



Množiny

Pojmy zavedené v 8. prednáške₍₁₎

- N-rozmerné polia
 - Dvojrozmerné polia – matica
 - definícia – typ[][] premenna
 - inicializácia – new typ[pocetRiadkov][pocetStlpcov]
 - práca s prvkami – premenna[riadok][stlpec]
- Pole polí
 - inicializácia – new typ[pocetRiadkov][]
 - inicializácia prvkov
 - pole[riadok] = new typ[pocetStlpcov]

Pojmy zavedené v 8. prednáške₍₂₎

- vnorené cykly
- this
- sekcie rozhrania
 - verejné – ľubovoľný objekt v okolí
 - neverejné – len objekt vo svojom súkromí

Cieľ prednášky

- základná práca s textovými súbormi
- množinové operácie v jazyku Java
- 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 – 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}
```

```
};
```

```
}
```

Sudoku – nová požiadavka

- Možnosť načítania zadania zo súboru
- Zadania stiahnuté z internetu
- Formát: matica čísel
 - prvky oddelené medzerami
 - riadky oddelené ukončením riadku

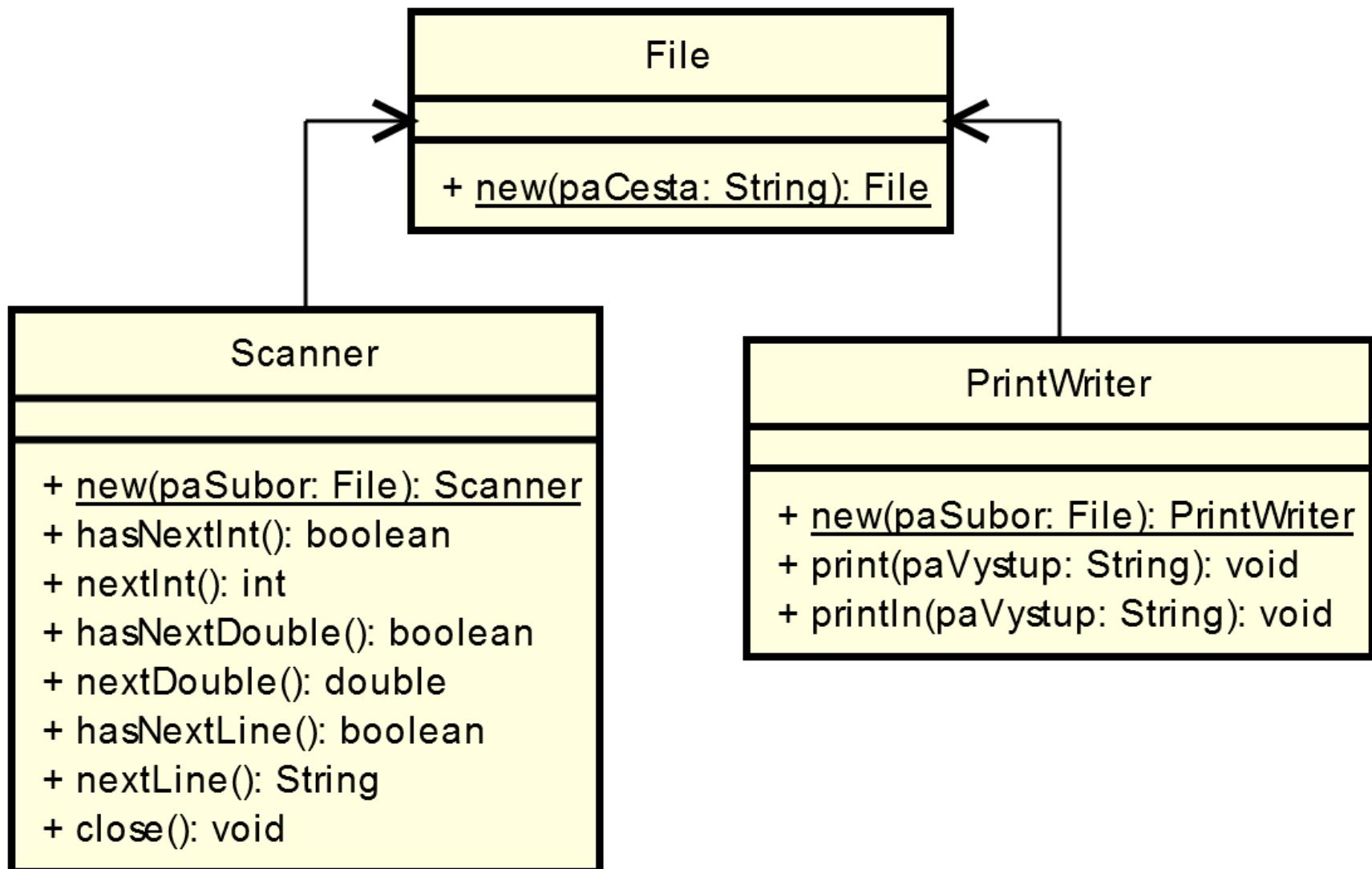
Súbor so zadaním



Práca so súbormi

- čítanie
 - otvorenie súboru na čítanie
 - postupné čítanie obsahu
 - zatvorenie súboru
- zápis
 - otvorenie súboru na zápis
 - postupný zápis nového obsahu
 - zatvorenie súboru

Práca so súbormi v jazyku Java



Sudoku – príkazy import

```
import java.util.Scanner;  
import java.io.File;  
import java.io.IOException;
```

Sudoku – metóda nacitajZoSuboru

```
public void nacitajZoSuboru(String paNazov)  
throws IOException  
{  
    ...  
}
```

Sudoku – metóda nacitajZoSuboru

File subor = **new** File(paNazov);

Scanner citac = **new** Scanner(subor);

otvor

```
for (int riadok = 0; riadok < 9; riadok++) {
```

```
    for (int stlpec = 0; stlpec < 9; stlpec++) {
```

```
        aMriezka[riadok][stlpec] = citac.nextInt();
```

```
}
```

```
}
```

čítaj

citac.close(); zatvor

Sudoku – nová požiadavka

- Možnosť zapísania aktuálneho stavu riešenia do súboru
- Využitie knižničnej triedy `java.io.PrintWriter`
- Formát zhodný s formátom zadania

Sudoku – príkazy import

- Využívame ďalšiu triedu, treba pridať import

```
import java.io.PrintWriter;
```

Sudoku – metóda zapisDoSuboru

```
public void zapisDoSuboru(String paNazov)  
throws IOException
```

```
{
```

```
...
```

```
}
```

Sudoku – metóda zapisDoSuboru

File subor = **new** File(paNazov);

PrintWriter zapisovac = **new** PrintWriter(subor);

otvor

```
for (int[] riadok : aMriezka) {  
    for (int policko : riadok) {  
        zapisovac.print(" " + policko);  
    }  
    zapisovac.println();  
}  
zapisovac.close(); zatvor
```

zapíš

Sudoku – metóda zapisDoSuboru

File subor = **new** File(paNazov);

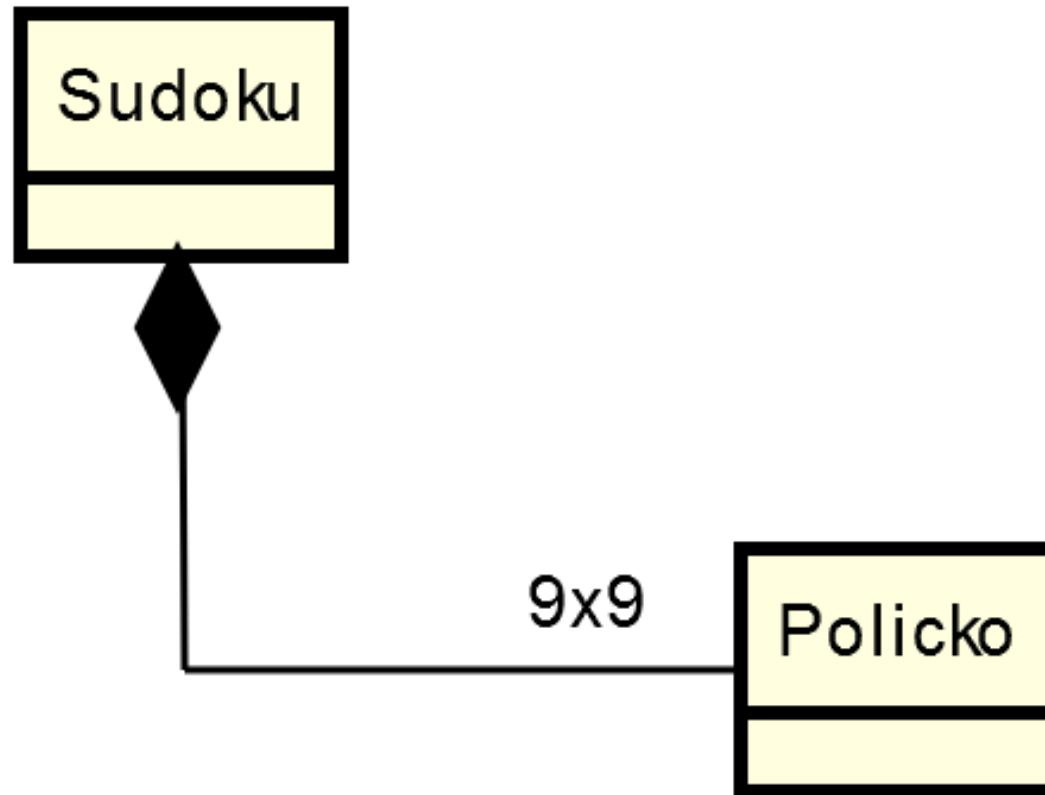
PrintWriter zapisovac = **new** PrintWriter(subor);

```
for (int[] riadok : aMriezka) {  
    for (int policko : riadok) {  
        zapisovac.print(" " + policko);  
    }  
    zapisovac.println();  
}  
zapisovac.close();
```

Políčko

- inicializácia – prázdne políčko
 - test, či je prázdne
 - vkladané číslo – testovanie pravidiel
 - výpis – bodka alebo číslo z <1, 9>
-
- teraz – inštancia triedy Sudoku
 - presun do inštancií triedy Policko

Zavedenie triedy Policko

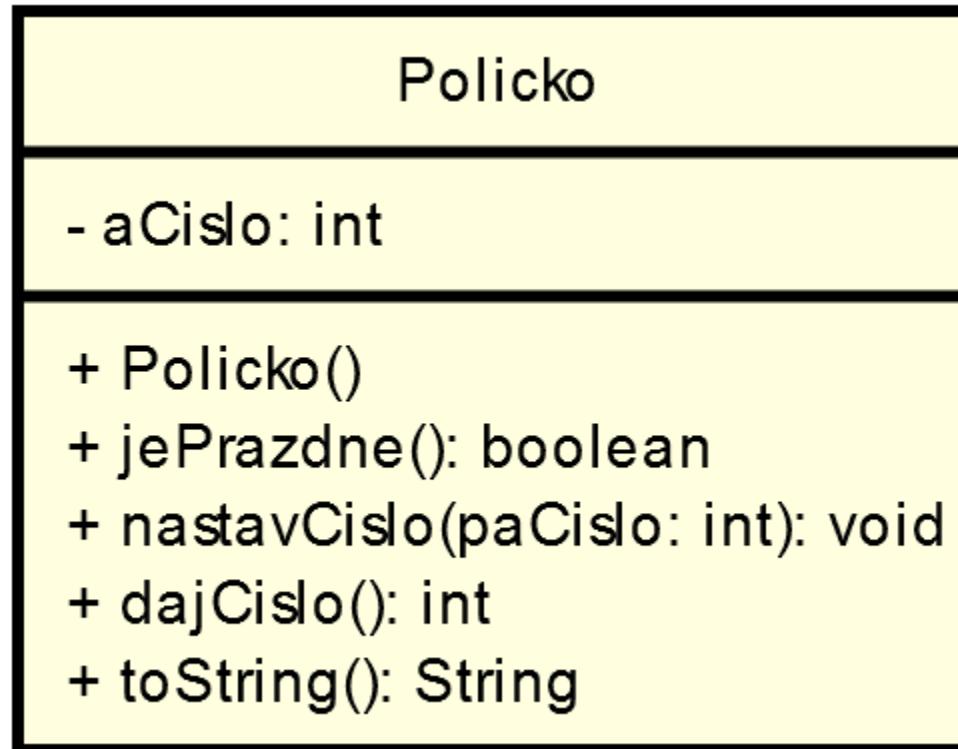


Políčko – rozhranie

Policko

- + new(): Policko
- + jePrazdne(): boolean
- + nastavCislo(paCislo: int): void
- + dajCislo(): int
- + toString(): String

Políčko – vnútorný pohľad



Policko – trieda

```
public class Policko
```

```
{  
    private int aCislo;
```

```
public Policko()
```

```
{  
    aCislo = 0;
```

```
}
```

```
...
```

```
}
```

Policko – jePrazdne

```
public boolean jePrazdne()  
{  
    return aCislo == 0;  
}
```

Policko – nastavCislo

```
public void nastavCislo(int paCislo)  
{  
    if (this.jePrazdne()) {  
        aCislo = paCislo;  
    }  
}
```

Policko – dajCislo

```
public int dajCislo()  
{  
    return aCislo;  
}
```

Policko – `toString`

```
public String toString()
{
    if (this.jePrazdne()) {
        return ".";
    } else {
        return "" + this.dajCislo();
    }
}
```

Metóda `toString`

- automatická konverzia objektov na reťazce
- ak trieda ne definuje svoju metódu `toString` má automatickú
 - vráti reťazcový identifikátor inštancie triedy
- využitie `toString`

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

```
String retazec = "" + policko;
```

Sudoku – trieda – stará

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

Sudoku – trieda – nová

```
public class Sudoku  
{  
    private Policko[][] aHraciePole;  
}
```

...

Sudoku – konštruktor – starý

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

Sudoku – konštruktor – nový

```
public Sudoku()
{
    aHraciePole = new Policko[9][9];
    for (int riadok = 0; riadok < 9; riadok++) {
        for (int stlpec = 0; stlpec < 9; stlpec++) {
            aHraciePole[riadok][stlpec] = new Policko();
        }
    }
}
```

Sudoku – vykresliMriezku – stará

```
for (int[] riadok : aMriezka) {  
    for (int policko : riadok) {  
        if (policko == 0) {  
            System.out.print(".");  
        } else {  
            System.out.print(policko);  
        }  
    }  
    System.out.println();  
}
```

Sudoku –vykresliMriezku – nová

```
for (Policko[] riadok : aMriezka) {  
    for (Policko policko : riadok) {  
        System.out.print(policko);  
    }  
    System.out.println();  
}
```

Sudoku – nová požiadavka

- automatické riešenie Sudoku
- Pridajte metódu `ries`, ktorá sa pokúsi vyriešiť Sudoku

Sudoku – možnosti riešenia

Najjednoduchšie automatické riešenie:

- Jednoduché pravidlo
 - Ak je množina kandidátov políčka jednoprvková, môžeme kandidáta použiť ako hodnotu políčka
- Množina kandidátov – množina všetkých čísel, ktoré možno do políčka vpísat’

Sudoku – kandidáti políčka

stípcoví
kandidáti
políčka

A 9x9 Sudoku grid with some pre-filled numbers. A blue box highlights the second column (columns 1-2). A blue arrow points from the text "stípcoví kandidáti políčka" to the second column. A blue rounded rectangle encloses the numbers 2, 3, 4, 6, and 9, which are the candidates for the empty cell at row 8, column 2.

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

2,3,4,
6,9

Sudoku – kandidáti políčka

riadkoví
kandidáti
políčka

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

4,
6, 7, 9

Sudoku – kandidáti políčka

blokoví
kandidáti
políčka

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

4,
5, 6, 9

Množiny kandidátov

- Množina riadkových kandidátov – pre každý riadok
- Množina stĺpcových kandidátov – pre každý stĺpec
- Množina blokových kandidátov – pre každý blok

Sudoku – kandidáti polička

- Platí:

kandidatiPolicka

=

stlpcoviKandidati

∩

riadkoviKandidati

∩

blokoviKandidati

Sudoku – kandidáti políčka

2	3	4	6	9				
4		6	7	9				
4	5	6		9				
4	6	9						

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



Práca s množinami

- Generická trieda HashSet<TPrvok>
- kontajner bez usporiadania prvkov
- nemožnosť pristúpiť cez index
- možnosť prechádzať pomocou foreach

Trieda HashSet – rozhranie

HashSet<TPrvok>

- + [new\(\): HashSet<TPrvok>](#)
- + add(paPrvok: TPrvok): boolean
- + remove(paPrvok: TPrvok): boolean
- + contains(paPrvok: TPrvok): boolean
- + size(): int
- + isEmpty(): boolean
- + containsAll(paMnozina: HashSet<TPrvok>): boolean
- + addAll(paMnozina: HashSet<TPrvok>): boolean
- + removeAll(paMnozina: HashSet<TPrvok>): boolean
- + retainAll(paMnozina: HashSet<TPrvok>): boolean

Matematické operácie s množinami

- množiny A, B
- prvok x

Prvok množiny

- $x \in A$

A.contains(x)

boolean jePrvkom = A.contains(x);

```
if (A.contains(x)) {  
    ...  
}
```

Množina je prázdna

- $A = \emptyset$

`A.isEmpty()`

`boolean jePrazdna = A.isEmpty();`

`if (A.isEmpty()) {`

`...`

`}`

Množina A je podmnožinou B

- $A \subseteq B$

B.containsAll(A)

boolean jePodmnozinou = B.containsAll(A);

if (B.containsAll(A)) {

...

}

Zjednotenie množín A a B

- $A \cup B \rightarrow C$

```
HashSet<TPrvok> C = new HashSet<TPrvok>();  
C.addAll(A);  
C.addAll(B);
```

Priek množín A a B

- $A \cap B \rightarrow C$

```
HashSet<TPrvok> C = new HashSet<TPrvok>();  
C.addAll(A);  
C.retainAll(B);
```

Rozdiel množín A a B

- $A \setminus B \rightarrow C$ alebo
- $A - B \rightarrow C$

```
HashSet<TPrvok> C = new HashSet<TPrvok>();  
C.addAll(A);  
C.removeAll(B);
```

Sudoku – trieda

```
public class Sudoku  
{  
    private Policko[][] aHraciePole;  
  
    private HashSet<Integer>[] aRiadkoviKandidati;  
    private HashSet<Integer>[] aStlpcoviKandidati;  
    private HashSet<Integer>[][] aBlokoviKandidati;  
  
    ...  
}
```

Sudoku – konštruktor

```
public Sudoku()
{
    aRiadkoviKandidati = new HashSet<Integer>[9];
    aStlpcoviKandidati = new HashSet<Integer>[9];
    aBlokoviKandidati = new HashSet<Integer>[3][3];
    ...
}
```

Chyba – generická trieda ako prvk pol'a

The screenshot shows a Java code editor window titled "Sudoku". The menu bar includes "Class", "Edit", "Tools", and "Options". The toolbar contains "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". A dropdown menu "Source Code" is open. The code editor displays the following Java code:

```
10 private HashSet<Integer>[] aRiadkoviKandidati;
11 private HashSet<Integer>[] aStlpcoviKandidati;
12 private HashSet<Integer>[][] aBlokoviKandidati;
13
14 public Sudoku()
15 {
16     aRiadkoviKandidati = new HashSet<Integer>[9];
17     aStlpcoviKandidati = new HashSet<Integer>[9];
18     aBlokoviKandidati = new HashSet<Integer>[3][3];
```

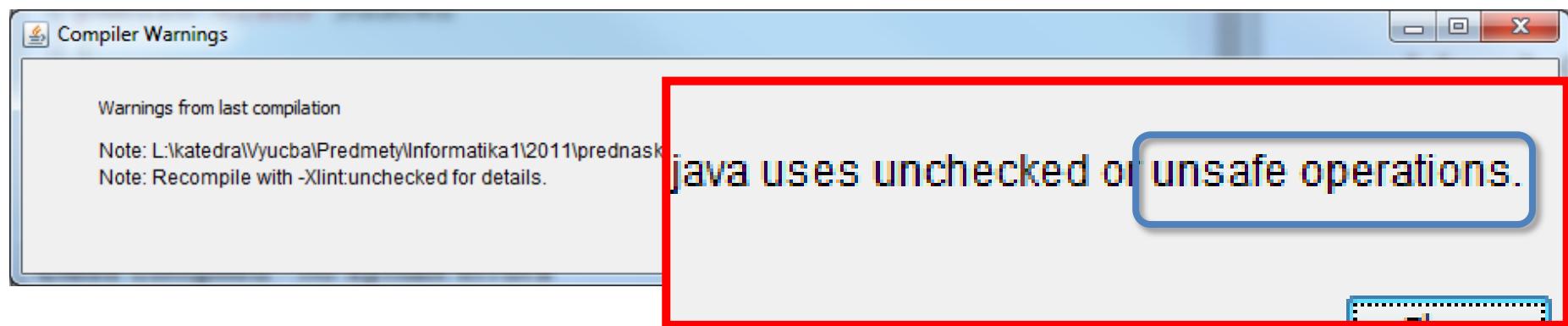
The line `aRiadkoviKandidati = new HashSet<Integer>[9];` is highlighted with a yellow background and a red border, indicating a syntax error. The status bar at the bottom left says "generic array creation".

Sudoku – konštruktor

```
public Sudoku()  
{  
    aRiadkoviKandidati = new HashSet[9];  
    aStlpcoviKandidati = new HashSet[9];  
    aBlokoviKandidati = new HashSet[3][3];  
}
```

...

Varovanie



Sudoku – konštruktor

```
@SuppressWarnings({"unchecked"})
public Sudoku()
{
    aRiadkoviKandidati = new HashSet[9];
    aStlpcoviKandidati = new HashSet[9];
    aBlokoviKandidati = new HashSet[3][3];
```

Sudoku – konštruktor – pokračovanie

- naplnenie všetkých množín kandidátov hodnotami $\langle 1, 9 \rangle$
- na doma

Práca s kandidátmi

- jednoduché riešenie
 - polička s jediným kandidátom
- každé poličko má trojicu množín kandidátov
 - riadok, stĺpec, blok
- kandidáti polička – prienik trojice
- poveríme poličko

Rozšírenie rozhrania triedy Policko

Policko

- + new(paR: HashSet<Int.>, paS: HashSet<Int.>, paB: HashSet<Int.>): Policko
- + jePrazdne(): boolean
- + nastavCislo(paCislo: int): void
- + dajCislo(): int
- + toString(): String
- + dajJedinehoKandidata(): Integer

Sudoku – konštruktor – pokračovanie

```
for (int riadok = 0; riadok < 9; riadok++) {  
    for (int stlpec = 0; stlpec < 9; stlpec++) {  
        aHraciePole[riadok][stlpec] = new Policko(  
            aRiadkoviKandidati[riadok],  
            aStlpcoviKandidati[stlpec],  
            aBlokoviKandidati[riadok/3][stlpec/3]  
        );  
    }  
}
```

Policko – trieda

```
public class Policko
```

```
{
```

```
    private HashSet<Integer> aRiadkoviKandidati;  
    private HashSet<Integer> aStlpcoviKandidati;  
    private HashSet<Integer> aBlokoviKandidati;
```

```
    private int aCislo;
```

Policko – konštruktor

```
public Policko(HashSet<Integer> aRiadkoviKandidati,  
               HashSet<Integer> paStlpcoviKandidati,  
               HashSet<Integer> paBlokoviKandidati)  
{  
    aCislo = 0;  
    aRiadkoviKandidati = paRiadkoviKandidati;  
    aStlpcoviKandidati = paStlpcoviKandidati;  
    aBlokoviKandidati = paBlokoviKandidati;  
}
```

Policko – dajJedinehoKandidata

```
public Integer dajJedinehoKandidata()
{
    HashSet<Integer> kati = new HashSet<Integer>();
    kati.addAll(aRiadkoviKandidati);
    kati.retainAll(aStlpcoviKandidati);
    kati.retainAll(aBlokoviKandidati);
}
```

...

Policko – dajJedinehoKandidata

...

```
if (kati.size() == 1) {  
    for (Integer cislo: kati) {  
        return cislo;  
    }  
}  
return null;  
}
```

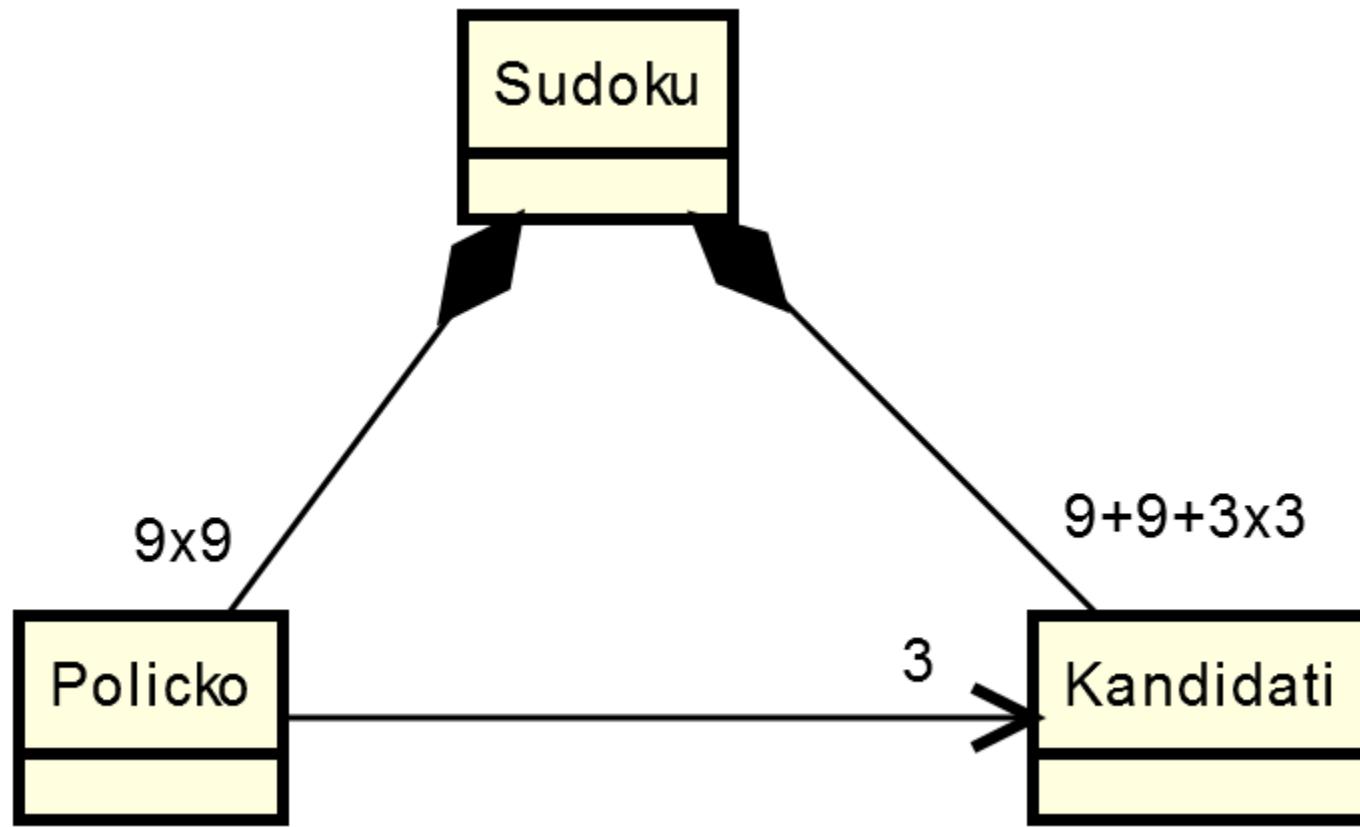
Sudoku – metóda ries

```
public void ries()  
{  
    for (Policko[] riadok : aHraciePole) {  
        for (Policko policko : riadok) {  
            // riešenie pre políčko  
        }  
    }  
}
```

Sudoku – riešenie pre políčko

```
if (policko.jePrazdne()) {  
    Integer kandidat = policko.dajJedinehoKandidata();  
    if (kandidat != null) {  
        policko.nastavCislo(kandidat);  
    }  
}
```

Kandidáti ako trieda



Kandidati – rozhranie

Kandidati

- + new(): Kandidati
- + jePrvok(paCislo: int): boolean
- + vyber(int: paCislo): void
- + prienik(paKandidati: Kandidati): void
- + toString(): String
- + dajPocetPrvkov(): int
- + dajJednehoKandidata(): int

Aké zmeny sú nutné?

- v triede Sudoku
- v triede Policko
- zvážte – DÚ

Ďakujem za pozornosť