

# Informatika 1

Kontajnery s pevným počtom prvkov



# Pojmy zavedené v 7. prednáške (1)

- skupiny objektov – kontajnery
  - ArrayList
- generické triedy
- obalovacie triedy
  - primitívne typy ako objekty
- cyklus for-each
- anonymné objekty

# Pojmy zavedené v 7. prednáške (2)

- práca so súbormi
  - trieda `java.io.File`
  - throws `java.io.IOException` – povinné
- čítanie zo súboru a terminálu
  - trieda `java.util.Scanner`
- zápis do súboru
  - trieda `java.io.PrintWriter`

# Cieľ prednášky

- kontajnery – polia
  - jednorozmerné
  - viacrozmerné
- vnorené cykly
- príklady: Štatistika pripojení, hra Sudoku

# Štatistika pripojení na web server

- web server eviduje intenzitu pripájania
- pre každé pripojenie zaznamenáva
  - dátum a čas v tvare päťice celých čísel
    - rok mesiac deň hodina minúta
  - klient (adresa počítača)
  - adresa požiadavky
- máme k dispozícii súbor záznamov weblog.txt
- úloha – vytvoriť štatistiku pripojení podľa hodín (bez ohľadu na deň)

# Log súbor

2021	09	01	07	45	166.254.109.216	/spravicky/16488
2021	09	01	08	40	173.137.190.129	/clanky/16763
2021	09	01	09	00	215.145.186.231	/clanky/7717
2021	09	01	09	50	190.232.201.202	/spravicky/43318
2021	09	01	10	04	109.97.94.95	/clanky/16189
2021	09	01	10	27	229.249.211.233	/clanky/13231
2021	09	01	10	59	165.246.158.207	/clanky/17016
2021	09	01	11	02	197.229.220.190	/clanky/22915
2021	09	01	11	04	217.220.209.238	/spravicky/13879
2021	09	01	11	06	85.128.39.62	/spravicky/13541
2021	09	01	11	35	186.246.148.205	/spravicky/23875

# Metóda citajZRiadkuHodinu

```
private int citajZRiadkuHodinu(Scanner citac) {  
    int rok = citac.nextInt();  
    int mesiac = citac.nextInt();  
    int den = citac.nextInt();  
    int hodina = citac.nextInt();  
    int minuta = citac.nextInt();  
    String klient = citac.next();  
    String ciel = citac.next();  
    citac.nextLine();  
    return hodina;  
}
```

## AnalyzatorLogSuboru – rozhranie

AnalyzatorLogSuboru

+ new(cesta: String): AnalyzatorLogSuboru  
+ vytlacHodinovePocty(): void



# Analýza (1)

- potrebujeme ukladať údaje pre hodiny 0-23
  - počet požiadaviek v danú hodinu
- kontajner – ArrayList
  - prvky: celé čísla

## Analýza (2)

- `ArrayList<Integer>` – kontajner na čísla
- hodiny 0 až 23 – 24 prvkov
- začiatočná hodnota každého prvku – 0
- vkladanie v cykle

## Vnútrotný pohľad

### AnalyzatorLogSuboru

- hodinovePocty: ArrayList<Integer>

+ AnalyzatorLogSuboru(cesta: String)

+ vytlačHodinovePocty(): void

- analyzujData(cesta: String): void

- citajZRiadkuHodinu(citac: Scanner): int

# Trieda AnalyzatorLogSuboru

```
import java.util.ArrayList;

public class AnalyzatorLogSuboru {
    private ArrayList<Integer> hodinovePocty;

    ...
}
```

# AnalyzatorLogSuboru – konštruktor

```
public AnalyzatorLogSuboru(String cesta) {  
    this.hodinovePocty = new ArrayList<Integer>();  
  
    for (int i = 0; i < 24; i++) {  
        this.hodinovePocty.add(0);  
    }  
  
    this.analyzujData(cesta);  
}
```

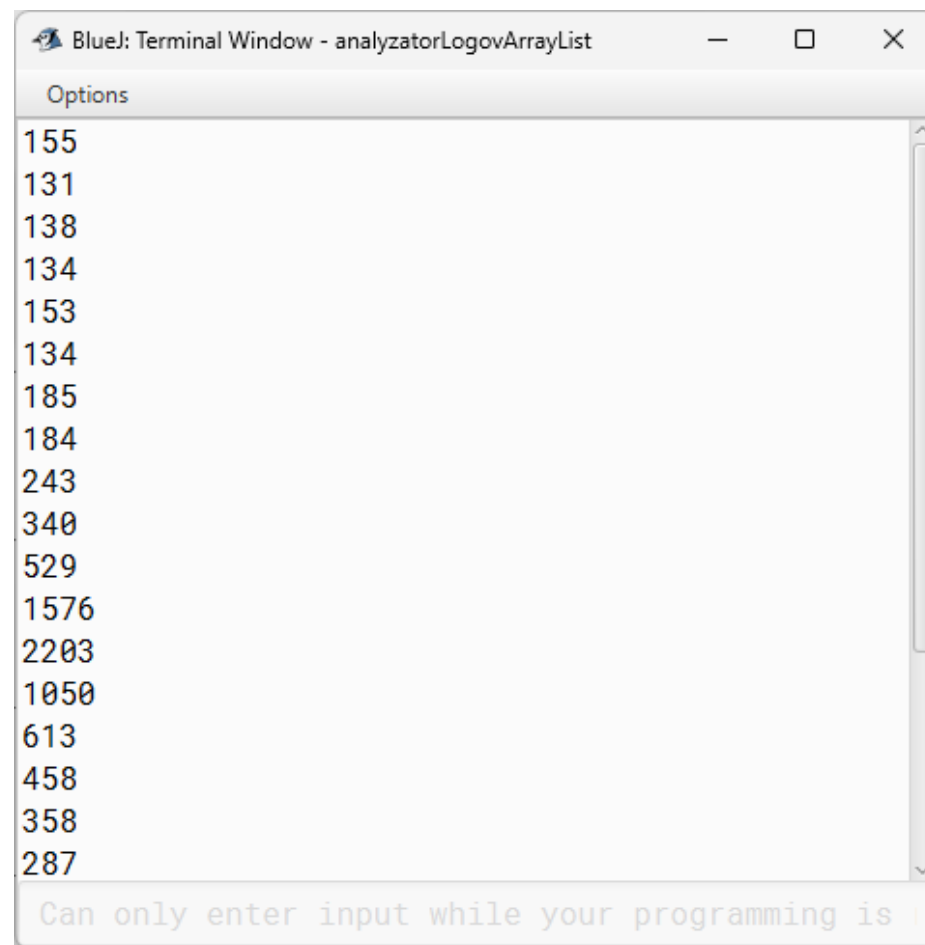
# AnalyzatorLogSuboru – analyzujData

```
private void analyzujData(String cesta) throws IOException {  
    Scanner citac = new Scanner(new File(cesta));  
    while (citac.hasNextLine()) {  
        int hodina = this.citajZRiadkuHodinu(citac);  
  
        int novyPocet = this.hodinovePocty.get(hodina) + 1;  
        this.hodinovePocty.set(hodina, novyPocet);  
    }  
}
```

# Metóda vytlačHodinovePocty

```
public void vytlacHodinovePocty() {  
    for (int pocet : this.hodinovePocty) {  
        System.out.println(pocet);  
    }  
}
```

# Výstup

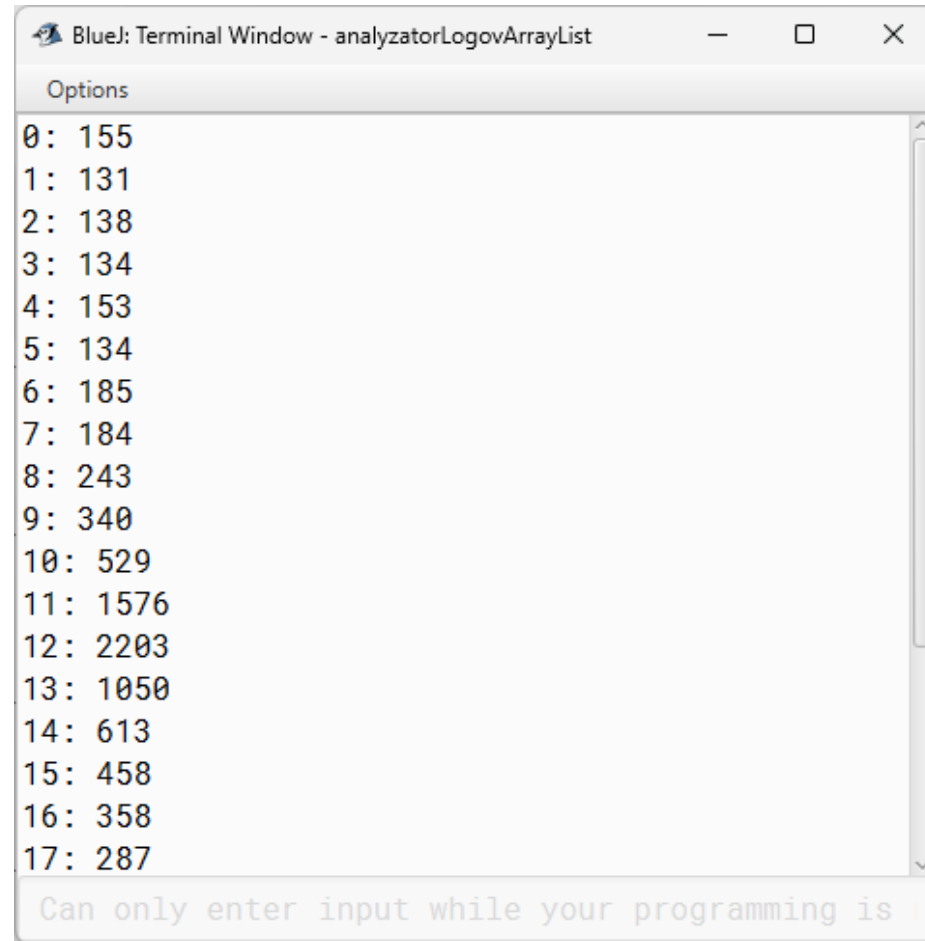


A screenshot of a BlueJ Terminal Window titled "BlueJ: Terminal Window - analyzatorLogovArrayList". The window has a standard macOS-style title bar with minimize, maximize, and close buttons. Below the title bar is a tab labeled "Options". The main area of the terminal displays a list of 18 integers, one per line, which are the elements of an ArrayList: 155, 131, 138, 134, 153, 134, 185, 184, 243, 340, 529, 1576, 2203, 1050, 613, 458, 358, and 287. A vertical scrollbar is visible on the right side of the text area. At the bottom of the window, there is a prompt area with the text "Can only enter input while your programming is" followed by a cursor.

```
BlueJ: Terminal Window - analyzatorLogovArrayList
Options
155
131
138
134
153
134
185
184
243
340
529
1576
2203
1050
613
458
358
287
Can only enter input while your programming is
```



# Požadovaný výstup



The image shows a screenshot of a BlueJ Terminal Window titled "BlueJ: Terminal Window - analyzatorLogovArrayList". The window has a standard macOS-style title bar with minimize, maximize, and close buttons. Below the title bar is a tab labeled "Options". The main area of the window displays a list of 18 integers, each preceded by an index from 0 to 17. The values are: 155, 131, 138, 134, 153, 134, 185, 184, 243, 340, 529, 1576, 2203, 1050, 613, 458, 358, and 287. A vertical scrollbar is visible on the right side of the text area. At the bottom of the window, there is a status bar with the text "Can only enter input while your programming is" followed by a cursor.

```
0: 155
1: 131
2: 138
3: 134
4: 153
5: 134
6: 185
7: 184
8: 243
9: 340
10: 529
11: 1576
12: 2203
13: 1050
14: 613
15: 458
16: 358
17: 287
```

Can only enter input while your programming is

# Tlač s hodinami

```
public void vytlacHodinovePocty() {  
    int hodina = 0;  
    for (int pocet : this.hodinovePocty) {  
        System.out.format("%d: %d%n", hodina, pocet);  
        hodina++;  
    }  
}
```

## Tlač s hodinami pomocou cyklu for

```
public void vytlacHodinovePocty() {  
    for (int hod = 0; hod < 24; hod++) {  
        System.out.format("%d %d%n", hod, this.hodinovePocty.get(hod));  
    }  
}
```

# Kontejnery s fixným počtom prvkov

- Diár – príklad na použitie kontejnerov s premenlivým počtom prvkov
  - zmena počtu prvkov v priebehu životného cyklu
- existujú situácie, keď sa počet prvkov nemení
  - počet hodín dňa je konštantný
- kontejnery pre takéto situácie – polia
  - iná syntax – historické dôvody
  - pole – najstarší kontejner

# Pole ako objekt – rovnaké vlastnosti

- pole – objektový typ – referencia na pole
- hodnota null – aj pre pole
- práca s poľom ako celkom – referencia na pole
- môže obsahovať ľubovoľný typ prvkov

# Pole ako objekt – odlišné vlastnosti

- definícia poľa
- vytvorenie poľa
- prístup k prvkom
- nie je generický typ

# Java – definícia poľa

```
typPrvkov[] menoPola;
```

- typ prvkov – ľubovoľný primitívny alebo objektový typ
- príklady:

```
int[] pocetPristupov;  
AutomatMHD[] automaty;
```

# Java – vytvorenie poľa

```
menoPola = new typPrvkov[pocetPrvkov];
```

- správa new – špecifická forma
- prvky sú inicializované na hodnotu 0
- príklady:

```
pocetPristupov = new int[24];  
automaty = new AutomatMHD[2];
```



# Java – definícia a vytvorenie poľa

```
typPrvkov[] menoPola = new typPrvkov[pocetPrvkov];
```

- spojenie definície a inicializácie do jedného príkazu
- príklady:

```
int[] pocetPristupov = new int[24];  
AutomatMHD[] automaty = new AutomatMHD[2];
```

# Java – definícia a vytvorenie poľa vymenovaním prvkov

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

- zoznamPrvkov – čiarkou oddelený zoznam prvkov vytváraného poľa
- príklady:

```
int[] mince = {1, 2, 5, 10, 20, 50};  
AutomatMHD[] automaty = {  
    new AutomatMHD(50), new AutomatMHD(20)  
};
```

# Java – vytvorenie poľa vymenovaním prvkov bez definície premennej

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

- zoznamPrvkov – čiarkou oddelený zoznam prvkov vytváraného poľa

- príklady:

```
mince = new int[] {1, 2, 5, 10, 20, 50};  
automaty = new AutomatMHD[] {  
    new AutomatMHD(50), new AutomatMHD(20)  
};
```

# Java – prístup k prvkom poľa (1)

- pomenovanie prvkov – meno poľa + index
  - premenná

```
menoPola[indexPrvku]
```

- $0 \leq \text{index prvku} < \text{počet prvkov}$
- prvok poľa – premenná
- operácie – pravidlá pre typ prvkov
- index – celočíselný aritmetický výraz

## Java – prístup k prvkom poľa (2)

- príklad zápis do poľa:

```
pocetPristupov[0] = 5;  
automaty[0] = new AutomatMHD(50);
```

- príklad čítanie z poľa:

```
System.out.println(pocetPristupov[0]);  
automaty[0].vytlacListok();
```

- príklad kombinovaný:

```
pocetPristupov[0]++;
```

# Java – pole a cyklus for

- dĺžka poľa = počet prvkov

```
menoPola.length
```

- výsledok – int

```
for (int i = 0; i < zoznam.length; i++) {  
    System.out.println(i + ": " + zoznam[i]);  
}
```

# Java – pole a cyklus foreach

- pole je možné prechádzať cyklom foreach

```
for (int prvok : zoznam) {  
    System.out.println(prvok) ;  
}
```

# Java – pole

- relačné operátory == a !=
  - porovnanie referencií
- príkaz priradenia
  - priradenie referencie, nie kópia všetkých prvkov poľa

```
int[] poleDruhe = pole;  
pole[1] = 5; // !!! zmenia sa „obe“ polia
```



# Trieda AnalyzatorLogSuboru

```
public class AnalyzatorLogSuboru {  
    private int[] hodinovePocty;  
  
    ...  
}
```

# AnalyzatorLogSuboru – konštruktor

```
public AnalyzatorLogSuboru(String cesta) {  
    this.hodinovePocty = new int[24];  
  
    this.analyzujData(cesta);  
}
```

# AnalyzatorLogSuboru – analyzujData

```
private void analyzujData(String cesta) throws IOException {  
    Scanner citac = new Scanner(new File(cesta));  
    while (citac.hasNextLine()) {  
        int hodina = this.citajZRiadkuHodinu(citac);  
  
        this.hodinovePocty[hodina]++;  
    }  
  
    citac.close();  
}
```

# Sudoku (1)

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 (2)

- hlavolam – v každých novinách
- cieľ: čiastočne vyplnenú mriežku doplniť tak, aby obsahovala každé číslo 1 až 9 práve raz v troch rôznych zoskupeniach.

# Sudoku (3)

- mriežka 9x9 políček
- tri typy zoskupení políček mriežky
  - riadky – 9 riadkov
  - stĺpce – 9 stĺpcov
  - bloky 3x3 – 9 blokov v zostave 3x3

# Sudoku – zoskupenia políček

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 – cieľ projektu

- podpora pre riešiteľa
- požadované funkcie:
  - zobrazenie mriežky
  - vloženie čísla do políčka
  - načítanie zadania zo súboru
- priebežná kontrola pravidiel



## Sudoku – rozhranie

### Sudoku

- + new(nazovZadania: String): Sudoku
- + vykresliMriezku(): void
- + nastavPolicko(riadok: int, stlpec: int, hodnota: int): void

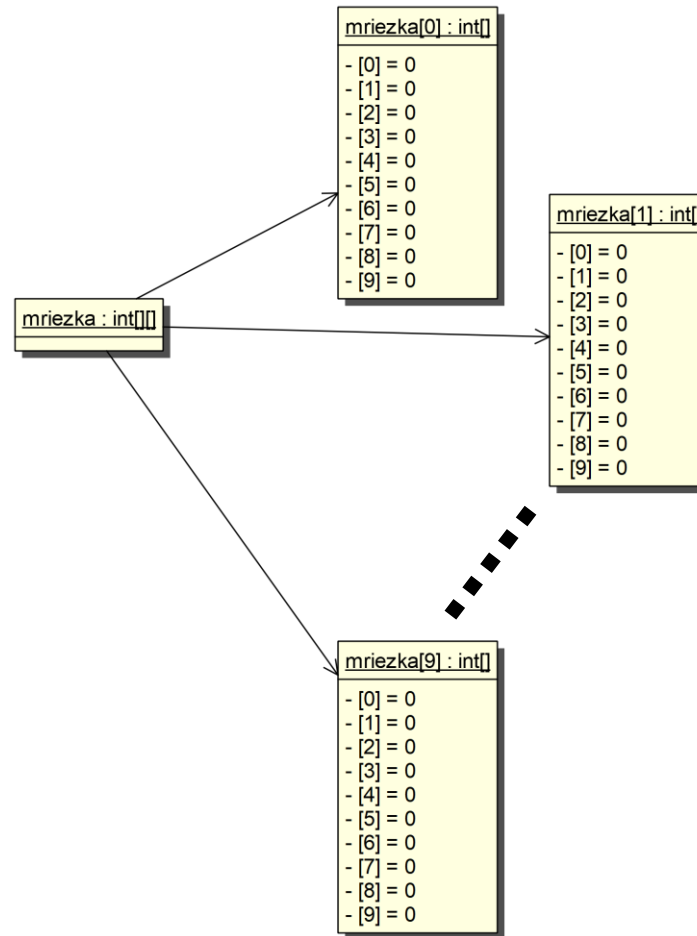
# Sudoku – mriežka

- 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
  - 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:

```
typPrvkov[] menoPola;
```

- definícia poľa polí

```
typPrvkov[][] menoPola;
```


typ prvkov

# Pole polí – vytvorenie

- Java – špeciálna syntax – historické dôvody

- vytvorenie poľa:

```
menoPola = new typPrvkov[pocetPrvkov];
```



- vytvorenie poľa polí

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



# Polia ako prvky iného poľa

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

mriezka[9] : int[][]

- [0] = null
- [1] = null
- [2] = null
- [3] = null
- [4] = null
- [5] = null
- [6] = null
- [7] = null
- [8] = null
- [9] = null

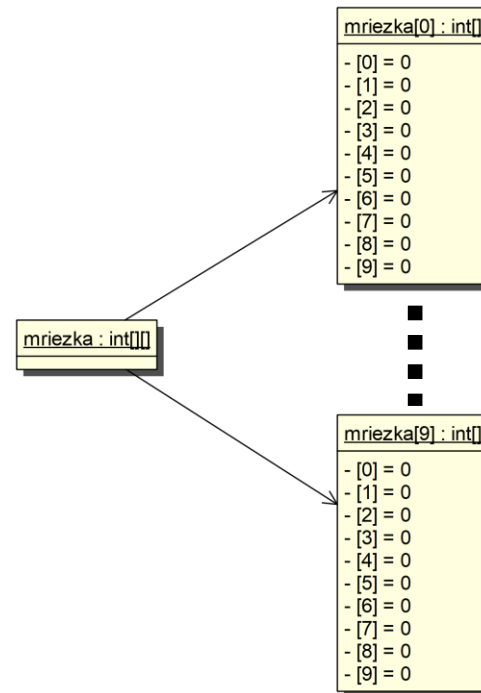
# Pole polí – inicializácia (1)

```
for (int i = 0; i < pole.length; i++) {  
    // pocetPrvkov - pocet prvkov vnoreného pola  
    pole[i] = new typPrvkov[pocetPrvkov];  
}
```



## Pole polí – inicializácia (2)

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



# Pole polí

- prvok-pole
- prvky rovnakých dĺžok → obdĺžniková forma (matica)
- rozmery – riadky a stĺpce
- nepravideľné viacrozmerné polia
  - jednotlivé riadky – rôzny počet prvkov

# Pole polí – matica

mriezka : int[][]										
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
- [0] =	0	0	0	0	0	0	0	0	0	0
- [1] =	0	0	0	0	0	0	0	0	0	0
- [2] =	0	0	0	0	0	0	0	0	0	0
- [3] =	0	0	0	0	0	0	0	0	0	0
- [4] =	0	0	0	0	0	0	0	0	0	0
- [5] =	0	0	0	0	0	0	0	0	0	0
- [6] =	0	0	0	0	0	0	0	0	0	0
- [7] =	0	0	0	0	0	0	0	0	0	0
- [8] =	0	0	0	0	0	0	0	0	0	0
- [9] =	0	0	0	0	0	0	0	0	0	0

# Java – prístup k prvkom poľa polí

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

```
menoPola[indexRiadku][indexStĺpca]
```

- $0 \leq \text{index riadku} < \text{počet riadkov}$
- $0 \leq \text{index stĺpca} < \text{počet stĺpcov}$
- prvok poľa – premenná
- operácie – pravidlá pre typ prvkov
- index riadku, index stĺpca – celočíselné aritmetické výrazy

# Java – prístup k prvkom

```
// 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];
```

# Vytvorenie poľa polí pomocou konštanty

- vytvorenie a inicializácia poľa:

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

- vytvorenie a inicializácia poľa polí:

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

# n-rozmerné pole

- dvojrozmerné pole – matica

```
int[][] matica;
```

- trojrozmerné pole

```
int[][][] kocka;
```

- ...



## Sudoku – vnútorný pohľad

### Sudoku

- mriezka: int[][]

+ Sudoku(nazovZadania: String)

+ vykresliMriezku(): void

+ nastavPolicko(riadok: int, stlpec: int, hodnota: int): void

- nacistajZadanie(nazovZadania: String): void

- mozeVlozit(riadok: int, stlpec: int, hodnota: int): boolean

# 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[][] mriezka;  
  
    ...  
}
```

# Sudoku – konštruktor

```
public Sudoku(String nazovZadania) {  
    this.mriezka = new int[9][9];  
    this.nacitajZadanie(nazovZadania);  
}
```

matica celých čísel – núl

# Sudoku – vykreslenie mriežky

- vypísanie na konzolu
- znaky:
  - „1“-„9“: Známe hodnoty v mriežke
  - „.“: Nevyplnená hodnota

# Sudoku – vykreslenie mriežky pomocou vnoreného cyklu

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

vnorený cyklus

# For-each pre viacrozmerne polia

- pole je objekt
- pole je kontajner
- na prechádzanie poľa teda môžeme použiť for-each
- prvkami dvojrozmerných polí sú polia – riadky

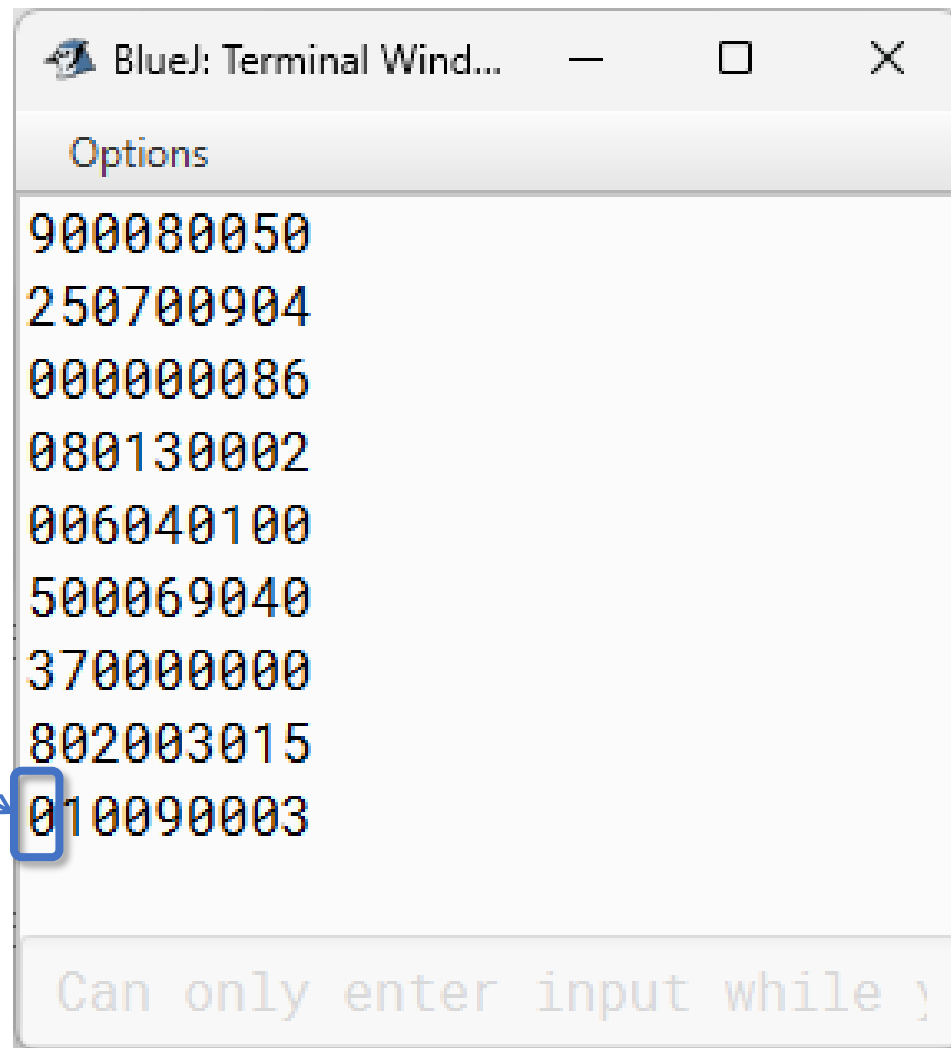
## For-each pre viacrozmerne polia – príklad

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



# Výsledok

mala  
byť  
bodka

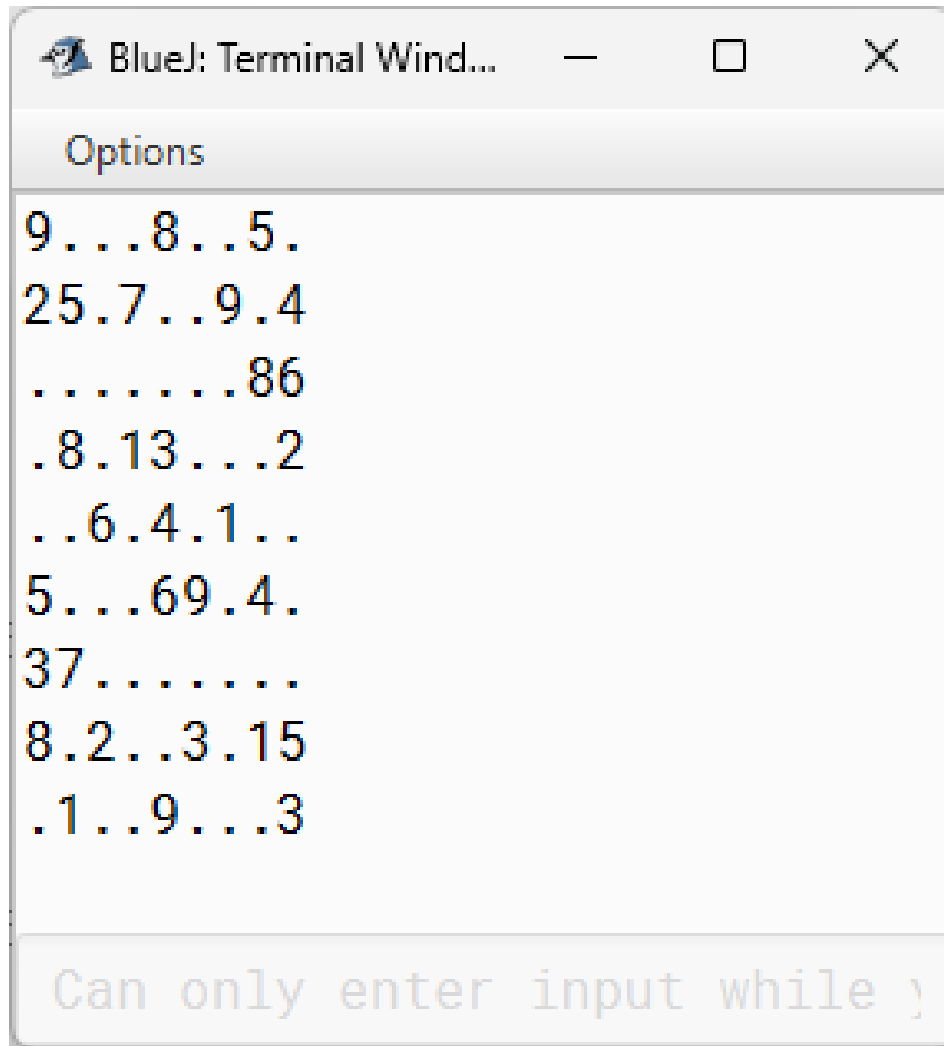


```
BlueJ: Terminal Wind...  
Options  
900080050  
250700904  
000000086  
080130002  
006040100  
500069040  
370000000  
802003015  
010090003  
Can only enter input while y
```

# Sudoku – metóda vykresliMriezku

```
public void vykresliMriezku() {  
    for (int[] riadok : this.mriezka) {  
        for (int policko : riadok) {  
            if (policko == 0) {  
                System.out.print(".");  
            } else {  
                System.out.print(policko);  
            }  
        }  
        System.out.println();  
    }  
}
```

# Výsledok



A screenshot of a BlueJ Terminal Window. The window title is "BlueJ: Terminal Wind...". It has standard window controls (minimize, maximize, close). Below the title bar is a tab labeled "Options". The main area of the window displays a sequence of numbers on ten lines: "9...8..5.", "25.7..9.4", "...86", ".8.13...2", "..6.4.1..", "5...69.4.", "37.....", "8.2..3.15", ".1..9...3". At the bottom of the window, there is a prompt "Can only enter input while )" in a light gray font.

```
BlueJ: Terminal Wind... — □ ×
Options
9...8..5.
25.7..9.4
...86
.8.13...2
..6.4.1..
5...69.4.
37.....
8.2..3.15
.1..9...3
Can only enter input while )
```

# Jednoduchá úloha

- doplňte deliace čiary pre bloky sudoku

```
BlueJ: Terminal Wind...  
Options  
9..|.8.|.5.  
25.|7..|9.4  
...|...|.86  
---+---+---  
.8.|13.|..2  
..6|.4.|1..  
5..|.69|.4.  
---+---+---  
37.|...|...  
8.2|..3|.15  
.1.|.9.|..3  
Can only enter input while ,
```

# Sudoku – metóda načítaj zadanie

- načíta zadanie zo súboru s príponou .sudoku
- testovacie sudoku

# zakladneZadanie.sudoku

9	0	0	0	8	0	0	5	0
2	5	0	7	0	0	9	0	4
0	0	0	0	0	0	0	8	6
0	8	0	1	3	0	0	0	2
0	0	6	0	4	0	1	0	0
5	0	0	0	6	9	0	4	0
3	7	0	0	0	0	0	0	0
8	0	2	0	0	3	0	1	5
0	1	0	0	9	0	0	0	3

# Sudoku – metóda nacistajZadanie

```
private void nacistajZadanie(String nazovZadania) {  
    Scanner vstup = new Scanner(new File(nazovZadania + ".sudoku"));  
  
    for (int riadok = 0; riadok < 9; riadok++) {  
        for (int stlpec = 0; stlpec < 9; stlpec++) {  
            this.mriezka[riadok][stlpec] = vstup.nextInt();  
        }  
    }  
  
    vstup.close();  
}
```

# Informatika 1

BONUS: ShapesGE





# ShapesGE

- Shapes-based Game Engine
- Na-tvaroch-založený Herný Engine
- drop-in náhrada za tvaryV4
  - rozhrania takmer rovnaké
  - vyšší výkon
  - nové možnosti

# Takmer rovnaké rozhrania

- treba import

```
import fri.shapesge.Stvorec;  
...
```

- chýbajú metódy pomalyPosun...
  - nefungujú správne
- chýba metóda Obrazok.zmenPolohu
  - v tvaroch nefunguje správne
- nové konštruktory s parametrami x,y (staré pre istotu zostali)
- zmena vlastností tvaru ho nepresunie navrch (treba skryť/zobraziť)
- možnosť zadať farbu ako #RRGGBB (staré názvy farieb fungujú)

# Vyšší výkon

- BlueJ Shapes (tvary a následne tvaryV2, tvaryV3, tvaryV4) sú len na výučbu
  - schválne naprogramované ako pomalé
  - naprogramované neoptimálne
- naprogramované nanovo
  - gameloop
  - FPS

# Nové možnosti

- zmena vlastností okna (rozmer, titulok, fullscreen support, ...)
- zmena max FPS
- zmena farby pozadia
- možnosť ukončenia krížikom
- úprava časovača (default 250ms)/pridanie nových časovačov
- podpora celej klávesnice a myši
- možnosť úpravy farebnej palety
- ...
- všetko opt-in

# Konfiguračný súbor

- sbge.ini v projektovom adresári
- sekcie
  - [Window]
  - [Colors]
  - [Timers]
  - [Keyboard]
  - [Mouse]

# Sekcia [Window]

```
[Window]
```

```
Width = 300
```

```
Height = 300
```

```
Title = Shapes Game Engine Demo
```

```
Background = white
```

```
FPS = 100
```

```
ShowInfo = true
```

```
Fullscreen = false
```

```
ExitOnClose = false
```

# Sekcia [Colors]

## [Colors]

red = #FF0000

blue = #0000FF

yellow = #FFFF00

green = #00FF00

magenta = #FF00FF

white = #FFFFFF

brown = #663300

black = #000000

## Sekcia [Colors]

- POZOR: jej uvedením v konfigu sa vymažú pôvodné farby
- každý riadok definícia farby
  - nazov = #RRGGBB



# Sekcia [Timers]

**[Timers]**

```
tik = 250
```

# Sekcia [Timers]

- POZOR: jej uvedením v konfigu sa vymažú pôvodné časovače
- každý riadok definícia časovača
  - selektor = cas v ms

# Sekcia [Keyboard]

## [Keyboard]

posunDole = pressed DOWN

posunHore = pressed UP

posunVlavo = pressed LEFT

posunVpravo = pressed RIGHT

aktivuj = pressed SPACE, pressed ENTER

zrus = pressed ESCAPE

# Sekcia [Keyboard]

- POZOR: jej uvedením v konfigu sa vymažú pôvodné definície kláves
- každý riadok definícia reakcie na klávesnicu
  - selektor = klavesa, klavesa, klavesa, ...
- definícia klávesy
  - pressed/released/typed
  - názov klávesy (podľa konštánt VK\_ v štandardnej knižnici)
  - <https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html>

# Sekcia [Mouse]

**[Mouse]**

vyberSuradnice = **clicked LEFT**

# Sekcia [Mouse]

- POZOR: jej uvedením v konfigu sa vymažú pôvodné definície myšky
- každý riadok definícia reakcie na myš
  - selektor = mys, mys, mys, ...
- definícia myši
  - pressed/released/clicked
  - názov tlačítka myši (LEFT/RIGHT/MIDDLE alebo BUTTONx, kde x je číslo)
  - <https://docs.oracle.com/javase/7/docs/api/java/awt/event/MouseEvent.html>

# Stiahnutie

- shapesGE sa dá stiahnuť na adrese  
<https://github.com/infjava/shapesge/releases/latest>
- dve verzie:
  - shapesGE – rozhranie v angličtine (rovnaké ako projekt shapes z BlueJ)
  - shapesGE-SK – rozhranie v slovenčine (rovnaké ako projekt tvaryV4)

# Inštalácia

- v hlavnom okne BlueJ menu Tools>Preferences...
- záložka Libraries, tlačidlo Add File
- vybrať stiahnutý súbor jar

