjem odseku case

break znotraj switch

```
switch (ocena) {  
    case 5:  
        System.out.println("odlično");  
    case 4:  
        System.out.println("prav dobro");  
        break;  
    case 3:  
        System.out.println("dobro");  
    case 2:  
        System.out.println("zadostno");  
    case 1:  
        System.out.println("nezadostno");  
        break;  
}
```

ocena	izpis
5	odlično prav dobro
4	prav dobro
3	dobro zadostno nezadostno
2	zadostno nezadostno
1	nezadostno

Število dni v mesecu

- program, ki prebere zaporedno številko meseca in izpiše število dni v tem mesecu v navadnem (neprestopnem) letu

```
int stDni = 0;
switch (mesec) {
    case 2:
        stDni = 28;
        break;
    case 4:
    case 6:
    case 9:
    case 11:
        stDni = 30;
        break;
    default:
        stDni = 31;
        break;
}
System.out.println(stDni);
```

Doseg spremenljivk

- spremenljivka, deklarirana znotraj odseka case, obstaja **do konca stavka switch**
 - ne samo do konca tistega odseka case!

```
switch (t) {  
    case 1:  
        int a = 3;  
        ...  
        break;  
  
    case 2:  
        int a = 4; // napaka pri prevajanju  
        ...  
        break;  
}
```


Doseg spremenljivk

- lahko pa stavke znotraj odseka case zapremo v blok

```
switch (t) {  
    case 1: {  
        int a = 3;  
        ...  
        break;  
    } // spremenljivka a ne obstaja več  
  
    case 2: {  
        int a = 4;  
        ...  
        break;  
    }  
}
```

Pogojni operator

- trojiški operator
- izraz s pogojnim operatorjem

pogoj ? *izraz_1* : *izraz_2*

- če je *pogoj* izpolnjen, se izračuna *izraz_1*
- sicer se izračuna *izraz_2*
- rezultat izračuna postane rezultat celotnega izraza

Pogojni operator

- pogojni operator lahko nadomesti nekatere pogojne stavke ...

```
if (tocke >= 50) {  
    System.out.println("opravil");  
} else {  
    System.out.println("padel");  
}
```

- ... vendar ne takole ...

```
(tocke >= 50) ? (System.out.println("opravil"))  
              : (System.out.println("padel"));
```

- ... ampak takole ...

```
System.out.println((tocke >= 50) ? ("opravil") : ("padel"));
```

- `System.out.println(...)` ni izraz, ker nima vrednosti

Primeri

- absolutna vrednost števila a

```
int absolutna = (a > 0) ? (a) : (-a);
```

- manjše izmed števil a in b

```
int manjse = (a < b) ? (a) : (b);
```

- najmanjše izmed števil a, b in c

```
int najmanjse = (a < b) ?  
                 (a < c ? a : c) :  
                 (b < c ? b : c);
```

Zanimivost pri operatorjih =, += itd.

- izraz ima vrednost
- stavek ima učinek
- *spremenljivka* = *izraz*
 - *izraz z učinkom*
 - učinek: vrednost izraza *izraz* se vpiše v spremenljivko
 - vrednost: nova vrednost spremenljivke *spremenljivka*
 - podobno velja za operatorje +=, *= itd.

```
int a = 3;  
int b = (a = 5) + 6;    // a: 5, b: 11  
a = (b -= 3) - 1;       // a: 7, b: 8
```

Pregled javanskih operatorjev

- tabela v knjigi!
- prednostni nivoji
 - npr. operatorji $*$, $/$ in $\%$ tvorijo višji prednostni nivo kot operatorja $+$ in $-$
 - izraz $3 + 4 * 5$ se izračuna kot $3 + (4 * 5)$
- asociativnost
 - pove, kako se združujejo operatorji na istem prednostnem nivoju
 - leva asociativnost
 - izraz $30 / 5 * 4$ se izračuna kot $(30 / 5) * 4$
 - desna asociativnost
 - izraz $a = b = c$ se izračuna kot $a = (b = c)$