Fakulta riadenia a informatiky ŽU


# Viacrozmerné pole

# Pojmy zavedené v 7. prednáške

- Obaľovacie triedy
  - Primitívne typy ako objekty
- Knižnice
  - balíčky
  - príkaz import

# Pojmy zavedené v 7. prednáške

- Cykly
  - for
  - do-while
- Operatory ++, --

# Pojmy zavedené v 7. prednáške

- Pole
  - kontajner s pevným počtom prvkov
  - definícia
  - Vytvorenie a inicializácia
  - práca s poľom ako celkom
  - práca s prvkami poľa
  - dĺžka poľa - length

# Pojmy zavedené v 7. prednáške

- Diagramy aktivít
  - foreach
  - while
  - for
  - do-while

# Cieľ

- Viacrozmerné polia

- Vnorené cykly
- this
- Sekcie rozhrania
- Základná práca so súbormi

- Projekt: Sudoku

# Sudoku

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 9 | | | | 8 | | | 5 | |
| 2 | 5 | | 7 | | | 9 | | 4 |
| | | | | | | | 8 | 6 |
| | 8 | | 1 | 3 | | | | 2 |
| | | 6 | | 4 | | 1 | | |
| 5 | | | | 6 | 9 | | 4 | |
| 3 | 7 | | | | | | | |
| 8 | | 2 | | | 3 | | 1 | 5 |
| | 1 | | | 9 | | | | 3 |

# Sudoku

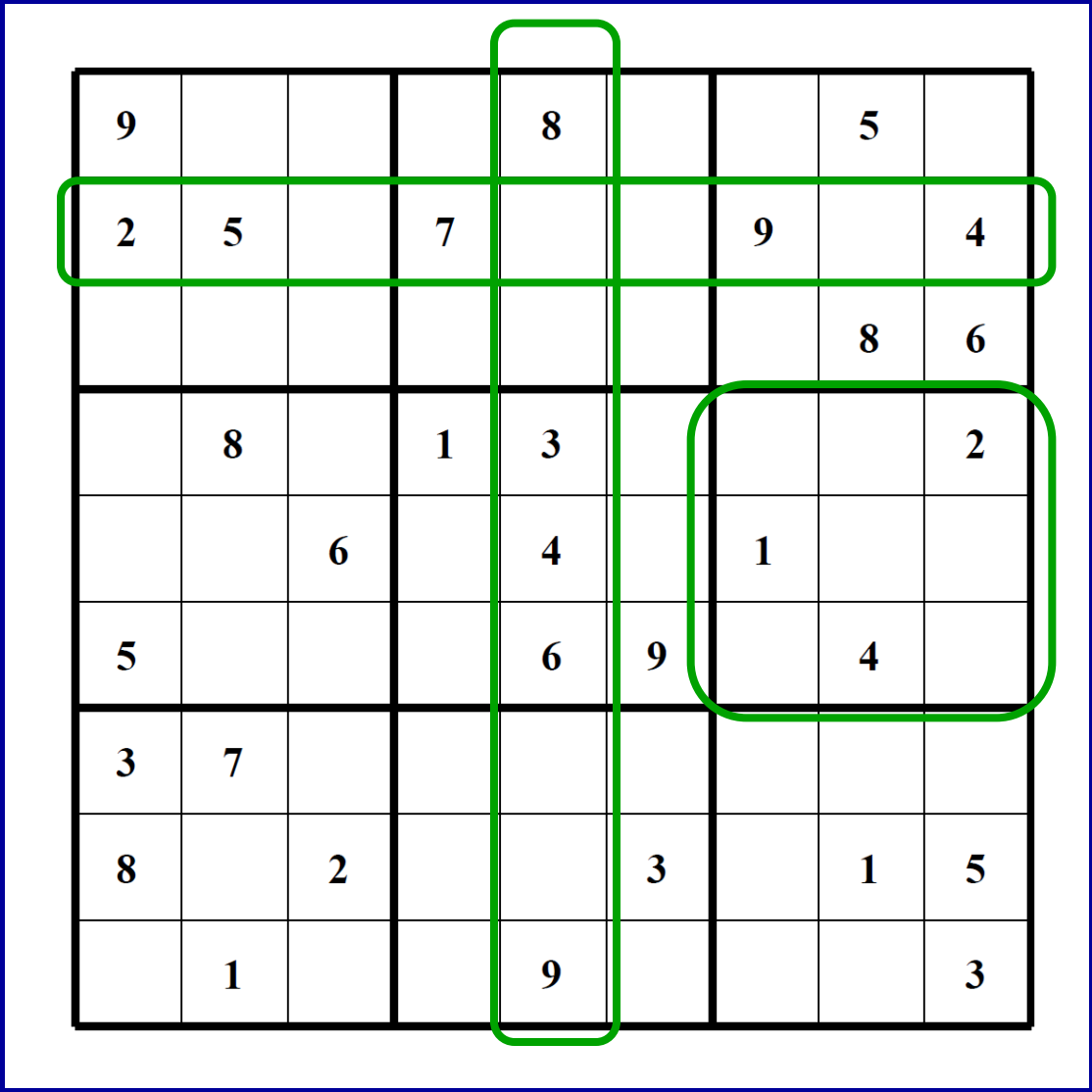
- Hlavolam – v každých novinách

- Cieľ: čiastočne vyplnenú mriežku doplniť tak, aby obsahovala každé číslo 1 až 9 práve raz v troch rôznych zoskupeniach.

# Sudoku

- Mriežka 9x9 políčok (aj iné rozmery)
- Tri typy zoskupení políčok mriežky
  - riadky – 9 riadkov
  - stĺpce – 9 stĺpcov
  - bloky 3x3 – 9 blokov v zostave 3x3

# Sudoku



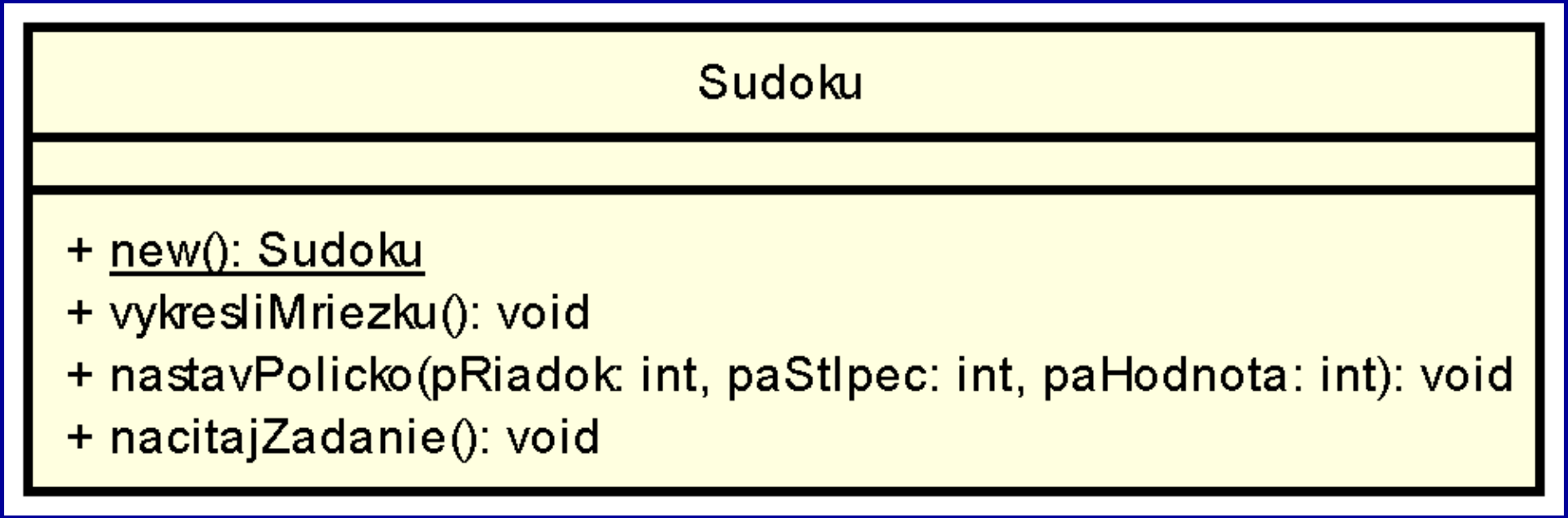
# Sudoku – 1. cieľ projektu

- Podpora pre riešiteľa

- Požadované funkcie:
  - zobrazenie mriežky
  - vloženie čísla do políčka
  - načítanie zadania

  - priebežná kontrola pravidiel

# Sudoku – rozhranie



# Sudoku – mriežka

- Doteraz – jednorozmerné problémy
  - Diár – zápis poznámok pod sebou
  - Analyzátor logu – hodiny idú po sebe
- Sudoku – dvojrozmerná mriežka
  - Má políčka vedľa seba aj pod sebou
- Podobné úlohy
  - Šach, Dáma, Skákaná – šachovnica
  - Krížovky
  - Maľované krížovky
  - Matematika – matice
  - ...

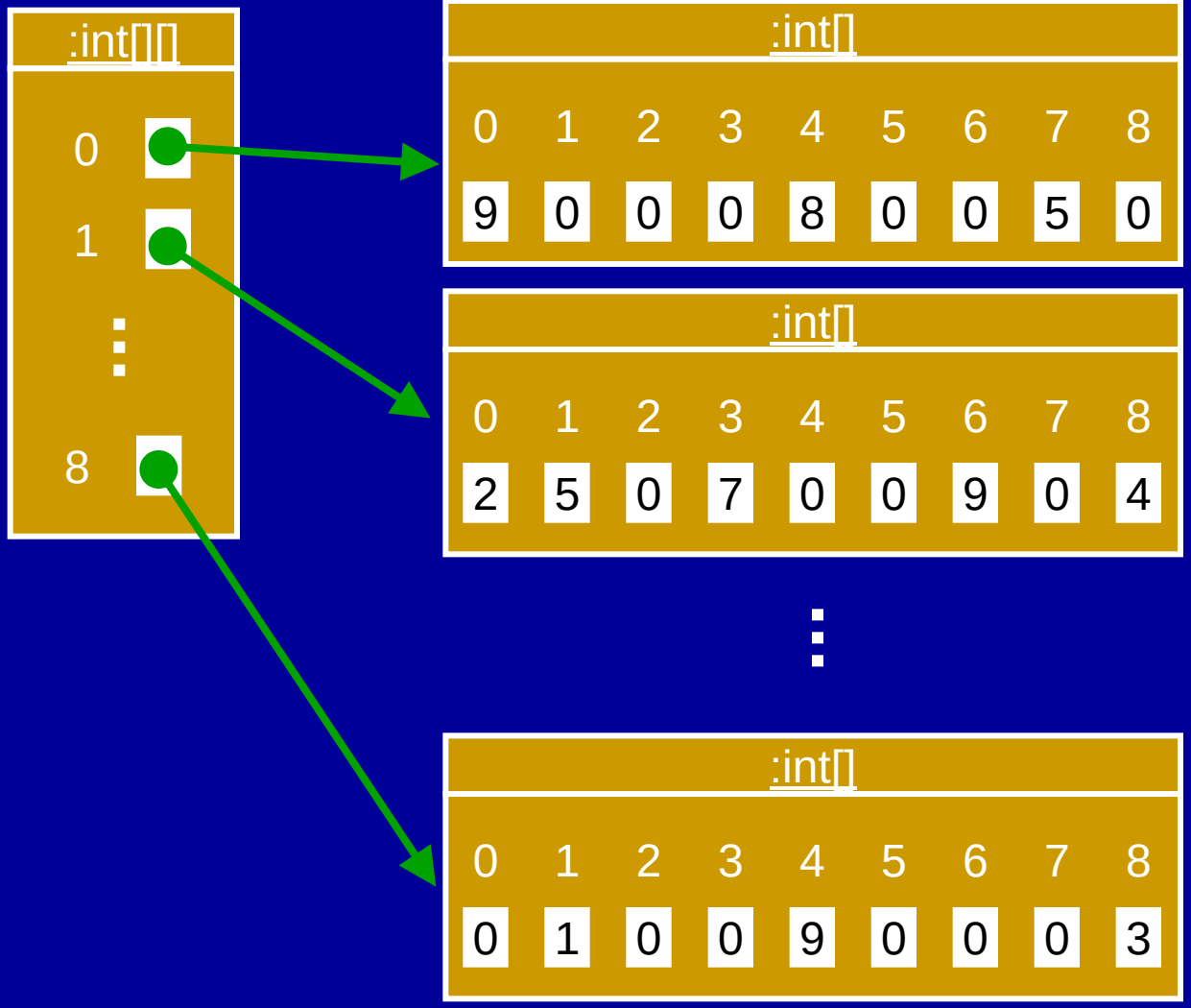
# Polia ako prvky iného poľa

- Prvkom poľa môže byť objekt
- Môže byť prvkom poľa iné pole?

- pole je objekt

  =>
- prvkami poľa môžu byť aj iné polia

# Polia ako prvky iného poľa



# Pole polí – definícia

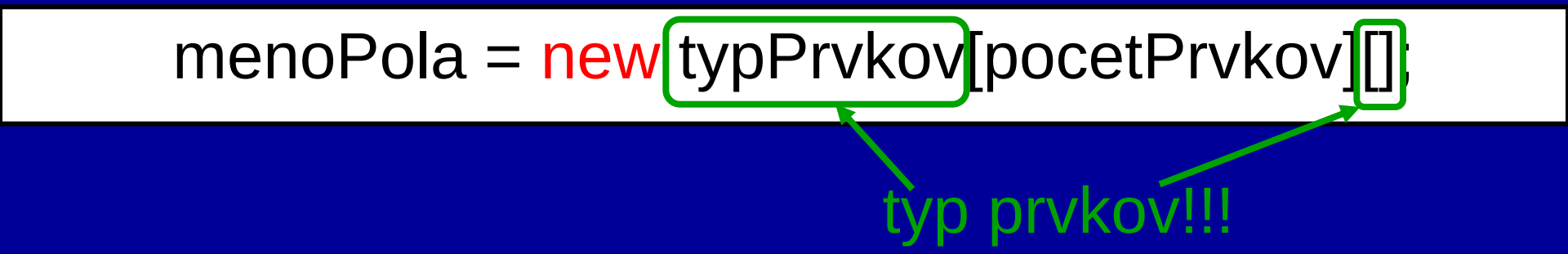
- Java – špeciálna syntax – historické dôvody
- Definícia poľa:


- Definícia poľa polí

# Pole polí – vytvorenie

- Java – špeciálna syntax – historické dôvody
- Vytvorenie poľa:

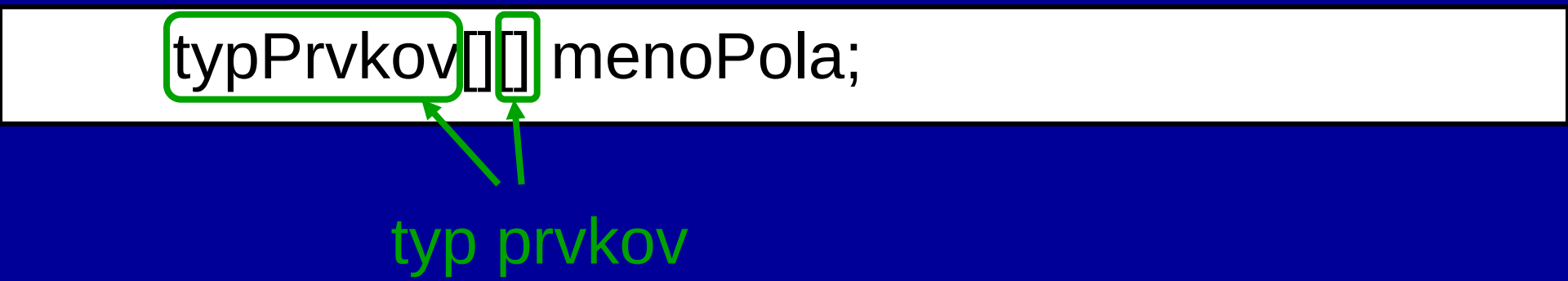

- Vytvorenie poľa polí



# Pole polí – definícia

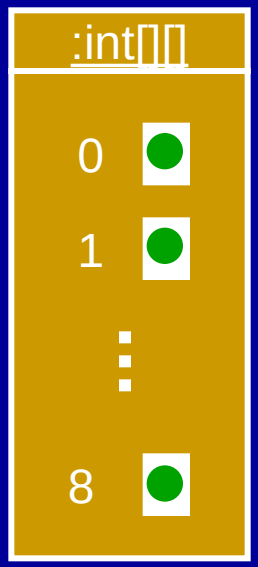
- Java – špeciálna syntax – historické dôvody
- Definícia poľa:


- Definícia poľa polí



# Polia ako prvky iného poľa

```
mriezka = new int[9][];
```



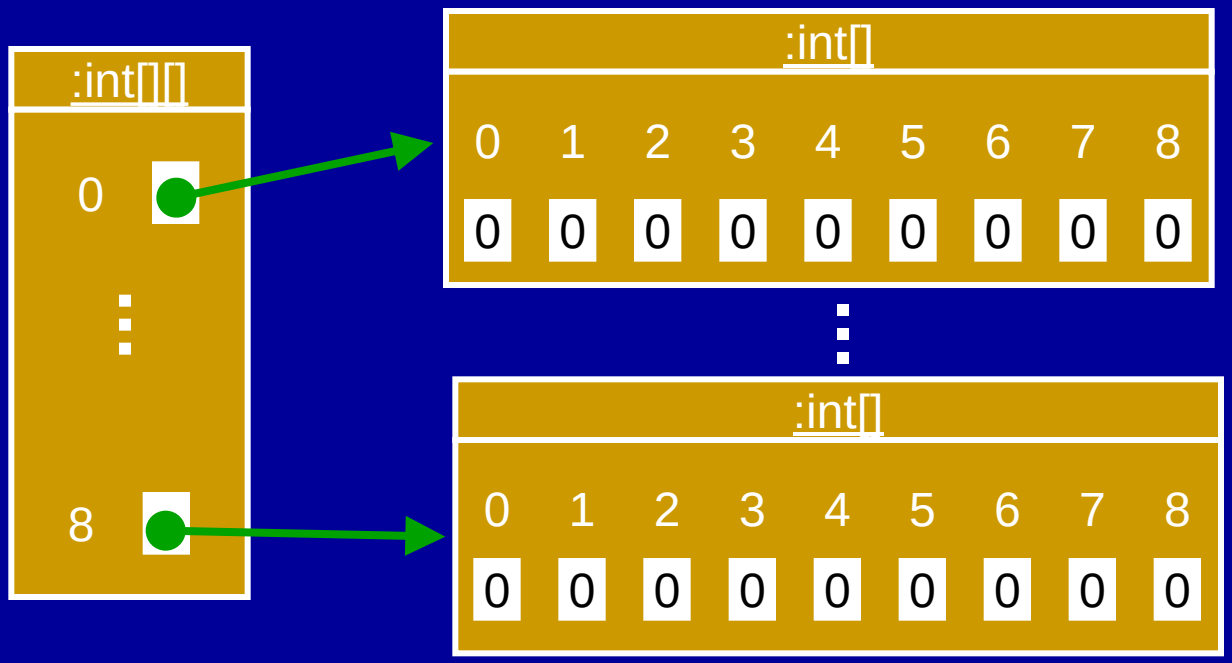
# Pole polí – inicializácia

```
for (int i = 0; i < pole.length; i++) {
    pole[i] = new typPrvkov[pocetPrvkov];
}
```

- pocetPrvkov – počet prvkov vnoreného poľa

# Pole polí – inicializácia

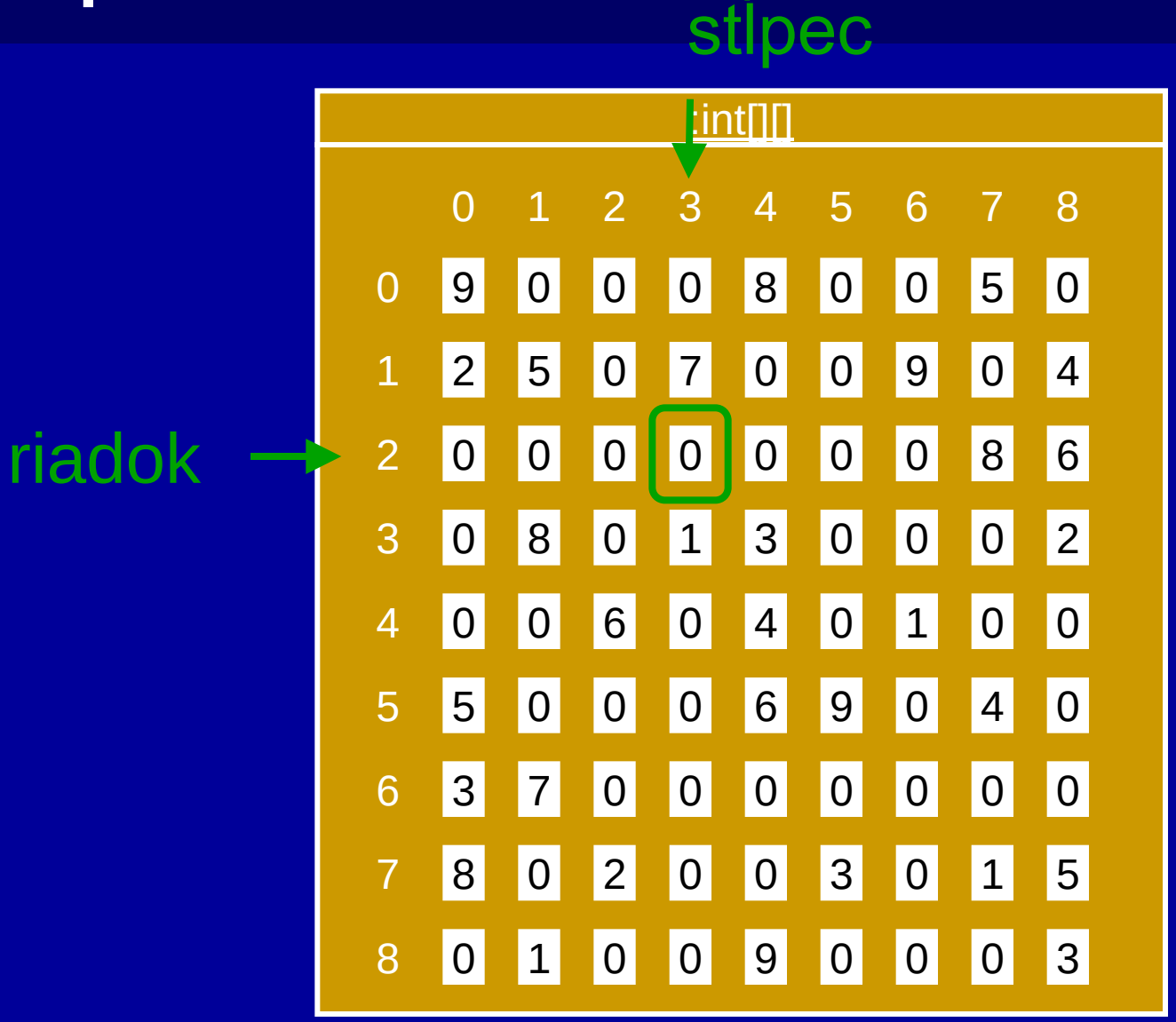
```
for (int i = 0; i < mriezka.length; i++) {
    mriezka[i] = new int[9];
}
```



# Pole polí

- prvok-pole
- prvky rovnakých rozmerov –> obdĺžniková forma (matica)
- rozmery – riadky a stĺpce
- nepravidelné viacrozmerné polia
  - jednotlivé riadky – rôzny počet prvkov

# Pole polí – matica



# Java – prístup k prvkom poľa polí

- pomenovanie prvkov – meno poľa + index riadku + index stlpca

```
menoPola[indexRiadku][indexStlpca]
```

- 0 ≤ index riadku < počet riadkov
- 0 ≤ index stĺpca < počet stĺpcov
- prvok poľa – premenná
- operácie – pravidlá pre typ prvkov
- index riadku, index stĺpca – celočíselné aritmetické výrazy

# Java – prístup k prvkom dvojrozmerného poľa

```
// 3. riadok, 4. stlpec
System.out.println(mriezka[2][3]);
```

# Java – vytvorenie poľa polí – matica

- zjednodušená syntax

```
typPrvkov[][] menoPola
        = new typPrvkov[pocetRiadkov][pocetStlpcov];
```

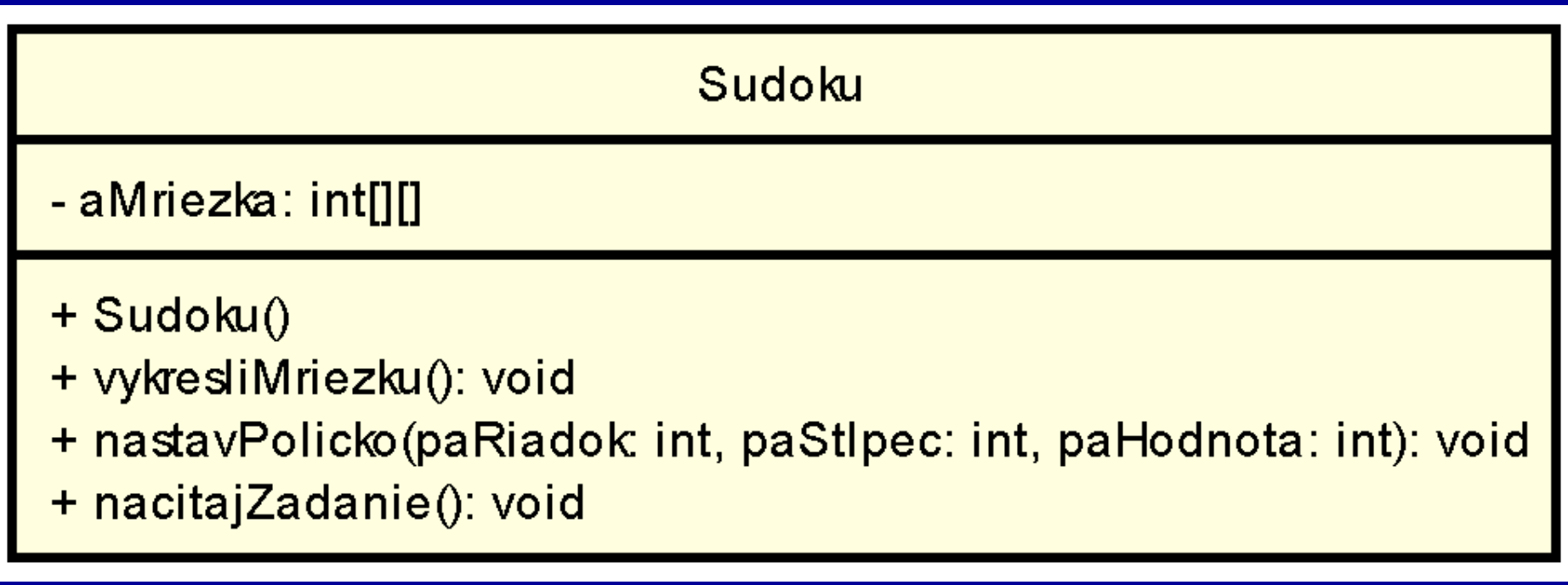
- príklad

```
int[][] mriezka = new int[9][9];
```

# n-rozmerné pole

- dvojrozmerné pole – matica
  - int[][] matica;
- trojrozmerné pole
  - int[][][] kocka;
- ...

# Sudoku – vnútorný pohľad



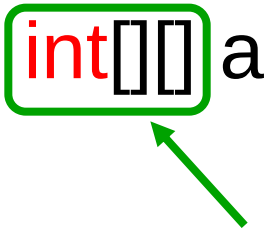
# Reprezentácia mriežky Sudoku

- Mriežka = matica
- prvky matice – čísla 1-9

- Nevyplnené hodnoty?
- Náhrada prázdneho políčka číslom mimo rozsahu 1-9
- Môžeme teda použiť číslo 0

# Sudoku – definícia triedy

```
public class Sudoku
{
    private int[][] aMriezka;
    ...
}
```



matica celých čísel

# Sudoku – konštruktor

```
public Sudoku()
{
    aMriezka = new int[9][9];
}
```

matica celých čísel - núl

# Sudoku – metóda načítaj zadanie

- Načíta zadanie vložené do zdrojových kódov
- Testovacie sudoku

# Vytvorenie poľa polí pomocou konštanty

- Vytvorenie a inicializácia poľa:

```
typPrvkov[] menoPola = {zoznamPrvkov};
```

- Vytvorenie a inicializácia poľa polí:

```
typPrvkov[][] menoPola = {
   {zoznamPrvkovPrvehoRiadku},
   {zoznamPrvkovDruhehoRiadku},
   ...
};
```

# Vytvorenie poľa pomocou konštanty

- S definíciou:

```
typPrvkov[] menoPola = {zoznamPrvkov};
```

- Bez definície:

povinné

```
menoPola = new typPrvkov[] {zoznamPrvkov};
```

- V prípade inicializácie s definíciou nie je new povinné – môže sa písať

```
typPrvkov[] menoPola =
   new typPrvkov[] {zoznamPrvkov};
```

nepovinné

# Sudoku – metóda nacitajZadanie

```
public void nacitajZadanie()
{
    aMriezka = new int[][] {
        {9, 0, 0, 0, 8, 0, 0, 5, 0},
        ...
        {8, 0, 2, 0, 0, 3, 0, 1, 5},
        {0, 1, 0, 0, 9, 0, 0, 0, 3}
    };
}
```

povinné

# Sudoku – vykreslenie mriežky

- Vypísanie na konzolu
- Znaky:
  - „1“-„9“: Známe hodnoty v mriežke
  - „.“: Nevyplnená hodnota

# Sudoku – metóda vykresliMriezku

```
public void vykresliMriezku()
{
    for (int riadok = 0; riadok < 9; riadok++) {
        this.vykresliRiadok(riadok);
    }
}
```

# Posielanie správ

- Digitálne hodiny – objekt celok posiela správy častiam
- Projekt Sudoku – trieda Sudoku
- objekt posiela správy sám sebe
- formát správy
  - adresát.selektor(parametre)
- adresát this – kľúčové slovo
- implicitný parameter každej metódy
- objekt sám seba označuje this (self)

# explicitné použitie this

```
public void vykresliMriezku()
{
    for (int riadok = 0; riadok < 9; riadok++) {
        this.vykresliRiadok(riadok);
    }
}
```

objekt si posiela správu

adresát

# implicitné použitie this

```
public void vykresliMriezku()
{
    for (int riadok = 0; riadok < 9; riadok++) {
        vykresliRiadok(riadok);
    }
}
```

this je podľa syntaxe nepovinné

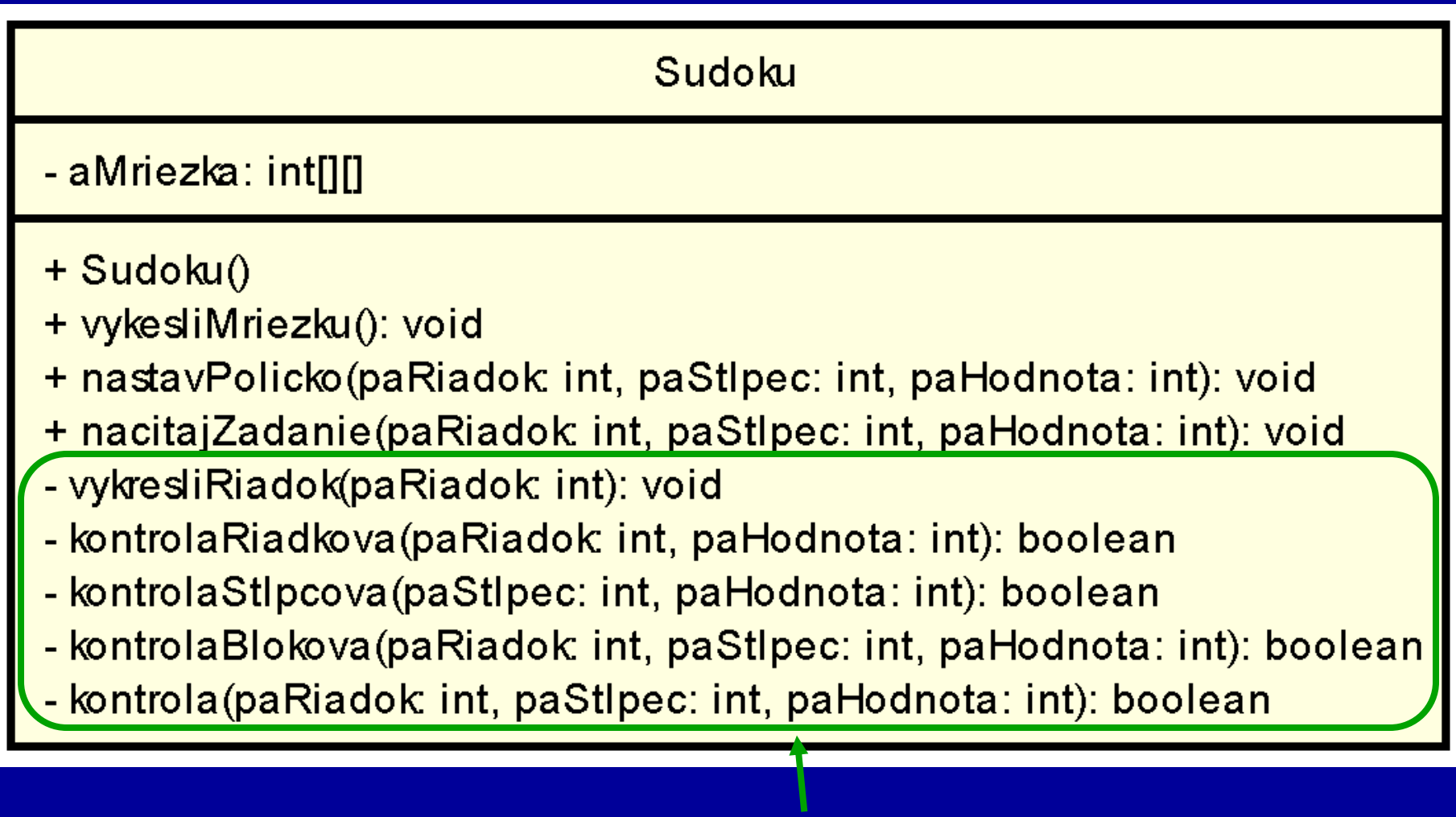
# Sudoku – metóda vykresliRiadok

```
public void vykresliRiadok(int paRiadok)
{
    for (int stlpec = 0; stlpec < 9; stlpec++) {
        System.out.print(aMriezka[paRiadok][stlpec]);
    }
    System.out.println();
}
```

# Sekcie rozhrania

- verejné – obsahuje správy, ktoré môže poslať ľubovoľný objekt
  - definícia triedy obsahuje metódy public
- neverejné – obsahuje správy, ktoré si môže poslať len objekt sám
  - definícia triedy obsahuje metódy private

# Sekcie rozhrania




existujú neverejné správy

# Sudoku – metóda vykresliRiadok

```
private void vykresliRiadok(int paRiadok)
{
    for (int stlpec = 0; stlpec < 9; stlpec++) {
        System.out.print(aMriezka[paRiadok][stlpec]);
    }
    System.out.println();
}
```

# Rozdeľuj a panuj

vykresliMriezku:

```
for (int riadok = 0; riadok < 9; riadok++) {
    this.vykresliRiadok(riadok);
}
```

vykresliRiadok:

```
for (int stlpec = 0; stlpec < 9; stlpec++) {
    System.out.print(aMriezka[paRiadok][stlpec]);
}
System.out.println();
```

# Urob všetko sám

```
public void vykresliMriezku()
{
    for (int riadok = 0; riadok < 9; riadok++) {
        for (int stlpec = 0; stlpec < 9; stlpec++) {
            System.out.print(aMriezka[riadok][stlpec]);
        }
        System.out.println();
    }
}
```

vnorený cyklus

# For-each pre polia

- Pole je objekt
- Pole je kontainer

- Na prechádzanie poľa teda môžeme použiť for-each

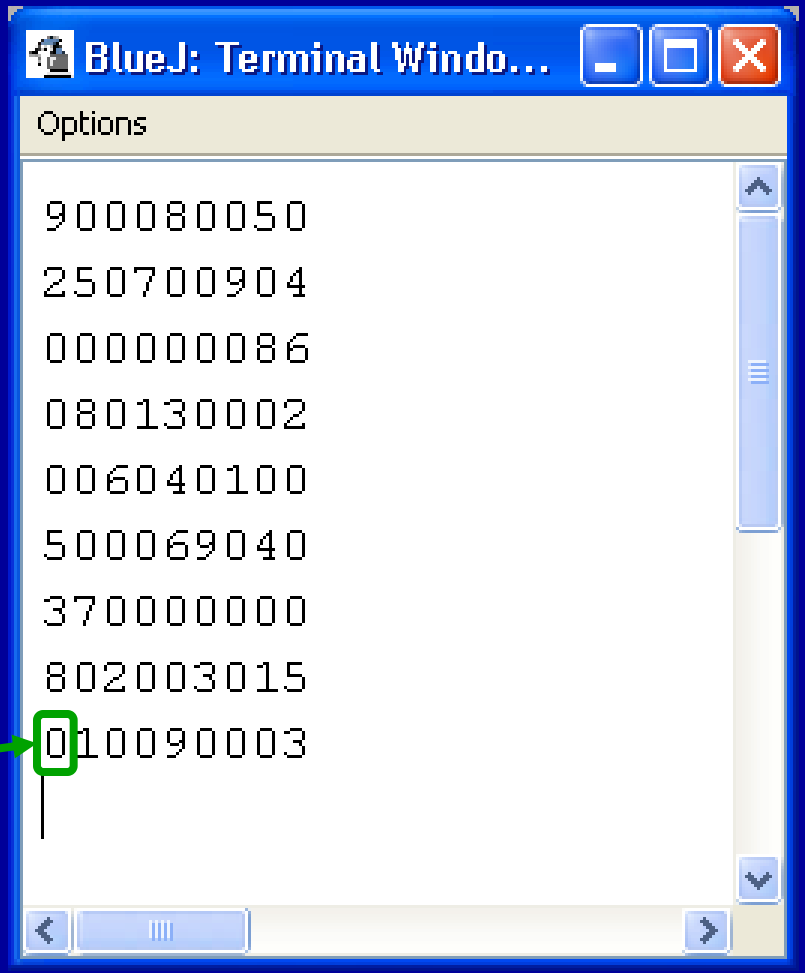
- Prvkami dvojrozmerných polí sú polia – riadky

# For-each pre polia - príklad

```
public void vykresliMriezku()
{
    for (int[] riadok : aMriezka) {
        for (int policko : riadok) {
            System.out.print(policko);
        }
        System.out.println();
    }
}
```

polia - kontajnery

# Výsledok



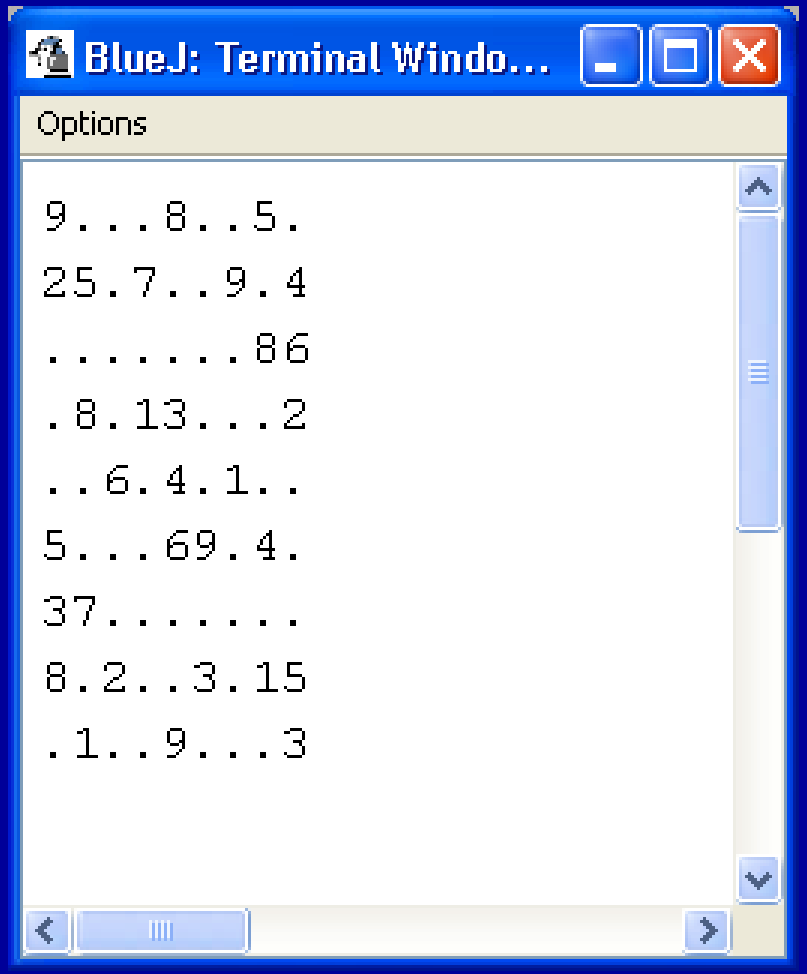

mala byť bodka

# Sudoku – metóda vykresliMriezku

```
System.out.print(policko);
```

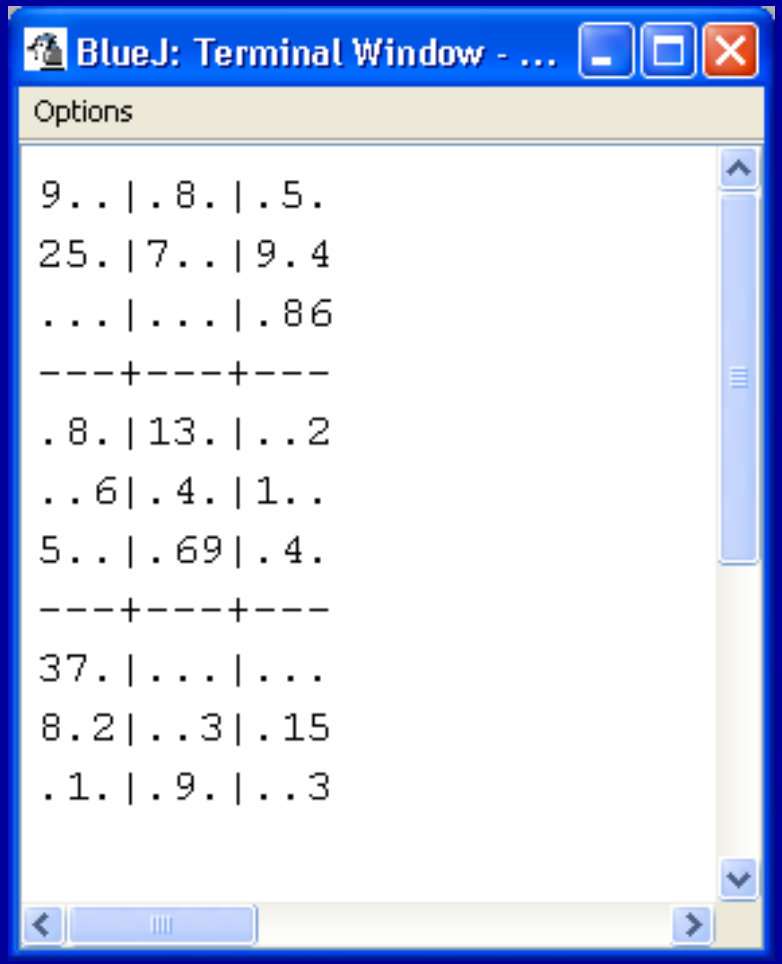
```
if (policko == 0) {
    System.out.print(".");
} else {
    System.out.print(policko);
}
```

# Výsledok



# Jednoduchá úloha

- doplňte deliace čiary pre bloky sudoku



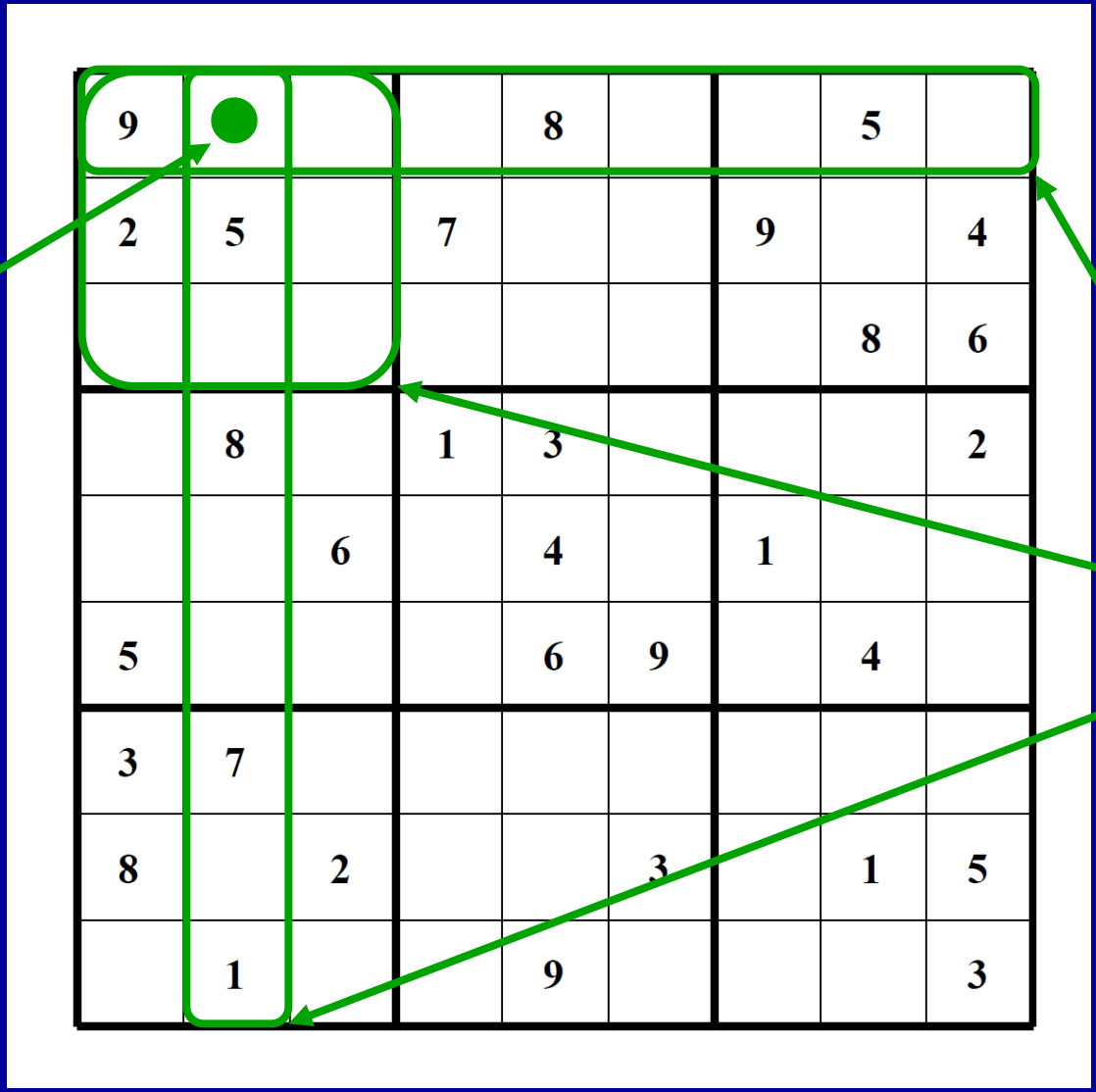
# Sudoku – metóda nastavPolicko

```
public void nastavPolicko
   (int paRiadok, int paStlpec, int paHodnota)
{
    if (paRiadok >= 0 && paRiadok < 9 &&
        paStlpec >= 0 && paStlpec < 9 &&
        paHodnota > 0 && paHodnota <= 9) {
            aMriezka[paRiadok][paStlpec] = paHodnota;
    }
}
```

# Priebežné kontroly pravidiel

- Kedy kontrolovať?
- Odpoveď: pri vkladaní čísla
- Typy kontrol:
  - riadková – nemôžem vložiť číslo, ktoré sa už v riadku nachádza
  - stĺpcová – nemôžem vložiť číslo, ktoré sa už v stĺpci nachádza
  - bloková – nemôžem vložiť číslo, ktoré sa už v bloku nachádza

# Sudoku – kontroly



# Sudoku – metóda kontrolaRiadkova

```
private boolean kontrolaRiadkova
   (int paRiadok, int paHodnota)
{
    for (int stlpec = 0; stlpec < 9; stlpec++) {
        if (aMriezka[paRiadok][stlpec] == paHodnota) {
            return false;
        }
    }
    return true;
}
```

predčasné ukončenie cyklu

# Sudoku – metóda kontrolaStlpcova

```
private boolean kontrolaStlpcova
   (int paStlpec, int paHodnota)
{
    for (int riadok = 0; riadok < 9; riadok++) {
        if (aMriezka[riadok][paStlpec] == paHodnota) {
            return false;
        }
    }
    return true;
}
```

# Sudoku – metóda kontrolaBlokova

```
private boolean kontrolaBlokova
   (int paRiadok, int paStlpec, int paHodnota)
{
    int startR = (paRiadok/3)*3;
    int startS = (paRiadok/3)*3;
    ...
}
```

# Sudoku – metóda kontrolaBlokova

```
...
for (int r = startR; r < (startR + 3); r++) {
   for (int s = startS; s < (startS + 3); s++) {
      if (aMriezka[r][s] == paHodnota) {
         return false;
      }
   }
}
return true;
...
```

# Sudoku – metóda kontrola

```
private boolean kontrola
   (int paRiadok, int paStlpec, int paHodnota)
{
    return
      this.kontrolaRiadkova(paRiadok, paHodnota) &&
      this.kontrolaStlpcova(paStlpec, paHodnota) &&
      this.kontrolaBlokova
          (paRiadok, paStlpec, paHodnota);
}
```

# Sudoku – metóda nastavPolicko

```
public void nastavPolicko
   (int paRiadok, int paStlpec, int paHodnota)
{
    if (paRiadok >= 0 && paRiadok < 9 &&
        paStlpec >= 0 && paStlpec < 9 &&
        paHodnota > 0 && paHodnota <= 9 &&
        this.kontrola(paRiadok, paStlpec, paHodnota)) {
            aMriezka[paRiadok][paStlpec] = paHodnota;
    }
}
```

Ďakujem za pozornosť