Fakulta riadenia a informatiky ŽU

Množiny

Pojmy zavedené v 8. prednáške

- 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

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

Cieľ

- základná práca so 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

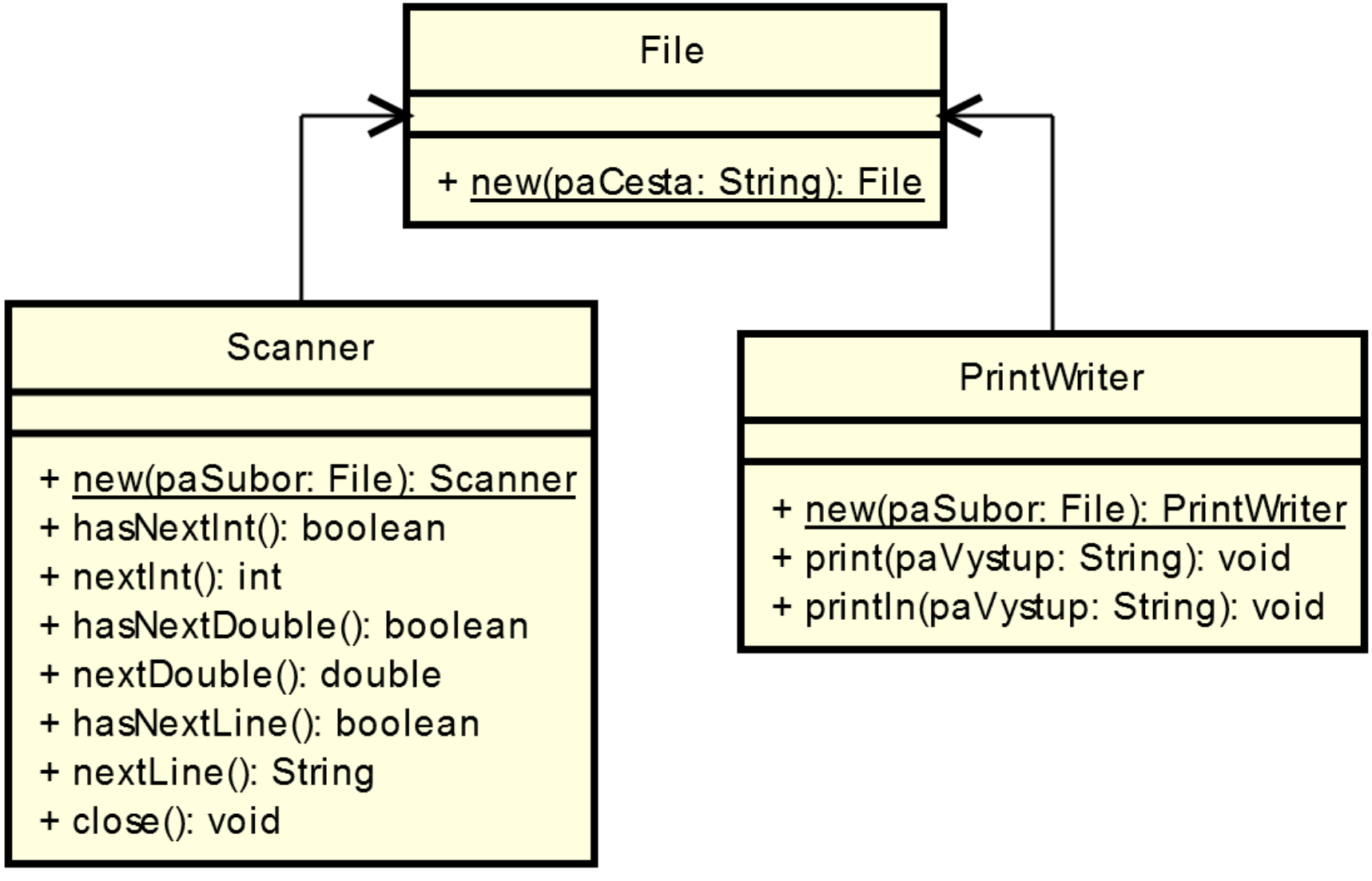
- Možnosť načítania zadania zo súboru
- Zadania stiahnuté z internetu
- Formát: matica čísel
  - prvky oddelené medzerami
  - riadky oddelené ukončením riadku

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);


for (int riadok = 0; riadok < 9; riadok++) {
    for (int stlpec = 0; stlpec < 9; stlpec++) {
        aMriezka[riadok][stlpec] = citac.nextInt();
    }
}
citac.close();
```

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

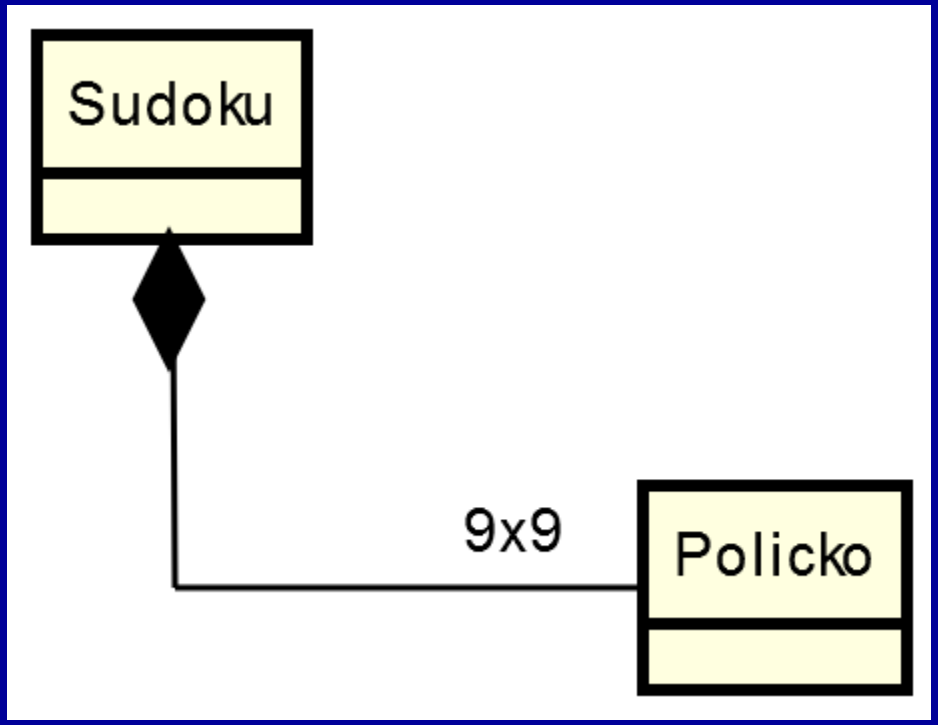
```
File subor = new File(paNazov);
PrintWriter zapisovac = new PrintWriter(subor);


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

Políčko

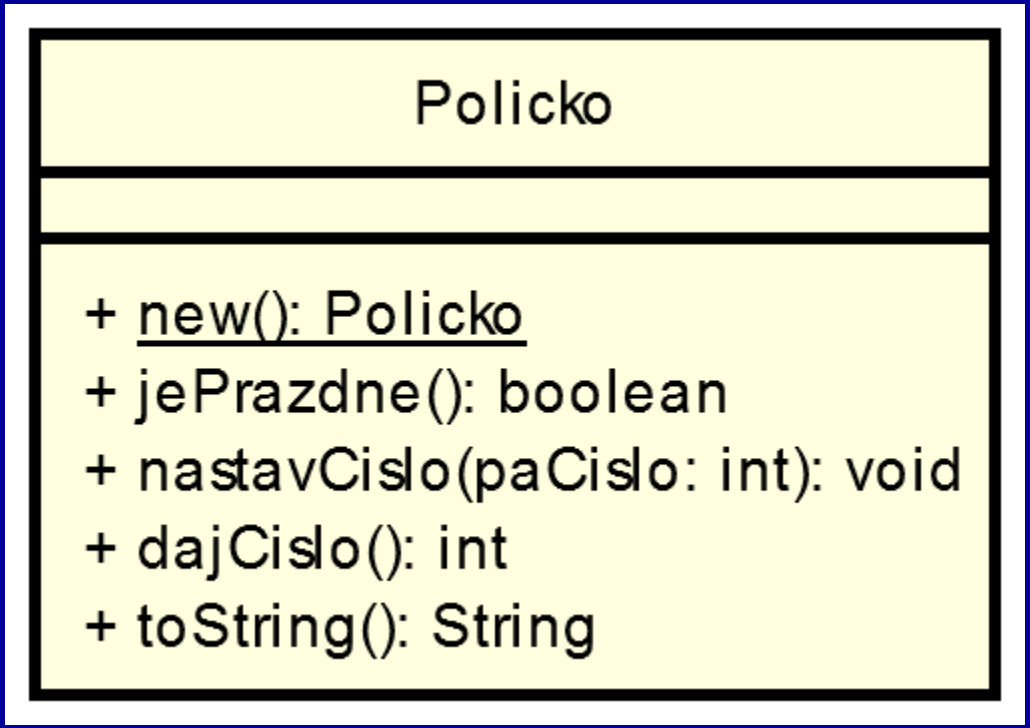
- inicializácia – prázdne políčko
- test, či je prázdne
- vkladané číslo – testovanie pravidiel
- výpis – bodka alebo číslo z <1, 9>

- teraz – inštancia triedy Sudoku
- presun do inštancií triedy Policko

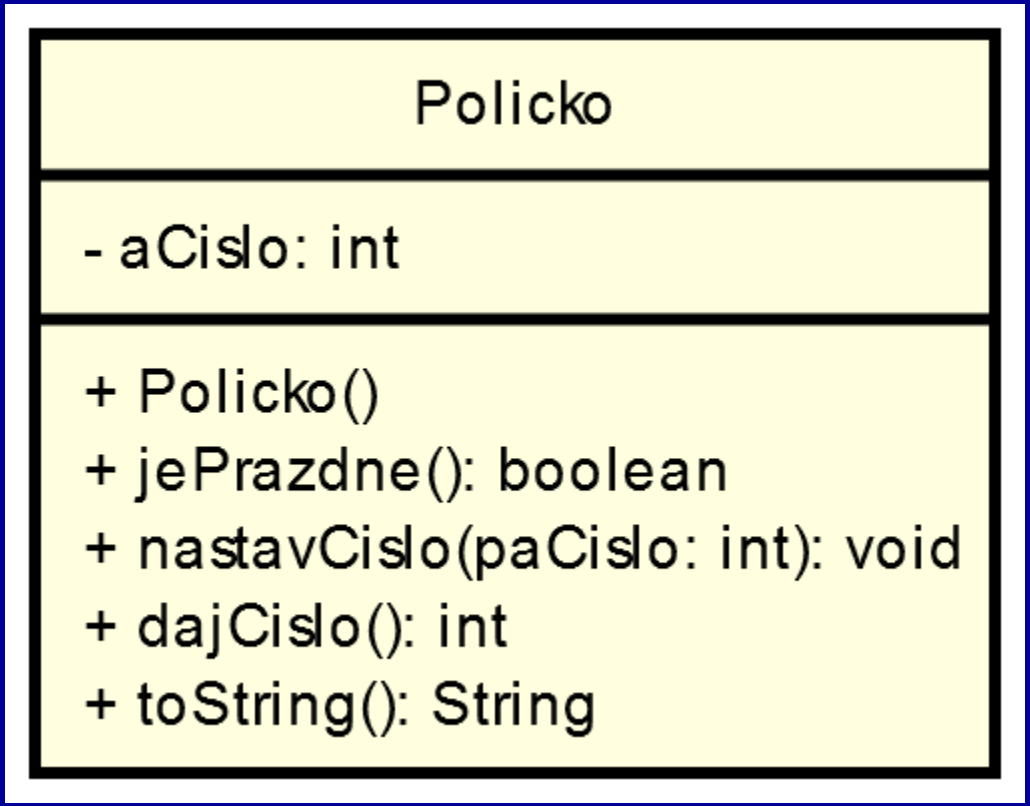
Zavedenie triedy Policko



Políčko – rozhranie



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 nedefinuje svoju metódu toString má automatickú
  - vráti reťazcový identifikátor inštancie triedy

```
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

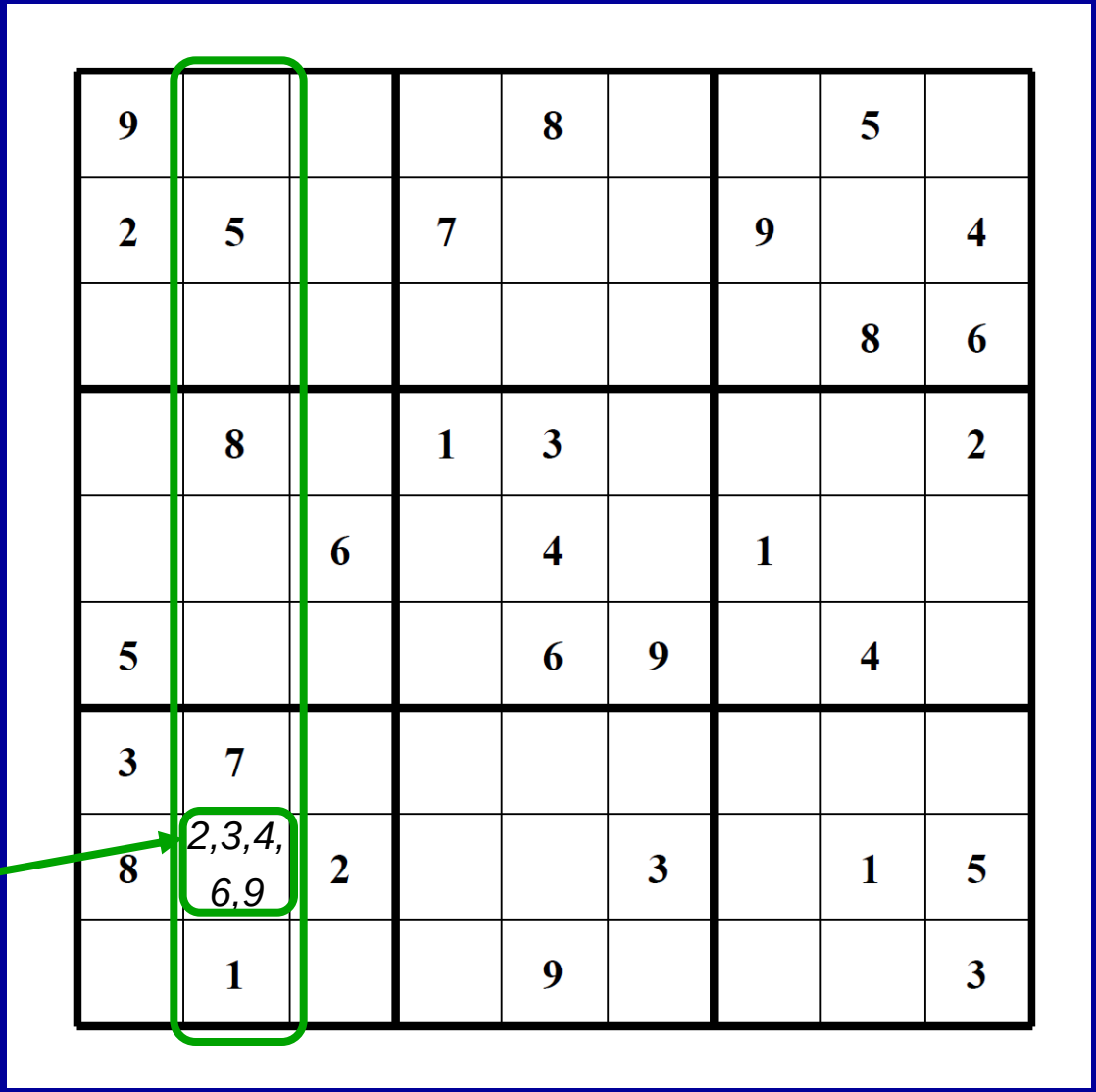
Sudoku

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 vpísať

Sudoku – kandidáti políčka



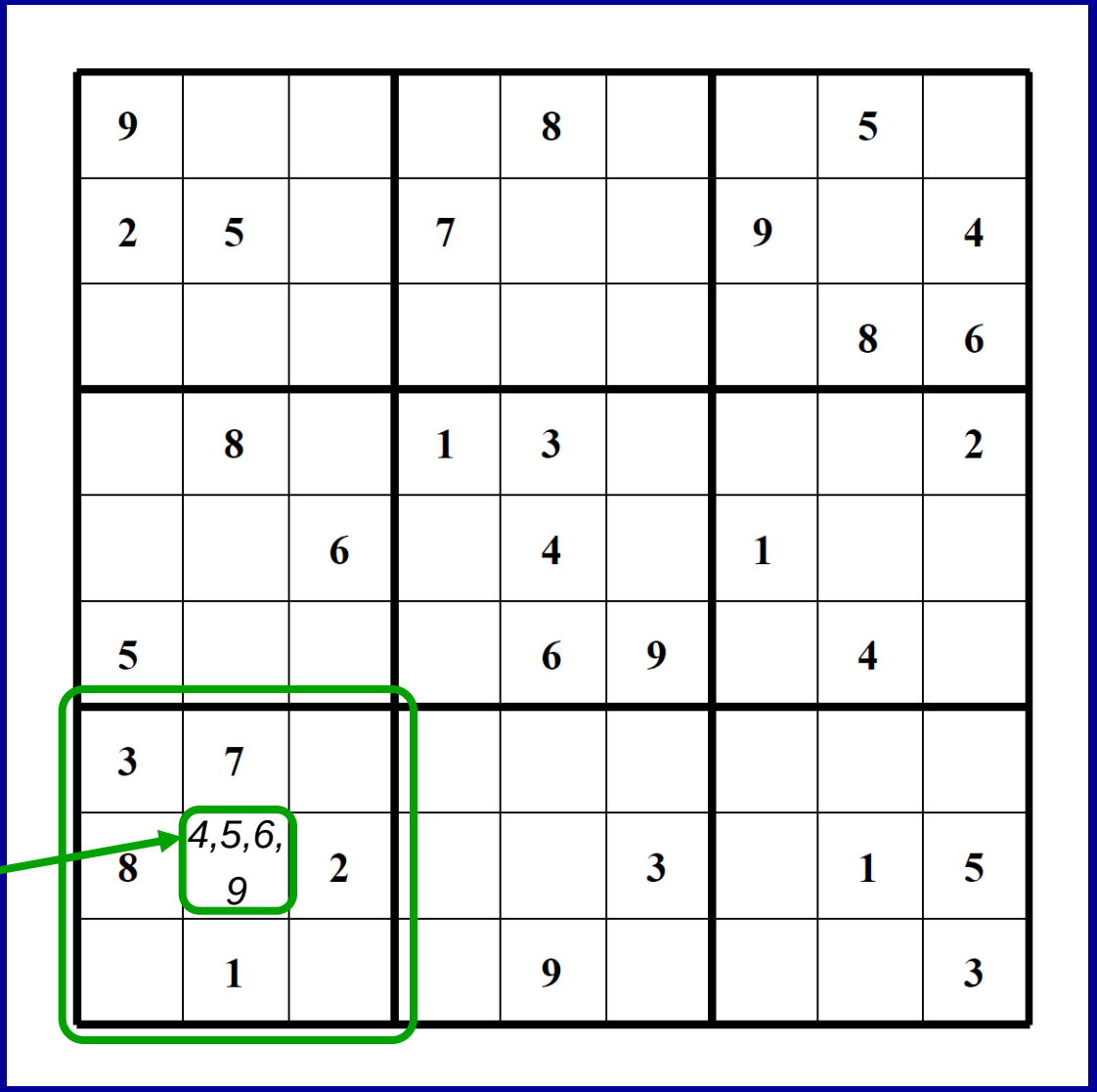
stĺpcoví kandidáti políčka

Sudoku – 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	*4,6,7,9*	2			3		1	5
	1			9				3

riadkoví kandidáti políčka

Sudoku – kandidáti políčka



Množiny kandidátov

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

Sudoku – kandidáti políčka

- Platí:

$$kandidatiPolicka = stlpcoviKandidati \cap riadkoviKandidati \cap blokoviKandidati$$

Sudoku – kandidáti políčka

2, 3, 4, 6, 9

4, 6, 7, 9

4, 5, 6, 9

==========

4, 6, 9

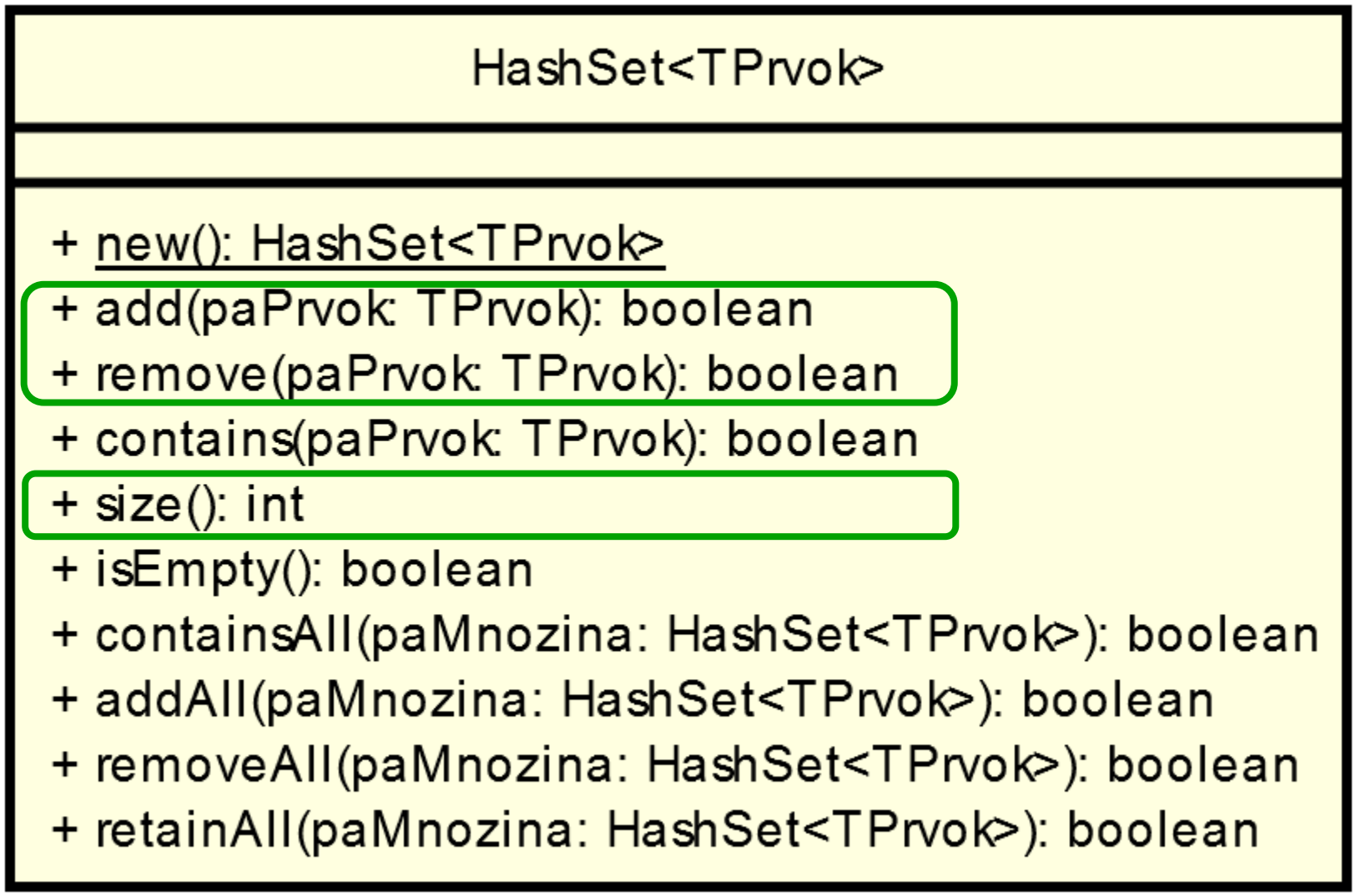
kandidáti políčka

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

Práca s množinami

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

Trieda HashSet – rozhranie



Matematické operácie s množinami

- máme množiny A, B
- máme prvok x

Prvok množiny

- $x \in A$

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

Množina je prázdna

- $A = \emptyset$

```
if (A.isEmpty())
   ...
```

Množina A je podmnožinou B

- $A \subseteq B$

```
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);
```

Prienik 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

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

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

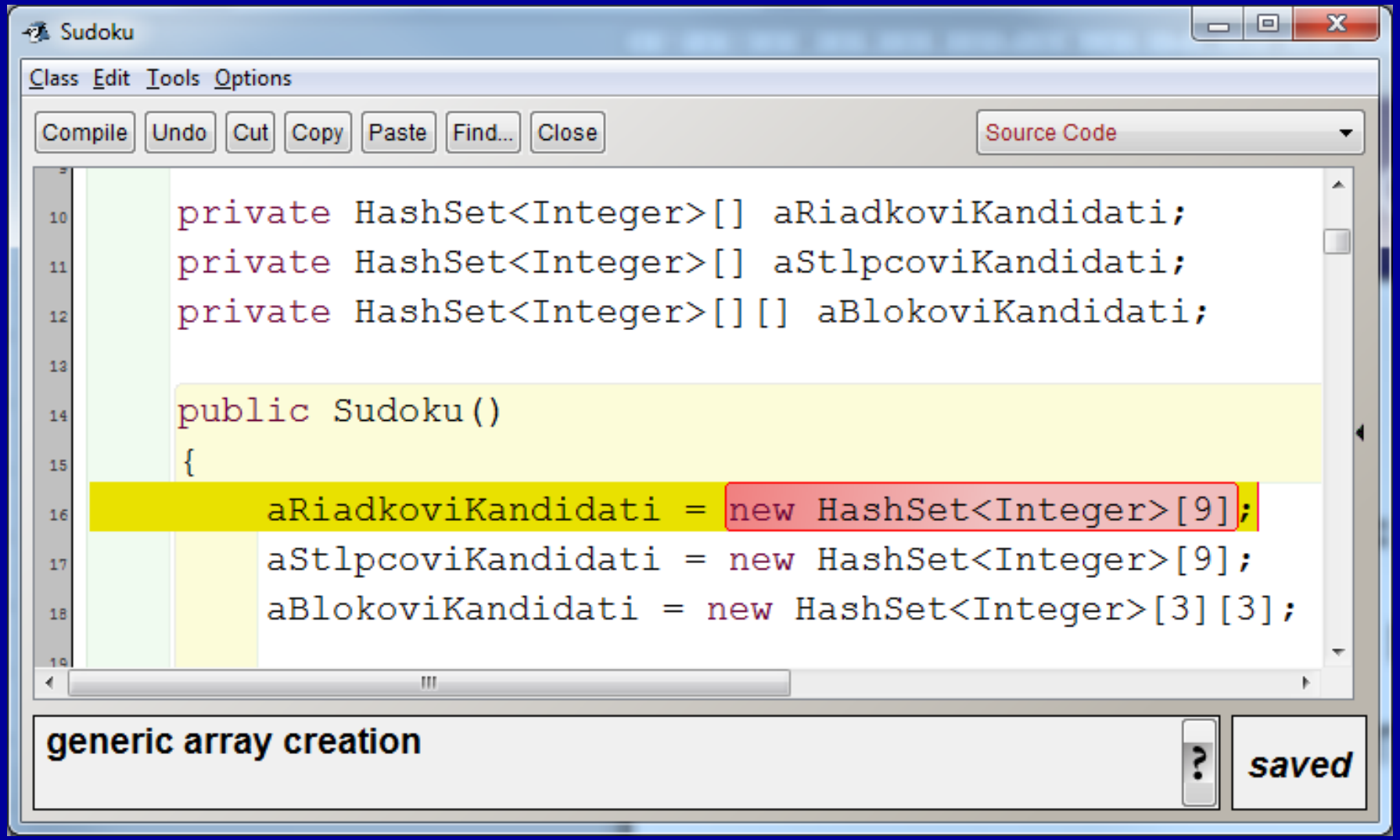
Sudoku – trieda

```
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];
```

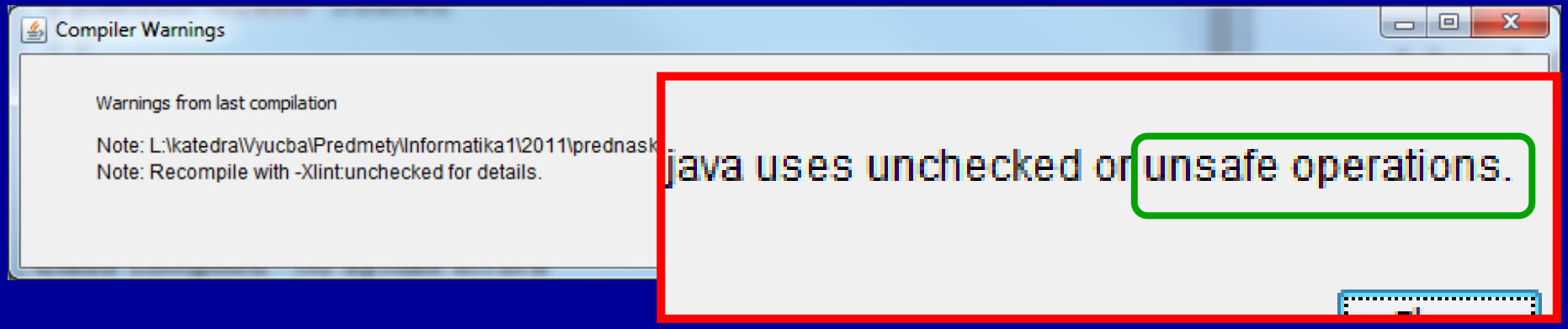
Chyba – generická trieda ako prvok poľa



Sudoku – konštruktor

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

Varovanie


Compiler Warnings
Warnings from last compilation
Note: L:\katedra\Vyucba\Predmety\Informatika1\2011\prednask
Note: Recompile with -Xlint:unchecked for details.
java uses unchecked or unsafe operations.

Sudoku – konštruktor

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

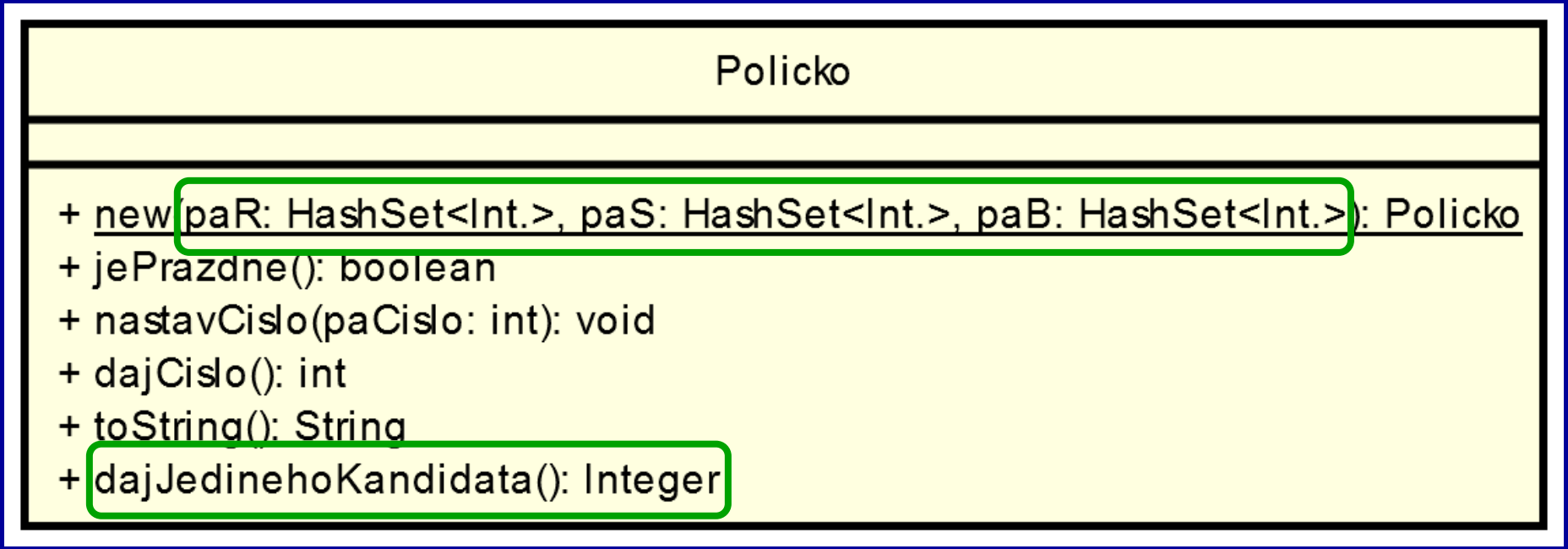
Sudoku – konštruktor – pokračovanie

- naplnenie všetkých množín kandidátov hodnotami <1, 9>
- na doma

Práca s kandidátmi

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

Rozšírenie rozhrania triedy Policko



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

Policko – dajJedinehoKandidata

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

Policko – dajJedinehoKandidata

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

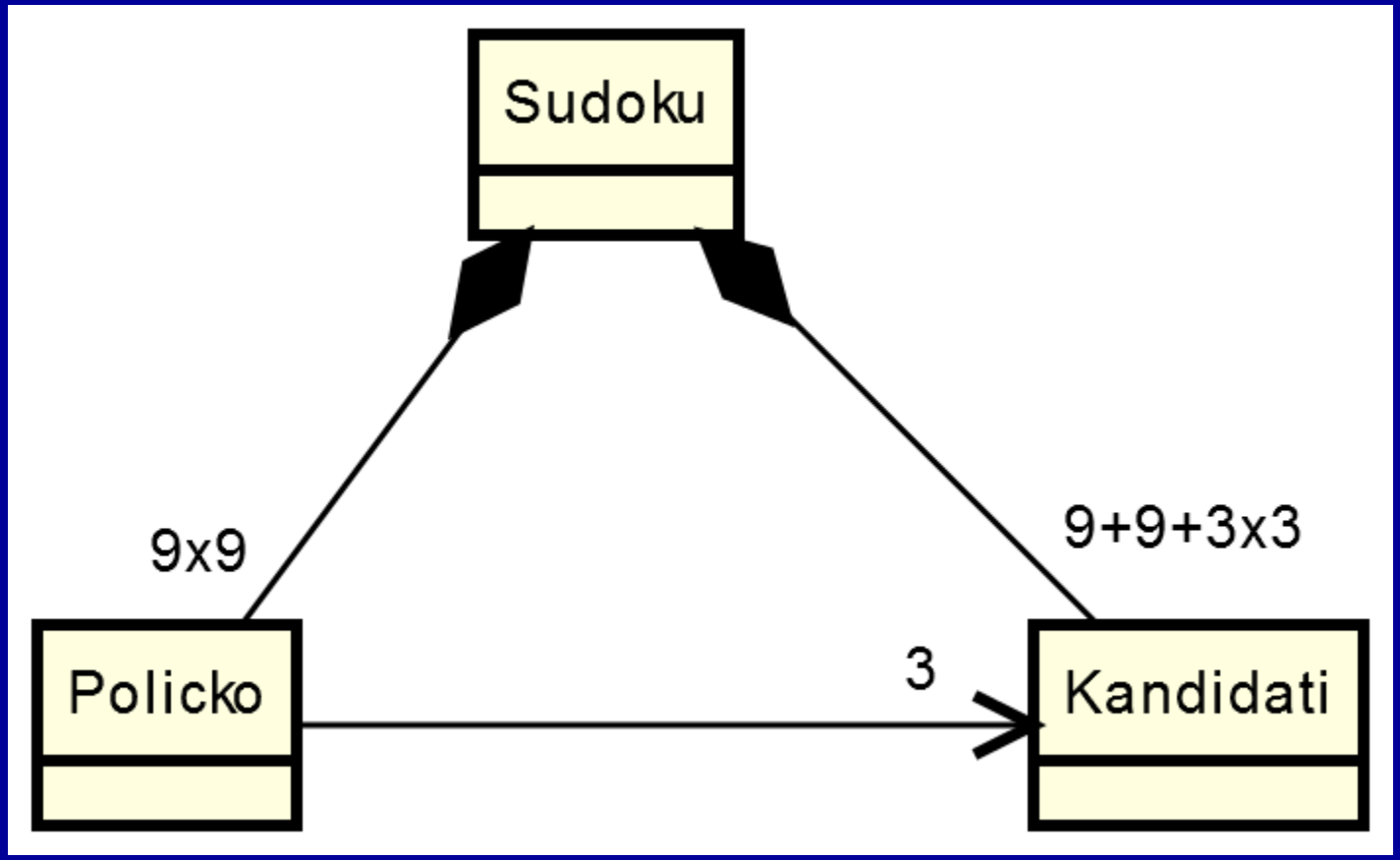
Sudoku – ries(1)

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

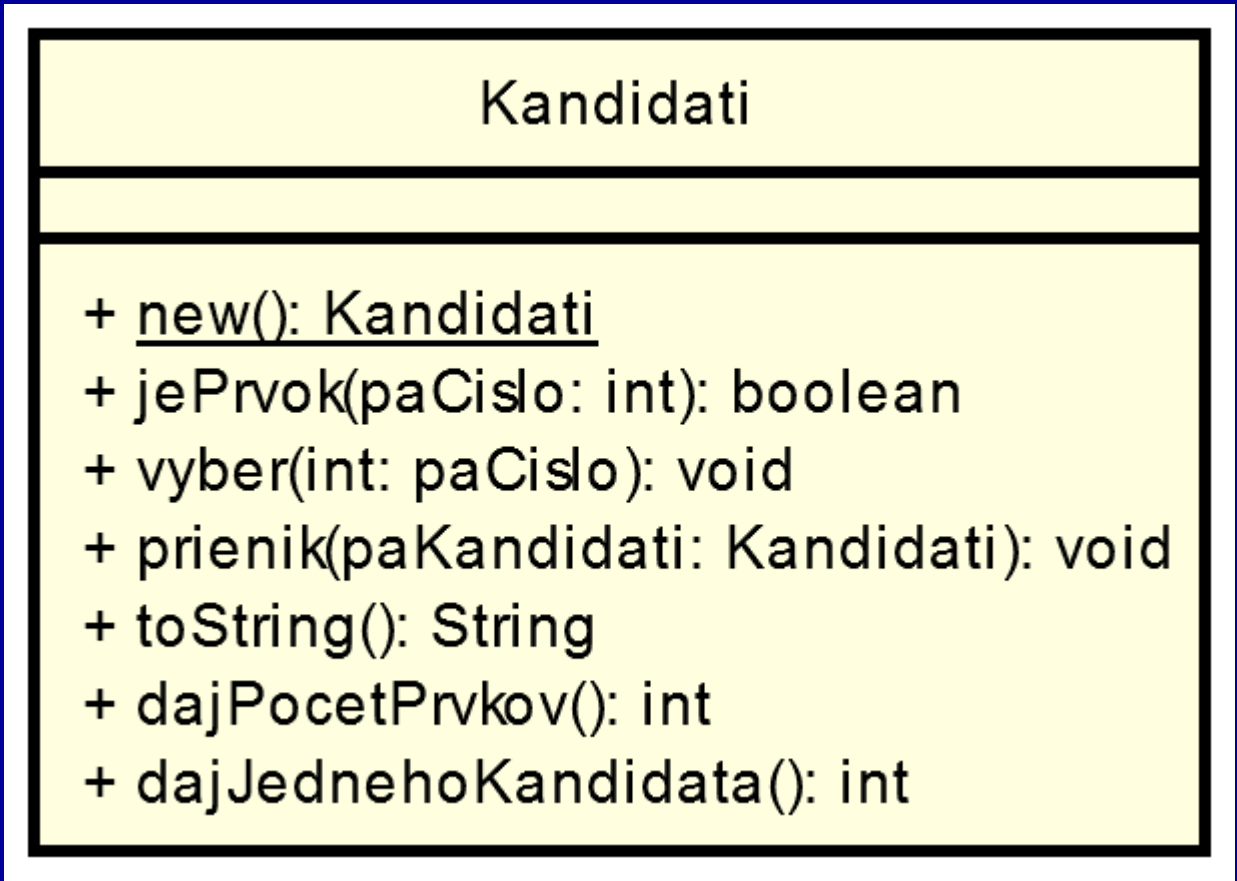
Sudoku – ries(2)

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

Kandidáti ako trieda



Kandidati – rozhranie



Kandidáti – konštruktor

```
public Kandidati()
{
    aKandidati = new HashSet<Integer>();
    for (int i = 1; i <= 9; i++) {
        aKandidati.add(i);
    }
}
```

Kandidáti – dajJednehoKandidata

```
public int dajJednehoKandidata()
{
    for (Integer cislo: aKandidati) {
        return cislo;
    }
    return 0;
}
```

Ďakujem za pozornosť