Uvod v programiranje

Fakulteta za informacijske študije

Računalništvo in spletne tehnologije

2019/20

17. februar 2020

Vsebina

- Krmilni stavki:
  - pogojni
  - zanke
  - vejitveni

Krmilni stavki

Stavki se v programu izvajajo po vrstnem redu (od zgoraj navzdol), razen če uporabimo krmilne stavke:

- Pogojni (odločitveni) stavki:
  - *if-then*
  - *if-then-else*
  - *switch*

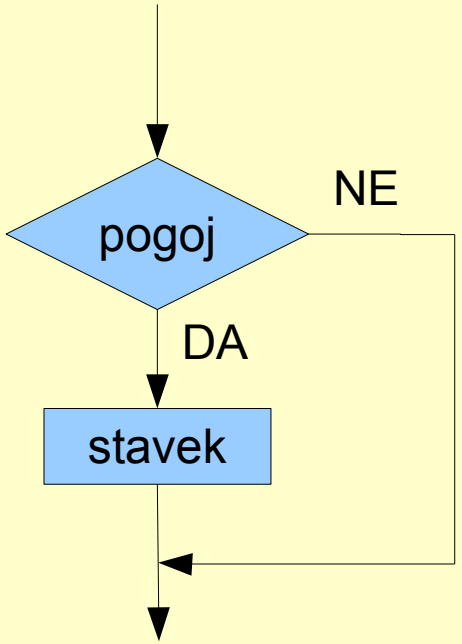
Krmilni stavki (nad.)

- Zanke:
  - *while*
  - *do-while*
  - *for*
- Vejitveni stavki:
  - *break*
  - *continue*
  - *return*

Pogojni stavek `'if-then'`

- Nam omogoča, da se del kode izvede samo v primeru, ko ima pogoj vrednost `true`. (Ko ima pogoj vrednost `false`, se izvajanje nadaljuje za stavkom *if-then*)

```
if (pogoj) {
   stavek;
}
```

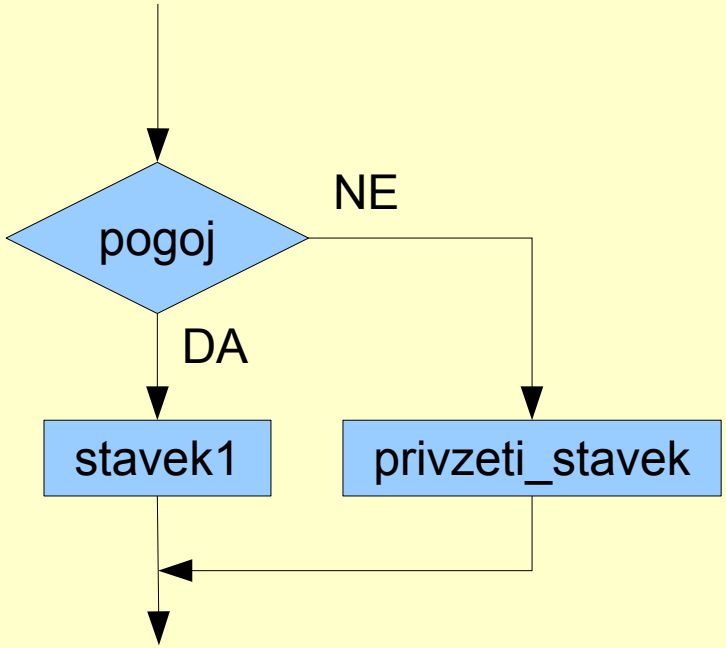


- Če je *stavek* enostaven, oklepaji niso obvezni
- *pogoj* mora biti izraz tipa `boolean`

Pogojni stavek 'if-then-else'

- Nam omogoča, da se del kode izvede tudi, ko ima pogoj vrednost `false`

```
if (pogoj) {
   stavek1;
} else {
   privzeti_stavek;
}
```



Primeri

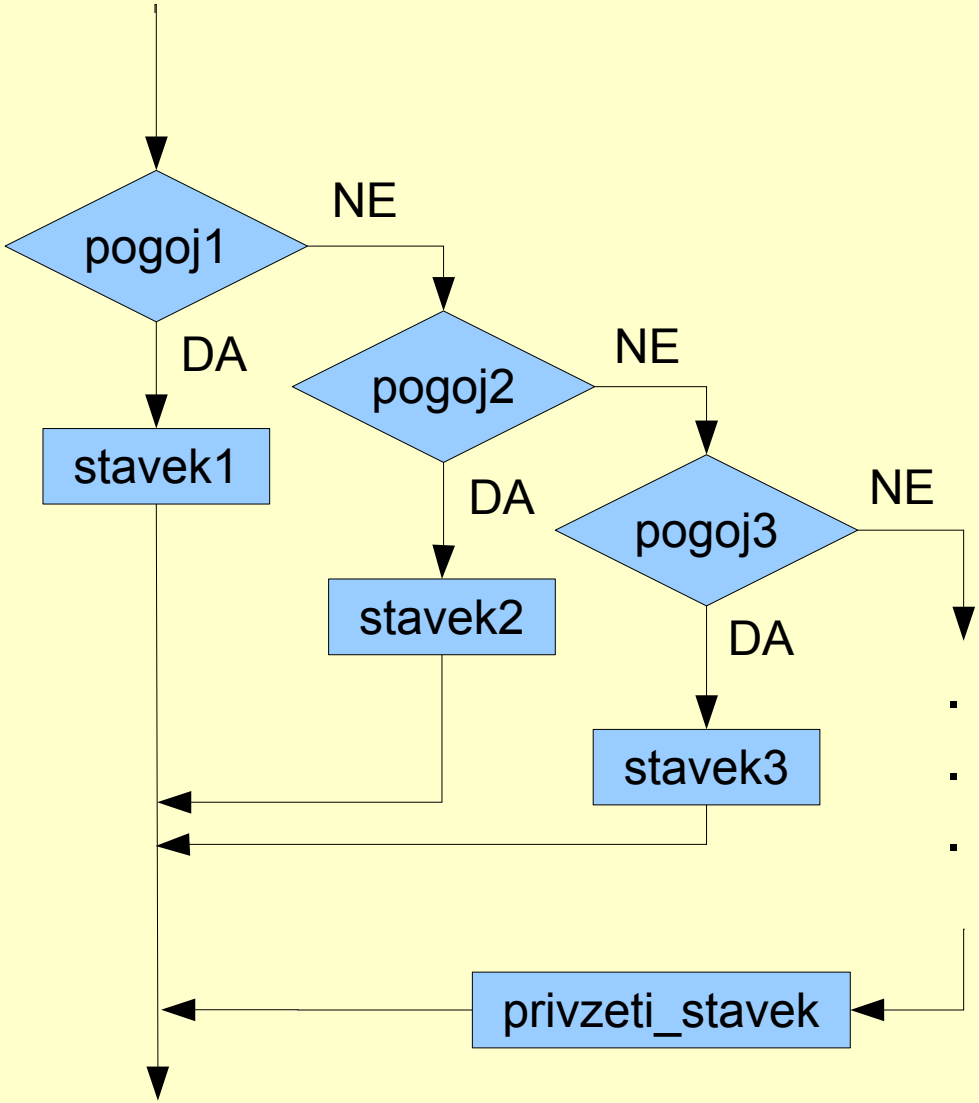
```
public class IfThen {
  public static void main(String[] args) {
    int i = (Integer.valueOf(args[0])).intValue();
    if (i > 5) {
      System.out.println("i je večji od 5!");
    }
  }
}

public class IfThenElse {
  public static void main(String[] args) {
    int i = (Integer.valueOf(args[0])).intValue();
    if (i > 5) {
      System.out.println("i je večji od 5!");
    } else {
      System.out.println("i je manjši ali enak 5!");
    }
  }
}
```

Gnezdenje stavkov 'if-then-else'

```
if (pogoj1) {
   stavek1;
} else if (pogoj2) {
   stavek2;
} else if (pogoj3) {
   stavek3;
} ...

   else {
   privzeti_stavek;
}
```



Primer

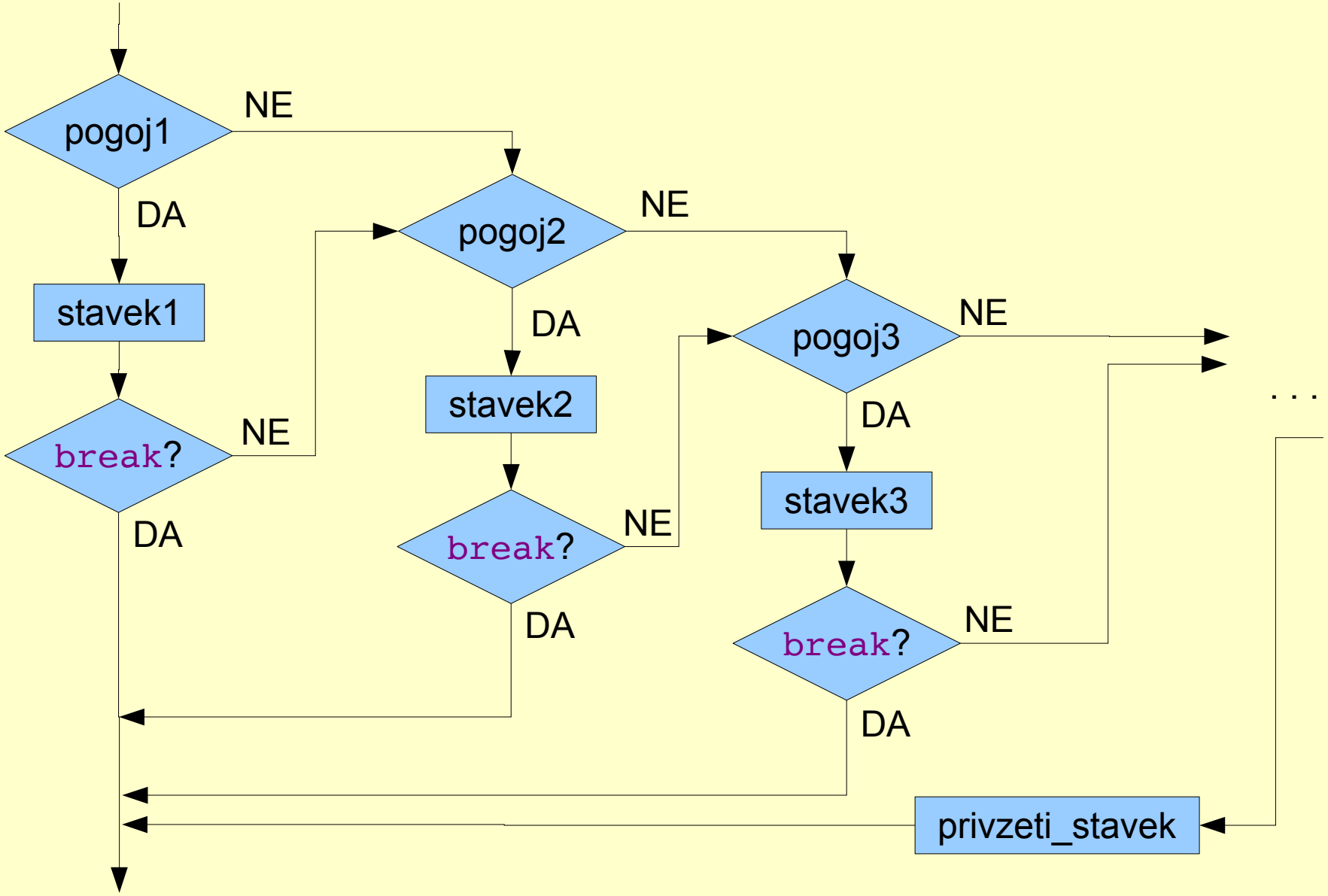
```
public class Oceni {
  public static void main(String[] args) {
    int tocke = (Integer.valueOf(args[0])).intValue();
    String ocena;
    if (tocke >= 90) {
      ocena = "10";
    } else if (tocke >= 80) {
      ocena = "9";
    } else if (tocke >= 70) {
      ocena = "8";
    } else if (tocke >= 60) {
      ocena = "7";
    } else if (tocke >= 50) {
      ocena = "6";
    } else {
      ocena = "Nezadostno";
    }
    System.out.println("Ocena = " + ocena);
  }
}
```

Stavek `'switch'`

```
switch (sprem) {
  case vred1: stavek1; [break;]
  case vred2: stavek2; [break;]
  case vred3: stavek3; [break;]
  ...
  case vredN: stavekN; [break;]
  default: privzeti_stavek; [break;]
}
```

- Dovoljeni tipi za spremenljivko `sprem` so: `byte`, `short`, `int`, `char`, njihovi t.i. 'ovojni' (wrapper) tipi (`Byte`, `Short`, `Integer`, `Character`) in t.i. 'oštevilčeni' (enumerated) tipi

Diagram poteka stavka '`switch`'



Primer

```
public class Switch {
  public static void main(String[] args) {
    int mesec = (Integer.valueOf(args[0])).intValue();
    switch (mesec) {
      case 1: System.out.println("januar"); break;
      case 2: System.out.println("februar"); break;
      case 3: System.out.println("marec"); break;
      case 4: System.out.println("april"); break;
      case 5: System.out.println("maj"); break;
      case 6: System.out.println("junij"); break;
      case 7: System.out.println("julij"); break;
      case 8: System.out.println("avgust"); break;
      case 9: System.out.println("september"); break;
      case 10: System.out.println("oktober"); break;
      case 11: System.out.println("november"); break;
      case 12: System.out.println("december"); break;
      default: System.out.println("napačen mesec"); break;
    }
  }
}
```

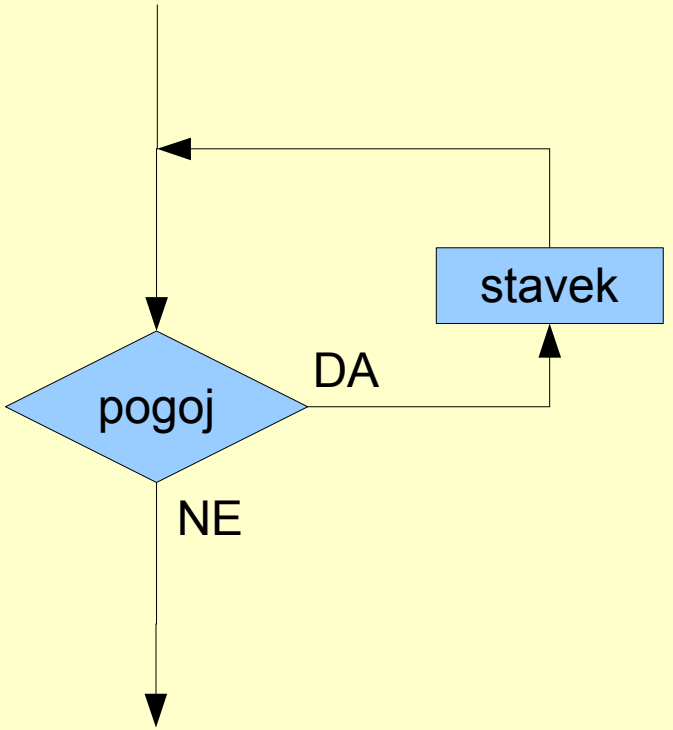
Primer 2

```
public class SteviloDniVMesecu {
  public static void main(String[] args) {
    int mesec = (Integer.valueOf(args[0])).intValue();
    int stDni = 0;
    switch (mesec) {
      case 1:
      case 3:
      case 5:
      case 7:
      case 8:
      case 10:
      case 12: stDni = 31; break;
      case 4:
      case 6:
      case 9:
      case 11: stDni = 30; break;
      case 2: stDni = 28; break;
      default: System.out.println("Napačen mesec"); break;
    }
    System.out.println("Število dni = " + stDni);
  }
}
```

Zanka `'while'`

- Najpreprostejša zanka, ki ponavlja določen del kode (*stavek*) dokler ima *pogoj* vrednost `true`

```
while (pogoj) {
   stavek;
}
```



- Pogoj se testira pred vsako izvedbo *stavka*

Primer

```
public class While {
  public static void main(String[] args) {
    int stevec = 1;
    while (stevec < 11) {
      System.out.println("Številec je: " + stevec);
      stevec++;
    }
  }
}
```

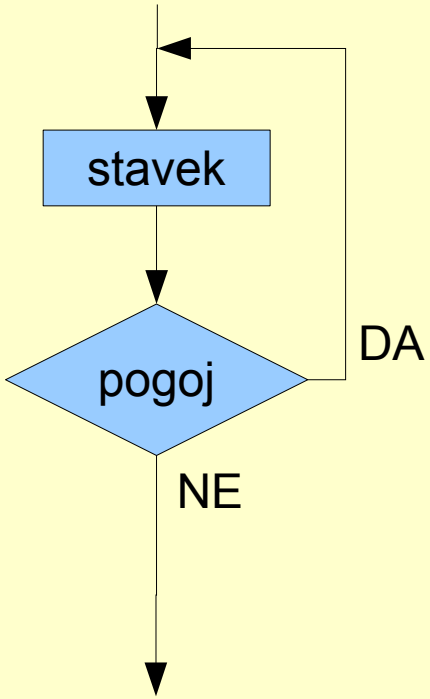
Neskončna zanka:

```
  while (true) {
    stavek;
  }
```

Zanka 'do-while'

- Podobna kot zanka *while,* le da se *pogoj* testira na koncu. Zato se *stavek* vedno izvede vsaj enkrat!

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



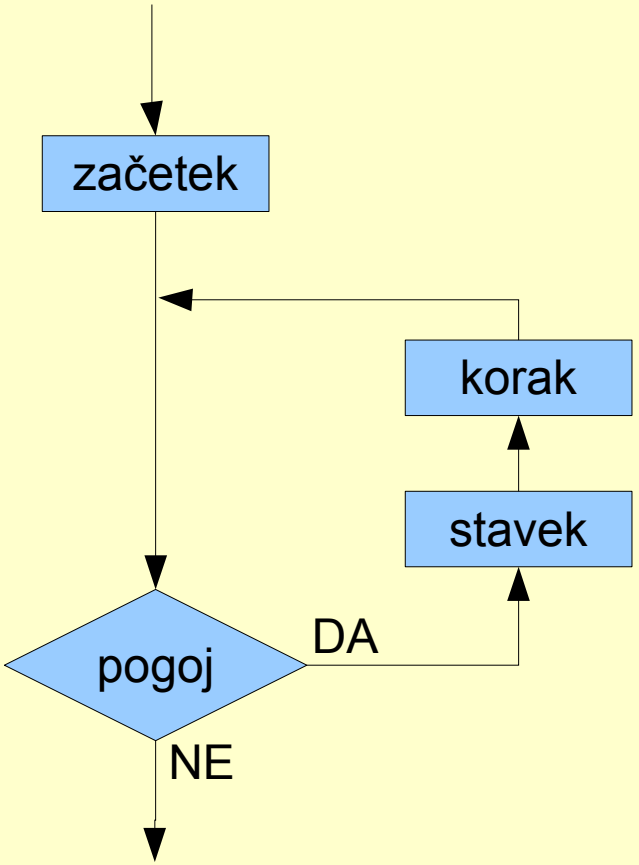
Primer

```
public class DoWhile {
  public static void main(String[] args) {
    int stevec = 1;
    do {
      System.out.println("Števec je: " + stevec);
      stevec++;
    } while (stevec < 11);
  }
}
```

Zanka 'for'

- Posebej priročna za 'sprehajanje' po množici vrednosti

```
for (začetek; pogoj; korak){
   stavek;
}
```



Primer

```
public class For {
  public static void main(String[] args) {
    for (int i = 1; i < 11; i++){
      System.out.println("Števec je: " + i);
    }
  }
}
```

- *začetek* se izvede samo na začetku
- Nato se preveri *pogoj*:
  - če ima vrednost `false`, se zanka konča
  - če ima vrednost `true` se po vrsti izvedeta stavka *stavek* in *korak*, nato se spet preveri *pogoj*
- Spremenljivka deklarirana znotraj stavka *začetek* (`i`) je dostopna le znotraj zanke *for!*

Izboljšana zanka `'for'`

- Za sprehajanje po tabelah (ali zbirkah - *Collections*) lahko uporabljamo t.i. *izboljšano* zanko *for*:

```
for (spremenljivka: tabela){
   stavek;
}
```

- Elementi *tabele* morajo biti enakega tipa kot *spremenljivka*

Primer

```
public class ForEach {
  public static void main(String[] args) {
    int[] tabela = {1,2,3,4,5,6,7,8,9,10};
    for (int i : tabela) {
      System.out.println("Števec je: " + i);
    }
  }
}
```

- Bolj pregledna kot standardna zanka *for*
- Na voljo le v *Javi* verzije 5.0 (1.5) ali novejši

Stavek `'break'`

- Omogoča, da predčasno prekinemo stavek *switch* ali zanko *while*, *do-while* ali *for*
- Izvajanje programa se nadaljuje za prekinjenim stavkom ali zanko
- Poznamo dva tipa stavka *break*:
  - *navadni (neoznačeni):* prekine notranjo zanko (prvo zanko okrog *break* stavka)
  - *označeni:* prekine poljubno zanko z oznako

Primer: navadni stavek 'break'

```
public class NavadniBreak {
  public static void main(String[] args) {
    int[] tabela = {1,2,3,4,5,6,7,8,9,10};
    int iskano = 5;
    int i;
    boolean nasel = false;
    for (i = 0; i < tabela.length; i++) {
      if (tabela[i] == iskano) {
        nasel = true;
        break;
      }
    }
    if (nasel) {
      System.out.println("Našel število " + iskano
                         + " na mestu " + i + ".");
    } else {
      System.out.println("Števila " + iskano
                        + " ni v tabeli.");
    }
  }
}
```

Primer: *označeni* stavek 'break'

```
public class OznaceniBreak {
  public static void main(String[] args) {
    int[][] tabela = {{1,2,3},{4,5,6},{7,8,9,10}};
    int iskano = 5;
    int i, j = 0;
    boolean nasel = false;
  zunanja:
    for (i = 0; i < tabela.length; i++) {
      for (j = 0; j < tabela[i].length; j++) {
        if (tabela[i][j] == iskano) {
          nasel = true;
          break zunanja;
        }
      }
    }
    if (nasel) {
      System.out.println("Našel število " + iskano
                         + " na mestu " + i + "," + j + ".");
    } else {
      System.out.println("Števila " + iskano + " ni v tabeli.");
    }
  }
}
```

Stavek `'continue'`

- Omogoča, da predčasno prekinemo trenutno ponovitev zanke *while*, *do-while* ali *for*
- Izvajanje programa se nadaljuje s testiranjem pogoja zanke katere ponovitev smo prekinili
- Podobno kot pri stavku *break* poznamo dva tipa stavka *continue*:
  - *navadni (neoznačeni):* prekine ponovitev notranje zanke (prve zanke okrog *continue* stavka)
  - *označeni:* prekine ponovitev poljubne zanke z oznako

Primer: navadni stavek `continue`

```
public class NavadniContinue {
  public static void main(String[] args) {
    String niz = "riba reže raci rep";
    int stRjev = 0;
    for (int i = 0; i < niz.length(); i++) {
      if (niz.charAt(i) != 'r')
        continue;
      stRjev++;
    }
    System.out.println("Našel " + stRjev +
                                 " r-je v nizu.");
  }
}
```

Primer: *označeni* stavek `continue`

```
public class OznaceniContinue {
  public static void main(String[] args) {
    String niz = "Tale niz vsebuje polno podnizov.";
    String podniz = "pod";
    boolean nasel = false;
    int zadnji = niz.length() - podniz.length();
  zunanja:
    for (int i = 0; i <= zadnji; i++) {
      int n = podniz.length();
      int j = i;
      int k = 0;
      while (n-- != 0) {
        if (niz.charAt(j++) != podniz.charAt(k++)) {
          continue zunanja;
        }
      }
      nasel = true;
      break zunanja;
    }
    System.out.println(nasel ? "Našel." : "Nisem našel.");
  }
}
```

Stavek `'return'`

- Zaključi izvajanje trenutne metode
- Izvajanje programa se nadaljuje za mestom od koder je bila metoda poklicana
- Če stavek *return* uporabimo znotraj metode *main*, zaključimo izvajanje programa
- Poznamo dve obliki stavka *return*:
  - takega, ki vrne vrednost

    `return` *vrednost*`;`
  - takega, ki ne vrne ničesar

    `return;`