Informatika 1

7

Polia objektov

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Pojmy zavedené v 6. prednáške(1)

- Skupiny objektov – kontajnery
  - ArrayList
- Generické triedy
- Cyklus
  - for-each
  - while
- Anonymné objekty

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Cieľ prednášky

- Kontajnery – polia
- Obaľovacie triedy
- Cyklus - for
- Diagramy UML pre cykly

- príklad: Štatistika pripojení

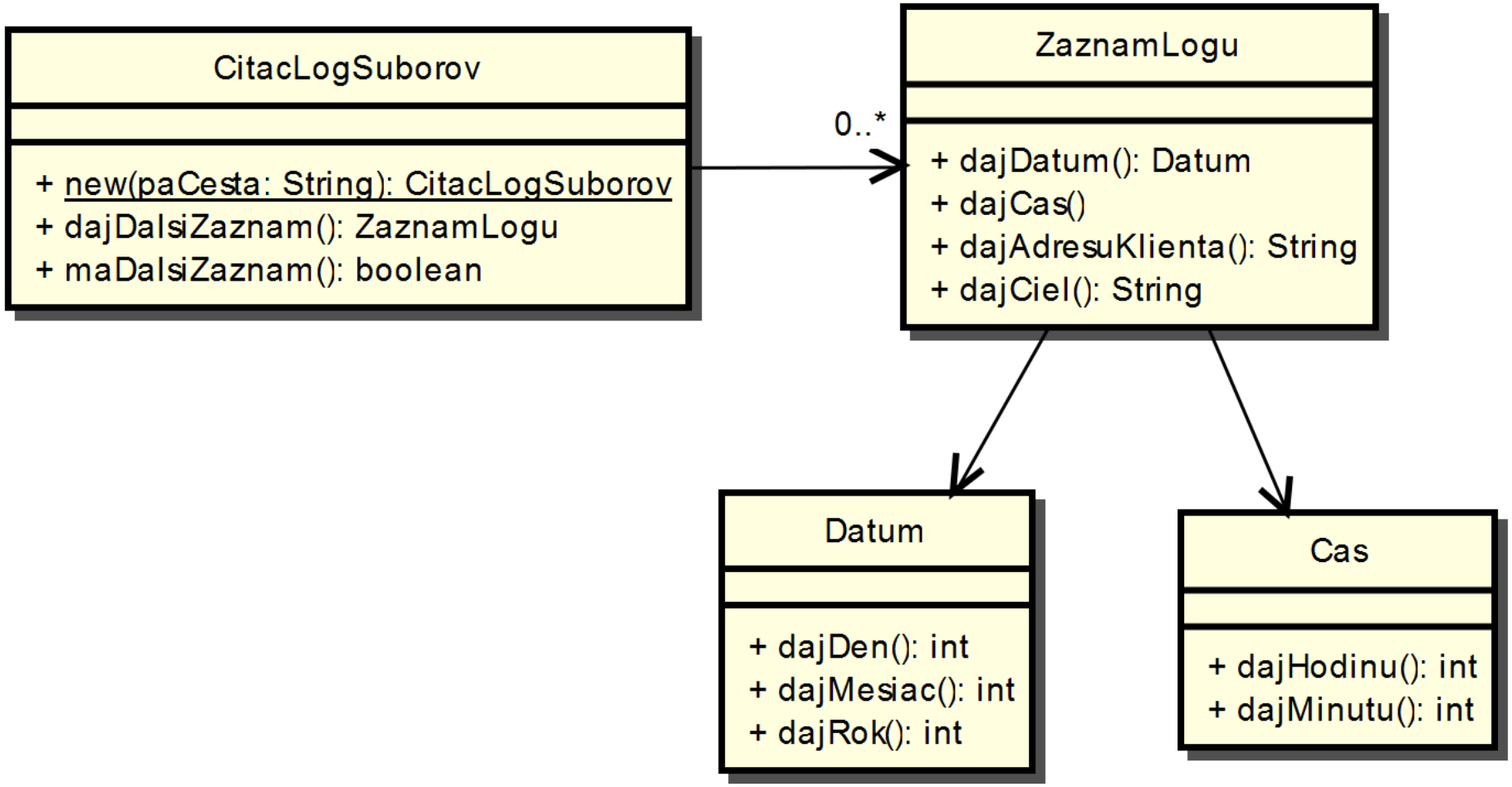
Štatistika pripojení na web server

- web server eviduje intenzitu pripájania
- pre každé pripojenie zaznamenáva
  - dátum a čas v tvare pätice 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

2010 09 01 07 45	166.254.109.216	/spravicky/16488
2010 09 01 08 40	173.137.190.129	/clanky/16763
2010 09 01 09 00	215.145.186.231	/clanky/7717
2010 09 01 09 50	190.232.201.202	/spravicky/43318
2010 09 01 10 04	109.97.94.95	/clanky/16189
2010 09 01 10 27	229.249.211.233	/clanky/13231
2010 09 01 10 59	165.246.158.207	/clanky/17016
2010 09 01 11 02	197.229.220.190	/clanky/22915
2010 09 01 11 04	217.220.209.238	/spravicky/13879
2010 09 01 11 06	85.128.39.62	/spravicky/13541
2010 09 01 11 35	186.246.148.205	/spravicky/23875
2010 09 01 11 40	244.162.121.146	/spravicky/49726
2010 09 01 11 44	109.97.94.95	/spravicky/38129
2010 09 01 11 45	96.201.235.232	/spravicky/42464
2010 09 01 11 49	180.184.204.217	/clanky/17001
2010 09 01 11 53	238.239.163.111	/clanky/6111

Práca s log súborom – rozhrania

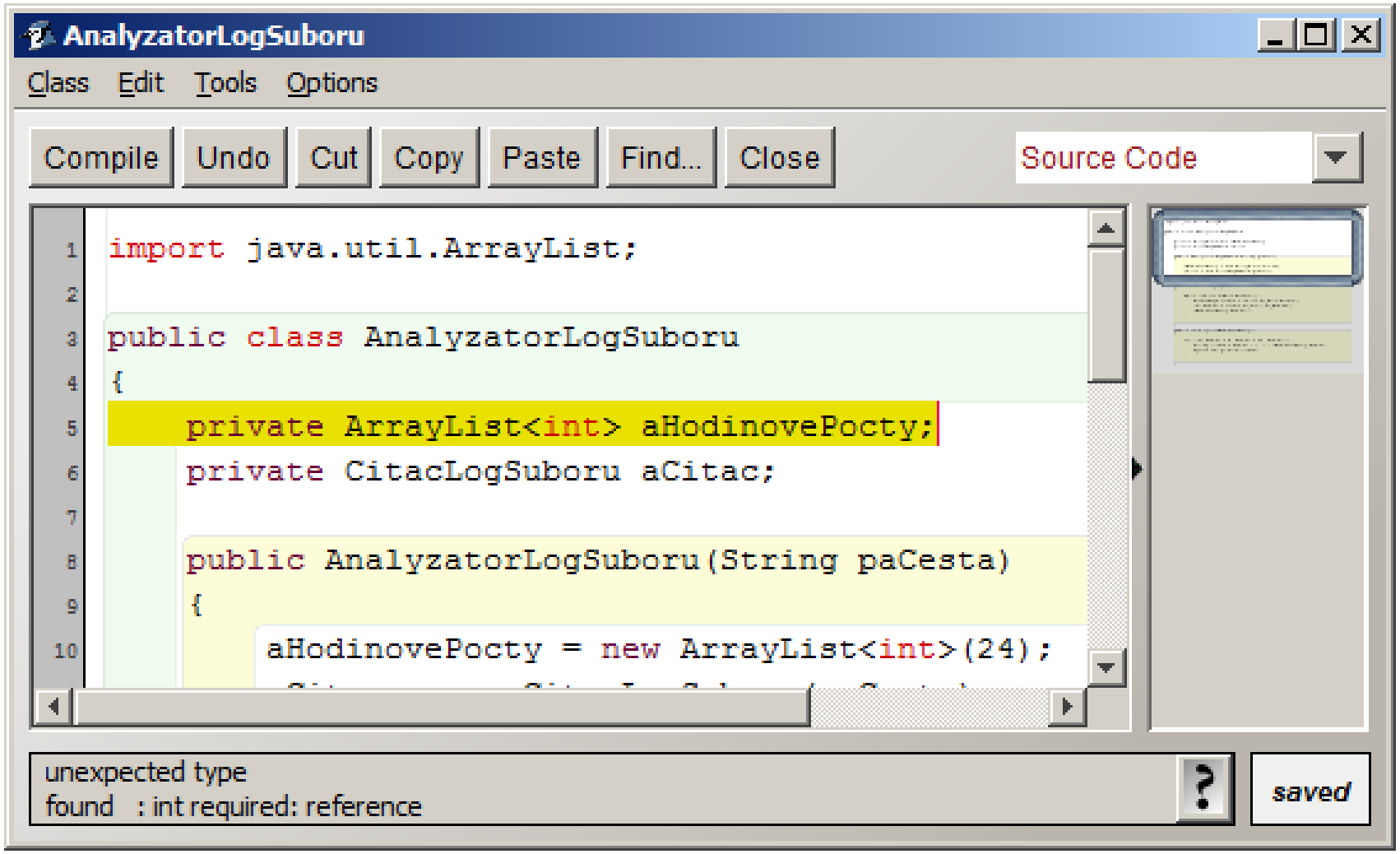


Analýza

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

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Chyba pri vytváraní kontajnera



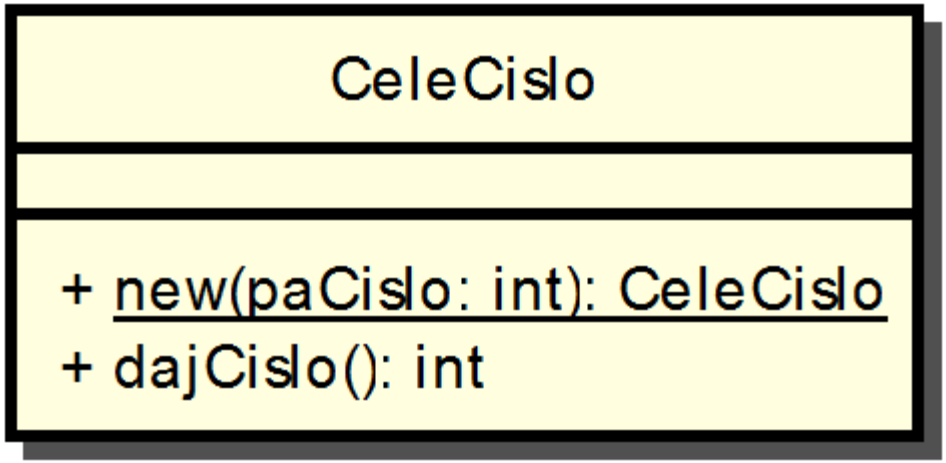
KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Problém s vytváraním kontajnera

- typ prvkov ArrayList – objektový typ
- naše prvky – čísla – primitívny typ

- riešenie – obaľovacie triedy

Obaľovacia trieda CeleCislo



Obaľovacie triedy v jazyku Java

primitívny typ	obaľovacia trieda
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Použitie obaľovacích tried

```
int celeInt = 15;

Integer obaleneCislo = new Integer(celeInt);

int druheCeleInt = obaleneCislo.intValue();
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Automatické konverzie

```
int celeInt = 15;

Integer obaleneCislo = celeInt;

int druheCeleInt = obaleneCislo;

obaleneCislo = obaleneCislo + 1;
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Obaľovacia trieda a ArrayList

```
ArrayList<Integer> cisla = new ArrayList<Integer>();

cisla.add(15);
cisla.add(cislo);
```

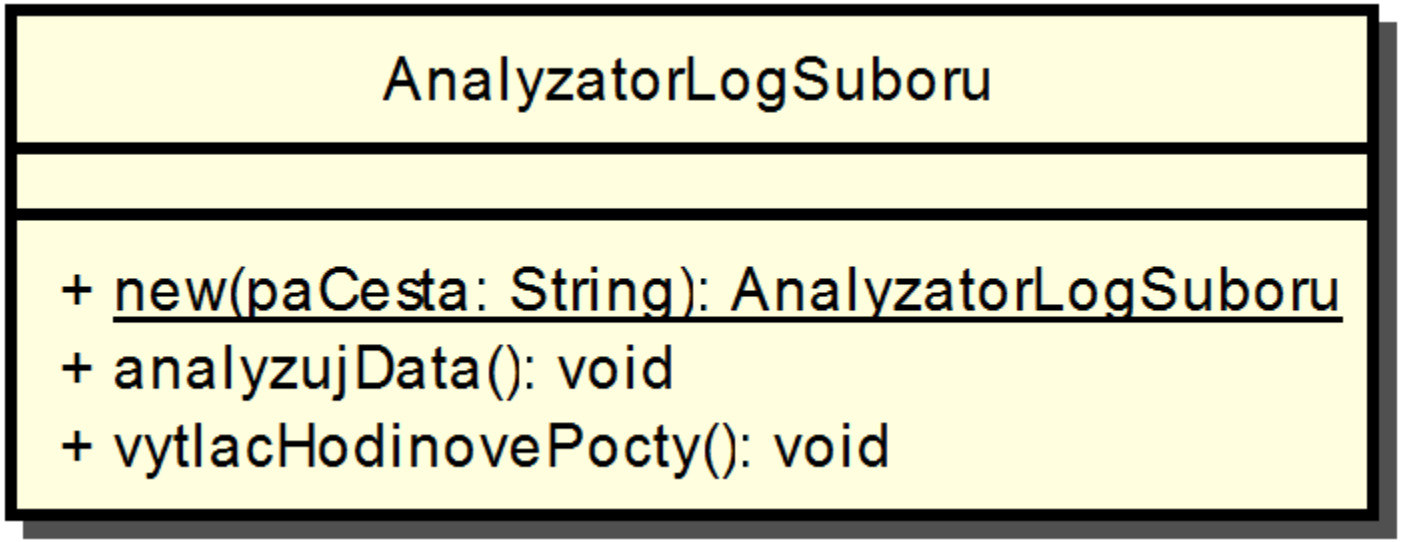

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Analýza

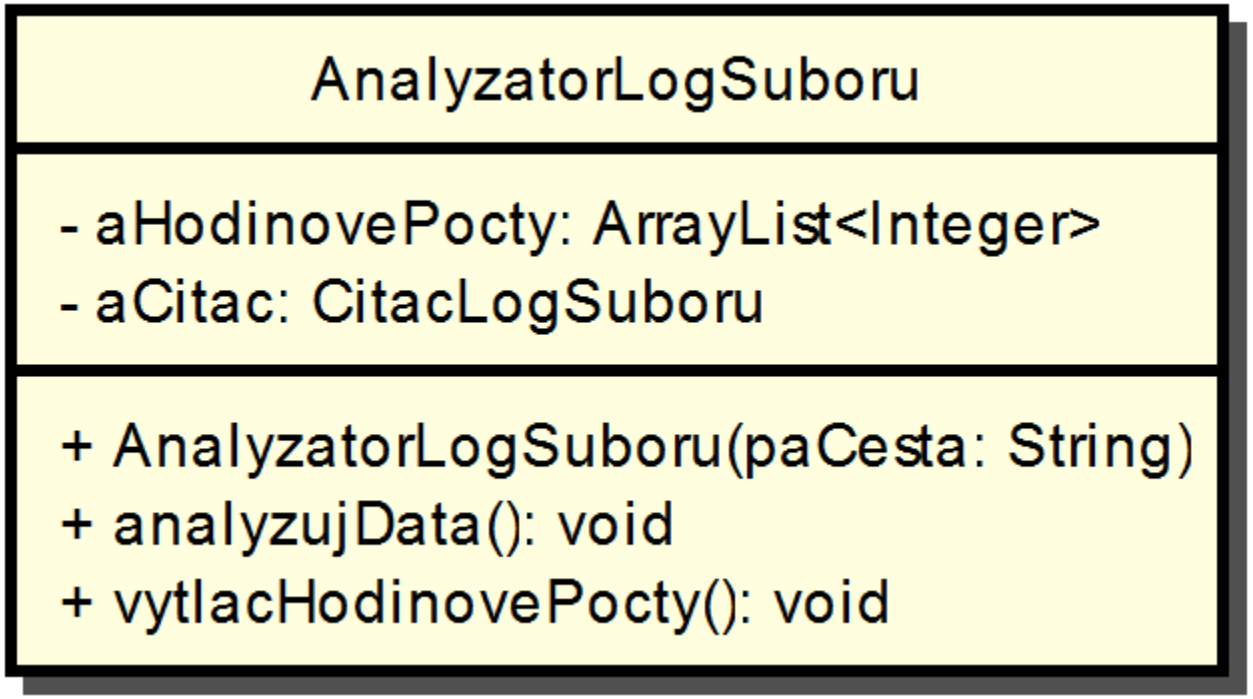
- ArrayList<Integer> – kontajner na čísla
- hodiny 0 až 23 –24 prvkov
- začiatočná hodnota každého prvku – 0

- vkladanie v cykle

AnalyzatorLogSuboru – rozhranie



Vnútorný pohľad



Trieda AnalyzatorLogSuboru

```
import java.util.ArrayList;

public class AnalyzatorLogSuboru
{
    private ArrayList<Integer> aHodinovePocty;
    private CitacLogSuboru aCitac;
    ...
}
```

Používanie knižníc(1)

- knižnice – rozširujúca funkčnosť
  - delí sa na balíčky
  - balíčky obsahujú triedy
  - poznáme len rozhrania
- štandardná knižnica – súčasť jazyka Java
  - String – balíček java.lang
  - ArrayList – balíček java.util

Používanie knižníc(2)

- príkaz import:

```
import nazovBalicka.NazovTriedy;
```

- príklad:

```
import java.util.ArrayList;
```

- pre triedy z balíčka java.lang netreba import

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

AnalyzatorLogSuboru – konštruktor(1)

```
public AnalyzatorLogSuboru(String paCesta)
{
    aCitac = new CitacLogSuboru(paCesta);
    aHodinovePocty = new ArrayList<Integer>();

    ...
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

AnalyzatorLogSuboru – konštruktor(2)

```
// inicializacia
int i = 0;
while (i < 24) {
    aHodinovePocty.add(0);
    i++;
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Cyklus for

- ďalší z cyklov
- pravidlo:
  - inicializuj premennú cyklu na zadanú hodnotu
  - vykonávaj kým platí podmienka
    - príkazy tela cyklu
    - príkaz kroku

```
for (inicializácia; podmienka; krok) {
   // telo cyklu
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Cyklus for - inicializácia

```
for (int i = 0; i < 24; i++) {
    // telo cyklu
}
```

- deklarácia a inicializácia premennej cyklu
- TypPrvku premennaCyklu = zaciatocnaHodnota
- zaciatocnaHodnota -> ľubovoľný výraz

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Cyklus for - podmienka

```
for (int i = 0; i < 24; i++) {
    // telo cyklu
}
```

- relačný výraz
- všeobecne – logický výraz
- podmienka skončenia cyklu
  - false – cyklus končí

Cyklus for – krok

```
for (int i = 0; i < 24; i++) {
    // telo cyklu
}
```

- príkaz, ktorým meníme premennú cyklu

Operátor ++

```
premenna ++;
```

- je ekvivalentné

```
premenna = premenna + 1;
```

- operátor inkrementácie – zväčšenia
- aplikovateľný na všetky číselné typy

Operátor --

```
premenna --;
```

- je ekvivalentné

```
premenna = premenna - 1;
```

- operátor dekrementácie – zmenšenia
- aplikovateľný na všetky číselné typy

Cyklus for a operátor ++

```
for (int i = 0; i < 24; i++) {
    // telo cyklu
}
```

- je ekvivalent

```
for (int i = 0; i < 24; i = i + 1) {
    // telo cyklu
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Cyklus for – ekvivalent cyklu while

```
for (int i = 0; i < 24; i++) {
    aHodinovePocty.add(0);
}
```

- je ekvivalent

```
int i = 0;
for (i < 24) {
    aHodinovePocty.add(0);
    i++;
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

AnalyzatorLogSuboru – konštruktor

```
public AnalyzatorLogSuboru(String paCesta)
{
    aCitac = new CitacLogSuboru(paCesta);
    aHodinovePocty = new ArrayList<Integer>();

    for (int i = 0; i < 24; i++) {
        aHodinovePocty.add(0);
    }
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

AnalyzatorLogSuboru – analyzujData

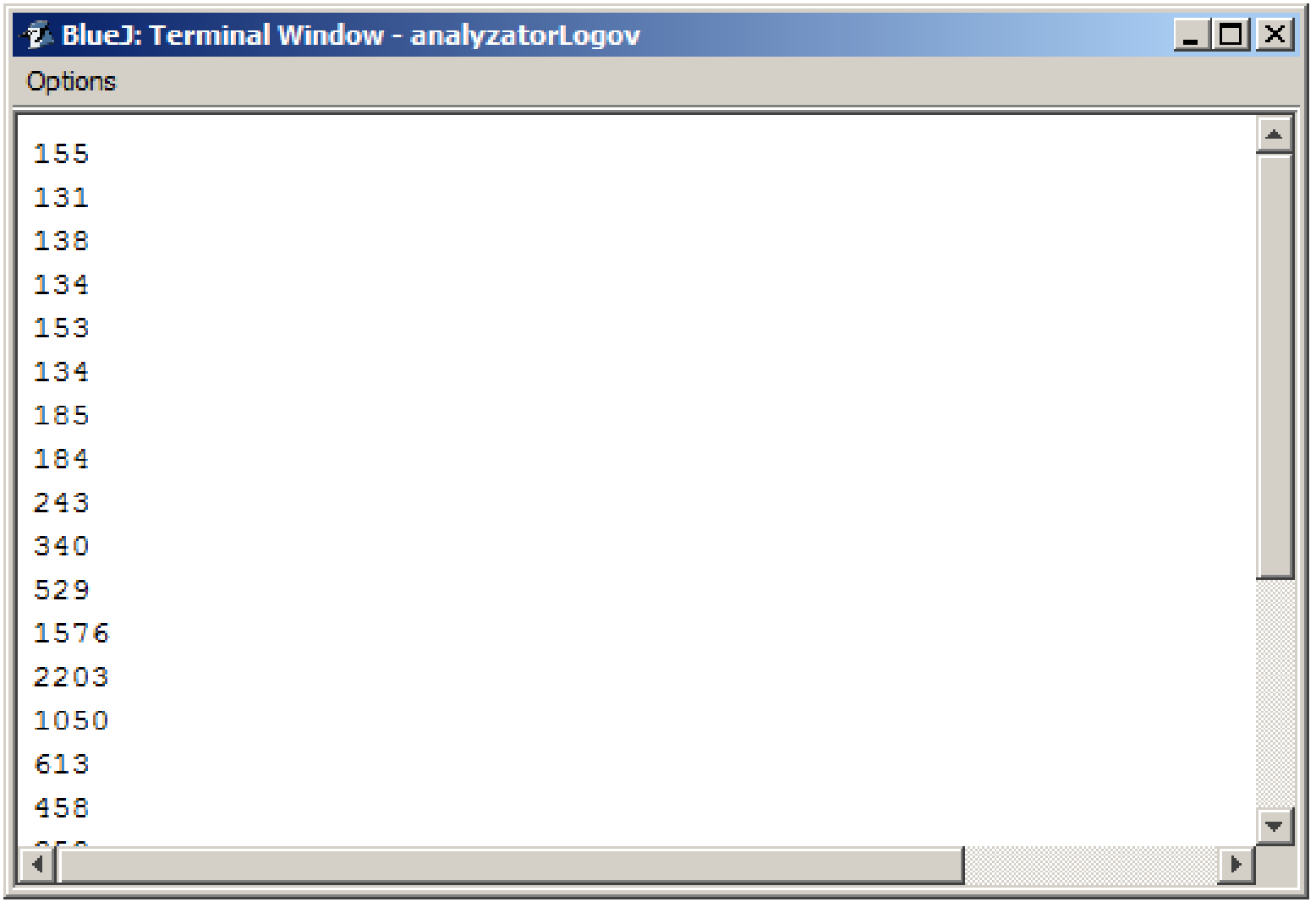
```
public void analyzujData()
{
   while (aCitac.maDalsiZaznam()) {
      ZaznamLogu zaznam = aCitac.dajDalsiZaznam();
      Cas cas = zaznam.dajCas();
      int hodina = cas.dajHodinu();
      int pocet = aHodinovePocty.get(hodina) + 1;
      aHodinovePocty.set(hodina, pocet);
   }
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

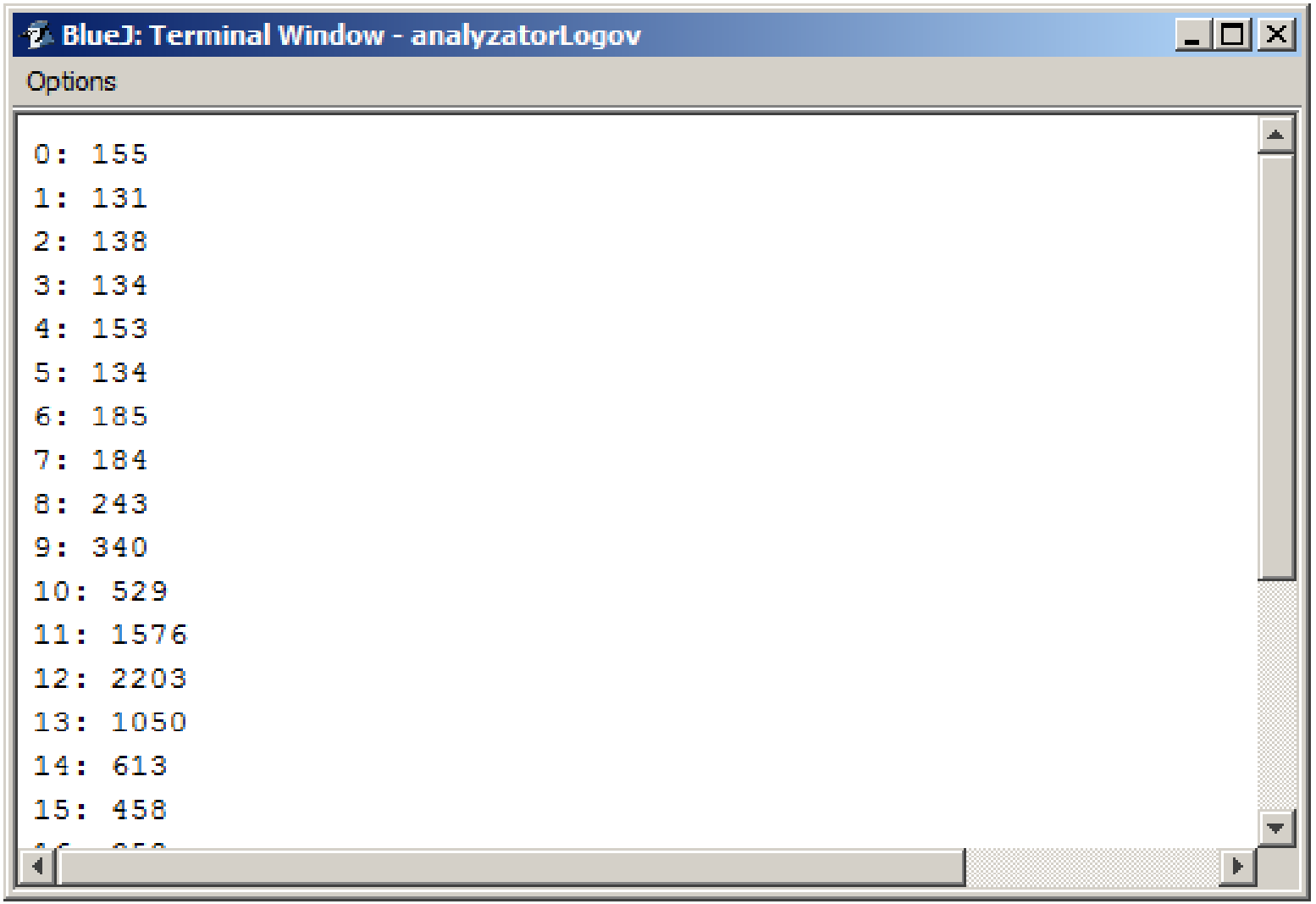
Metóda vytlacHodinovePocty

```
public void vytlacHodinovePocty()
{
    for (Integer pocet : aHodinovePocty) {
        System.out.println(pocet);
    }
}
```

Výstup


BlueJ: Terminal Window - analyzatorLogov
Options
155
131
138
134
153
134
185
184
243
340
529
1576
2203
1050
613
458

Požadovaný výstup



Tlač s hodinami

```
public void vytlacHodinovePocty()
{
    int hodina = 0;
    for (Integer pocet : aHodinovePocty) {
        String riadok = hodina + ": " + pocet;
        System.out.println(riadok);
        hodina++;
    }
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Tlač s hodinami pomocou cyklu for

```
public void vytlacHodinovePocty()
{
    for (int hodina = 0; hodina < 24; hodina++) {
        String riadok = hodina + ": " +
                                    aHodinovePocty.get(hodina);
        System.out.println(riadok);
    }
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

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 kontajner – 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 kontajner – 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;
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

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];
```

Inicializácia poľa vymenovaním prvkov

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

- zoznamPrvkov – čiarkou oddelený zoznam prvkov vytváraného poľa

- príklady:

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

Java – prístup k prvkom poľa(1)

- pomenovanie prvkov – meno poľa + index

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

- 0 ≤ index prvku < 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 Automat(50);
```

- príklad čítanie z poľa:

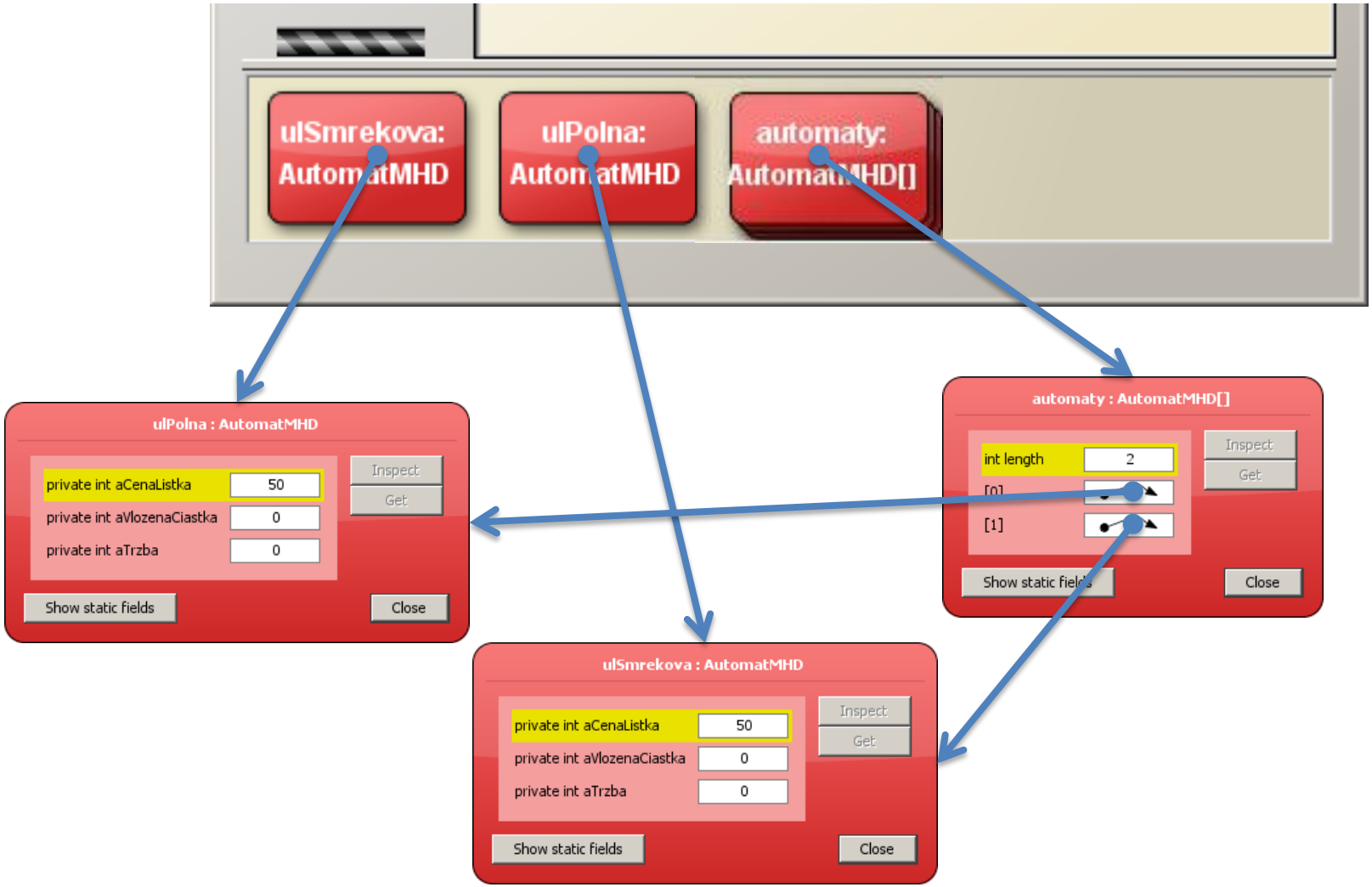
```
System.out.println(pocetPristupov[0]);
automaty[0].vytlacListok();
```

- príklad kombinovaný:

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


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Prvky poľa ako premenné



AutomatMHD[] automaty = {ulSmrekova, ulPolna};

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

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]);
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

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
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Trieda AnalyzatorLogSuboru

```
public class AnalyzatorLogSuboru
{
    private int[] aHodinovePocty;
    private CitacLogSuboru aCitac;


    ...
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

AnalyzatorLogSuboru – konštruktor

```
public AnalyzatorLogSuboru(String paCesta)
{
    aHodinovePocty = new int[24];
    aCitac = new CitacLogSuboru(paCesta);
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

AnalyzatorLogSuboru – analyzujData

```
public void analyzujData()
{
   while (aCitac.maDalsiZaznam()) {
      ZaznamLogu zaznam = aCitac.dajDalsiZaznam();
      Cas cas = zaznam.dajCas();
      int hodina = cas.dajHodinu();
      aHodinovePocty[hodina]++;
}
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Typy a polia

- definícia premennej

```
typPremennej nazovPremennej
```

- návratová hodnota

```
typ nazovMetody(parametre)
```

- typy
  - primitívne – int, float, boolean, ...
  - objektové – názov triedy: String
  - objektové – polia: typPrvku[]: int[], String[]

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Pole a prvok poľa

```
int[] mince = {1, 2, 5, 10, 20, 50};
AutomatMHD[] automaty = {ulSmrekova, ulPolna};
```

- int[] – int
  - mince – mince[3]
- AutomatMHD[] – AutomatMHD
  - automaty – automaty[1]
  - ulPolna – automaty
  - ulPolna – automaty[1]

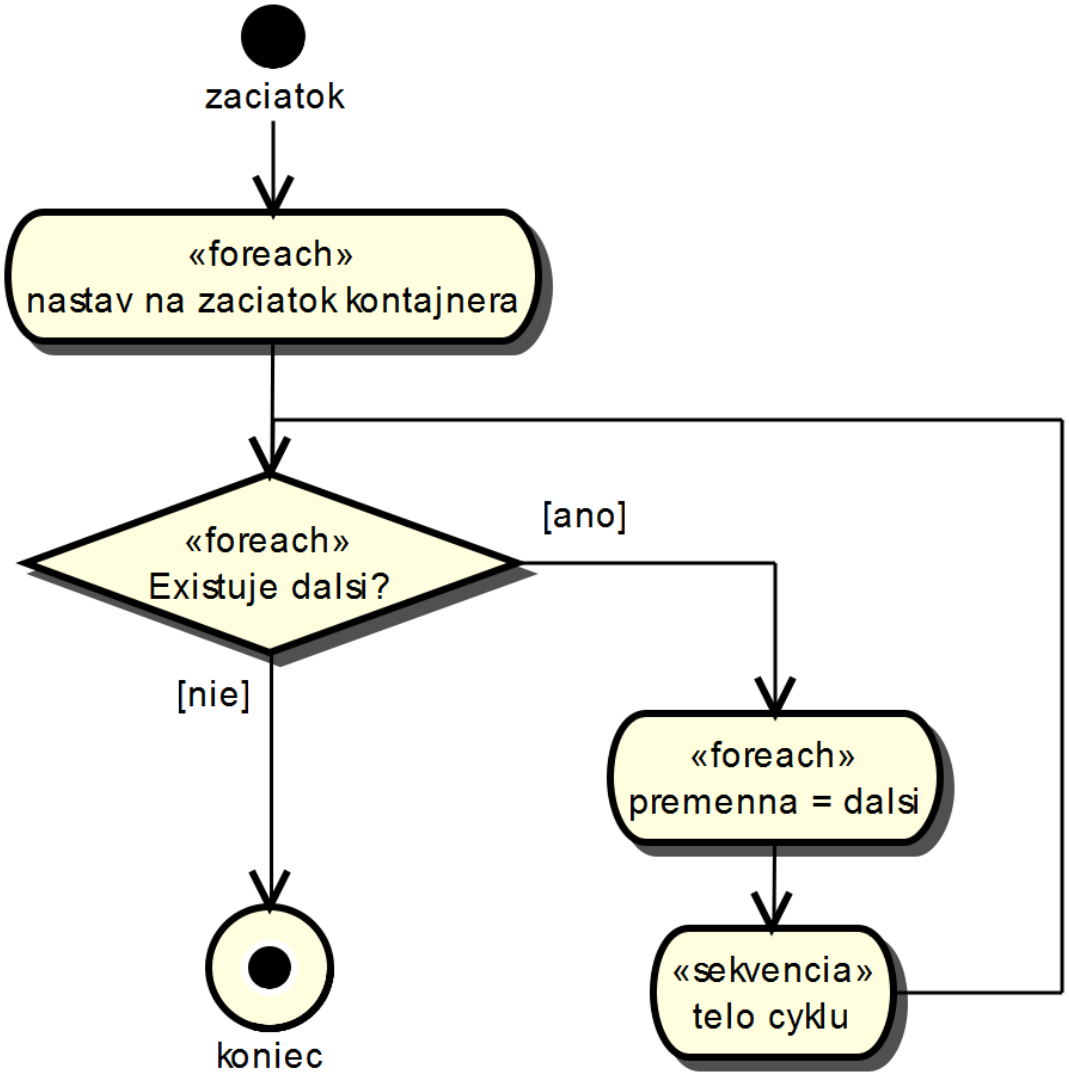
Cykly v UML

- Algoritmy – Diagramy aktivít
- Cykly
  - Žiadny špeciálny zápis
  - Modelovanie pomocou podmienky
  - Typ cyklu – stereotyp

Cyklus foreach

```
for (TypPrvku premenna : kontajner) {
    // telo cyklu
}
```

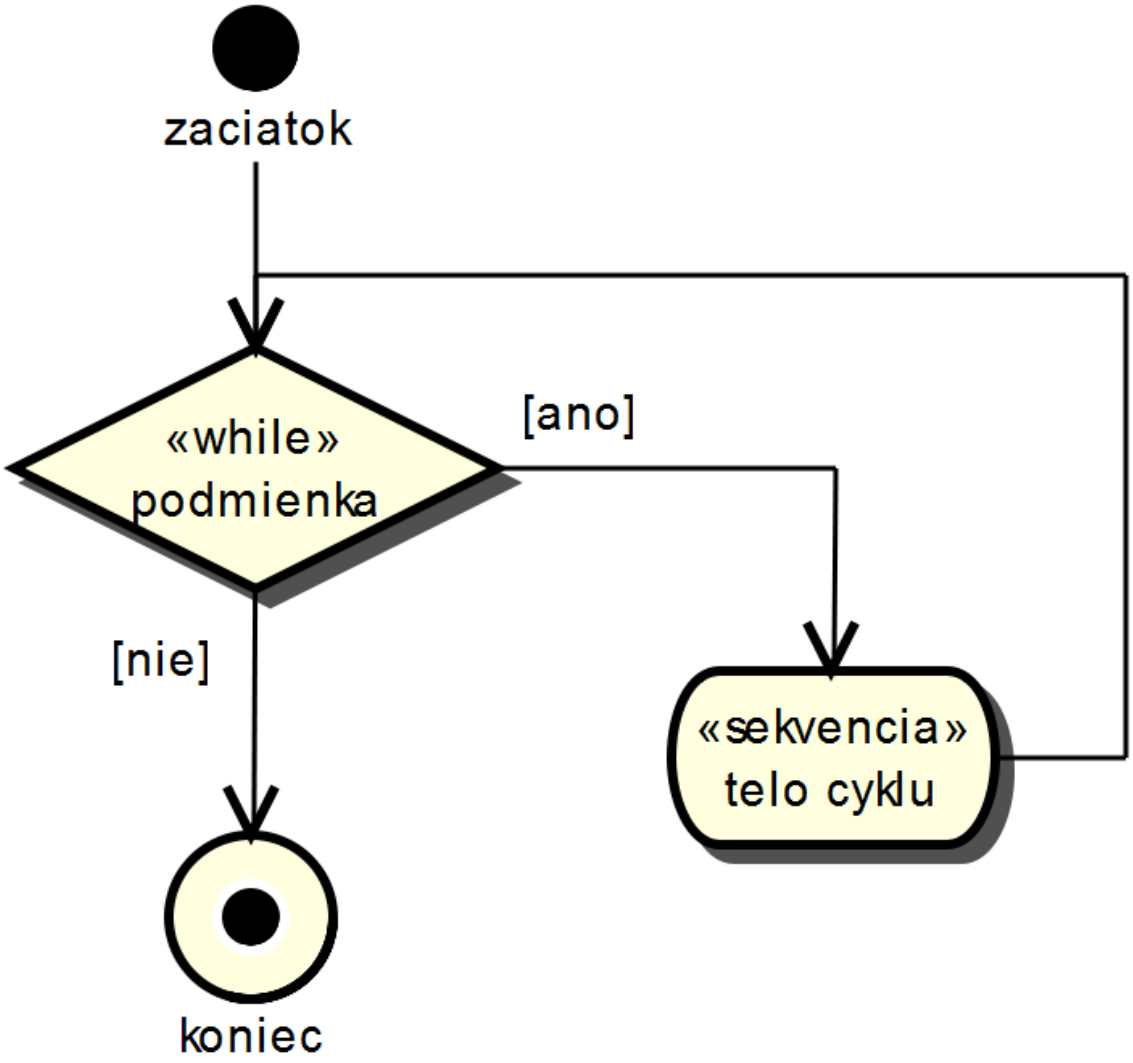
Cyklus foreach v UML



Cyklus while

```
while (podmienka) {
    // telo cyklu
}
```

Cyklus while v UML

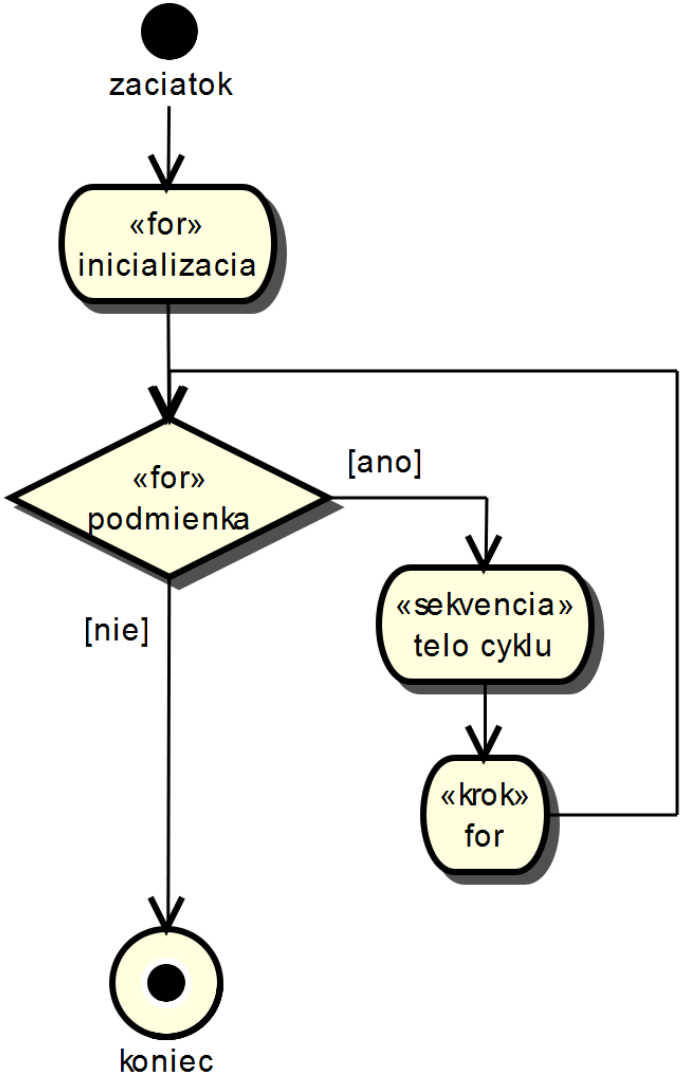


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Cyklus for

```
for (inicializácia; podmienka; krok) {
    // telo cyklu
}
```

Cyklus for v UML



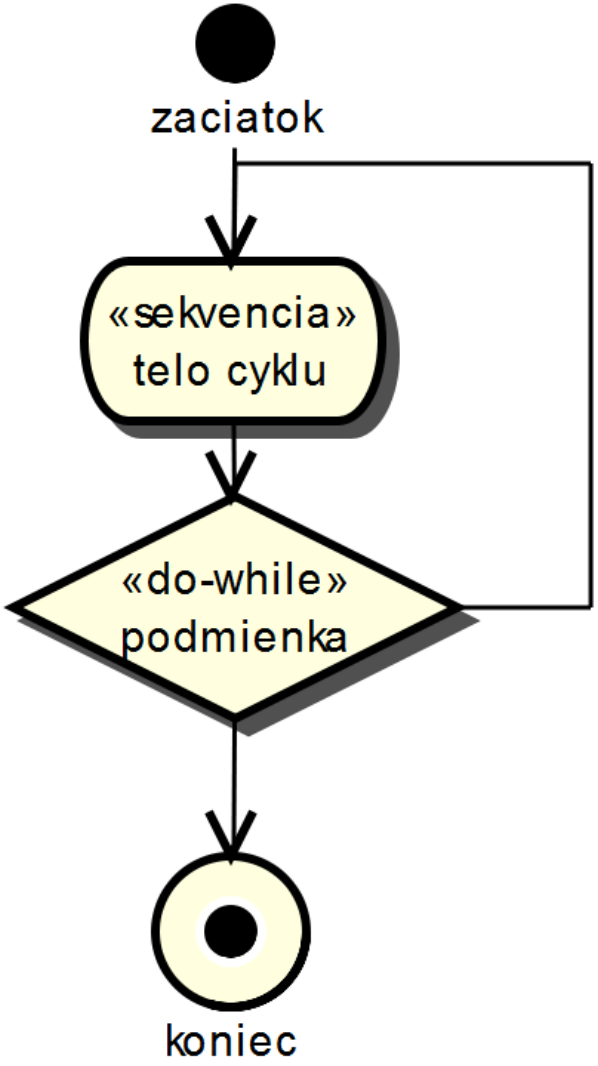
Cyklus do-while

- Ďalší z cyklov
- Pravidlo:
  - vykonávaj kým platí podmienka
  - telo sa vykoná aspoň jeden krát

```
do {
   // telo cyklu
} while (podmienka)
```


KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Cyklus do-while v UML



KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Vďaka za pozornosť