Informatika 1

Skladanie objektov

doc. Ing. Ján Janech, PhD.
16. 10. 2023

ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Pojmy zavedené v 3. prednáške (1)

- algoritmus
  - vlastnosti
  - procesor

Pojmy zavedené v 3. prednáške (2)

- štruktúrované programovanie
  - základné konštrukčné prvky
    - postupnosť – sekvencia
    - vetvenie
    - cykly
  - grafické znázornenie v UML

Pojmy zavedené v 3. prednáške (3)

- štruktúrované programovanie v jazyku Java
  - vetvenie
    - úplný/neúplný príkaz if
    - príkaz switch
  - cykly
    - for
    - while
    - do-while
  - príkaz bloku
  - príkazy na riadenie cyklu
    - break
    - continue
  - predčasné ukončenie vykonávania metódy príkazom return

Pojmy zavedené v 3. prednáške (4)

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

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 – celok
- motor, prevodovka, kolesá – 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

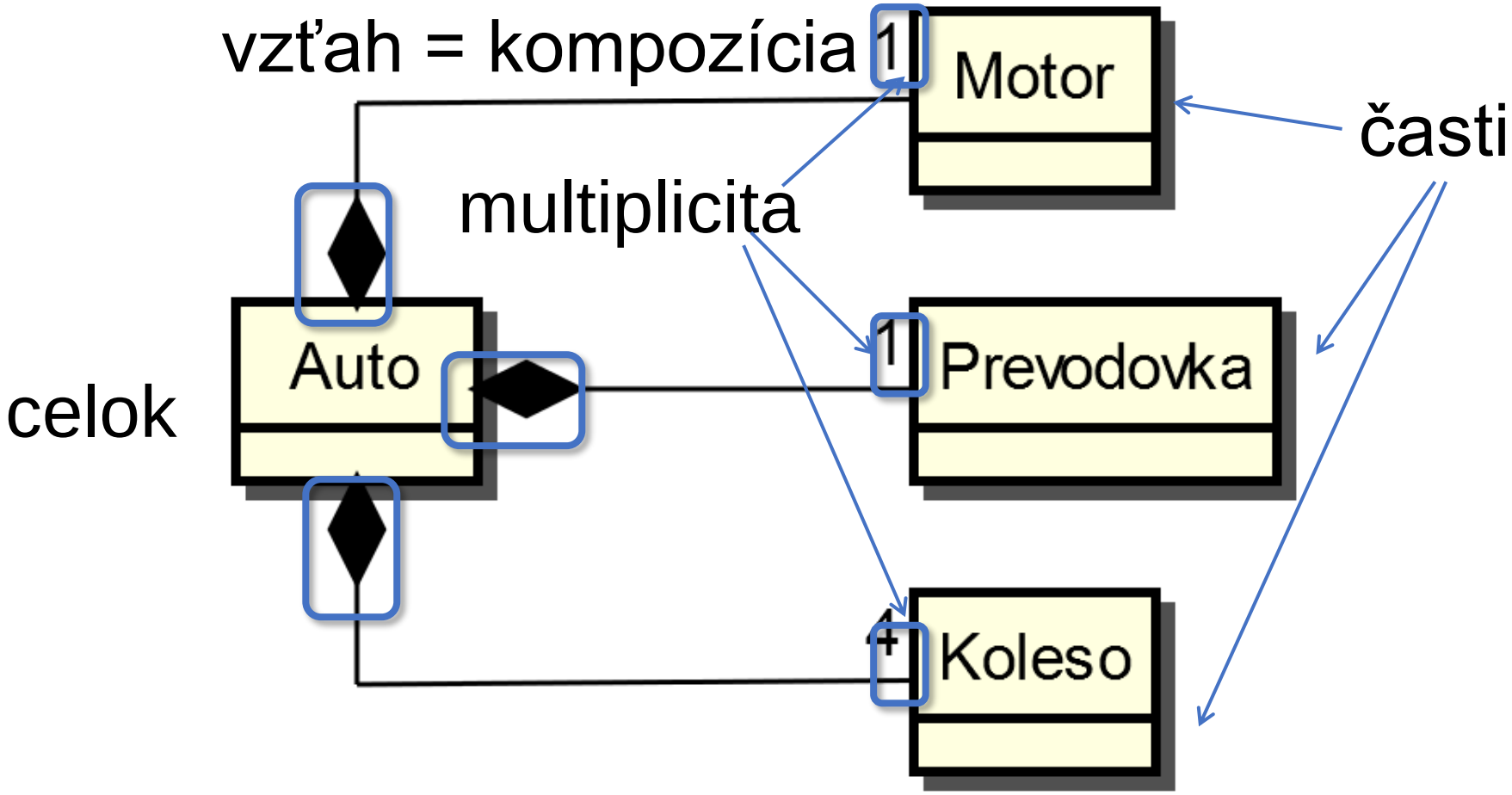
- úloha celku – organizovanie spolupráce častí

Kompozícia – skladanie objektov (4)

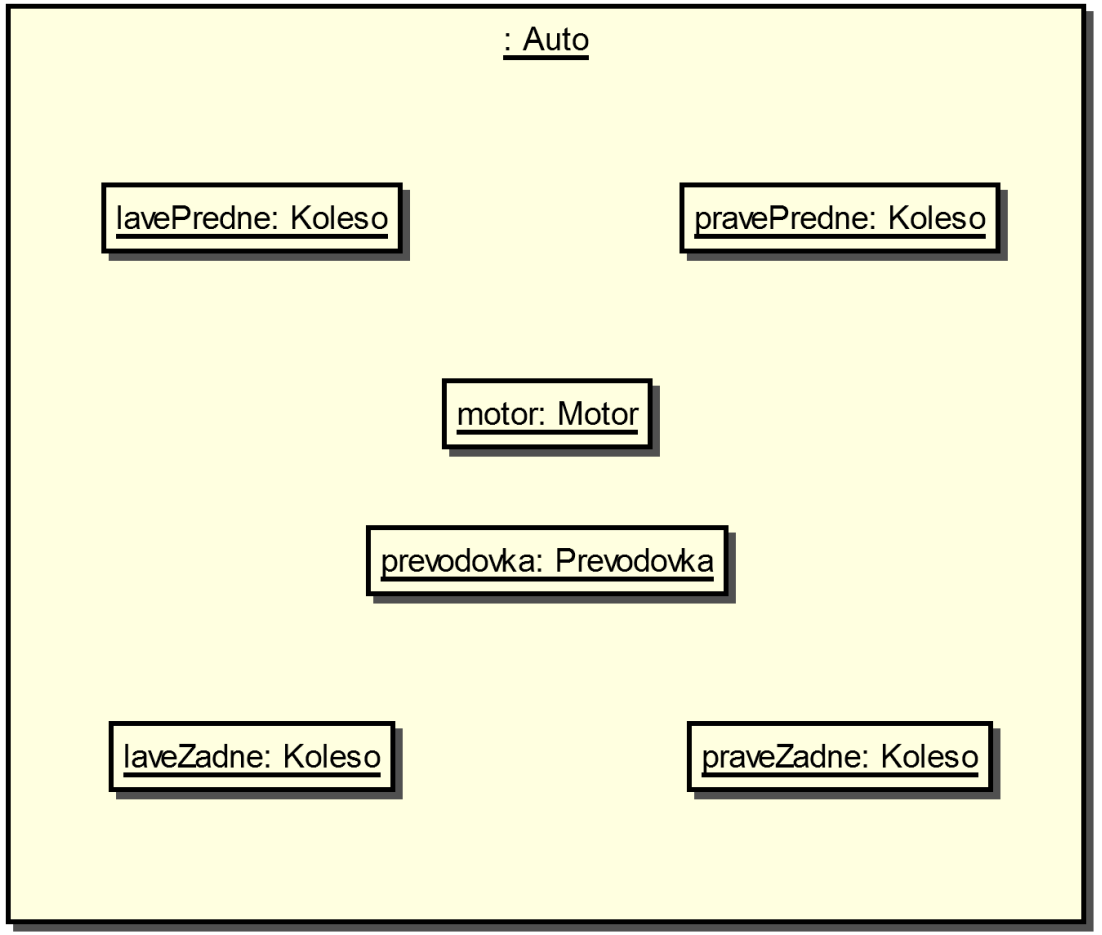
- charakteristika kompozície
- spoločný životný cyklus
  - spoločný vznik – celok (+ časti)
    - vznik časti – súčasť vzniku celku
  - služby – len celok
    - vonkajší pohľad – rozhranie auta
  - spoločný zánik – celok (+ časti)

- zodpovednosť celku za časti

Kompozícia v UML



Kompozícia – diagram objektov

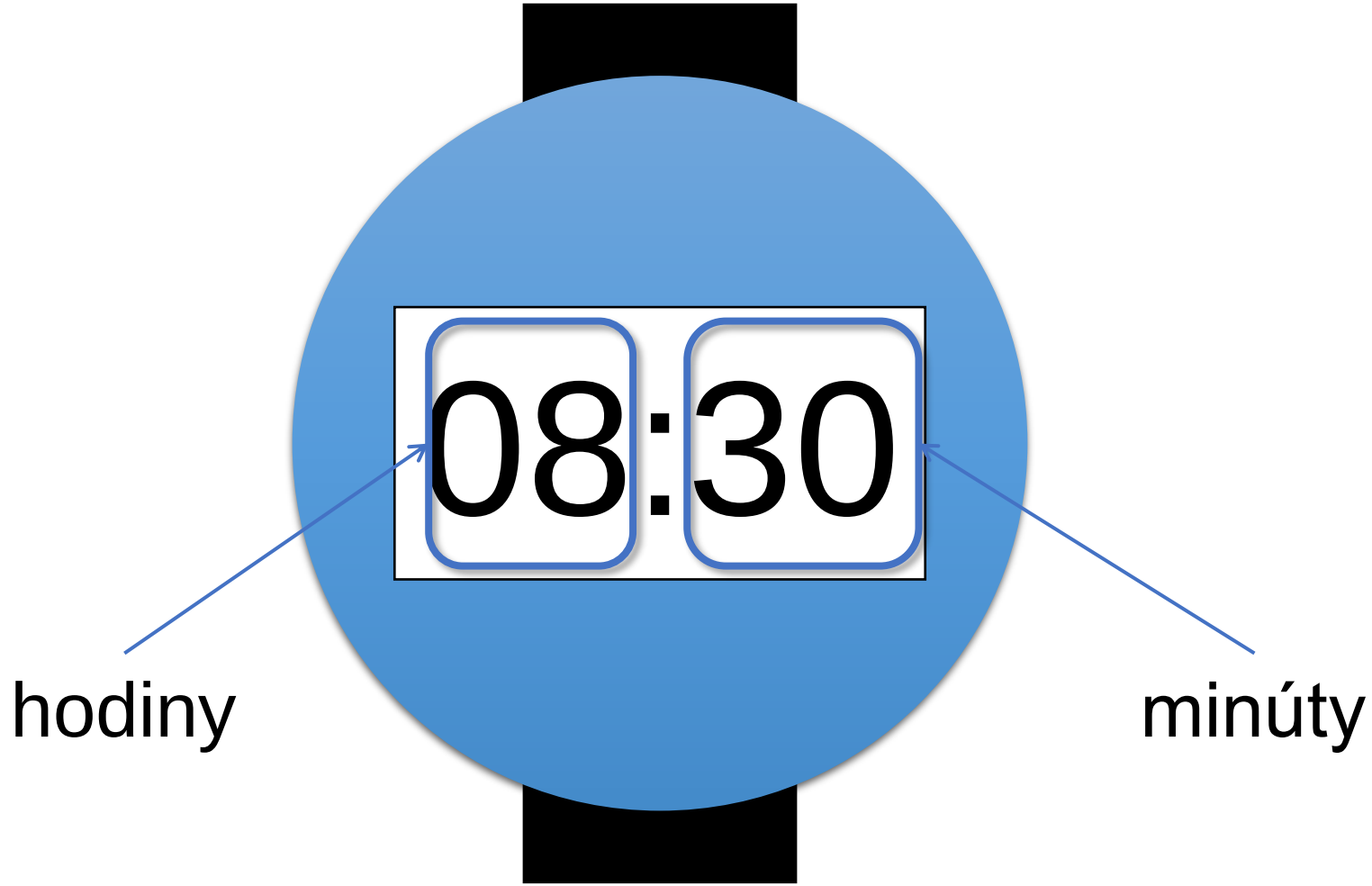


Projekt digitálne hodiny

- 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

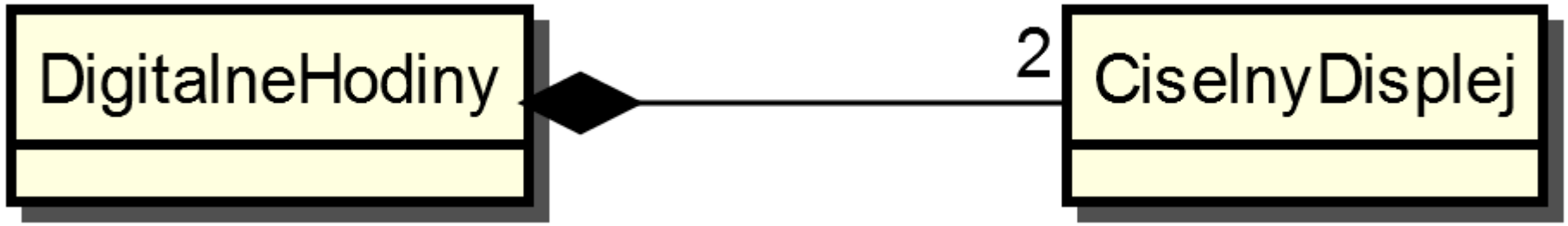
- krok pre hodiny: 1 hodina
- krok pre minúty: 1 minúta
- maximum pre hodiny: 23
- maximum pre minúty: 59

Riešenie rozdielov

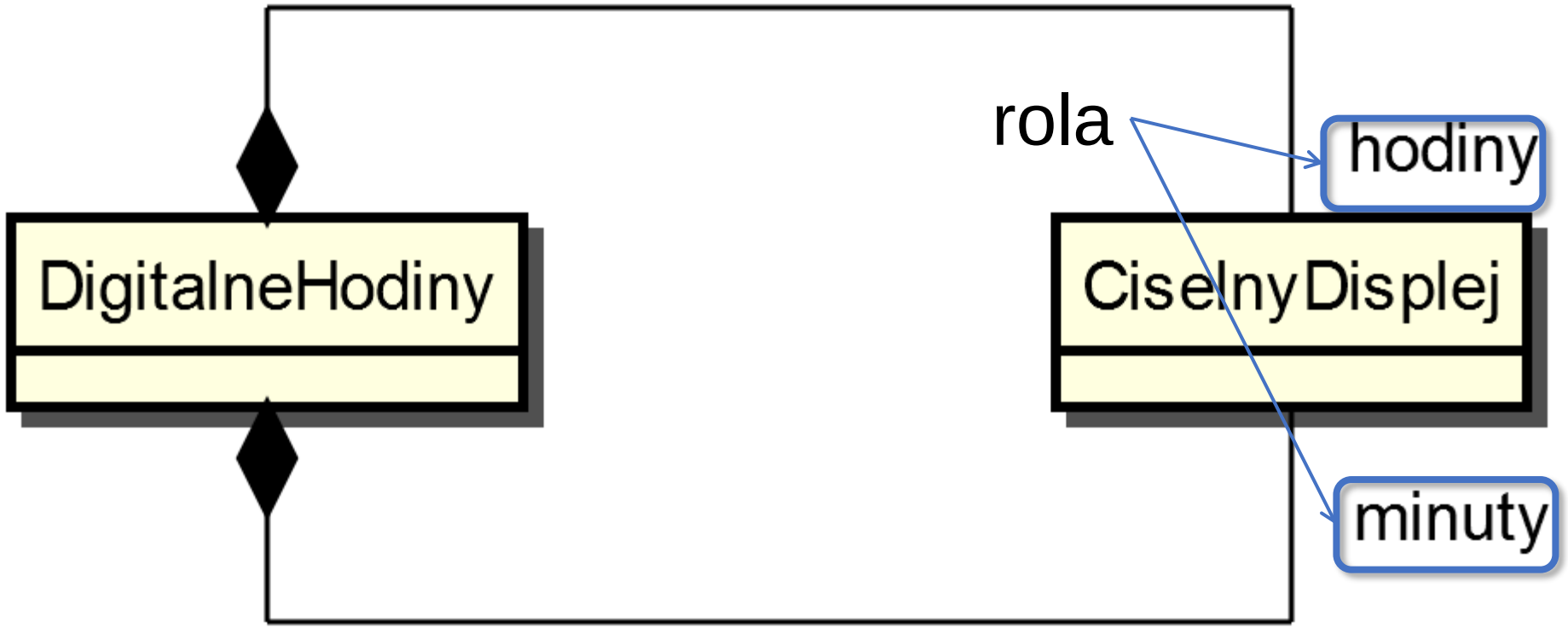
- dve triedy (MinutovyDisplej, HodinovyDisplej)

- spoločná trieda (CiselnyDisplej)
  - rôznosť krokov
    - oba kroky o 1
    - rôzne jednotky – úroveň interpretácie
  - rôznosť maxima
    - nastaviteľné maximum
    - parametre konštruktora

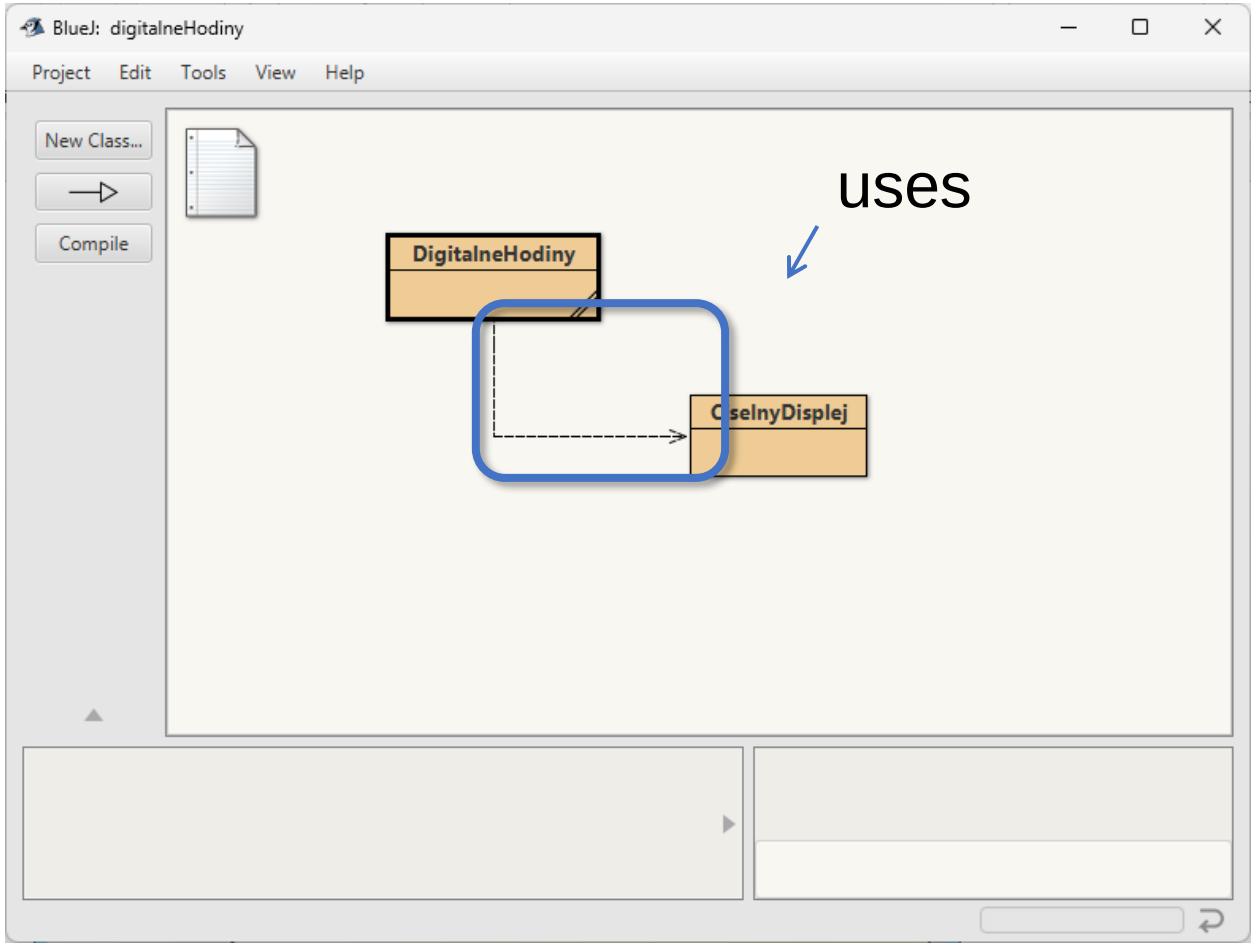
Digitálne hodiny v UML(1)



Digitálne hodiny v UML(2)



Digitálne hodiny v Bluej



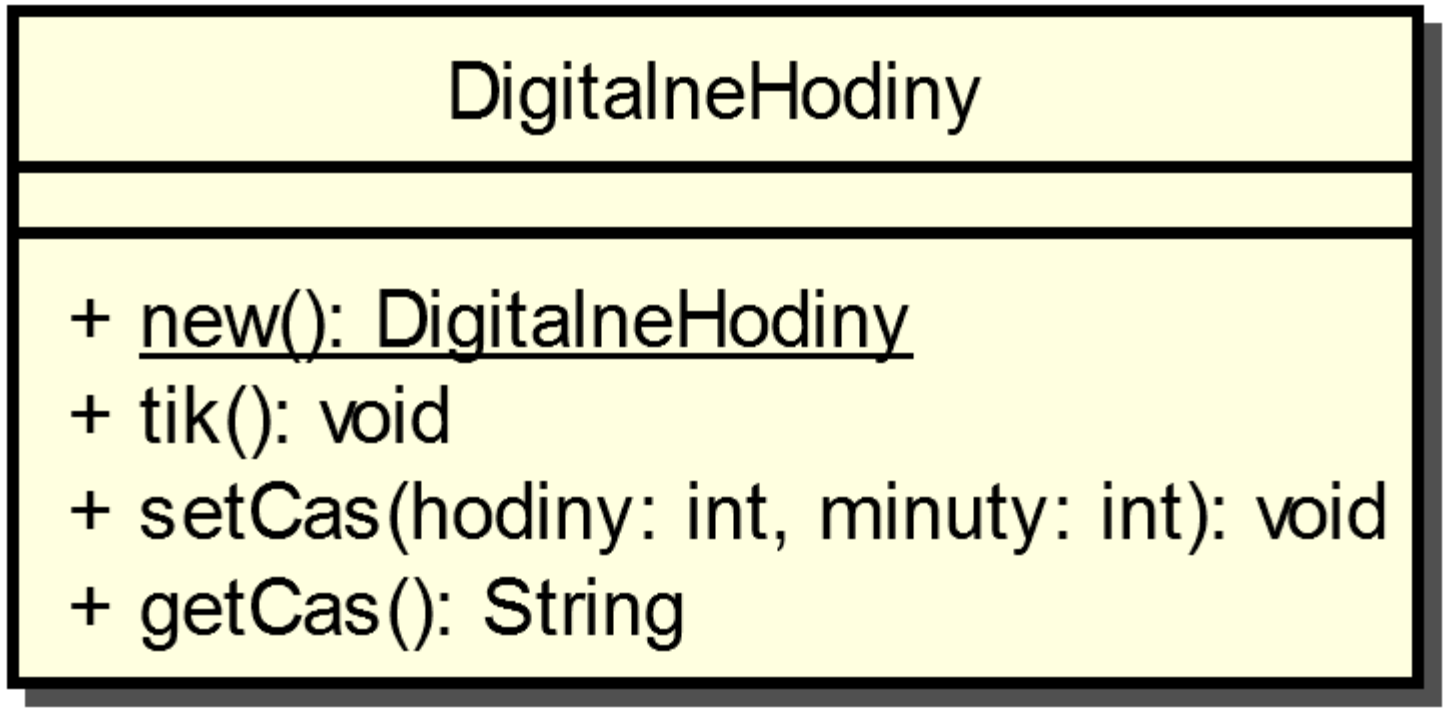
Charakteristiky číselného displeja

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

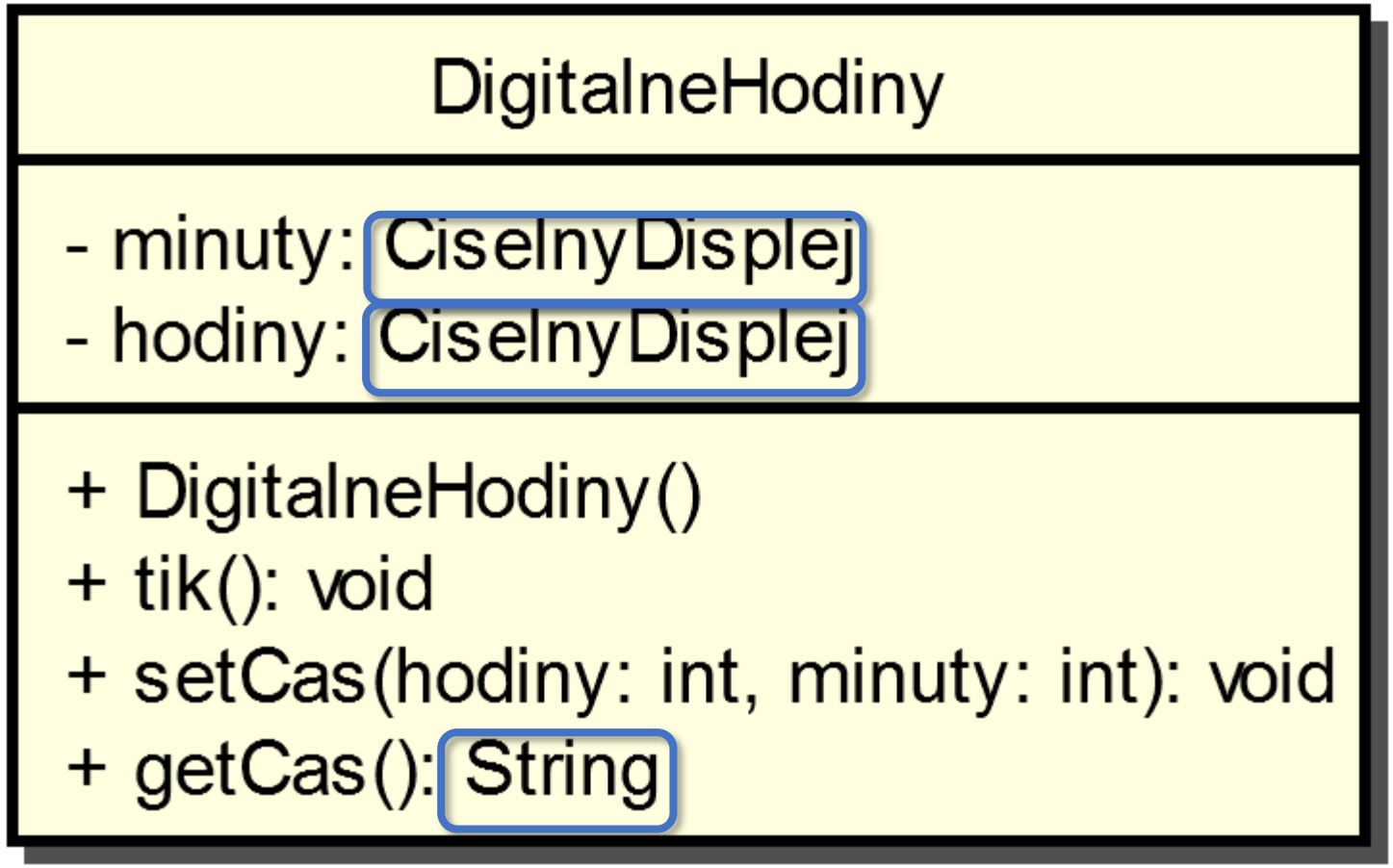
Úlohy digitálnych hodín – celku

- 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
    - hodiny : CiselnyDisplej
  - typ návratovej hodnoty
    - getCas() : String

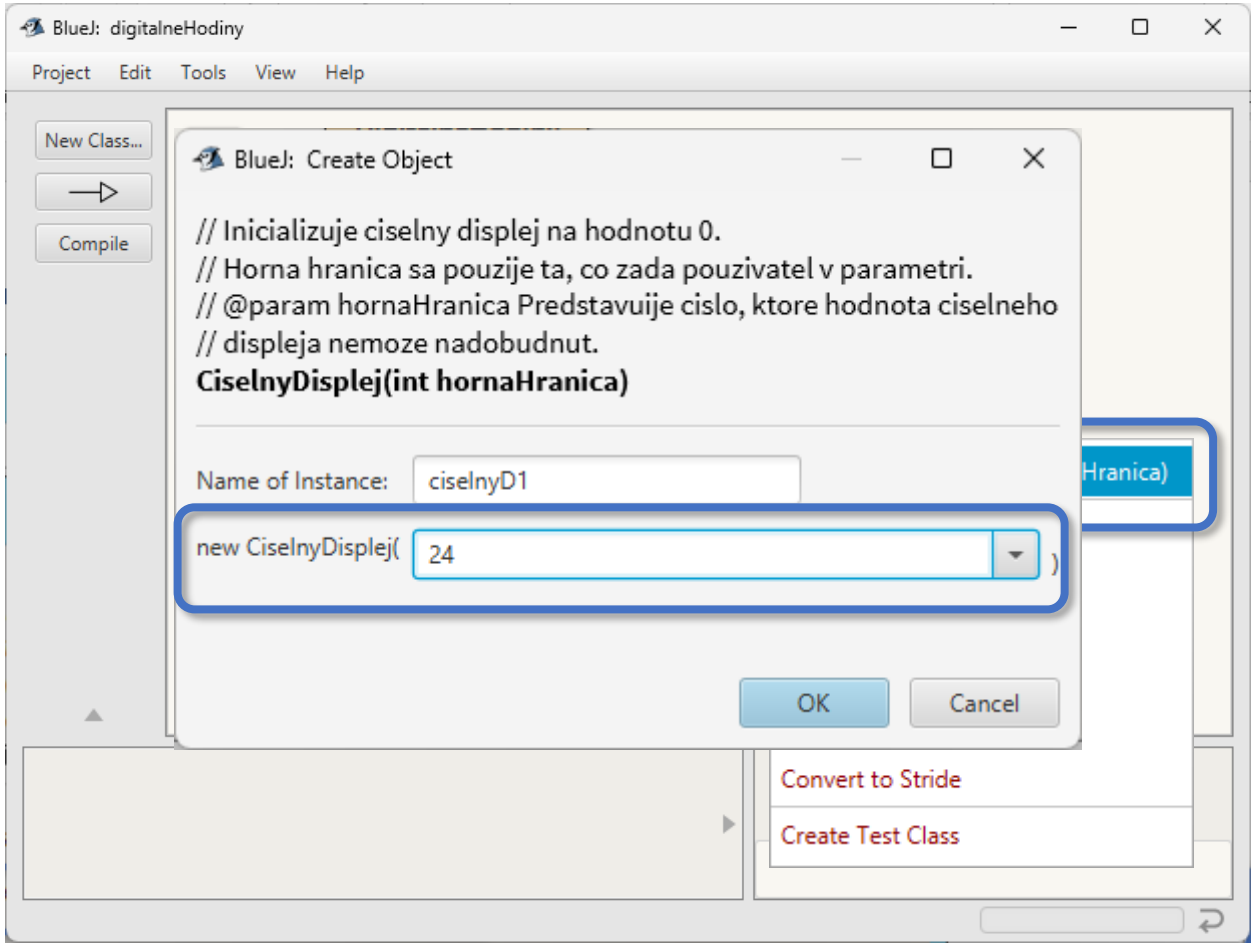
DigitalneHodiny – Java

```
public class DigitalneHodiny {
    private CiselnyDisplej hodiny;
    private CiselnyDisplej minuty;

    ...
}
```

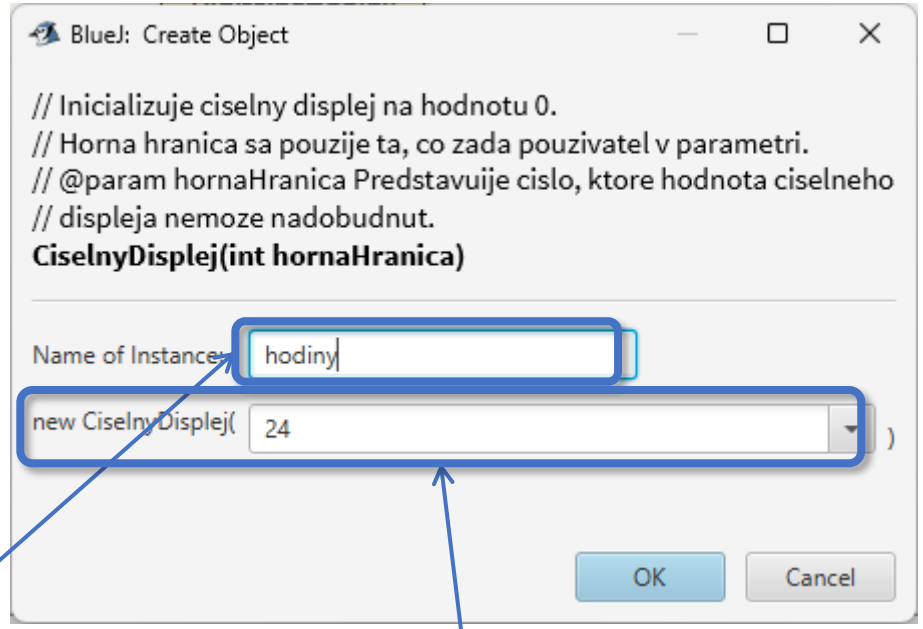
DigitalneHodiny – konštruktor – Java

```
public DigitalneHodiny() {
    this.hodiny = new CiselnyDisplej(24);
    this.minuty = new CiselnyDisplej(60);
}
```

Správa triede „new“

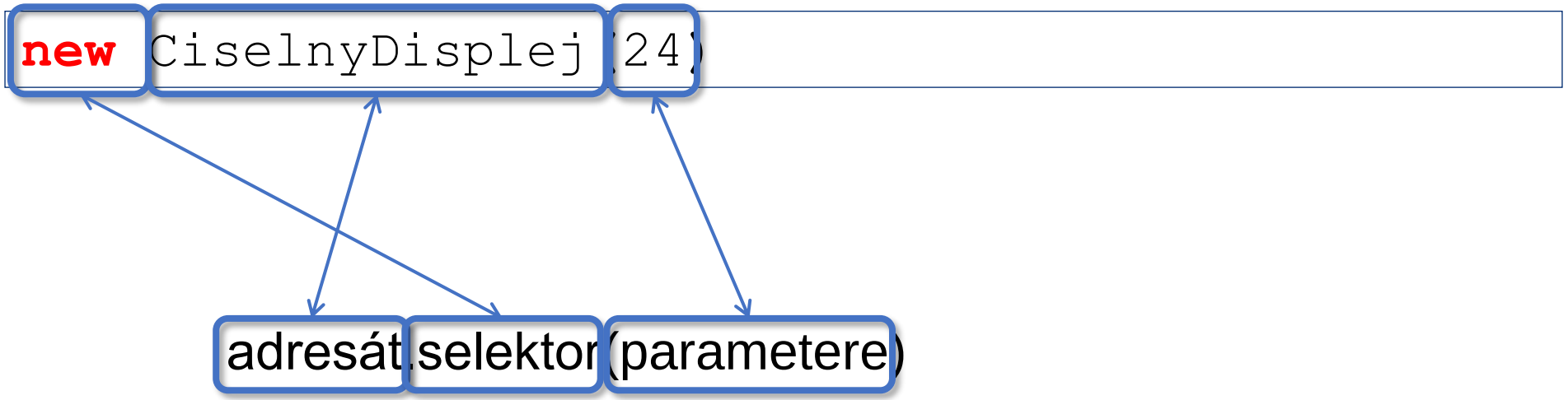


Správa triede „new“




```
this.hodiny = new CiselnyDisplej(24);
```

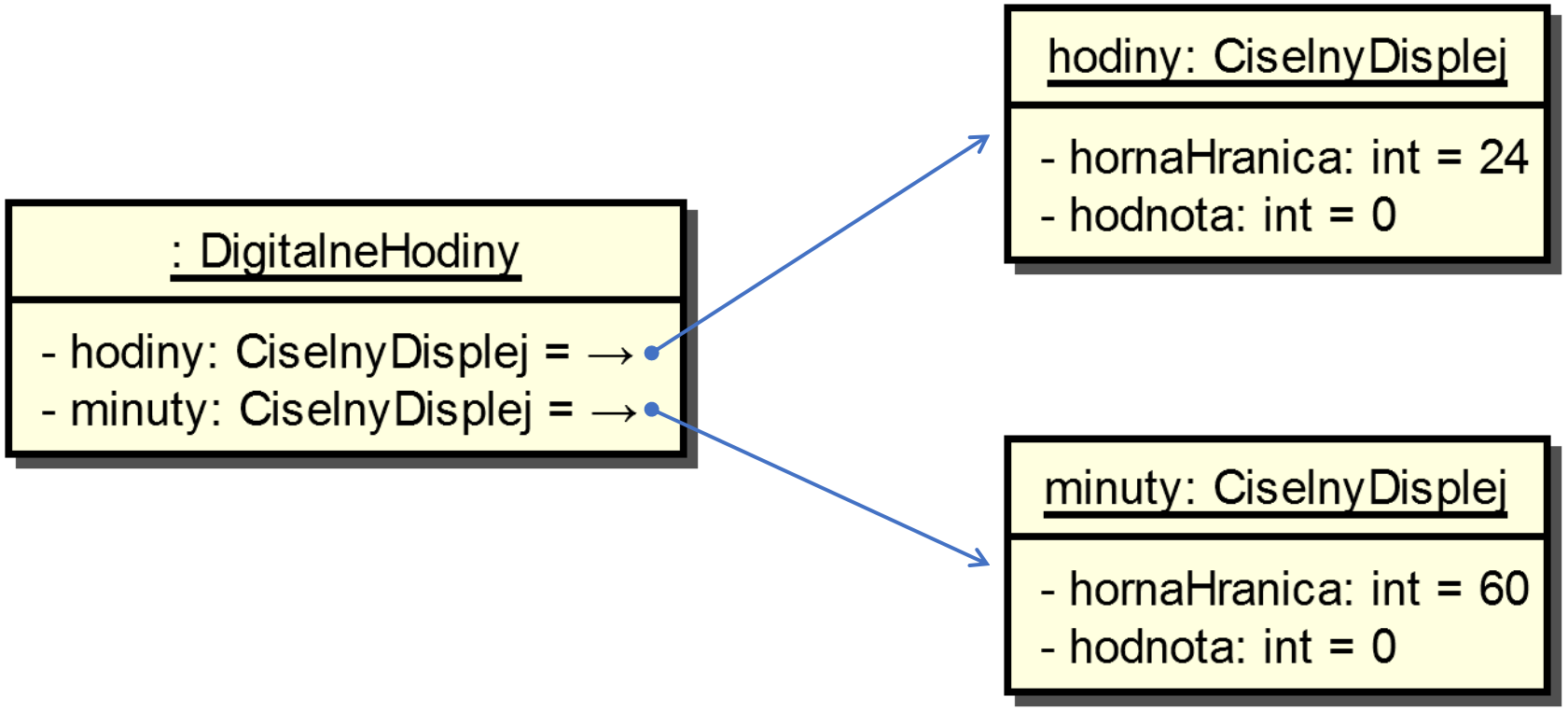
Štruktúra správy „new“



new = operátor v jazyku Java

DigitalneHodiny – konštruktor

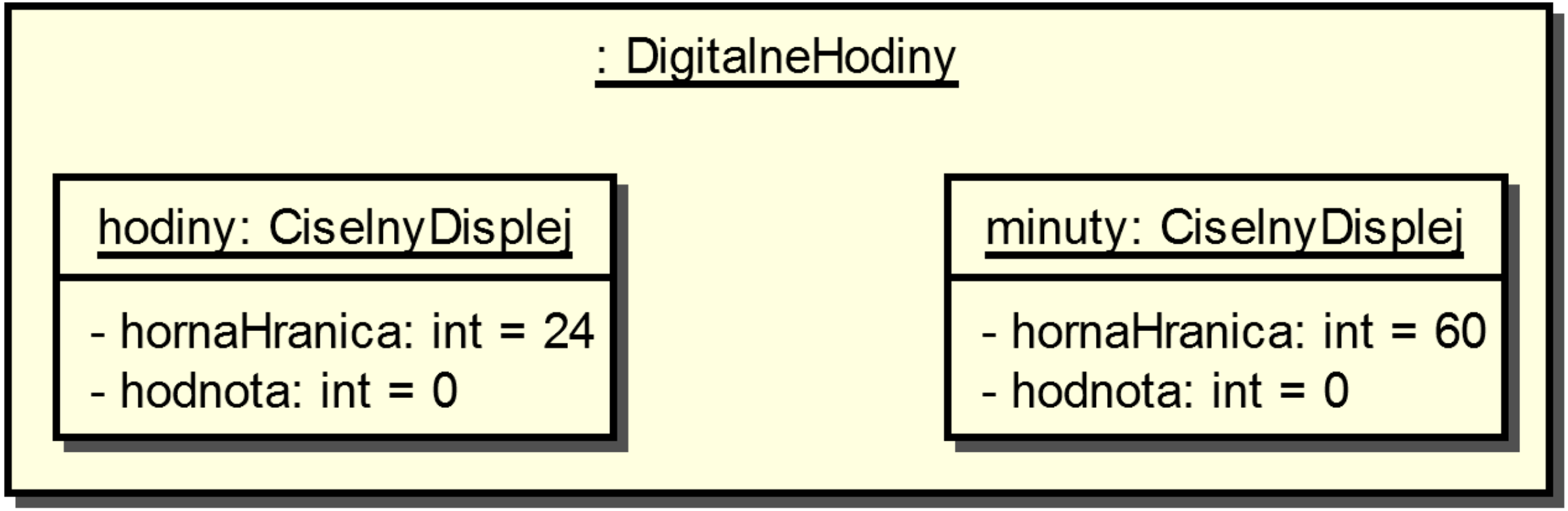
```
this.hodiny = new CiselnyDisplej(24);
this.minuty = new CiselnyDisplej(60);
```



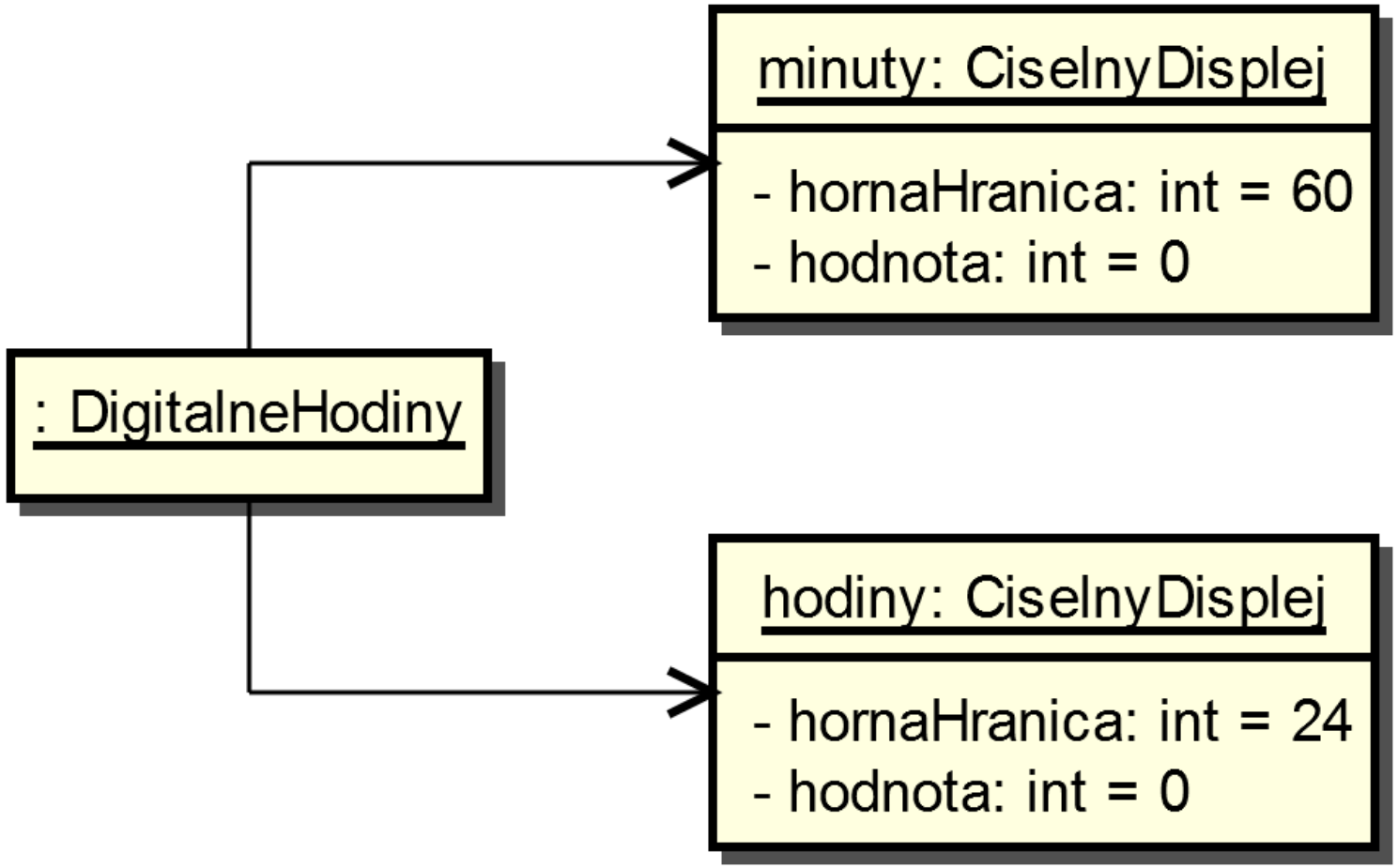
Referencie

- 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)



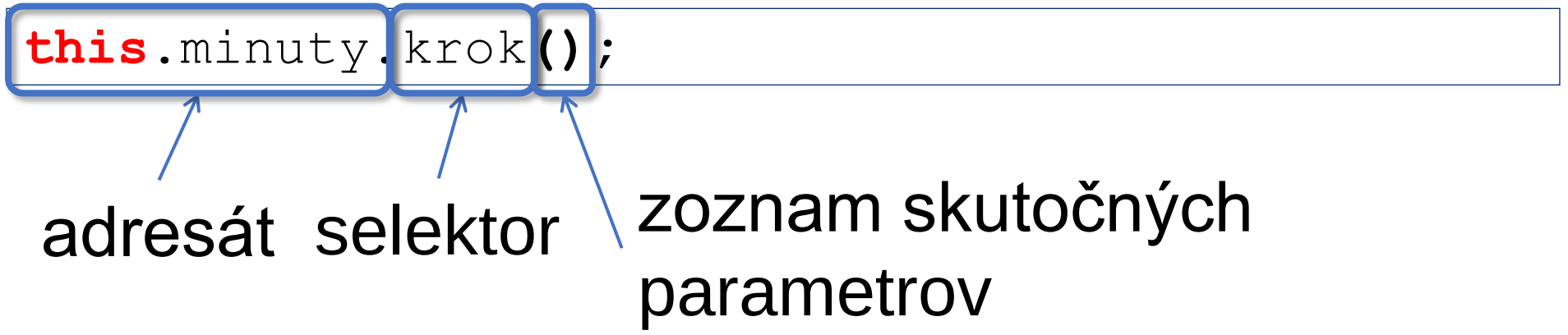
DigitalneHodiny – tik – UML

```
public void tik() {
    this.minuty.krok();

    if (this.minuty.getHodnota() == 0) {
        this.hodiny.krok();
    }
}
```

Príkaz na poslanie správy

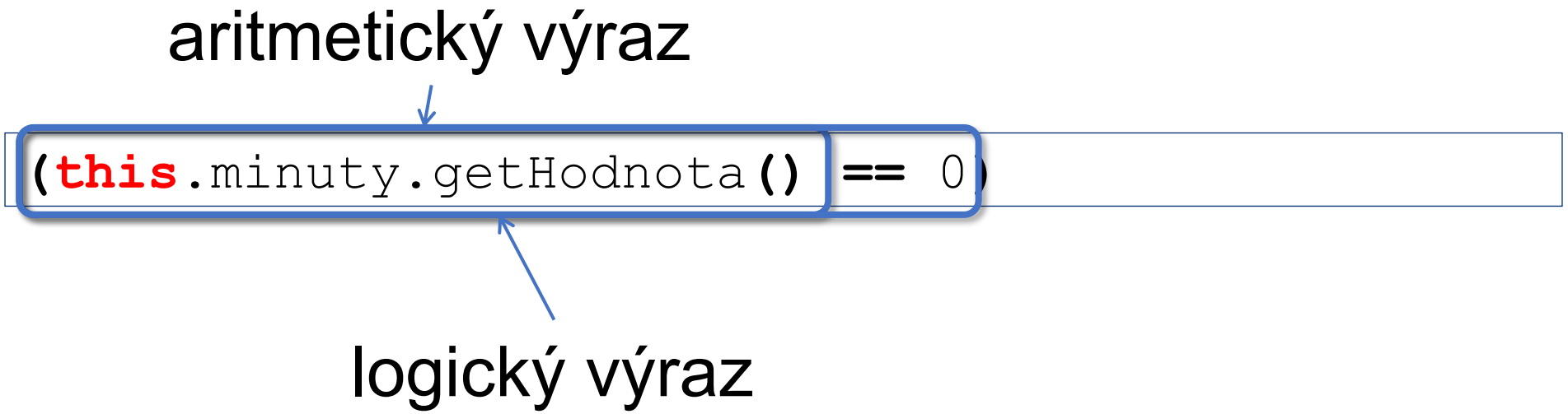
- objekt.sprava(skutocneParametre);
  - každý zo skutočných parametrov môže byť výraz
- vždy samostatný príkaz

- príklad:



Správy s návratovou hodnotou

- typ návratovej hodnoty
  - primitívny
  - objektový
- správa s návratovou hodnotou – výraz



Objektový výraz

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

```
this.hodiny = new CiselnyDisplej(24);
```

objektový výraz

Metóda getCas

```
public String getCas()
```

Literál typu String

- reťazcový literál

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

- prázdny reťazec

```
""
```

- literál – realizovaný ako inštancia triedy String

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
```

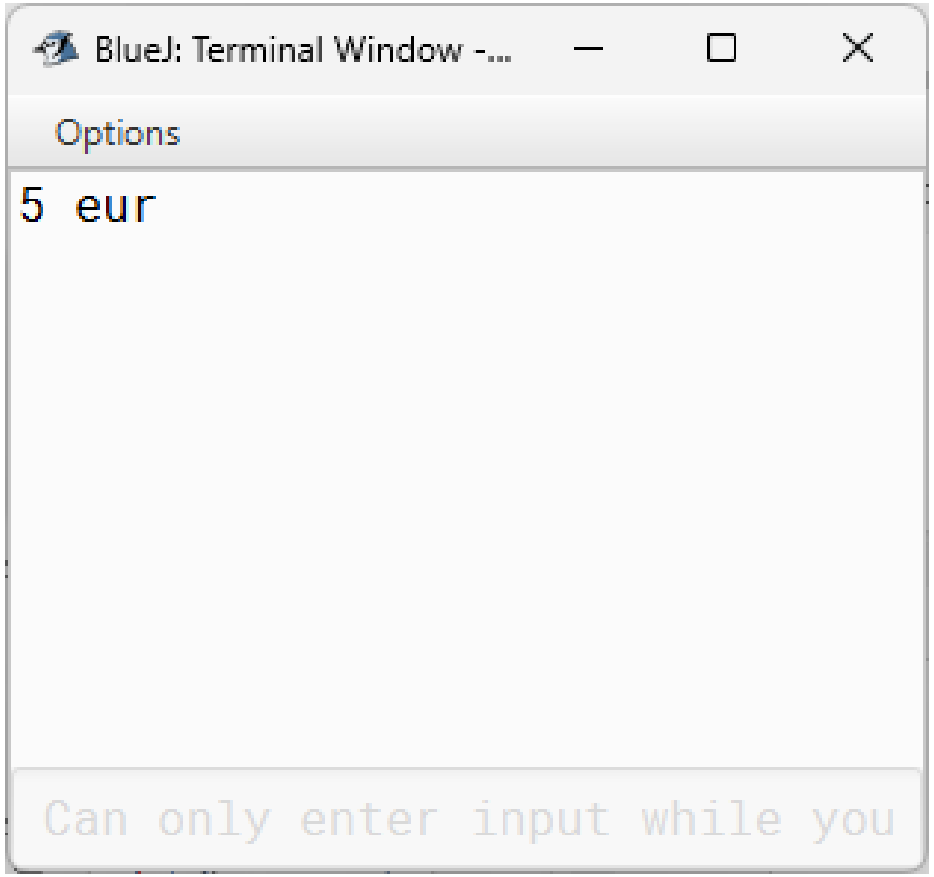
- aspoň jeden operand je reťazcový
- hodnota reťazcového výrazu – reťazec
- reťazcový operand
  - literál
  - premenná typu String
  - návratová hodnota typu String
  - reťazcový výraz

Spájanie reťazcov (3)

- iný ako reťazcový operand sa automaticky prekonvertuje na reťazec
  - primitívne typy sa konvertujú automaticky
  - objektovým typom sa pošle správa toString()
- vyhodnocovanie výrazu – zľava doprava

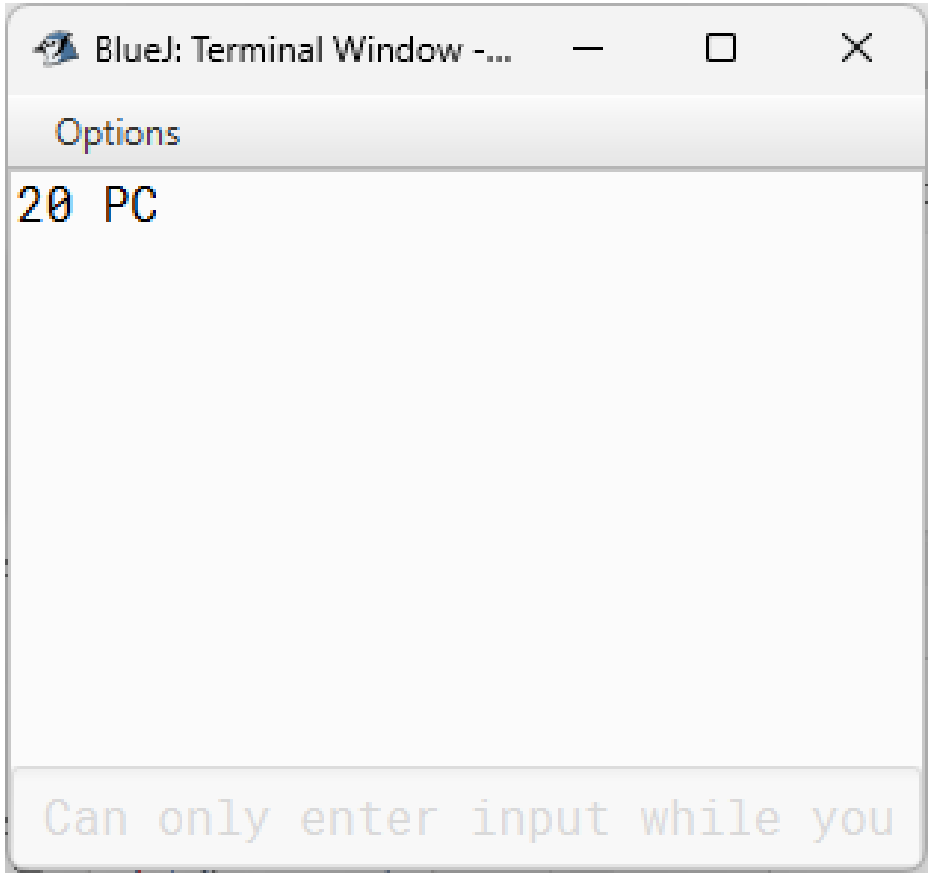
Spájanie reťazcov (1)

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



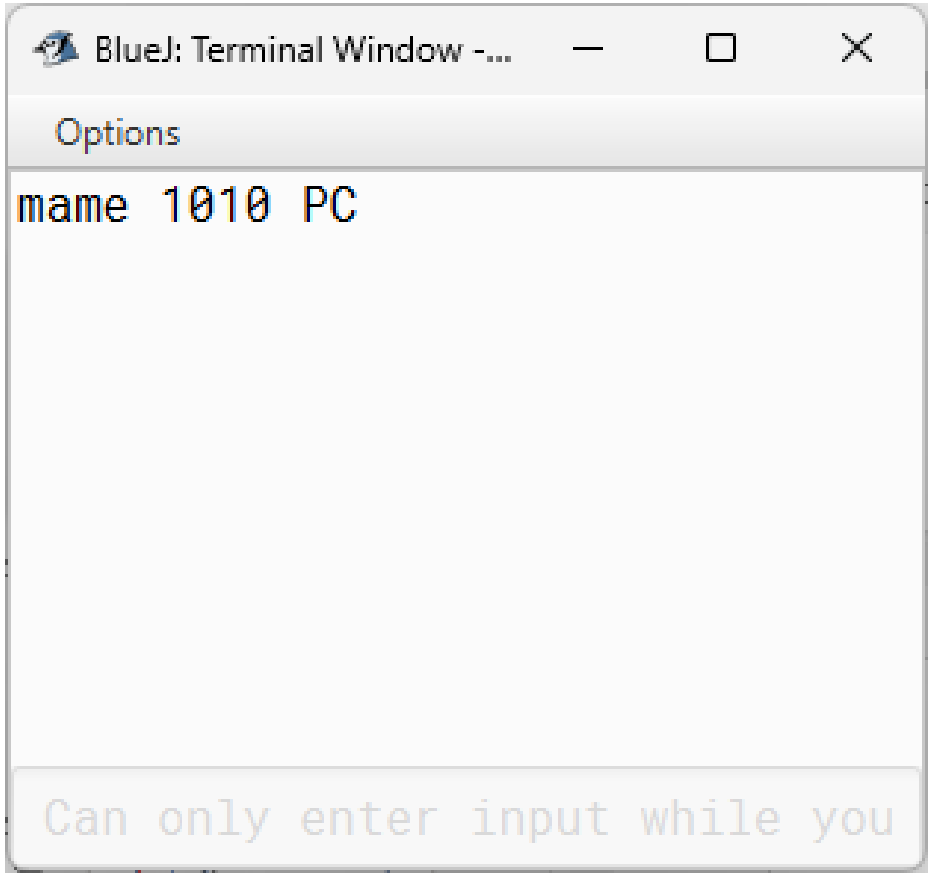
Spájanie reťazcov (2)

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



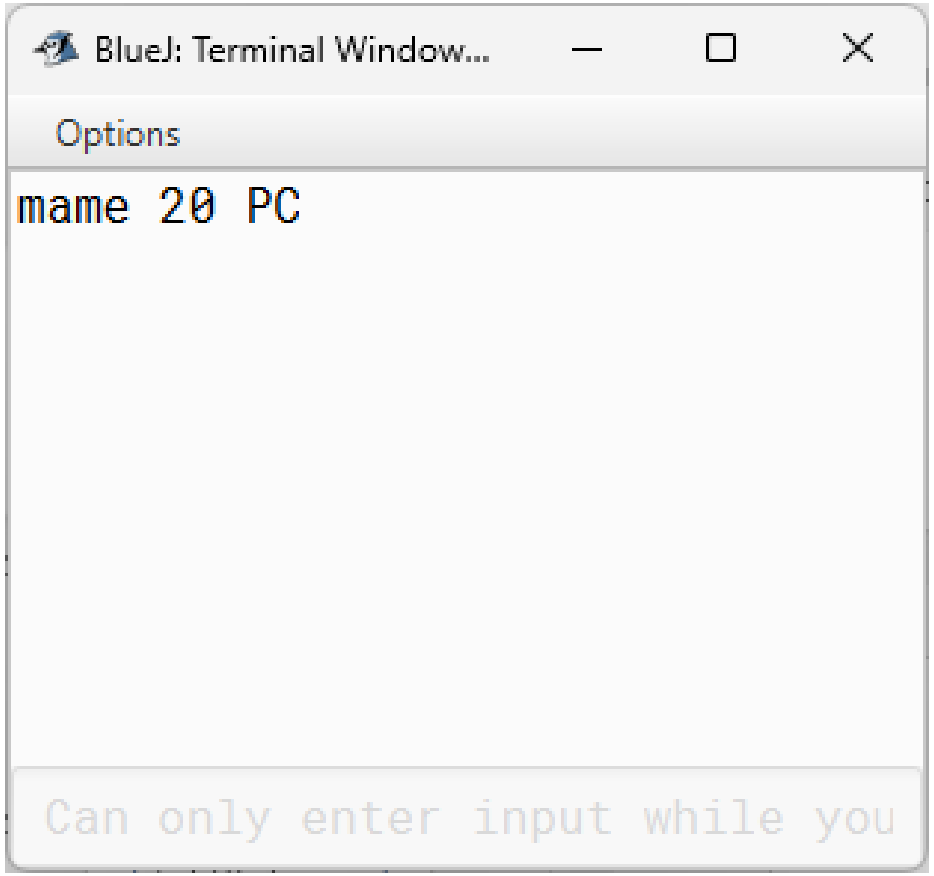
Spájanie reťazcov (3)

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



Spájanie reťazcov (4)

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



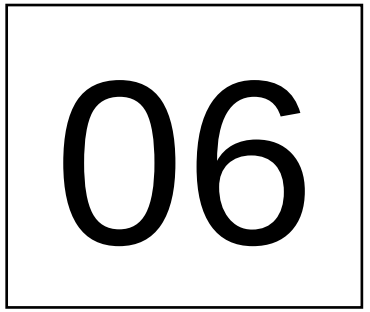
Metóda getCas

```
public String getCas() {
    return this.hodiny.getHodnotaAkoRetazec() + ":"
            + this.minuty.getHodnotaAkoRetazec();
}
```

CiselnyDisplej – getHodnotaAkoRetazec

```
public String getHodnotaAkoRetazec()
```

- vracia hodnotu ako dvojciferné číslo
- nula na začiatku



Riešenie problému s nulou – Java(2)

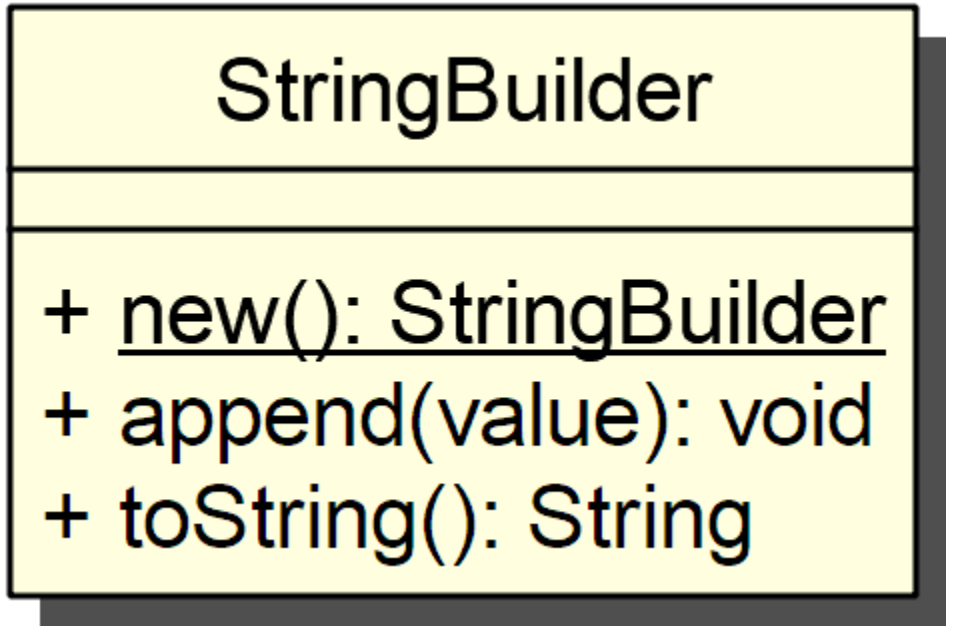
```
public String getHodnotaAkoRetazec() {
    if (this.hodnota < 10) {
        return "0" + this.hodnota;
    } else {
        return "" + this.hodnota;
    }
}
```

Ďalšie spôsoby spájania reťazcov

- StringBuilder
- String.format/System.out.format

StringBuilder

- objekt na vytváranie dlhších reťazcov



StringBuilder – použitie

```
int pocet = 10 + 10;
StringBuilder retazec = new StringBuilder();
retazec.append("Mame ");
retazec.append(pocet);
retazec.append(" PC");
System.out.println(retazec.toString());
```

Dôvod pre použitie StringBuilder

- reťazcový operátor + sa aj tak prekladá na použitie StringBuilder
- rýchlejšie spájanie viac reťazcov
  - Shlemiel The Painter Algorithm

StringBuilder vs. operátor +

```
String a = "a";
// StringBuilder sb = new StringBuilder();
// sb.append(a); sb.append(a);
// String b = sb.toString();
String b = a + a;


// StringBuilder sb = new StringBuilder();
// sb.append(b); sb.append(a);
// String c = sb.toString();
String c = b + a;
…
```

String.format

- formátovanie reťazca

```
String.format("formátovací reťazec", hodnoty)
```

Formátovací znak	Význam
%d	Celé číslo
%s	Reťazec
%f	Desatinné číslo
%n	Koniec riadku

Formát	Význam
%5znak	Zarovnanie vpravo na 5 znakov
%-5znak	Zarovnanie vľavo na 5 znakov
%5.2f	Zarovnanie vpravo na 5 znakov, 2 des. miesta

String.format – použitie

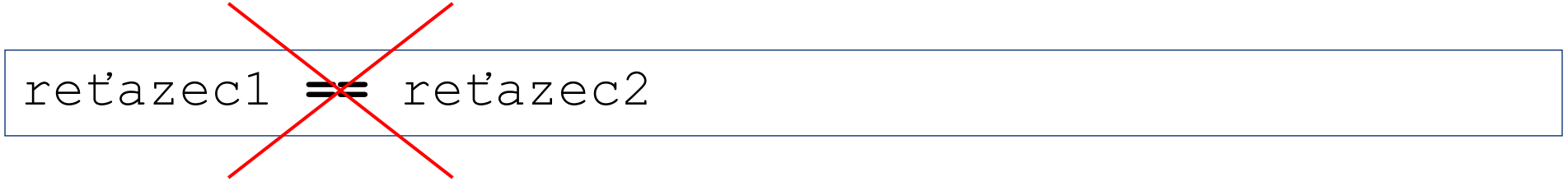
```
int pocet = 10 + 10;
String retazec = String.format("Mame %d PC", pocet);
System.out.println(retazec);

// resp. iba pre vypis
System.out.format("Mame %d PC%n", pocet);
```

getHodnotaAkoRetazec pomocou String.format

```
public String getHodnotaAkoRetazec() {
    return String.format("%02d", this.hodnota);
}
```

Dôležité – porovnávanie reťazcov

- operátor == porovnáva hodnoty
- String = objektový typ
- hodnotou objektového typu je referencia
- porovnanie reťazcov – správa equals



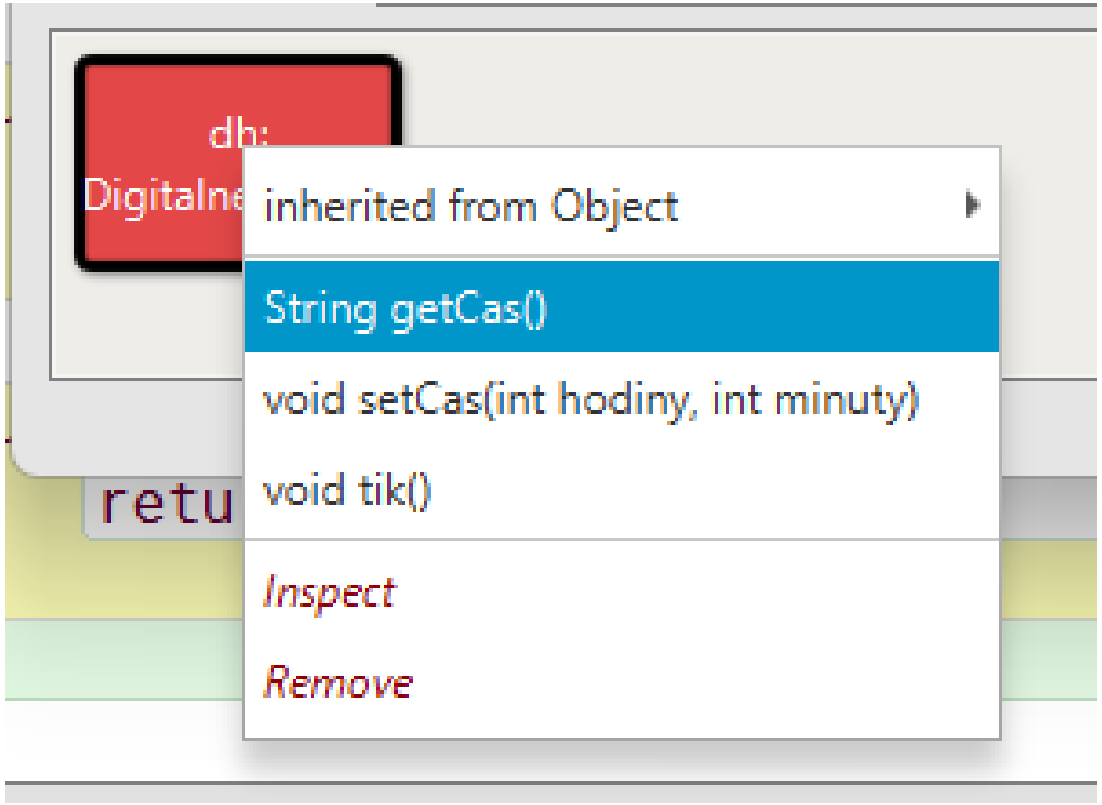

```
reťazec1.equals(reťazec2)
```

Zásobník

- rámec – frame
  - zoznam lokálnych premenných
  - vykonávaný príkaz
  - this
- pri poslaní správy si procesor rámec odloží
  - po návrate z vykonania metódy sa do neho vráti

Zásobník – príklad

Rámec	Dáta
(BlueJ)	dh



Zásobník – príklad

Rámec	Dáta
(BlueJ)	dh
DigitalneHodiny.getCas	this = dh

```
public String getCas() {
    return this.hodiny.getHAR()
           + ":" +
           this.minuty.getHAR();
}
```

Pozn.: HAR = HodnotaAkoRetazec

Zásobník – príklad

Rámec	Dáta
(BlueJ)	dh
DigitalneHodiny.getCas	this = dh
CiselnyDisplej.getHAR	this = dh.hodiny

```
public String getHAR() {
    return
        String.format(
            "%02d",
            this.hodnota
        );
}
```

Pozn.: HAR = HodnotaAkoRetazec

Zásobník – príklad

Rámec	Dáta
(BlueJ)	dh
DigitalneHodiny.getCas	this = dh
CiselnyDisplej.getHAR	this = dh.hodiny
String.format	format = "%02d" arg = 9
	...

```
public static String format(
    String format, int arg) {
        return
            new Formatter()
            .format(
                format,
                arg
            )
            .toString();
}
```

Pozn.: HAR = HodnotaAkoRetazec

Zásobník – príklad

Rámec	Dáta
(BlueJ)	dh
DigitalneHodiny.getCas	this = dh
CiselnyDisplej.getHAR	this = dh.hodiny String.format = "09"

```
public String getHAR() {
    return
        String.format(
            "%02d",
            this.hodnota
        );
}
```

Pozn.: HAR = HodnotaAkoRetazec

Zásobník – príklad

Rámec	Dáta
(BlueJ)	dh
DigitalneHodiny.getCas	this = dh hodiny.getHAR = "09"

```
public String getCas() {
  return this.hodiny.getHAR()
         + ":" +
         this.minuty.getHAR();
}
```

Pozn.: HAR = HodnotaAkoRetazec

Zásobník – príklad

Rámec	Dáta
(BlueJ)	dh
DigitalneHodiny.getCas	this = dh hodiny.getHAR = "09"
CiselnyDisplej.getHAR	this = dh.minuty

```
public String getHAR() {
    return
        String.format(
            "%02d",
            this.hodnota
        );
}
```

Pozn.: HAR = HodnotaAkoRetazec

Zásobník – príklad

Rámec	Dáta
(BlueJ)	dh
DigitalneHodiny.getCas	this = dh hodiny.getHAR = "09"
CiselnyDisplej.getHAR	this = dh.minuty
String.format	format = "%02d" arg = 30 ...

```
public static String format(
    String format, int arg) {
        return
            new Formatter()
            .format(
                format,
                arg
            )
            .toString();
}
```

Pozn.: HAR = HodnotaAkoRetazec

Zásobník – príklad

Rámec	Dáta
(BlueJ)	dh
DigitalneHodiny.getCas	this = dh hodiny.getHAR = "09"
CiselnyDisplej.getHAR	this = dh.minuty String.format = "30"

```
public String getHAR() {
    return
        String.format(
            "%02d",
            this.hodnota
        );
}
```

Pozn.: HAR = HodnotaAkoRetazec

Zásobník – príklad

Rámec	Dáta
(BlueJ)	dh
DigitalneHodiny.getCas	this = dh hodiny.getHAR = "09" minuty.getHAR = "30"

```
public String getCas() {
    return this.hodiny.getHAR()
           + ":" +
           this.minuty.getHAR();
}
```

Pozn.: HAR = HodnotaAkoRetazec

Zásobník – príklad

Rámec	Dáta
(BlueJ)	dh dh.getCas = "9:30"