Informatika 1

4

Skladanie objektov

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

Pojmy zavedené v 3. prednáške(1)

- algoritmus
  - vlastnosti
  - procesor

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

Pojmy zavedené v 3. prednáške(2)

- štruktúrované programovanie
  - základné konštrukčné prvky
    - postupnosť – sekvencia
    - vetvenie – alternatíva
  - grafické znázornenie v UML
- vetvenie v jazyku Java – príkaz if
  - blok

Pojmy zavedené v 3. prednáške(3)

- Výrazy
  - aritmetický
    - aritmetické operátory – unárne, binárne
  - logický
    - relačné operátory
  - priorita operátorov
  - zátvorky vo výrazoch
  - typová kompatibilita

Pojmy zavedené v 3. prednáške(4)

- identifikátory
  - pravidlá jazyka Java
  - konvencie UML
  - konvencie použité v príkladoch

Cieľ prednášky

- skladanie objektov
- komunikácia medzi objektmi pomocou správ
- objektové typy
- trieda String

- príklad: digitálne hodiny

Modularizácia a abstrakcia

- jednoduchá úloha – jeden objekt
- zložitá úloha – rozklad na menšie podúlohy, časti – viac ako jeden objekt
  - rozdeľuj a panuj
  - divide and conquer
  - divide et impera
- modularizácia – rozklad na menšie časti – moduly
- abstrakcia – zanedbanie vnútorných detailov jednotlivých častí

Príklad – auto

- časti auta – motor, prevodovka, kolesá,...
  - motor – zdroj sily pohybu auta
  - prevodovka – zmena veľkosti a smeru sily prenášanej na kolesá
  - kolesá – pohyb auta po ceste, zatáčanie
  - ...

Kompozícia – skladanie objektov(1)

- každá časť – špecifická úloha
- auto – spolupráca častí
- vonkajší pohľad - auto ako celok
- okolie auta - len auto ako celok

Kompozícia – skladanie objektov(2)

- auto – objekt
- motor, prevodovka, kolesá – objekty

- auto – objekt – celok
- motor, prevodovka, kolesa – objekty – časti

Kompozícia – skladanie objektov(3)

- vytváranie zložených objektov – skladanie
- skladanie objektov – kompozícia

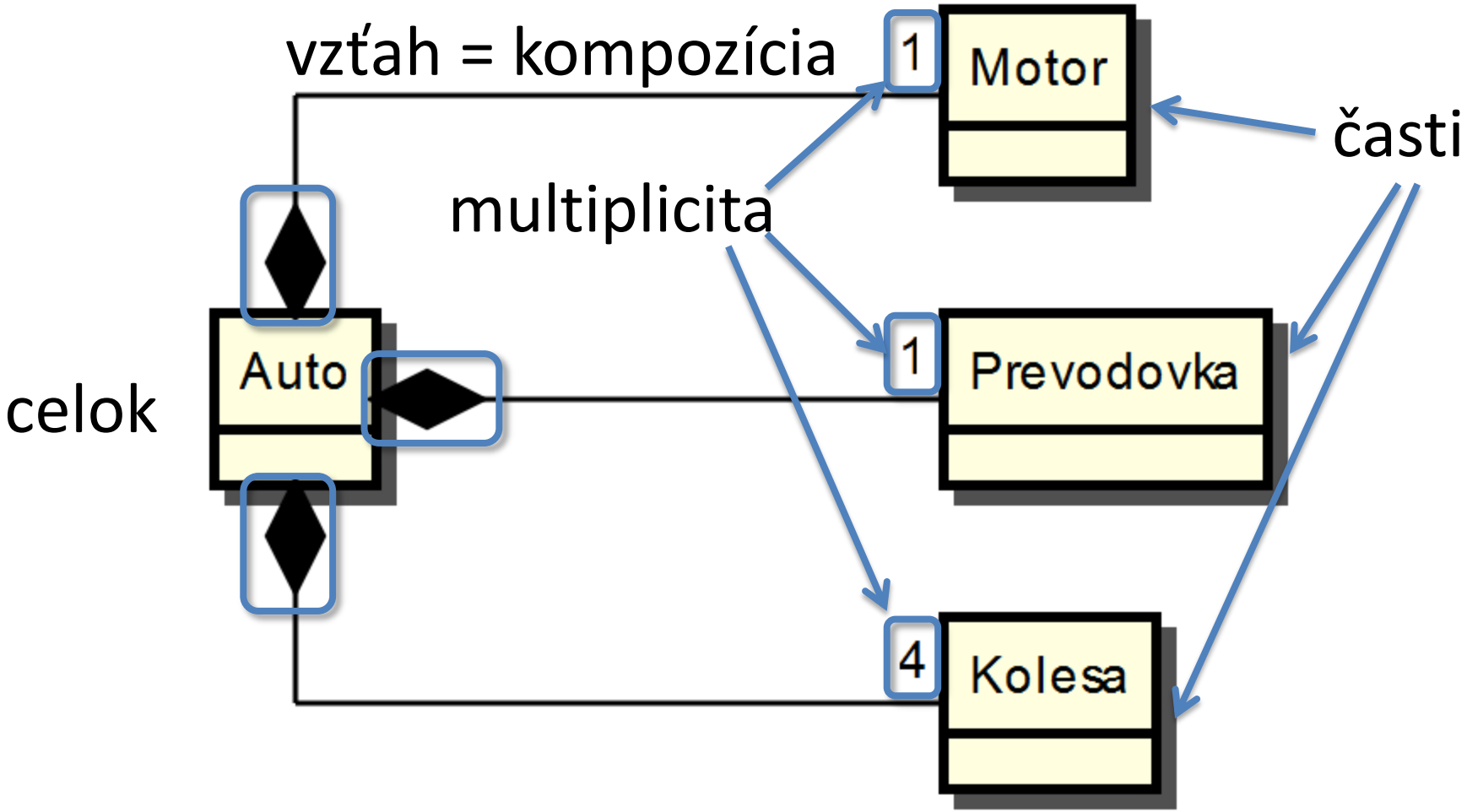
- kompozícia – závislosť celku a jeho častí
  - celok – nadriadený objekt
  - časť – podriadený objekt

- úloha celku – organizovanie spolupráce častí

Kompozícia – skladanie objektov(4)

- charakteristika kompozície
- spoločný životný cyklus
  - spoločný vznik – celok (+ časti)
    - vznik časti – súčasť vzniku celku
  - služby – len celok
    - vonkajší pohľad – rozhranie auta
  - spoločný zánik – celok (+ časti)
- zodpovednosť celku za časti

Kompozícia v UML



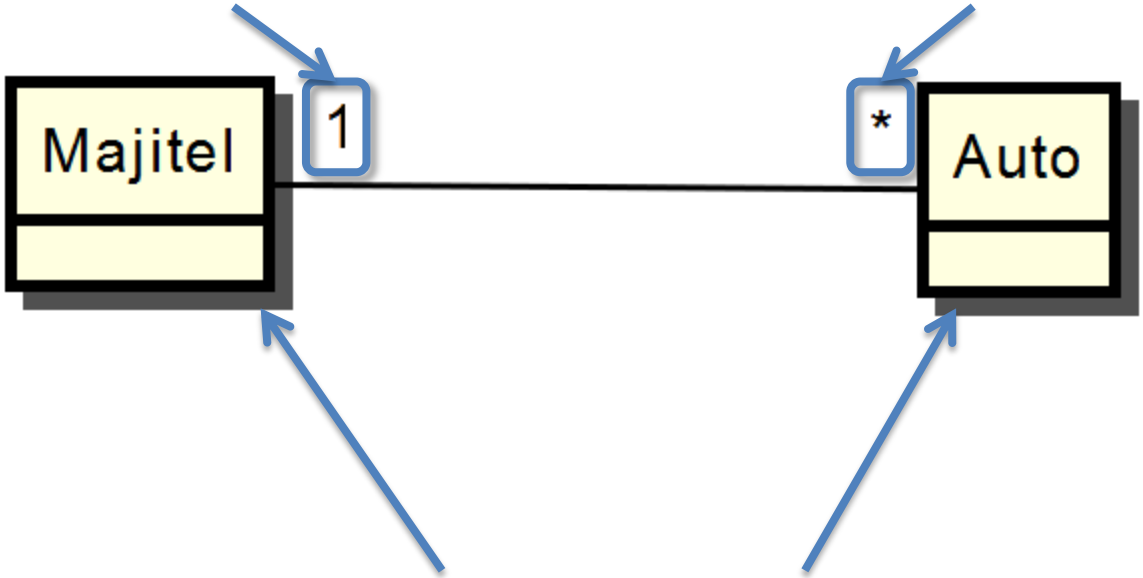
Asociácia – spolupráca objektov

- asociácia – ľubovoľná spolupráca dvoch objektov
  - príklady: klient a banka, učiteľ a študent

- charakteristika asociácie – nezávislé životné cykly oboch objektov

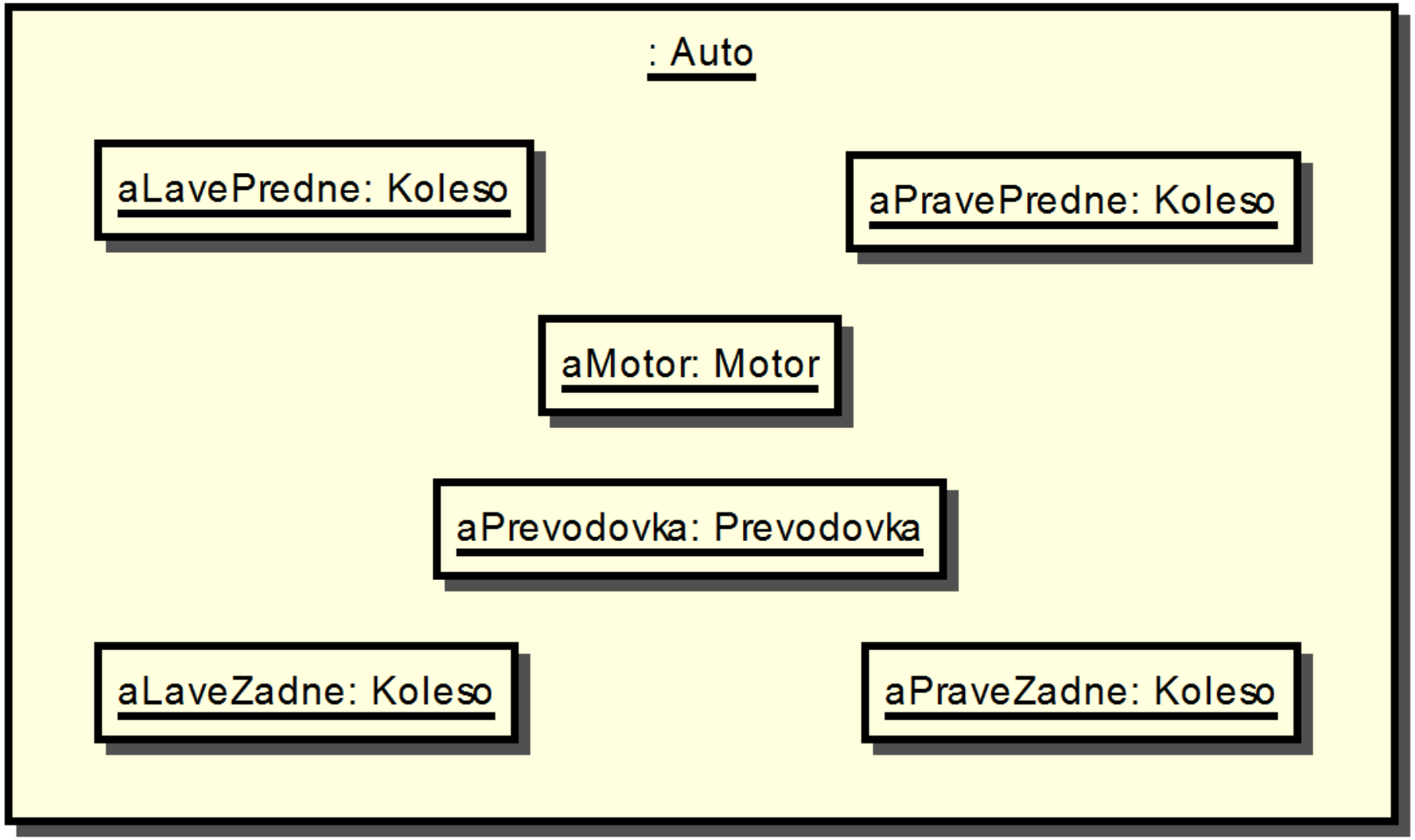
Asociácia v UML

jeden majiteľ môže mať viac aut



rovnocenné objekty

Diagram objektov – UML

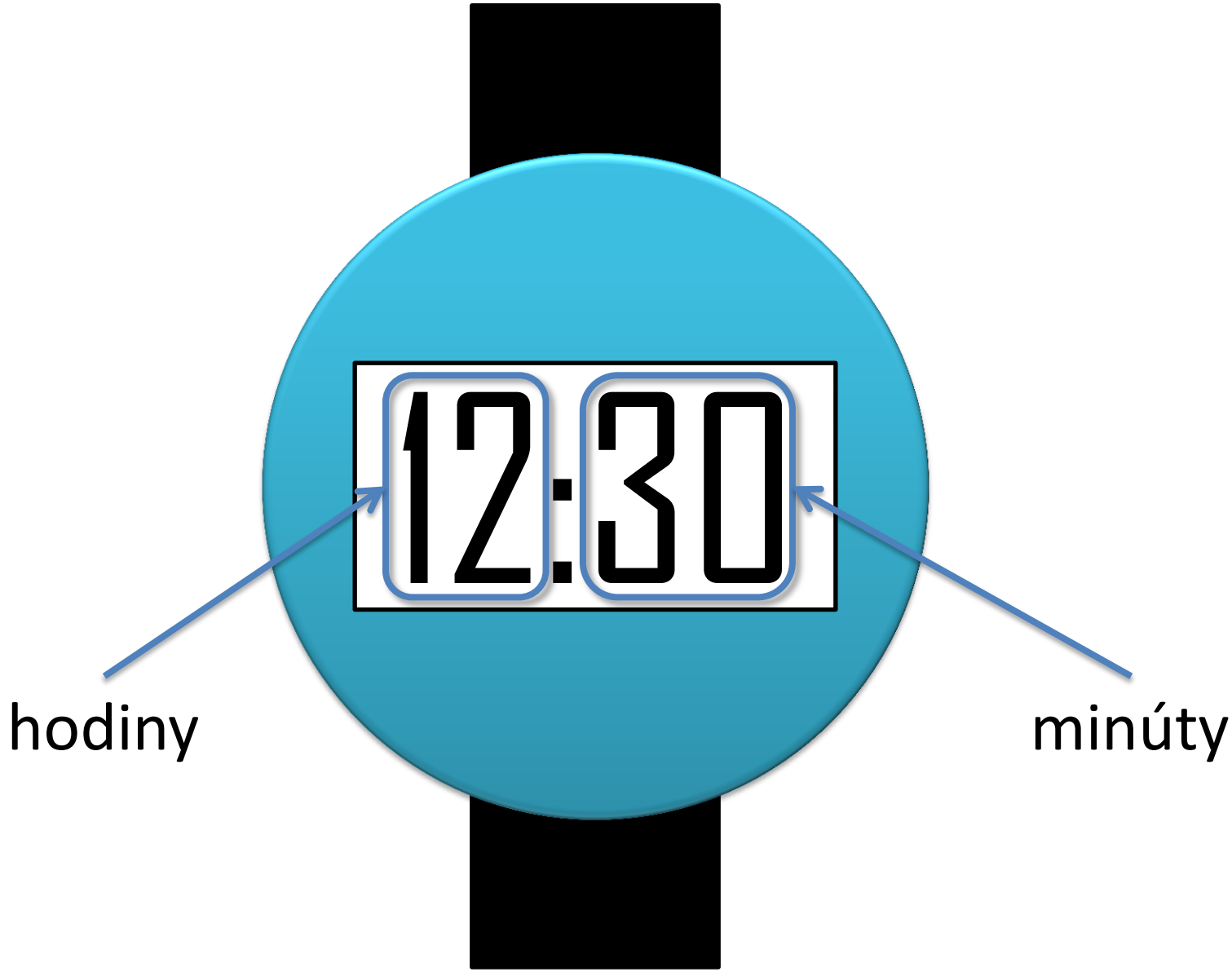


Projekt digitálne hodiny

- Požadované služby:
  - zobrazujú aktuálny čas 24-hod. formát
    - 00:00 – polnoc
    - 23:59 – minúta pred polnocou
- dajú sa nastaviť na požadovaný čas
- „tikajú“ – plynutie času – časový krok
  - krok 1 minúta

- trieda vytvorí inštanciu hodín
  - začiatočný čas: 00:00

Príklad: digitálne hodiny



Návrhy riešenia

- jediný objekt – podobne ako automat MHD

- kompozícia
  - digitálne hodiny – celok
    - v rozhraní bude mať požadované služby
  - minúty – časť pre prácu s minútami
  - hodiny – časť pre prácu s hodinami

Charakteristika minút

- „plynú“ – posunú sa o 1 minútu
  - najnižšia hodnota 00
  - najvyššia hodnota 59
- po uplynutí celej hodiny začínajú znovu od 00
- vždy dvojciferné číslo – vedúca nula
- dajú sa nastaviť na požadovanú hodnotu z <00, 59>

Charakteristika hodín

- „plynú“ – posunú sa o 1 hodinu
  - najnižšia hodnota 00
  - najvyššia hodnota 23
- po uplynutí celého dňa začínajú znovu od 00
- vždy dvojciferné číslo – vedúca nula
- dajú sa nastaviť na požadovanú hodnotu z <00, 23>

Minúty/hodiny – rovnaké vlastnosti

- plynú
- najnižšia hodnota 00
- po dosiahnutí maxima pokračujú od 00
- formátovanie v tvare dvojciferného čísla
- dajú sa nastaviť na požadovanú hodnotu

Minúty/hodiny – rozdiely

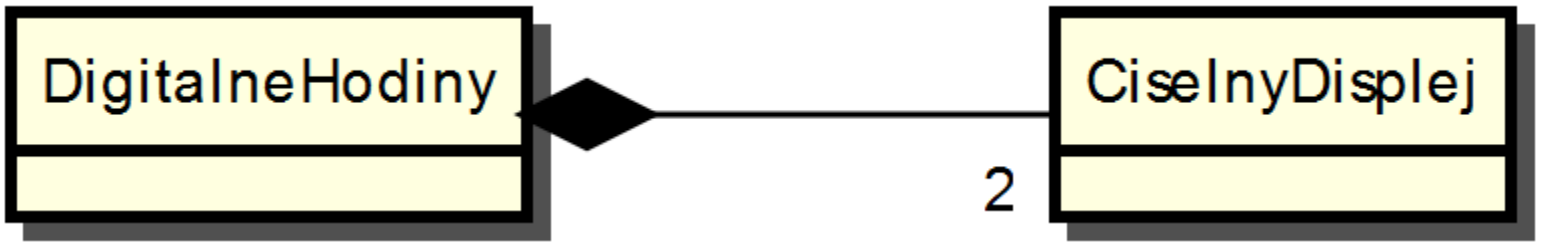
- krok pre hodiny: 1 hodina
- krok pre minúty: 1 minúta
- maximum pre hodiny: 23
- maximum pre minúty: 59

Riešenie rozdielov

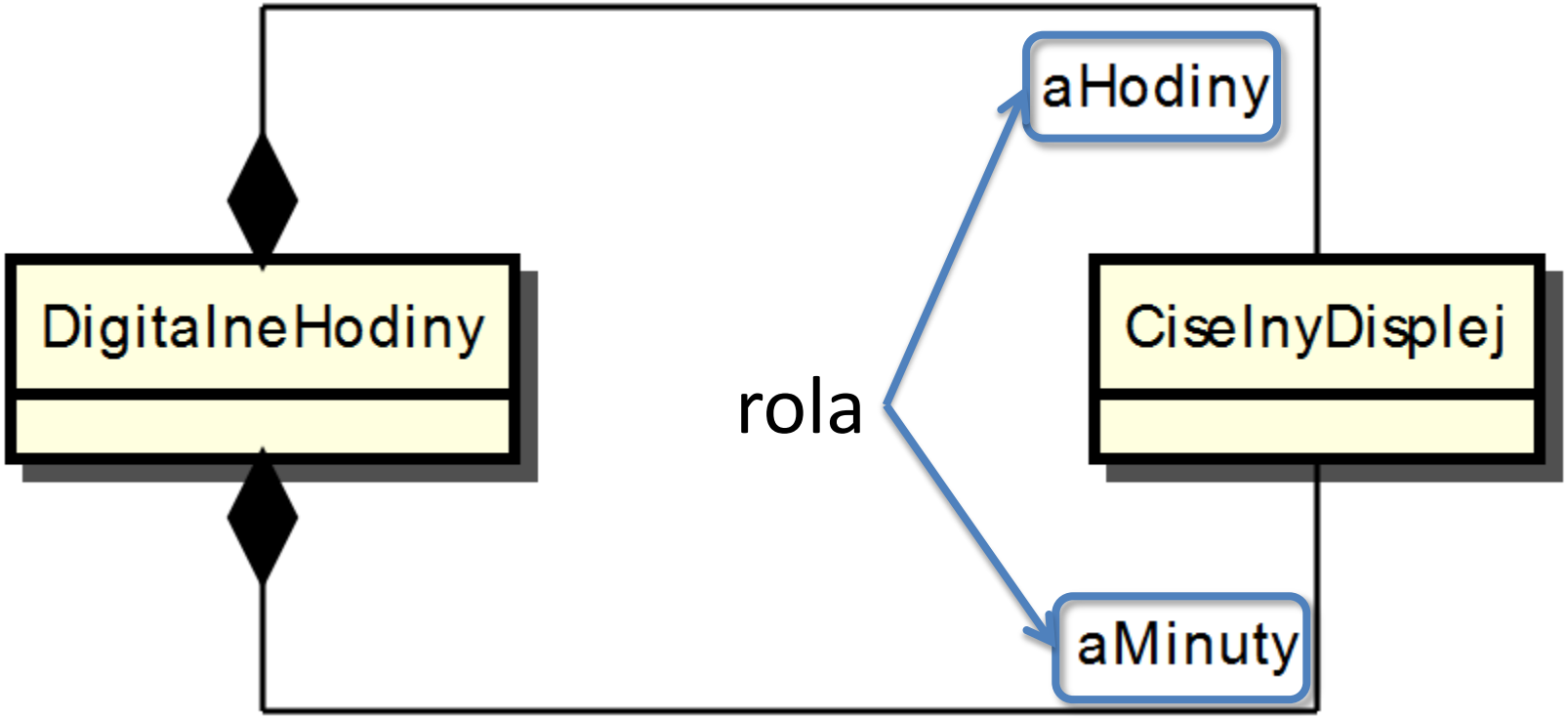
- dve triedy

- spoločná trieda, parametre
  - rôznosť krokov
    - oba kroky o 1
    - rôzne jednotky – úroveň interpretácie
  - rôznosť maxima
    - nastaviteľné maximum

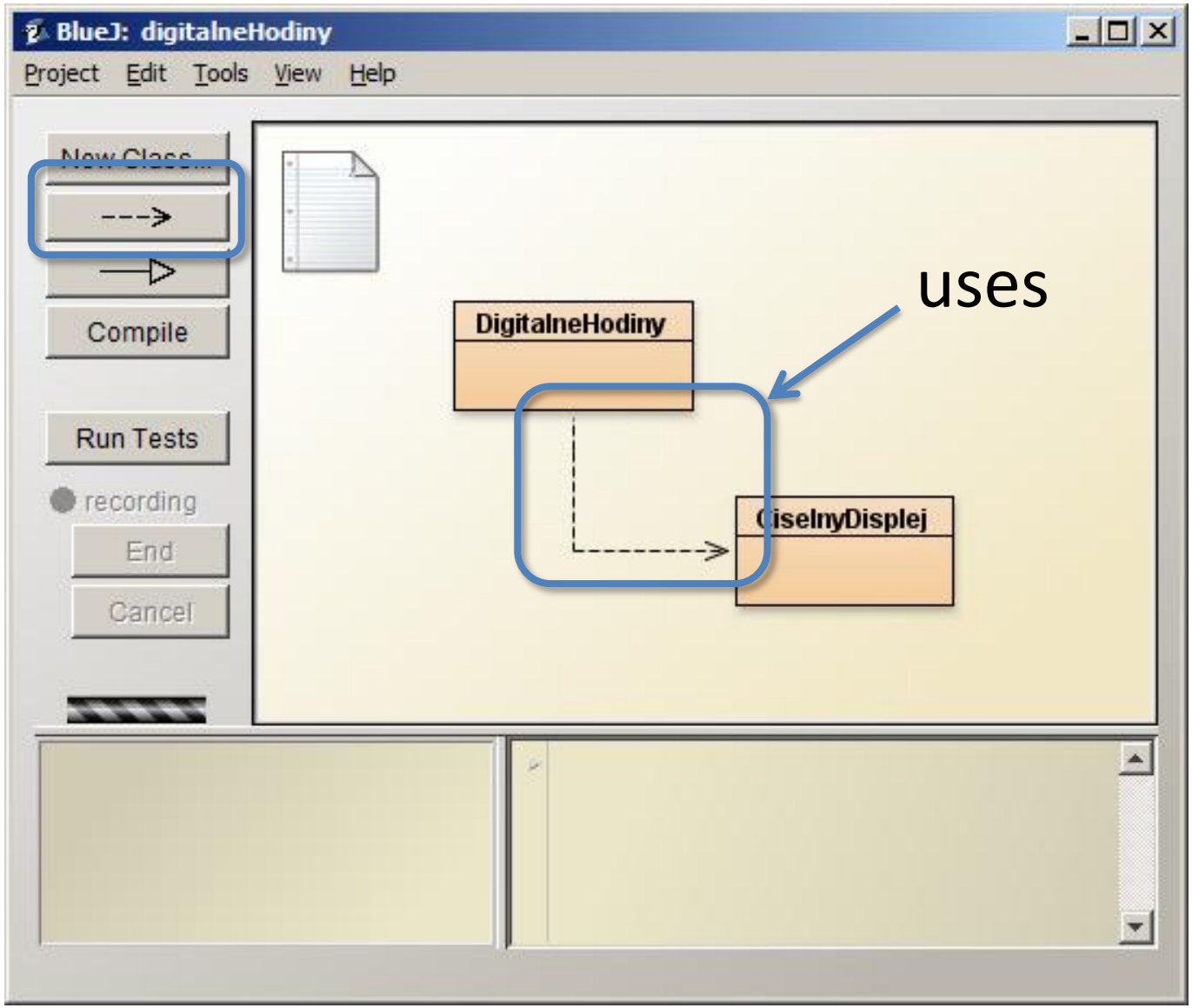
Digitálne hodiny v UML(1)



Digitálne hodiny v UML(2)



Digitálne hodiny v Bluej



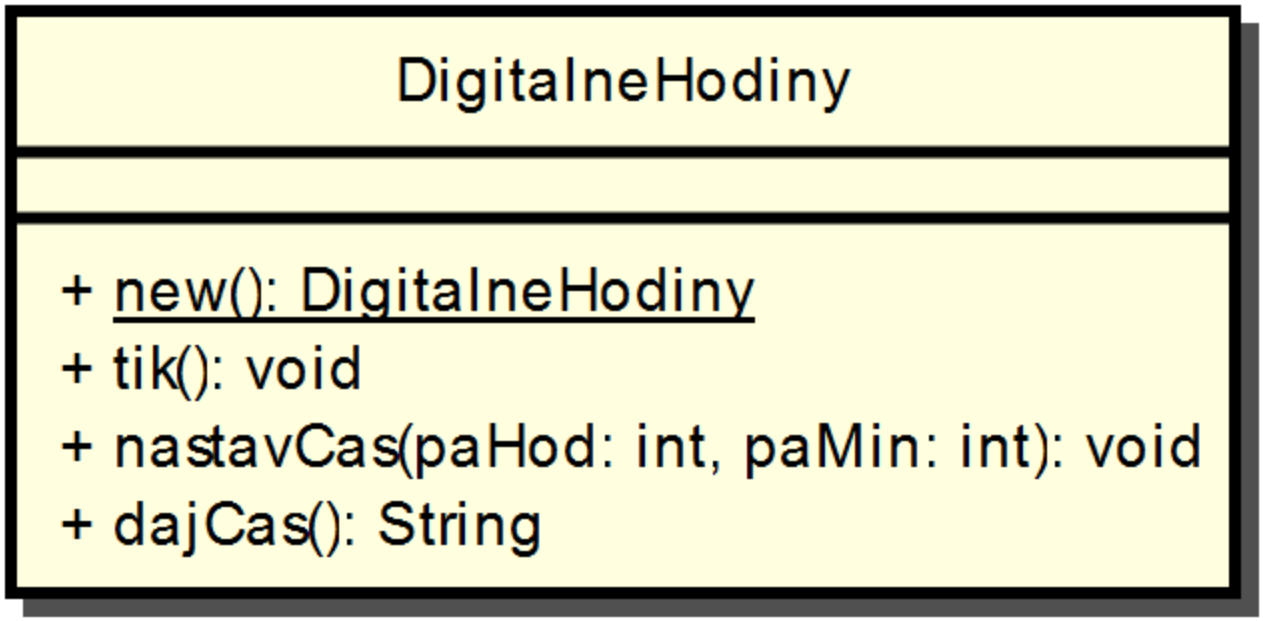
Charakteristiky číselného displeja

- počíta kroky
- aktuálny stav vráti vo formáte dvojciferného čísla
- po dosiahnutí nastaveného maxima začína od 00
- nastaviteľný na požadovanú hodnotu z <00, max.>

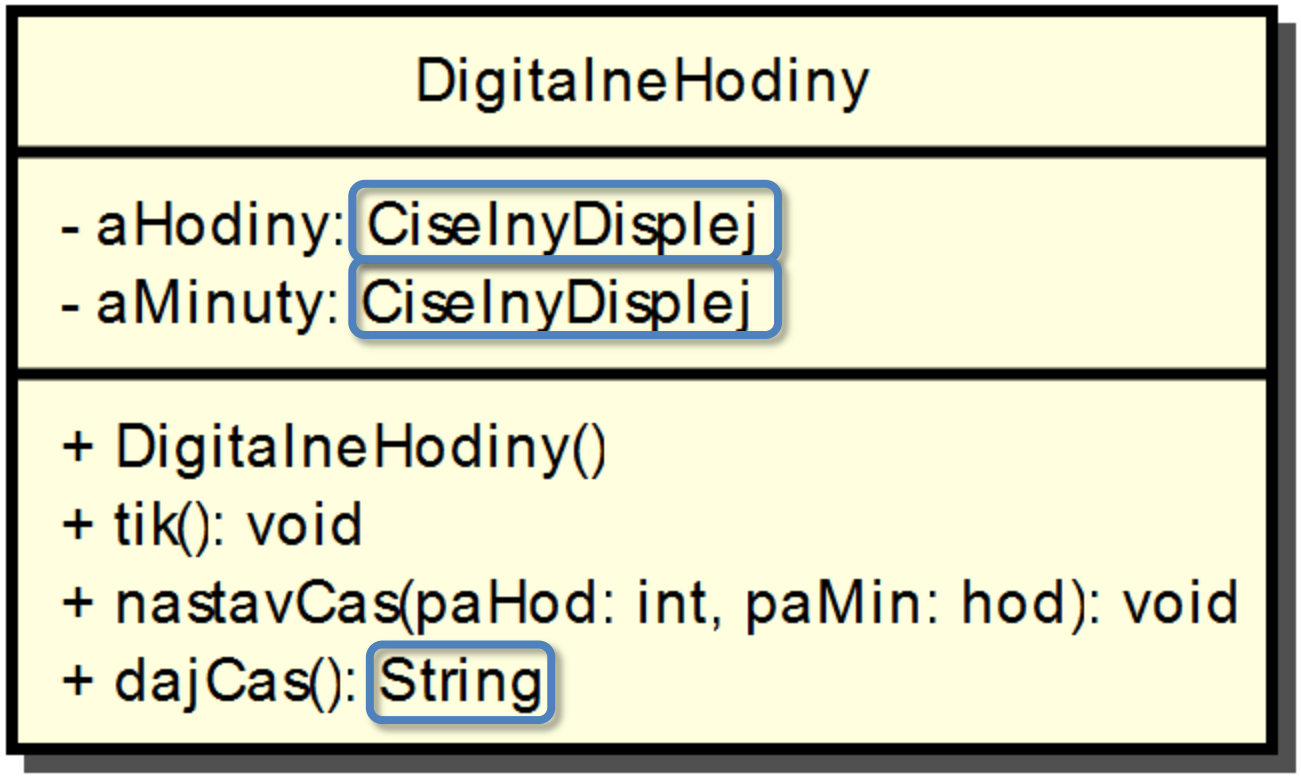
Úlohy digitálnych hodín – celku

- vytvorenie displejov – častí
- vzťah hodín a minút – rôznosť jednotky
  - po uplynutí 60 minút krok pre hodiny
- nastavenie maxima

DigitalneHodiny – rozhranie



DigitalneHodiny – vnútorný pohľad



Objektové typy

- typ = názov triedy
  - trieda ako typ

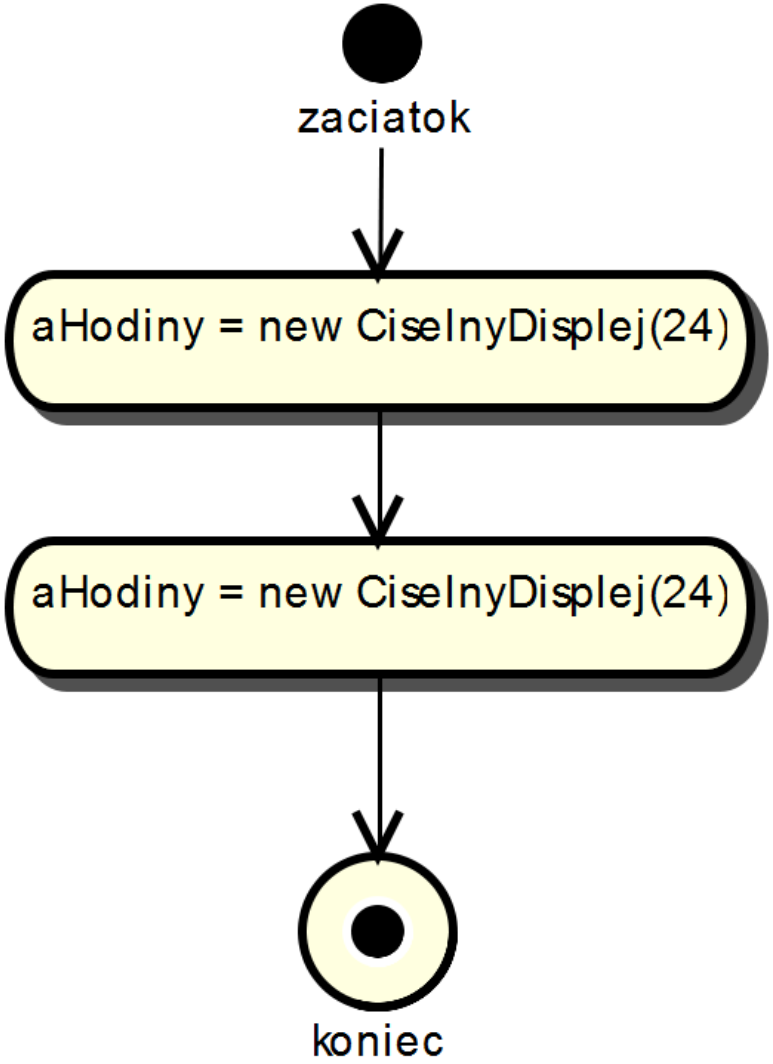
- príklady:
  - typ premennej
    - aHodiny : CiselnyDisplej
  - typ návratovej hodnoty
    - dajCas() : String

DigitalneHodiny – Java

```
public class DigitalneHodiny
{
    private CiselnyDisplej aHodiny;
    private CiselnyDisplej aMinuty;

    ...
}
```

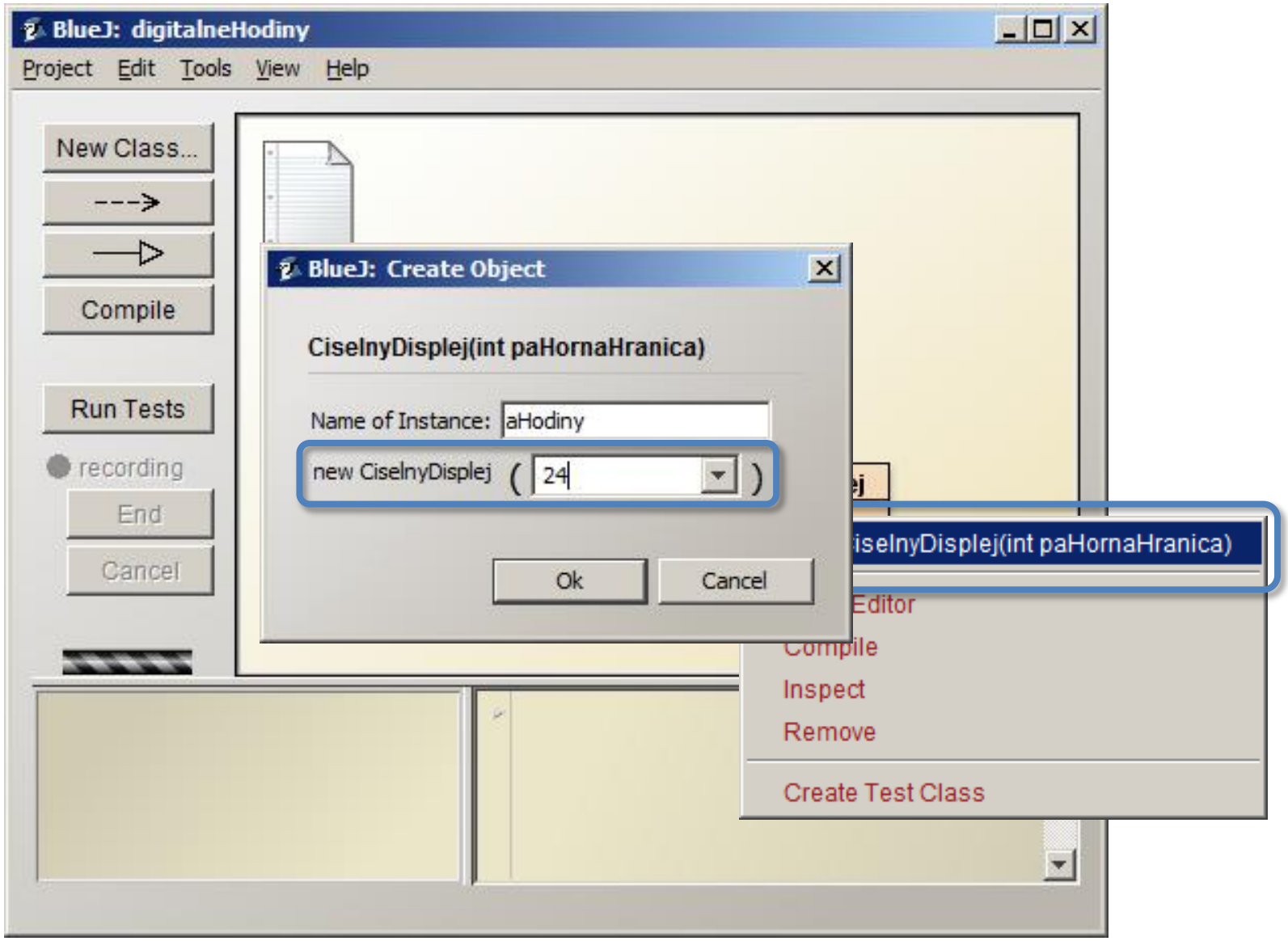
DigitalneHodiny – konštruktor – UML



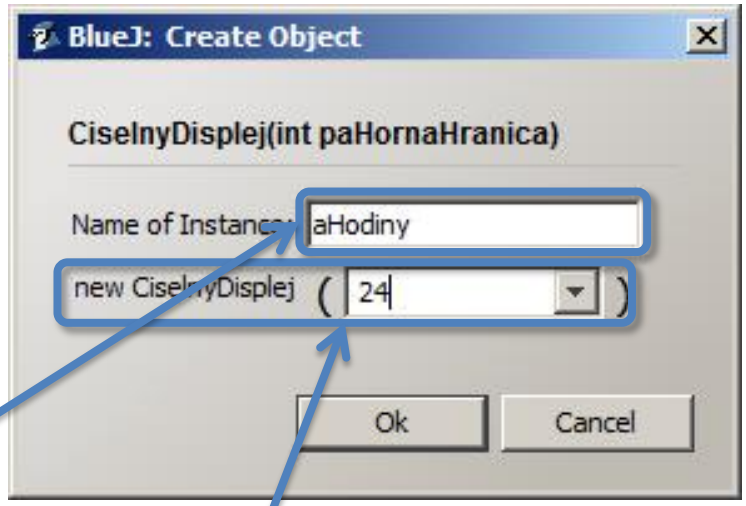
DigitalneHodiny – konštruktor – Java

```
public DigitalneHodiny()
{
    aHodiny = new CiselnyDisplej(24);
    aMinuty = new CiselnyDisplej(60);
}
```

Správa triede „new“

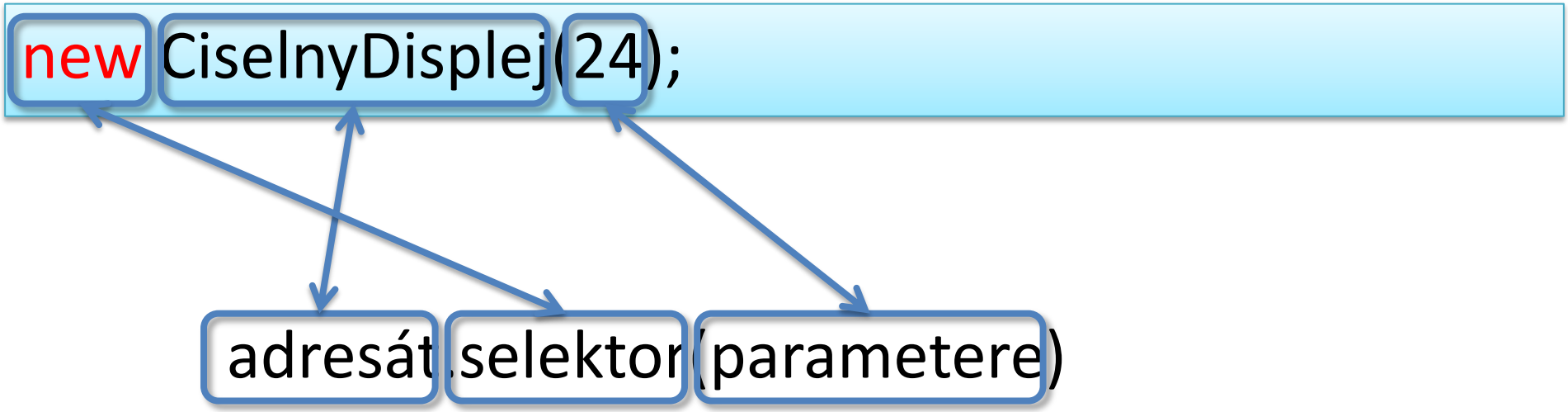


Správa triede „new“




```
aHodiny = new CiselnyDisplej(24);
```

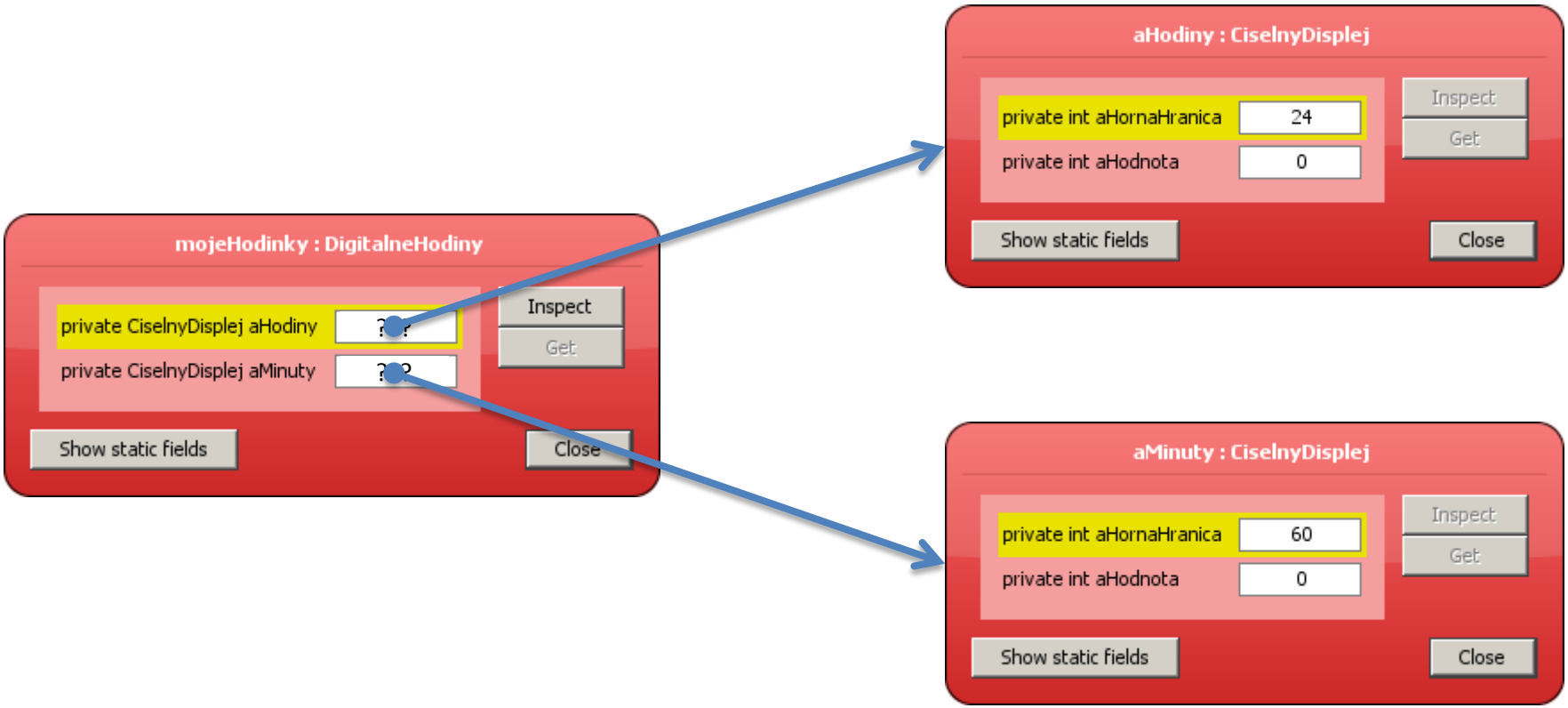
Štruktúra správy „new“



new = operátor v jazyku Java

DigitalneHodiny – konštruktor

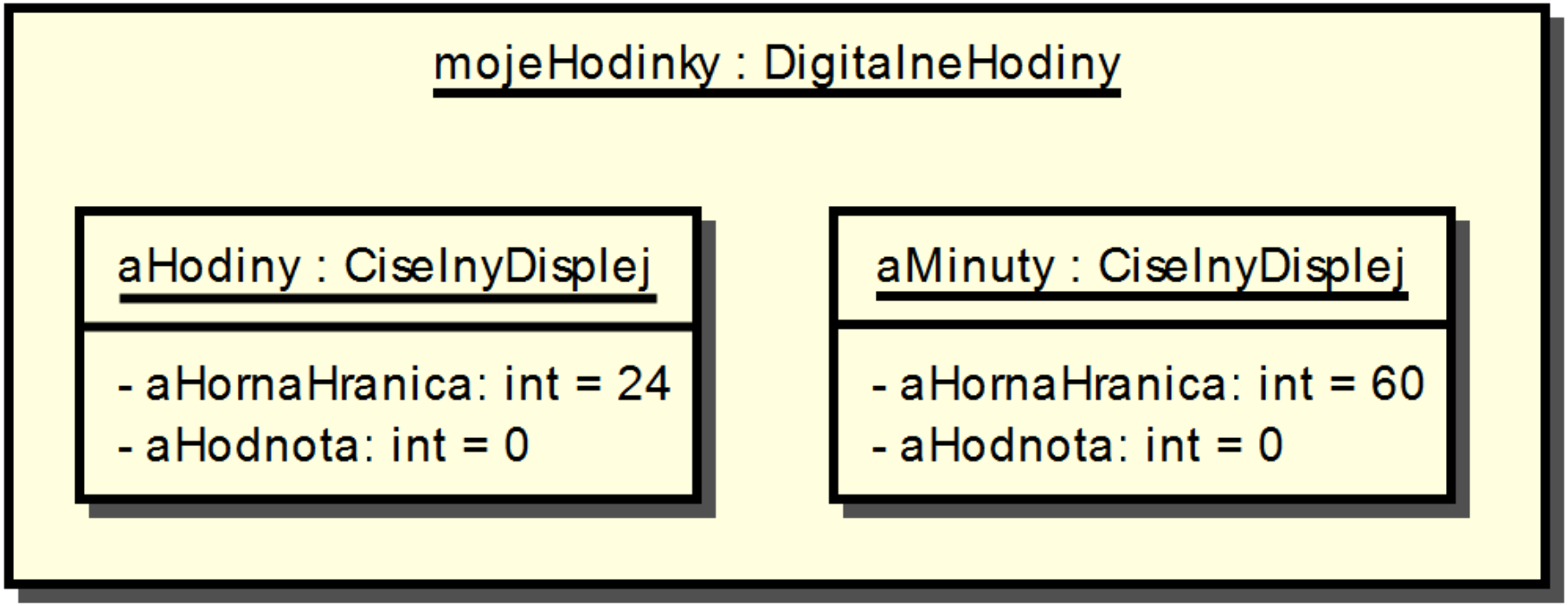
```
aHodiny = new CiselnyDisplej(24);
aMinuty = new CiselnyDisplej(60);
```



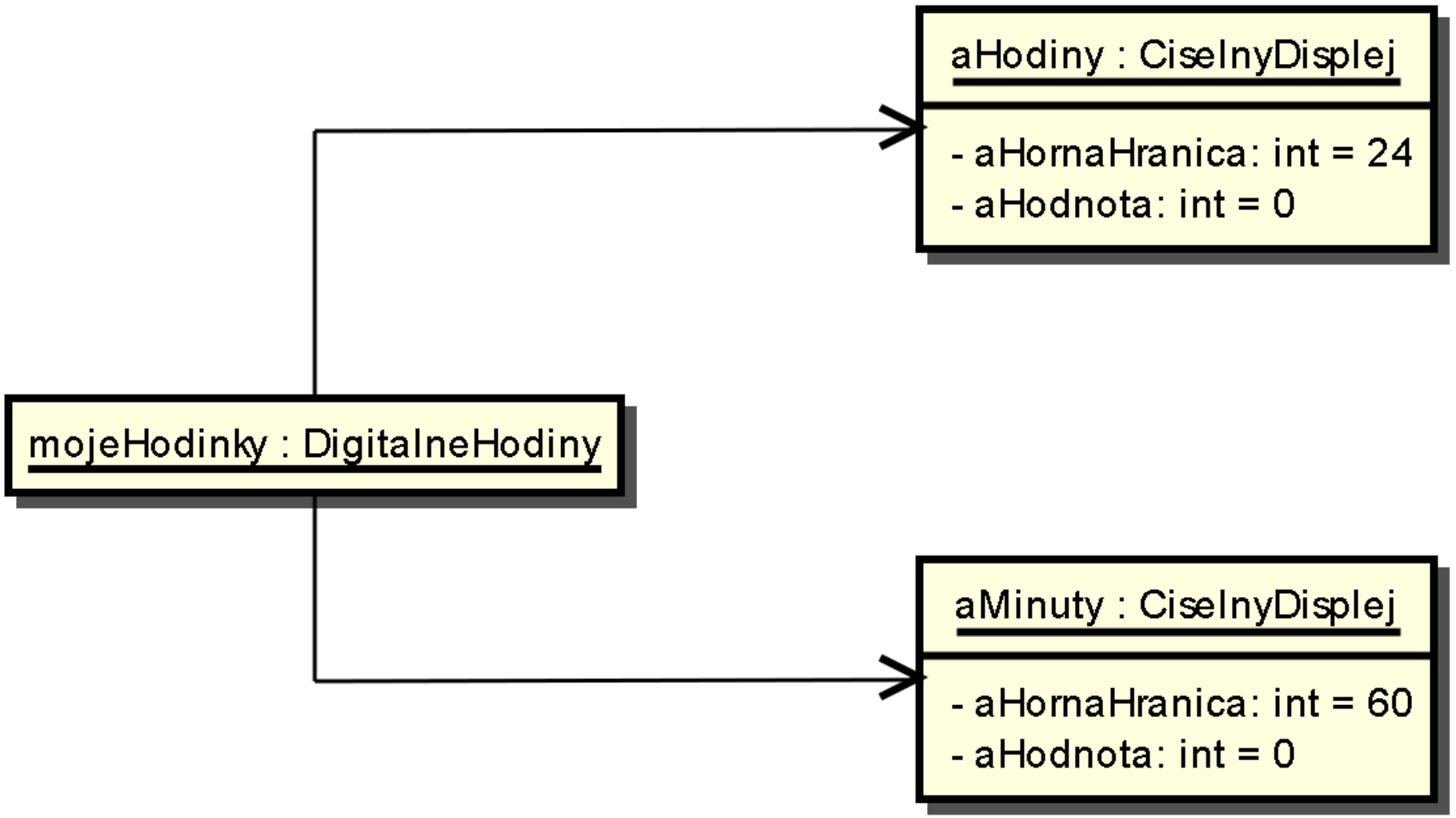
Referencie

- premenná uchováva hodnotu
- hodnota objektového typu – referencia
- rozdiel v používaní primitívnych typov a objektových typov
  - primitívne typy – hodnota vo výrazoch
  - objektové typy
    - hodnota vo výrazoch
    - adresát v správe

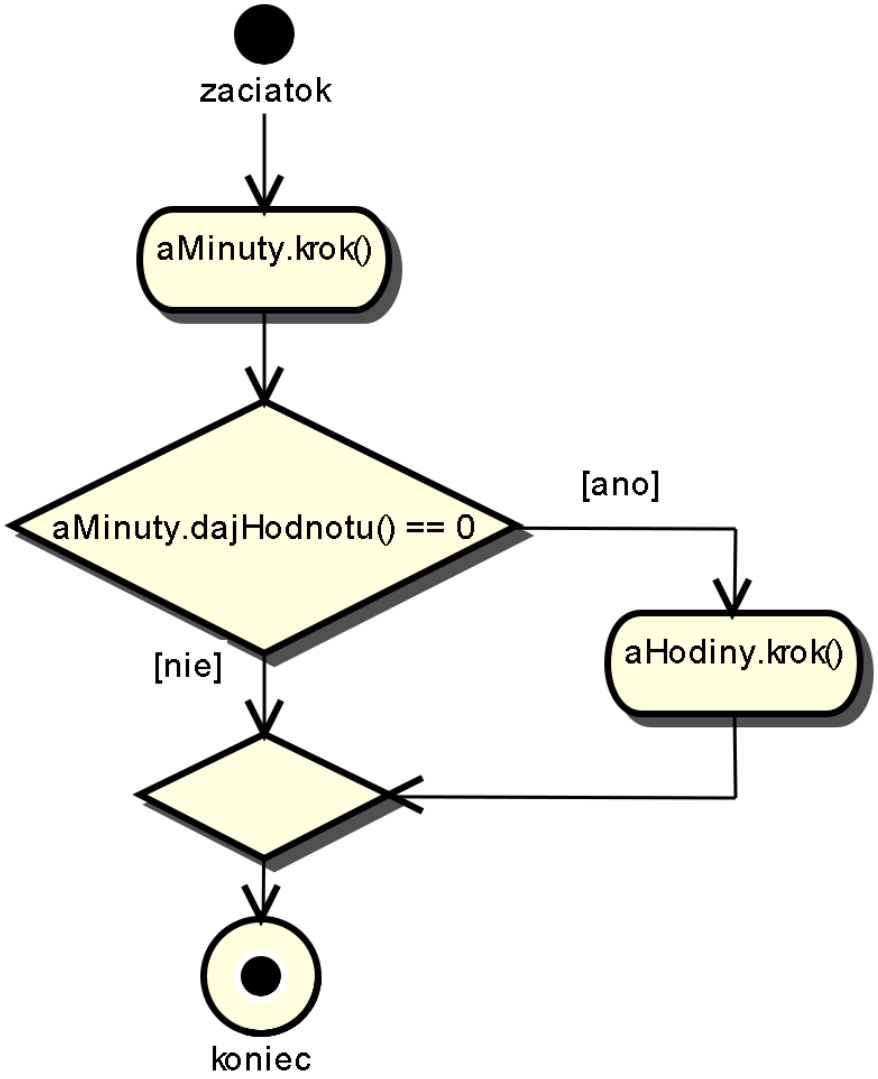
DigitalneHodiny – diagram objektov(1)



DigitalneHodiny – diagram objektov(2)



DigitalneHodiny – tik – UML



DigitalneHodiny – tik – UML

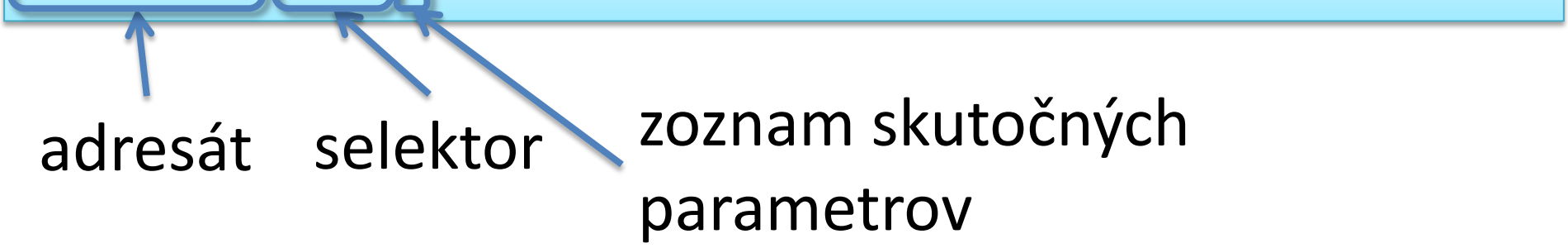
```
public void tik()
{
    aMinuty.krok();

    if (aMinuty.dajHodnotu() == 0) {
       aHodiny.krok();
    }
}
```

Príkaz na poslanie správy

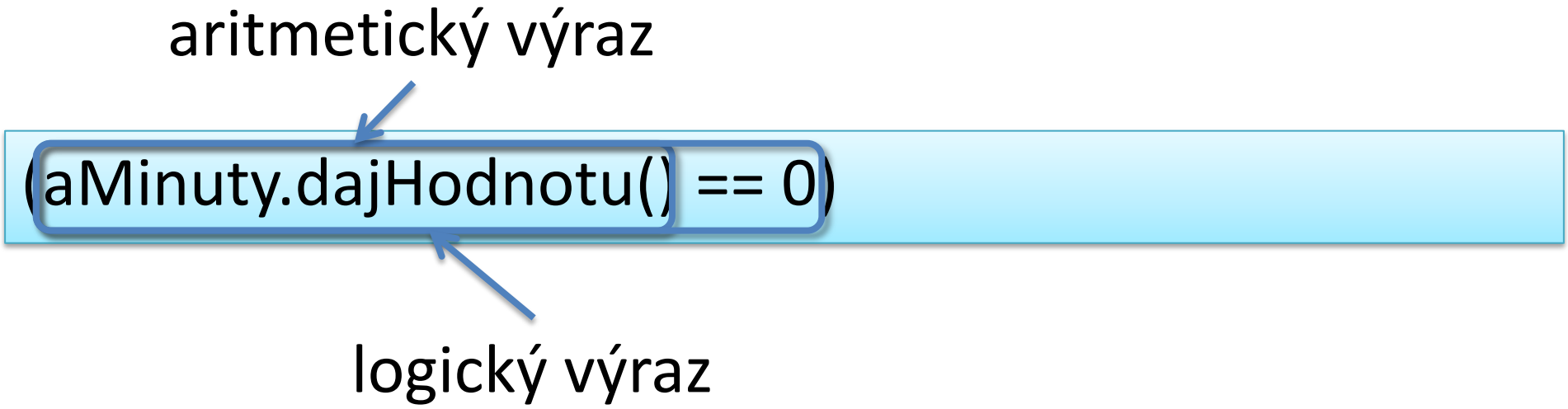
- objekt.sprava(skutocneParametre);
  - každý zo skutočných parametrov môže byť výraz
- správa nemá výstupný parameter
- vždy samostatný príkaz
- príklad:

```
aMinuty.krok();
```



správy s výstupným parametrom

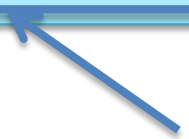
- typ výstupného parametra (návratová hodnota)
  - primitívny
  - objektový
- správa s výstupným parametrom – výraz



Objektový výraz

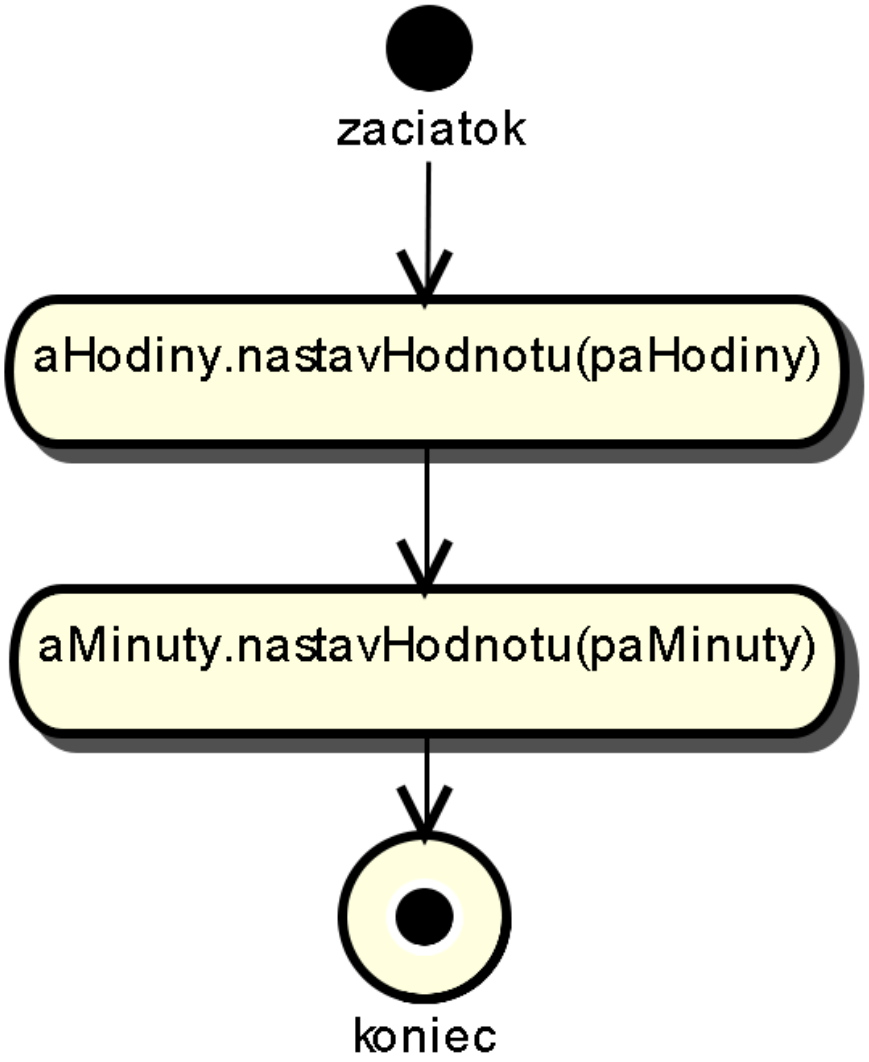
- výsledok vyhodnotenia objektového výrazu – referencia na objekt
- návratová hodnota správy „new“ – referencia na inštanciu danej triedy

```
aHodiny = new CiselnyDisplej(24);
```



objektový výraz

DigitalneHodiny – nastavCas – UML



DigitalneHodiny – nastavCas – Java

```
public void nastavCas(int paHodina, int paMinuta)
{
    aHodiny.nastavHodnotu(paHodina);
    aMinuty.nastavHodnotu(paMinuta);
}
```

Metóda dajCas

```
public String dajCas()
```

Literál typu String

- reťazcový literál

```
"ľubovoľný text v Unicode uzavretý do dvojice úvodzoviek"
```

- prázdny reťazec

```
""
```

- literál – realizovaný ako inštancia triedy String

Premenná typu String

- atribút

```
private String aPriezvisko;
```

- formálny parameter

```
public void zmenPriezvisko(String paPriezvisko)
```

- lokálna premenná

```
String paPriezvisko;
```

Spájanie reťazcov(1)

- Java používa operátor + pre spájanie reťazcov

- spojením dvoch reťazcov

```
"Žilinská"  + " univerzita"
```

- vzniká nový reťazec (nová inštancia triedy String)

```
"Žilinská univerzita"
```

Spájanie reťazcov(2)

- reťazcový výraz

```
prvyOperand + druhyOperand
```

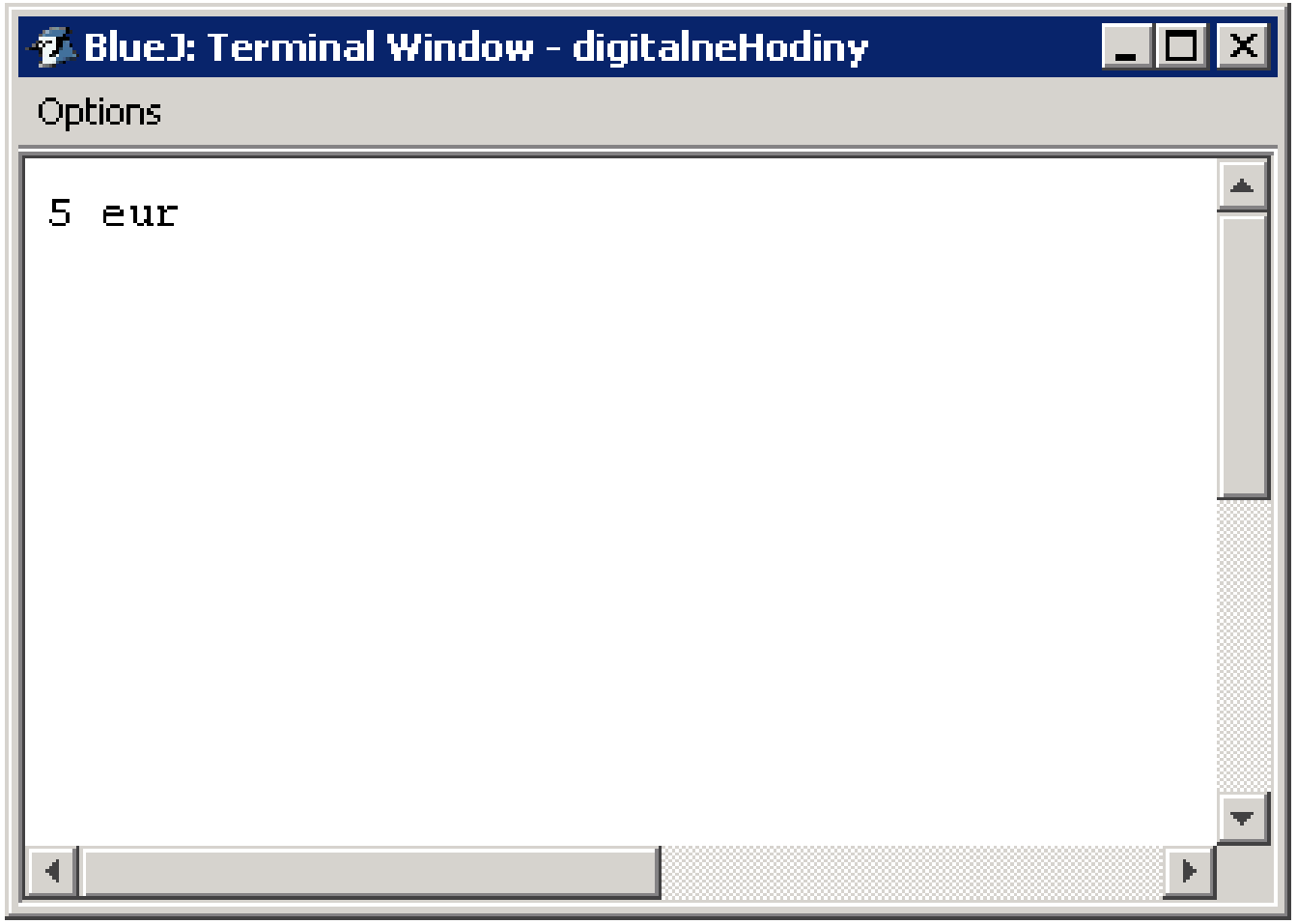
- aspoň jeden operand je reťazcový
- hodnota reťazcového výrazu – reťazec
- reťazcový operand
  - literál
  - premenná typu String
  - návratová hodnota typu String
  - reťazcový výraz

Spájanie reťazcov(3)

- iný ako reťazcový operand sa automaticky prekonvertuje na reťazec
- vyhodnocovanie výrazu – zľava doprava

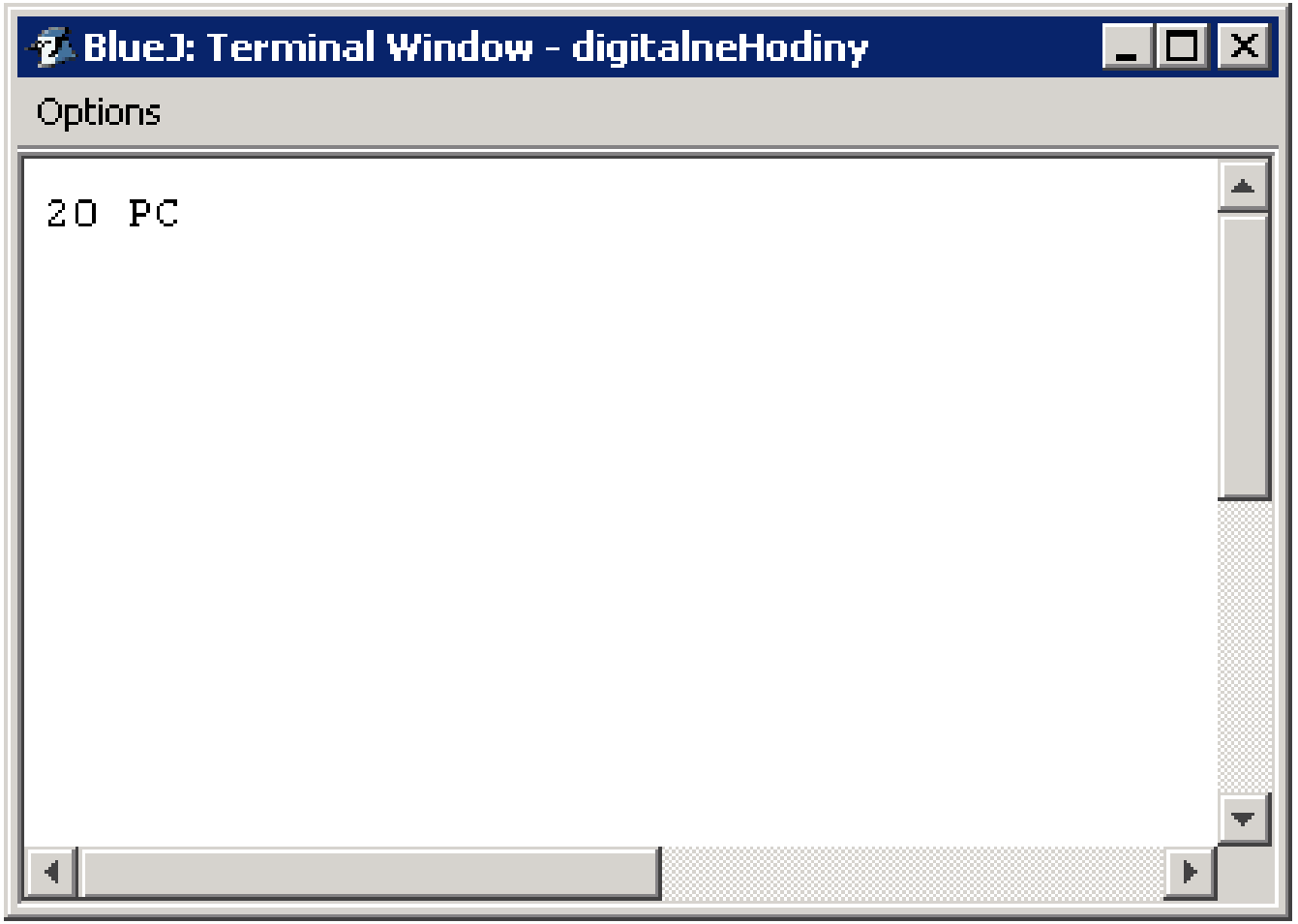
Spájanie reťazcov(1)

```
System.out.println(5 + " eur")
```



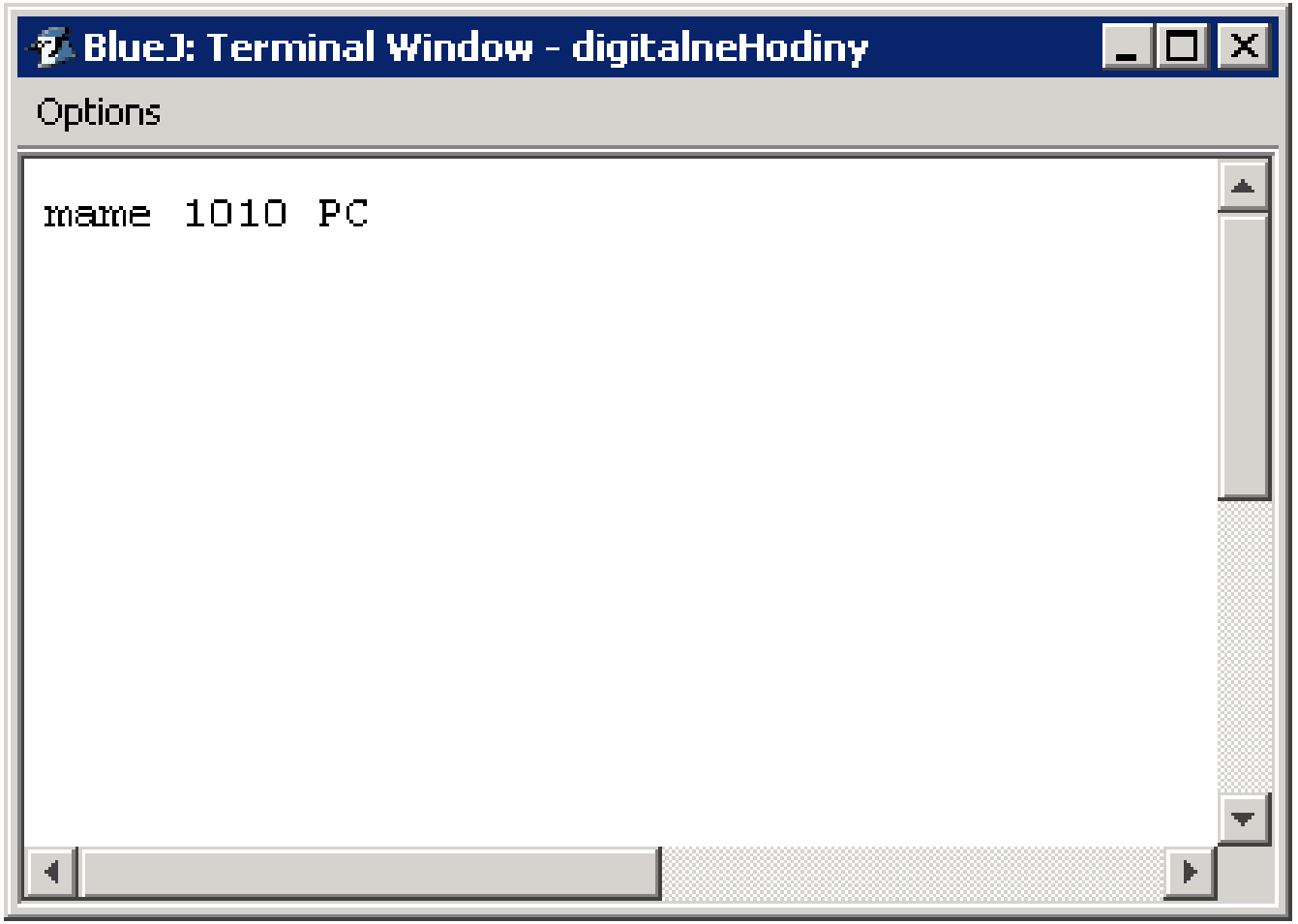
Spájanie reťazcov(2)

```
System.out.println(10 + 10 + " PC")
```



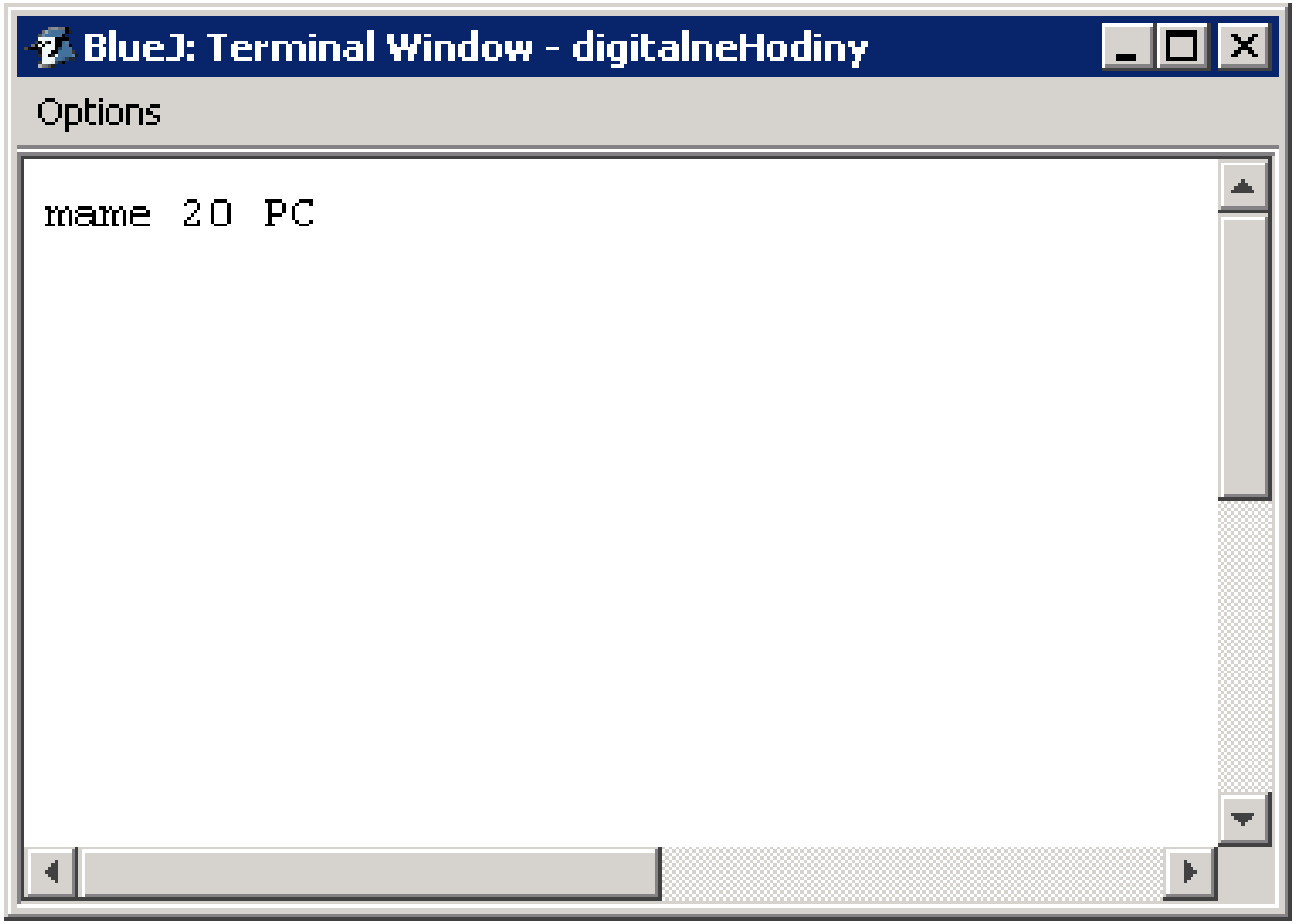
Spájanie reťazcov(3)

```
System.out.println("mame "+ 10 + 10 + " PC")
```



Spájanie reťazcov(4)

```
System.out.println("mame "+ (10 + 10) + " PC")
```



Metóda dajCas

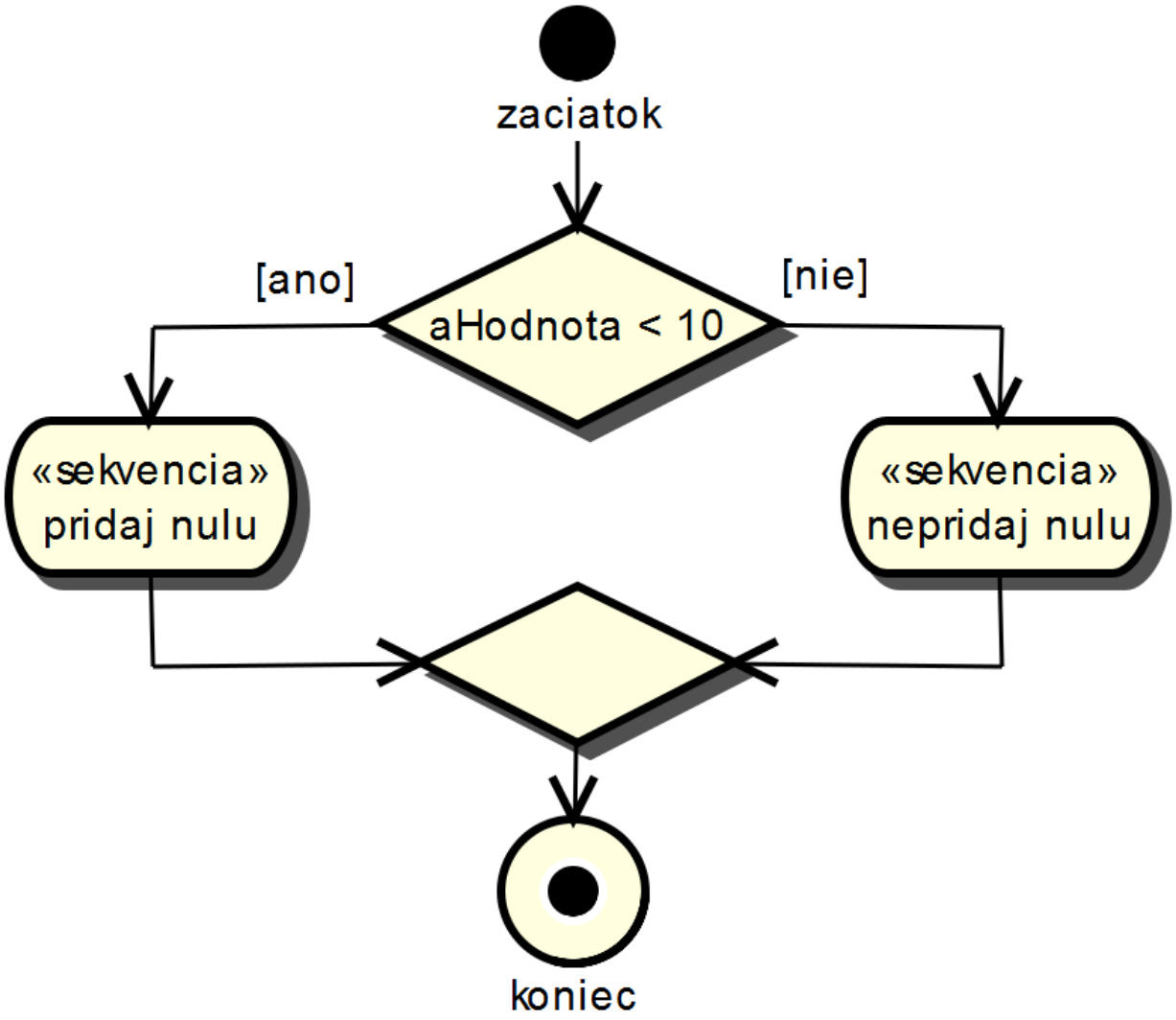
```
public String dajCas()
{
    return aHodiny.dajHodnotuAkoRetazec() + ":"
                        + aMinuty.dajHodnotuAkoRetazec();
}
```

CiselnyDisplej – dajHodnotuAkoRetazec

```
public String dajHodnotuAkoRetazec()
```

- vracia hodnotu ako dvojciferné číslo
- nula na začiatku

Riešenie problému s nulou – UML



Riešenie problému s nulou – Java(1)

```
public String dajHodnotuAkoRetazec()
{
    String hodnotaAkoRetazec;
    if (aHodnota < 10) {
        hodnotaAkoRetazec = "0" + aHodnota;
    } else {
        hodnotaAkoRetazec = "" + aHodnota;
    }
    return hodnotaAkoRetazec;
}
```

Riešenie problému s nulou – Java(2)

```
public String dajHodnotuAkoRetazec()
{
    if (aHodnota < 10) {
        return "0" + aHodnota;
    } else {
        return "" + aHodnota;
    }
}
```

Metóda krok

```
public void krok()
{
    aHodnota = (aHodnota + 1) % aHornaHranica;
}
```

Priraďovací príkaz

```
aHodnota = (aHodnota + 1) % aHornaHranica;
                          0 + 1
                            1     % 24
                                  1
aHodnota = 1
```

```
int hodnota = aHodnota + 1;
aHodnota = hodnota % aHornaHranica;
```

Metóda krok$_{(1)}$


```
aHodnota = (aHodnota + 1) % aHornaHranica;
```

Metóda krok$_{(2)}$


```
aHodnota = (aHodnota + 1) % aHornaHranica;
```

Metóda krok(3)


```
aHodnota = (aHodnota + 1) % aHornaHranica;
```

Vďaka za pozornosť