



Spolupráca objektov

Pojmy zavedené v 4. prednáške₍₁₎

- Modularizácia a abstrakcia
- Kompozícia – skladanie objektov
 - spoločné životné cykly
 - UML
- Asociácia – spolupráca objektov
 - nezávislé životné cykly
 - UML

Pojmy zavedené v 4. prednáške₍₂₎

- Objektové typy
- Referencie
- Diagram objektov

Pojmy zavedené v 4. prednáške₍₃₎

- Príkaz na poslanie správy
 - Príkaz na poslanie správy "new" triede
 - Príkaz na poslanie správy bez návratovej hodnotu
 - Príkaz na poslanie správy s návratovou hodnotou
- Objektový výraz
 - výstupný parameter správy objektového typu
 - špeciálne: reťazcový výraz

Pojmy zavedené v 4. prednáške₍₄₎

- Reťazce
 - reťazcové literály
 - spájanie reťazcov
 - reťazcové výrazy
 - trieda String

Cieľ prednášky

- Logické výrazy
- Relačné výrazy s objektmi
- Spolupráca objektov – asociácia

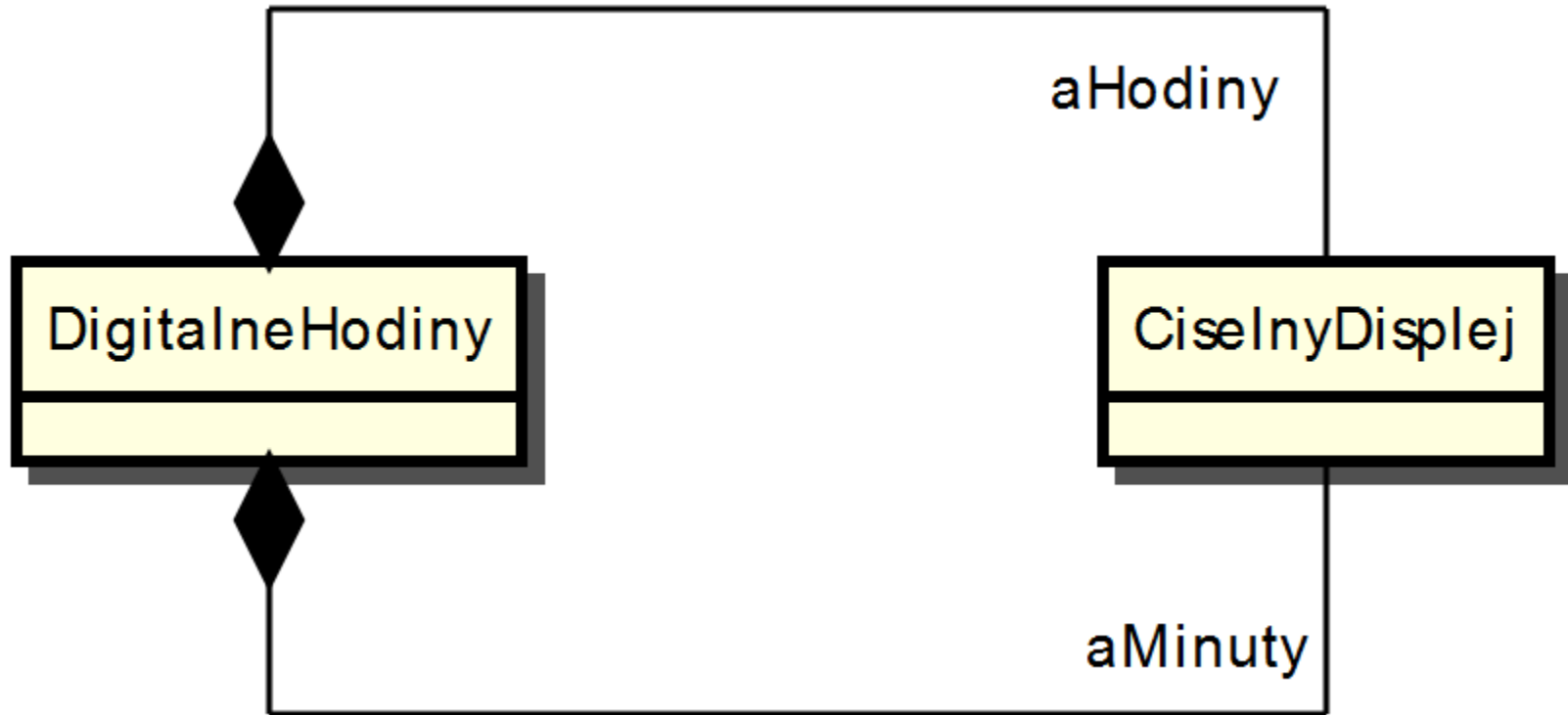
- príklad: Digitálne hodiny

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

Digitálne hodiny – verzia 1



DigitalneHodiny – vnútorný pohľad

DigitalneHodiny

- aHodiny: CiselyDisplej
- aMinuty: CiselyDisplej

- + DigitalneHodiny()
- + tik(): void
- + nastavCas(paHod: int, paMin: hod): void
- + dajCas(): String

CiselnyDisplej – vnútorný pohľad

CiselnyDisplej

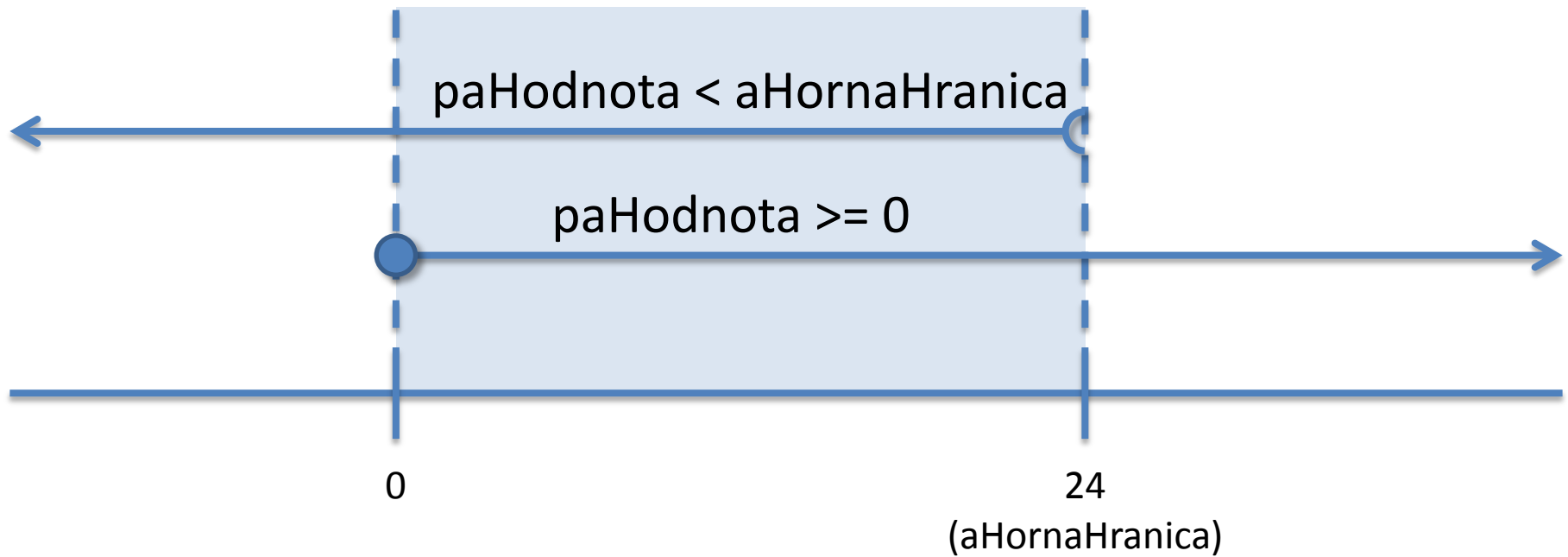
- aHodnota: int
- aHornaHranica: int

- + CiselnyDisplej()
- + dajHodnotu(): int
- + nastavHodnotu(int: paHodnota): void
- + dajHodnotuAkoRetazec(): String
- + krok(): void

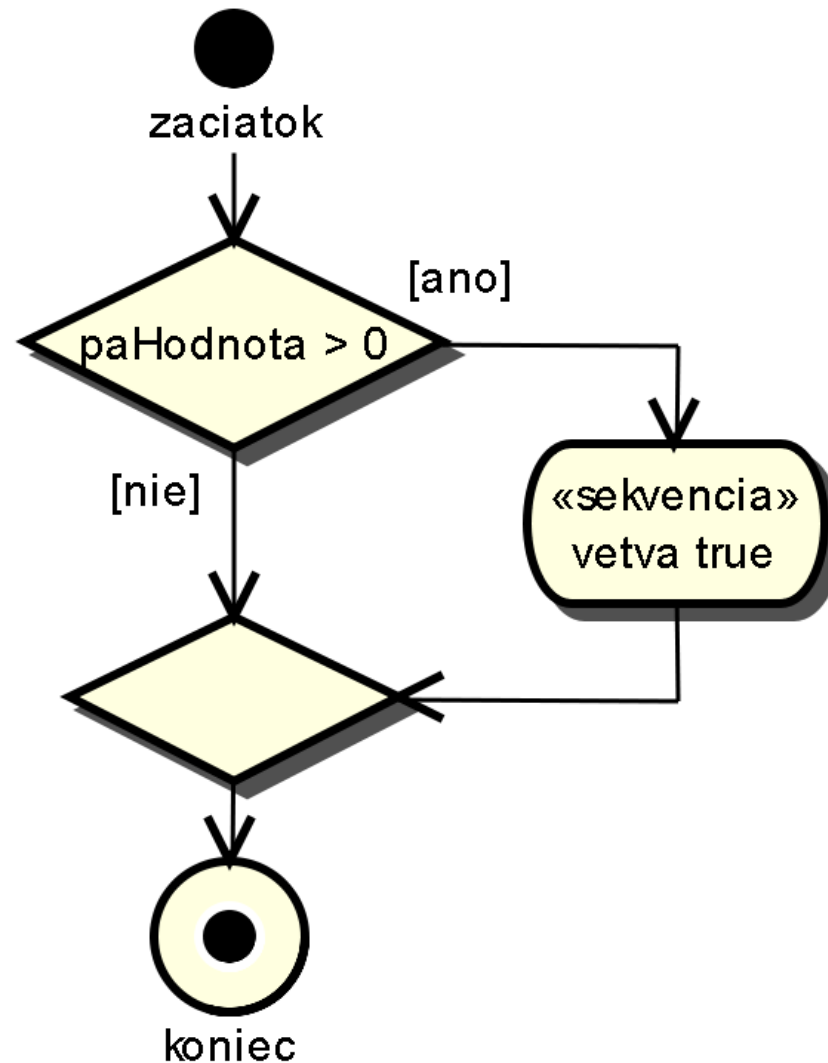
Metóda nastavHodnotu

- zmení hodnotu na displeji
- nesmie nastaviť hodnotu mimo rozsah
 - min 0
 - max aHornaHranica

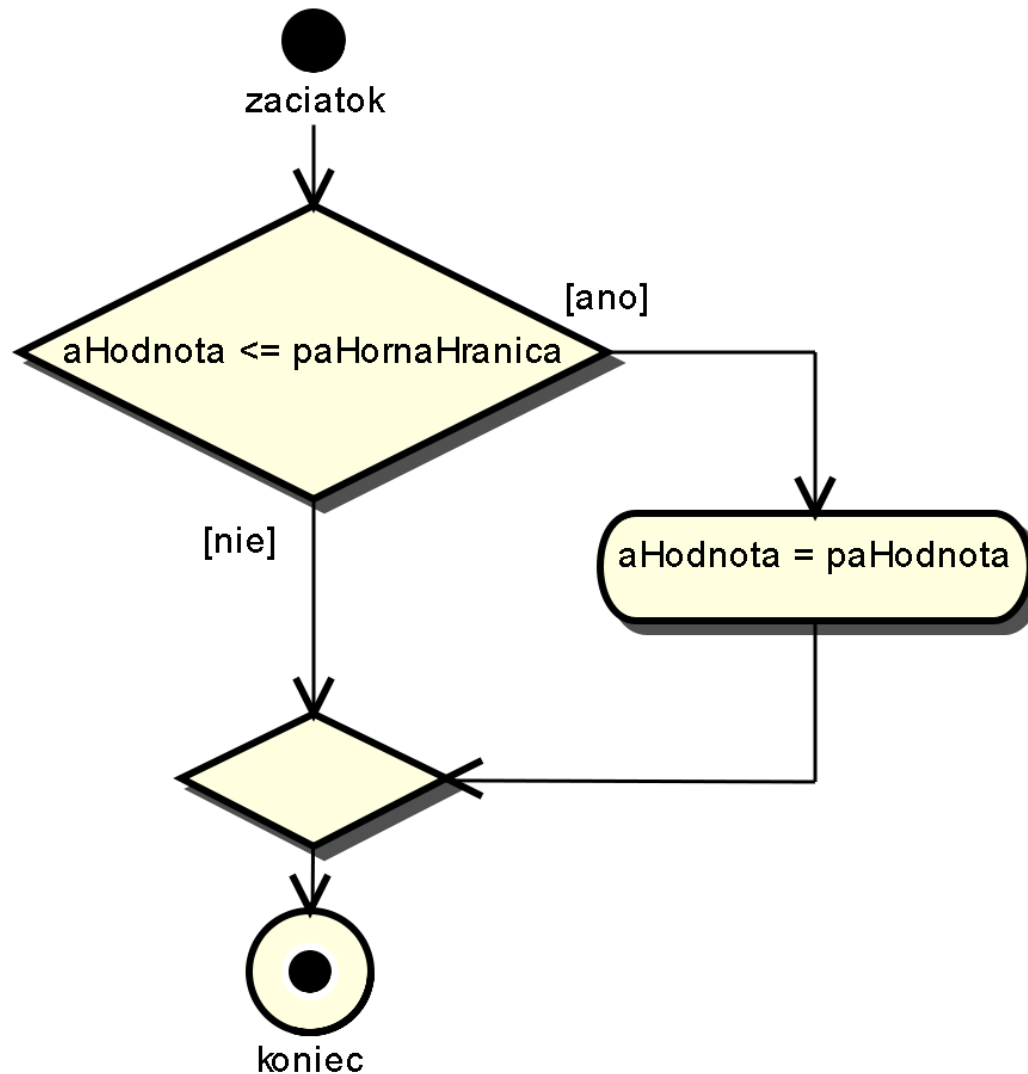
Zložená podmienka



Zložená podmienka – prvá časť



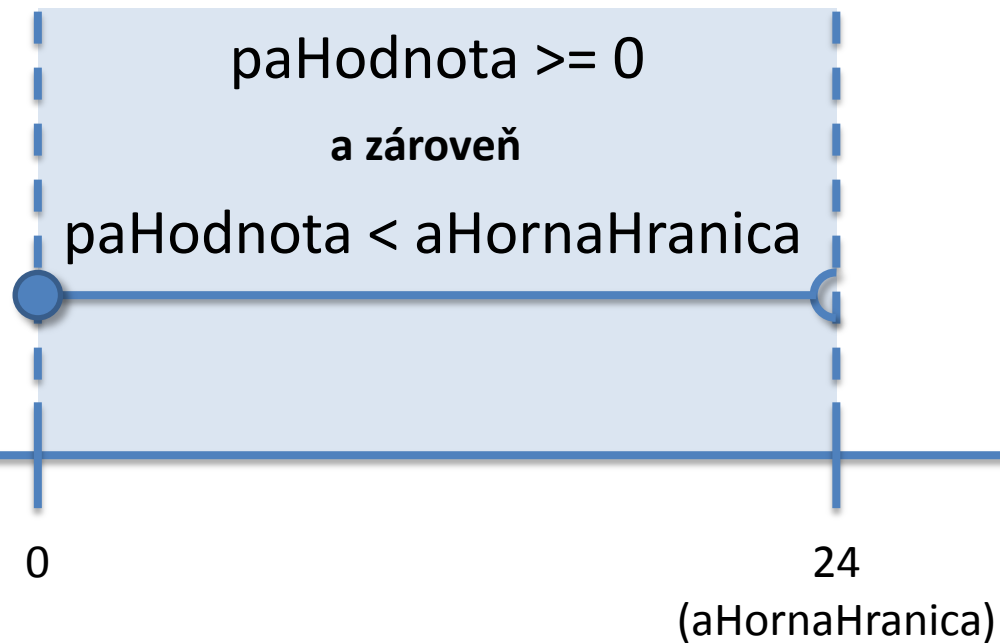
Zložená podmienka – druhá časť



Metóda nastavHodnotu

```
public void nastavHodnotu(int paHodnota)
{
    if (paHodnota >= 0) {
        if (paHodnota < aHornaHranica) {
            aHodnota = paHodnota;
        }
    }
}
```

Zložená podmienka



Logické operátory₍₁₎

- matematické formy
 - $x \in \langle a, b \rangle$
 - $a \leq x \leq b$
 - $a \leq x \wedge a \leq b$
 - $a \leq x$ a súčasne $b \leq y$

Logické operátory₍₂₎

- programovací jazyk Java
 - $a \leq x$ a súčasne $b \leq y$

$a \leq x$ **&&** $x \leq b$

Metóda nastavHodnotu

```
public void nastavHodnotu(int paHodnota)
{
    if ((paHodnota >= && paHodnota < aHornaHranica)) {
        aHodnota = paHodnota;
    }
}
```

&&

Metóda nastavHodnotu

```
public void nastavHodnotu(int paHodnota)
{
    boolean vyhovuje = (paHodnota >= 0)
                        && (paHodnota < aHornaHranica);
    if (vyhovuje) {
        aHodnota = paHodnota;
    }
}
```

Logické operátory

operácia	názov	matematika	Java
negácia	not	\bar{a}	!a
logický súčin	and	$a \wedge b$	a && b
logický súčet	or	$a \vee b$	a b

Priorita logických operátorov

priorita	operátory
najvyššia	unárne +, -, !
	*, /, %
	binárne +, -
	<, <=, >, >=
	==, !=
	<u>&&</u>
najnižšia	<u> </u>

Použitie logických operátorov

- unárny operátor !

operátor operand

- binárne operátory && a ||

prvyOperand operátor druhyOperand

- operandy – vždy logická, typ boolean
- hodnota logického výrazu – logická, typ boolean

Pravdivostné tabuľky

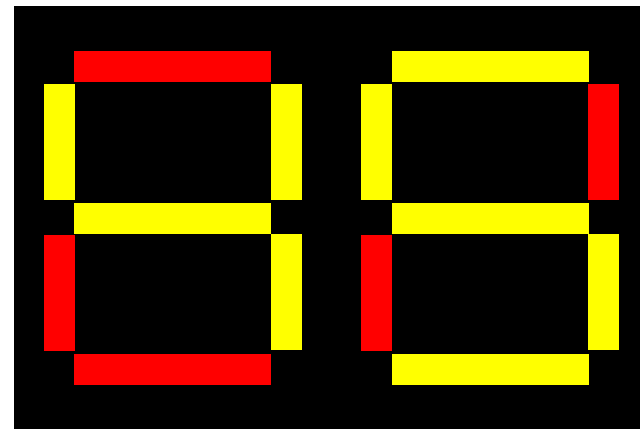
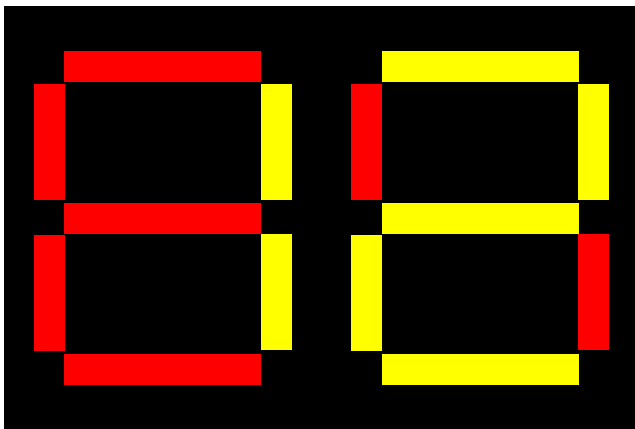
&& (and)	false	true
false	false	false
true	false	true

 (or)	false	true
false	false	true
true	true	true

! (not)	false	true
	true	false

Úloha

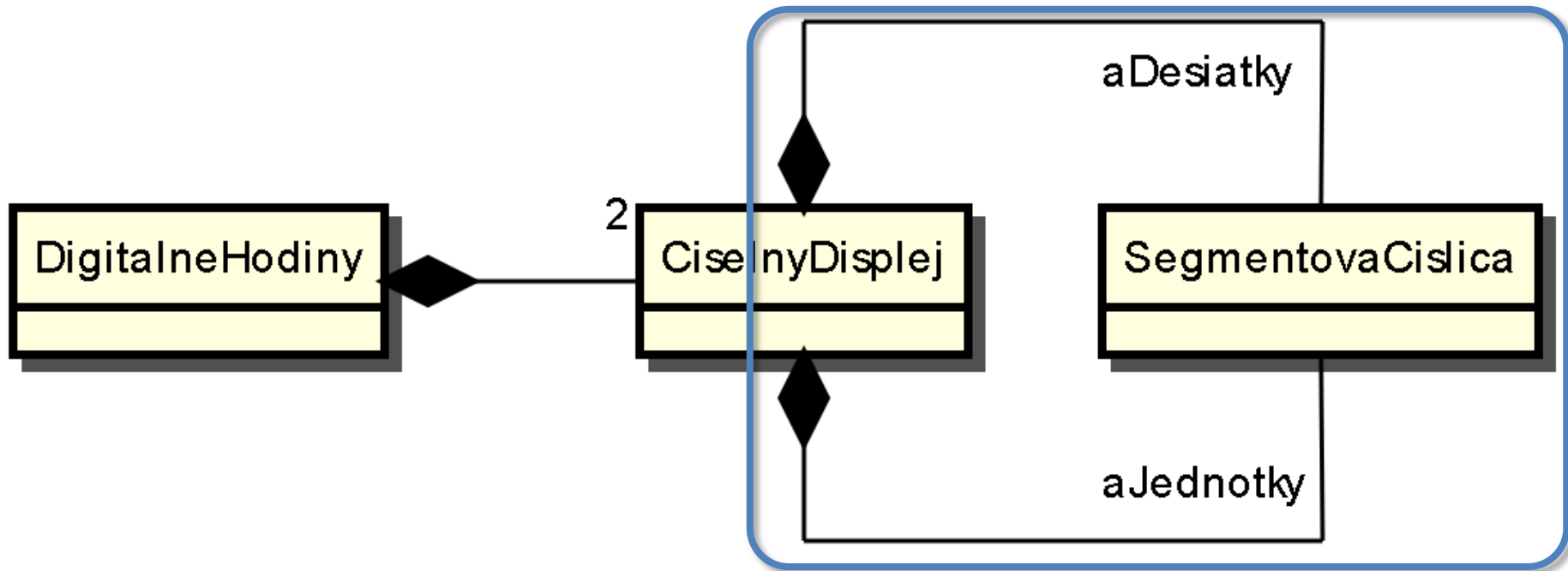
- zobrazit hodiny pomocou sedem segmentového displeja



Riešenie – prvá možnosť

- kompozícia – skladanie
 - SegmentovaCislica pre desiatky
 - SegmentovaCislica pre jednotky
- nutnosť určenia pozície
 - každá číslica je na inej pozícii
 - každý číselný displej je na inej pozícii

Segmentová číslica – skladanie



CiselnyDisplej – rozhranie

CiselnyDisplej

- + new(paHornaHranica: int, paX: int, paY: int): CiselnyDisplej
- + dajHodnotu(): int
- + nastavHodnotu(paHodnota: int): void
- + dajHodnotuAkoRetazec(): String
- + krok(): void

CiselnyDisplej – vnútorný pohľad

CiselnyDisplej

- aHornaHranica: int
- aHodnota: int
- aDesiatky: SegmentovaCislica
- aJednotky: SegmentovaCislica

- + CiselnyDisplej(paHornaHranica: int, paX: int, paY: int)
- + dajHodnotu(): int
- + nastavHodnotu(paHodnota: int): void
- + dajHodnotuAkoRetazec(): String
- + krok(): void

CiselnyDisplej – konštruktor

```
public CiselnyDisplej(int paHornaHranica,  
                    int paX, int paY)  
{  
    aHornaHranica = paHornaHranica;  
    aHodnota = 0;  
    aDesiatky = new SegmentovaCislica(paX, paY);  
  
    int posunX = 120 + 10;  
    aJednotky = new SegmentovaCislica(  
        paX + posunX, paY);  
}
```

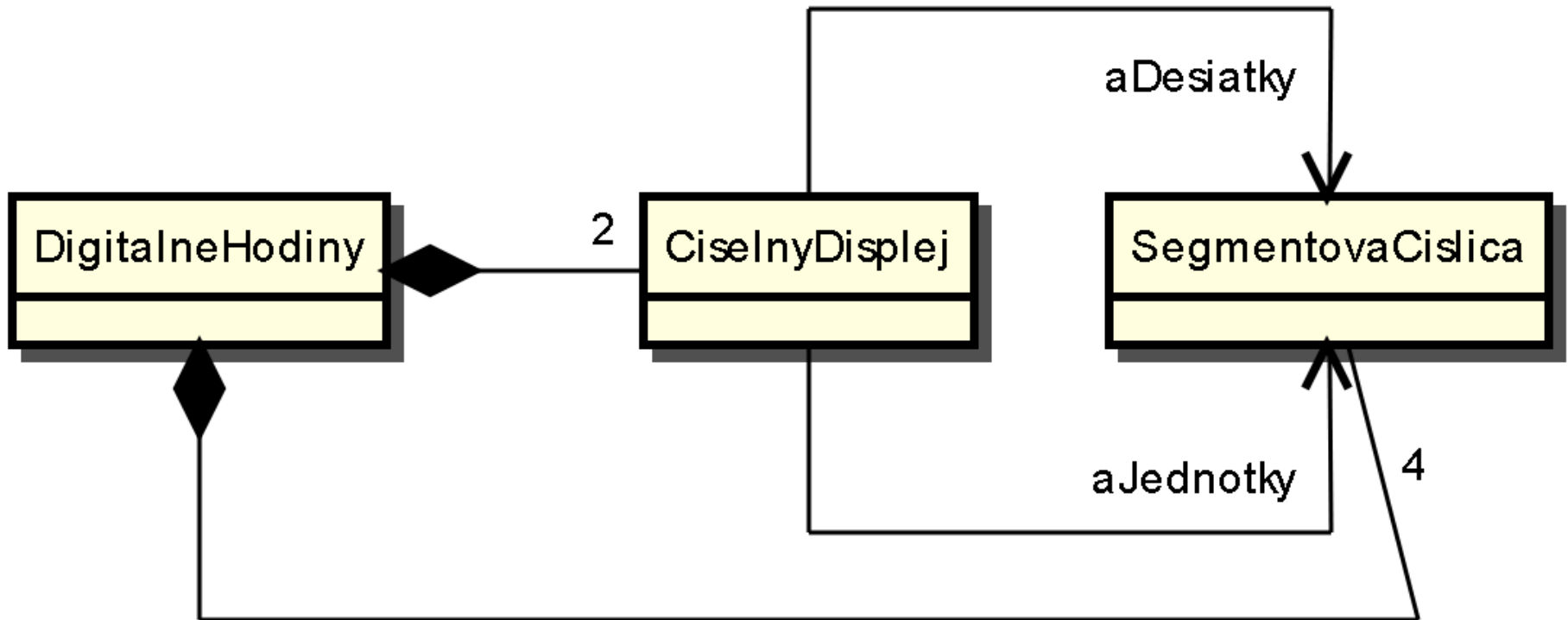
CiselnyDisplej – krok

```
public void krok()  
{  
    aHodnota = (aHodnota + 1) % aHornaHranica;  
    aDesiatky.zobraz(aHodnota / 10);  
    aJednotky.zobraz(aHodnota % 10);  
}
```

Riešenie – druhá možnosť

- asociácia – spolupráca
 - inštanciu SegmentovaCislica vytvárajú DigitalneHodiny
 - CiselnyDisplej spolupracuje so SegmentovaCislica
 - CiselnyDisplej dostane číslicu ako parameter konštruktora
- pozíciu číslic určujú hodiny

SegmentovaCislica – asociácia



CiselnyDisplej – rozhranie

CiselnyDisplej

```
+ new(paHhranica: int, paDesiatky: Cislica, paJednotky: Cislica): CiselnyDisplej  
+ dajHodnotu(): int  
+ nastavHodnotu(paHodnota: int): void  
+ dajHodnotuAkoRetazec(): String  
+ krok(): void
```

CiselnyDisplej – vnútorný pohľad

CiselnyDisplej

- aHodnota: int
- aHornaHranica: int
- aDesiatky: SegmentovaCislica
- aJednotky: SegmentovaCislica

- + CiselnyDisplej(paHHranica: int, paDesiatky: Cislica, paJednotky: Cislica)
- + dajHodnotu(): int
- + nastavHodnotu(paHodnota: int): void
- + dajHodnotuAkoRetazec(): String
- + krok(): void

CiselnyDisplej – konstruktor

```
public CiselnyDisplej(int paHornaHodnota,  
                    Cislica paDesiatky, Cislica paJednotky)  
{  
    aHornaHranica = paHornaHranica;  
    aHodnota = 0;  
    aDesiatky = paDesiatky;  
    aJednotky = paJednotky;  
}
```

CiselnyDisplej – krok

```
public void krok()  
{  
    aHodnota = (aHodnota + 1) % aHornaHranica;  
    if (aDesiatky != null && aJednotky != null) {  
        aDesiatky.zobraz(aHodnota / 10);  
        aJednotky.zobraz(aHodnota % 10);  
    }  
}
```

Hodnota null

- null – objektový literál
- null = referenciu neodkazuje na žiadny objekt
- pre referenciu na inštanciu ľubovoľnej triedy

Relačné výrazy s objektmi₍₁₎

- relačné operátory pre čísla
 - < (menšie ako), <= (menšie, alebo rovné),
> (väčšie ako), >= (väčšie, alebo rovné), == (rovné),
!= (nerovné)
- relačné operátory pre objekty
 - == (rovné), != (nerovné)
- porovnanie referencií na objekty

```
referencia1 == referencia2
```

Relačné výrazy s objektmi₍₂₎

- == porovnanie identity dvoch objektov
 - true – dve referencie na ten istý objekt
 - false – dve referencie na dva rôzne objekty

- != opak operátora ==

Porovnanie hodnôt objektov

- porovnanie hodnôt/stavov dvojice objektov?
- porovnanie dvojice rovnakého typu
- porovnanie zhody ich stavov
- špeciálna správa – metóda equals

Porovnávanie reťazcov – String

- správa equals
- testuje zhodnosť dvoch reťazcov
- hlavička metódy

```
public boolean equals(String paRetazec)
```

- príklad:

```
System.out.println("Zilinska".equals("univerzita"));
```

String – „==“ a equals₍₁₎

- dve rôzne referencie na ten istý reťazec
- dva rôzne reťazce – rovnaký obsah
- relačný operátor „==“
 - == – porovnanie referencií na reťazce
 - true – referencie na ten istý reťazec
- správa equals
 - equals – porovnanie obsahov reťazcov
 - true – rovnaký obsah dvoch reťazcov

String – „==“ a equals₍₂₎

- reťazec1 == reťazec2 – true
- \Rightarrow reťazec1.equals(reťazec2) – true
- POZOR: opačná implikácia neplatí
- ~~• reťazec1.equals(reťazec2) – true~~
- ~~• \Rightarrow reťazec1 == reťazec2 – true~~

String – „==“ a equals₍₃₎

```
String nazov = "Zilinska univerzita";  
// "zilinska univerzita"  
String nazovA = nazov.toLowerCase();  
// "zilinska univerzita"  
String nazovB = nazov.toLowerCase();
```

- dva rôzne objekty s rovnakým stavom
 - nazovA.equals(nazovB) – true
 - nazovA == nazovB – false

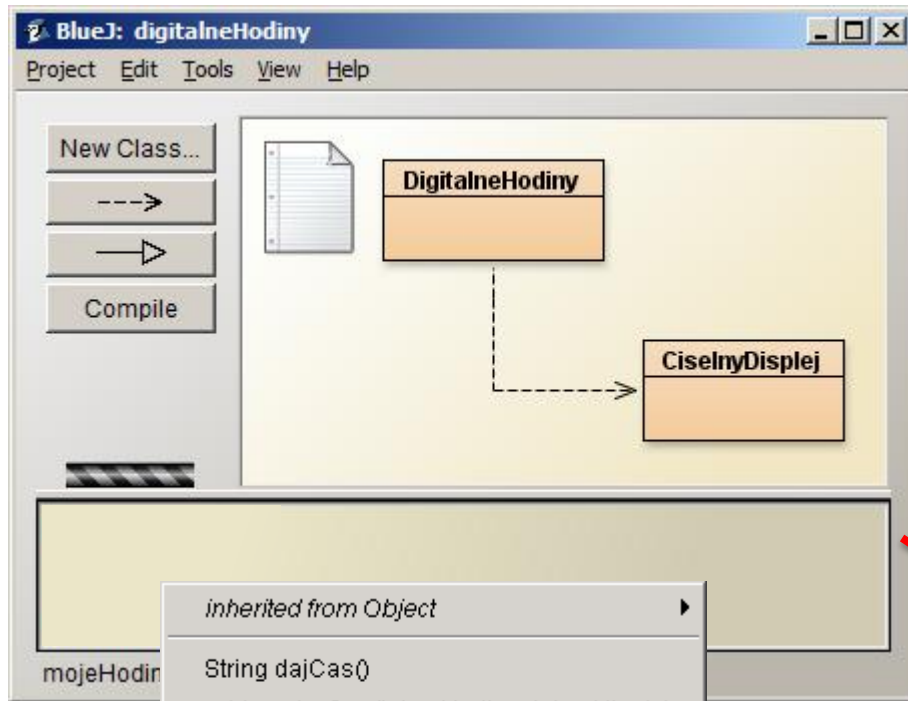
Zánik inštancie

- životný cyklus inštancie
 - vznik inštancie – špeciálna správa „new“ triede
 - poskytovanie služieb – prijímanie správ a reakcie na ne
 - zánik inštancie – ???

Zánik inštancie

- o zrušenie inštancie sa stará zberná služba – garbage collector
- kedy?
 - zberná služba ruší objekt v prípade, že naň neexistuje žiadna referencia
 - null = referencia neodkazuje na žiadny objekt

Zánik objektov pri kompozícii



mojeHodinky

- inherited from Object
- String dajCas()
- void nastavCas(int paHodina, int paMinuta)
- void tik()
- Inspect
- Remove

mojeHodinky : DigitalneHodiny

private CiselnyDisplej aHodiny 24

private CiselnyDisplej aMinuty 0

Show static fields

Inspect

Get

Close

aHodiny : CiselnyDisplej

private int aHodnaHranica 24

private int aHodnota 0

Show static fields

Inspect

Get

Close

aMinuty : CiselnyDisplej

private int aHodnaHranica 60

private int aHodnota 0

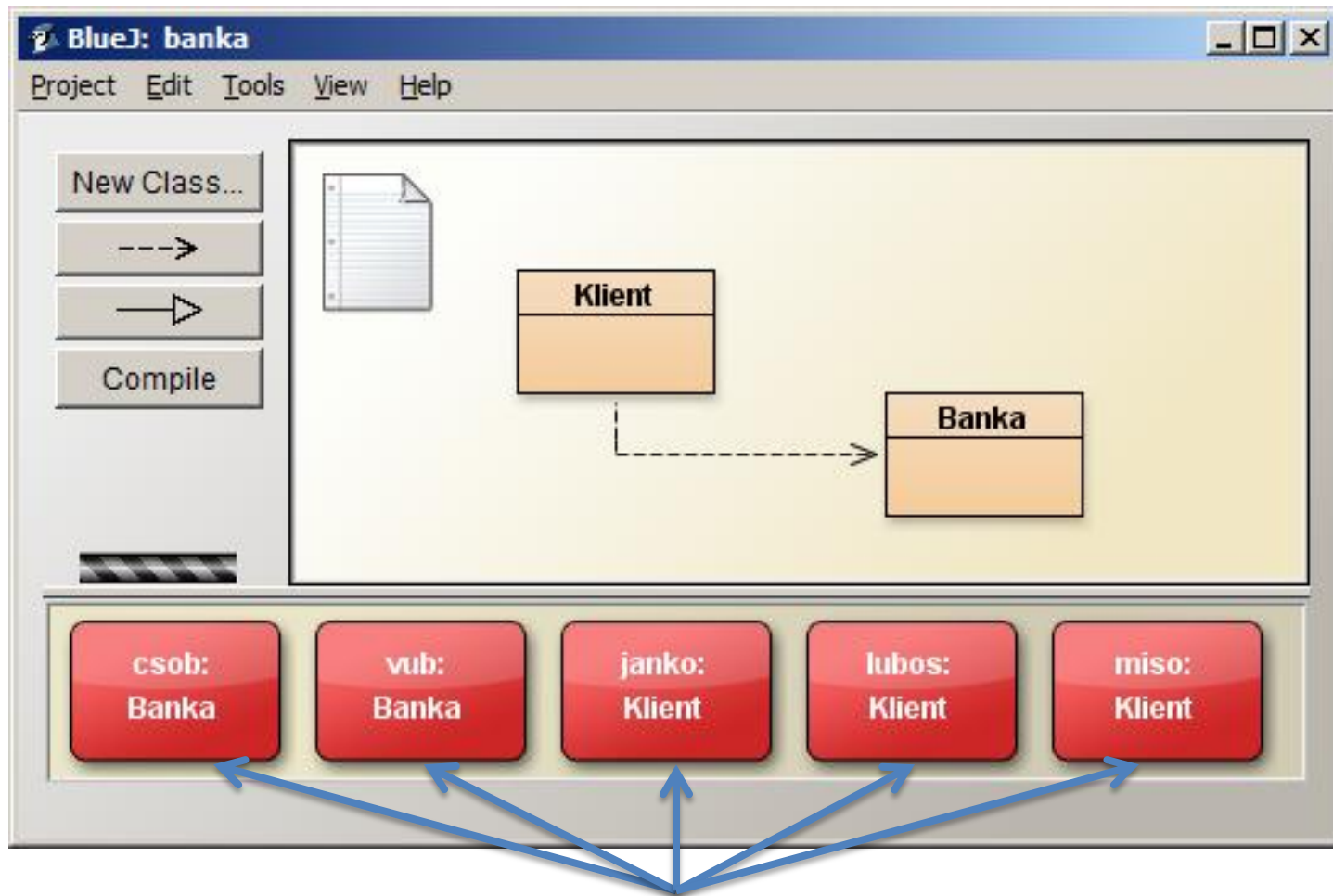
Show static fields

Inspect

Get

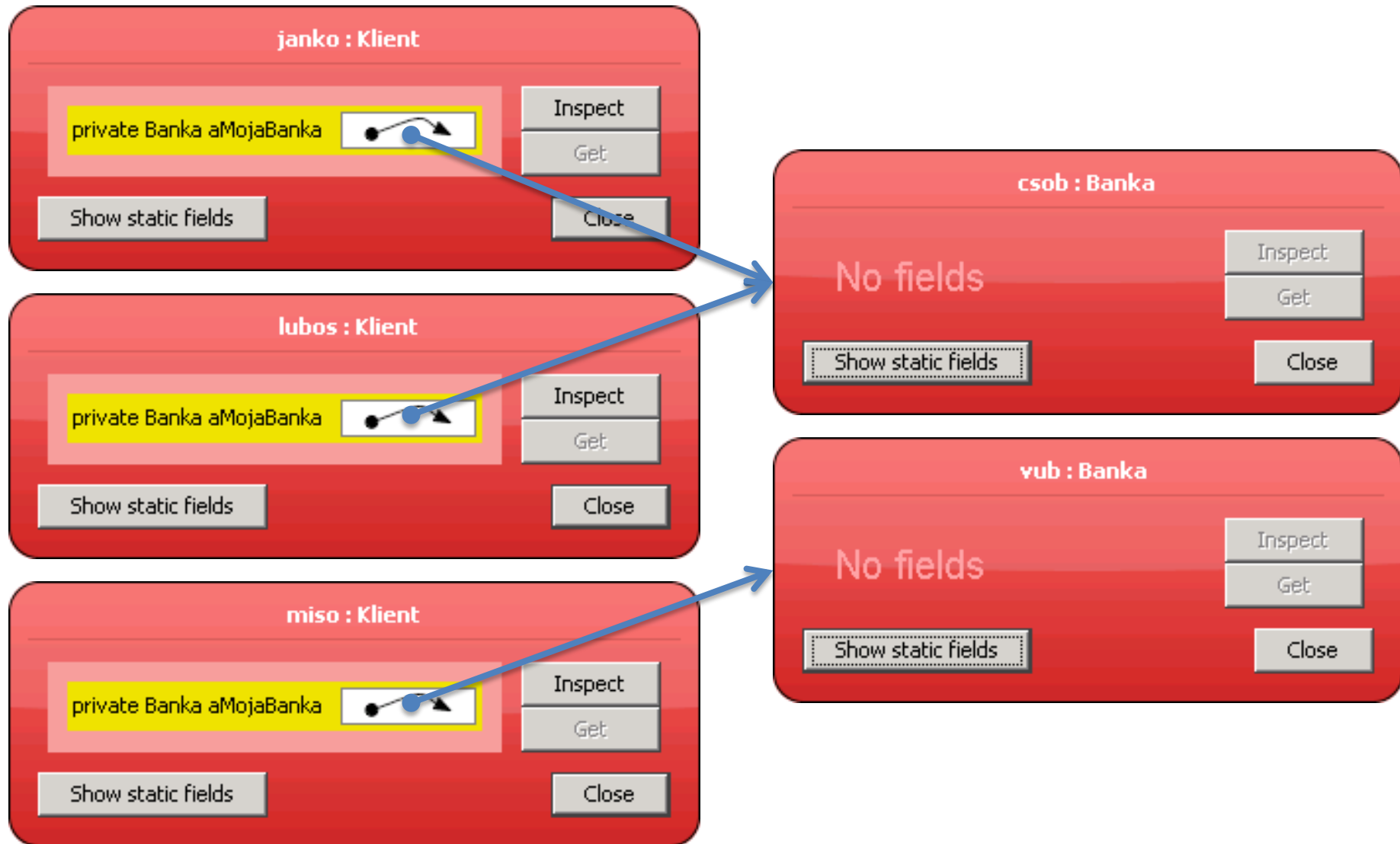
Close

Referencie v BlueJ

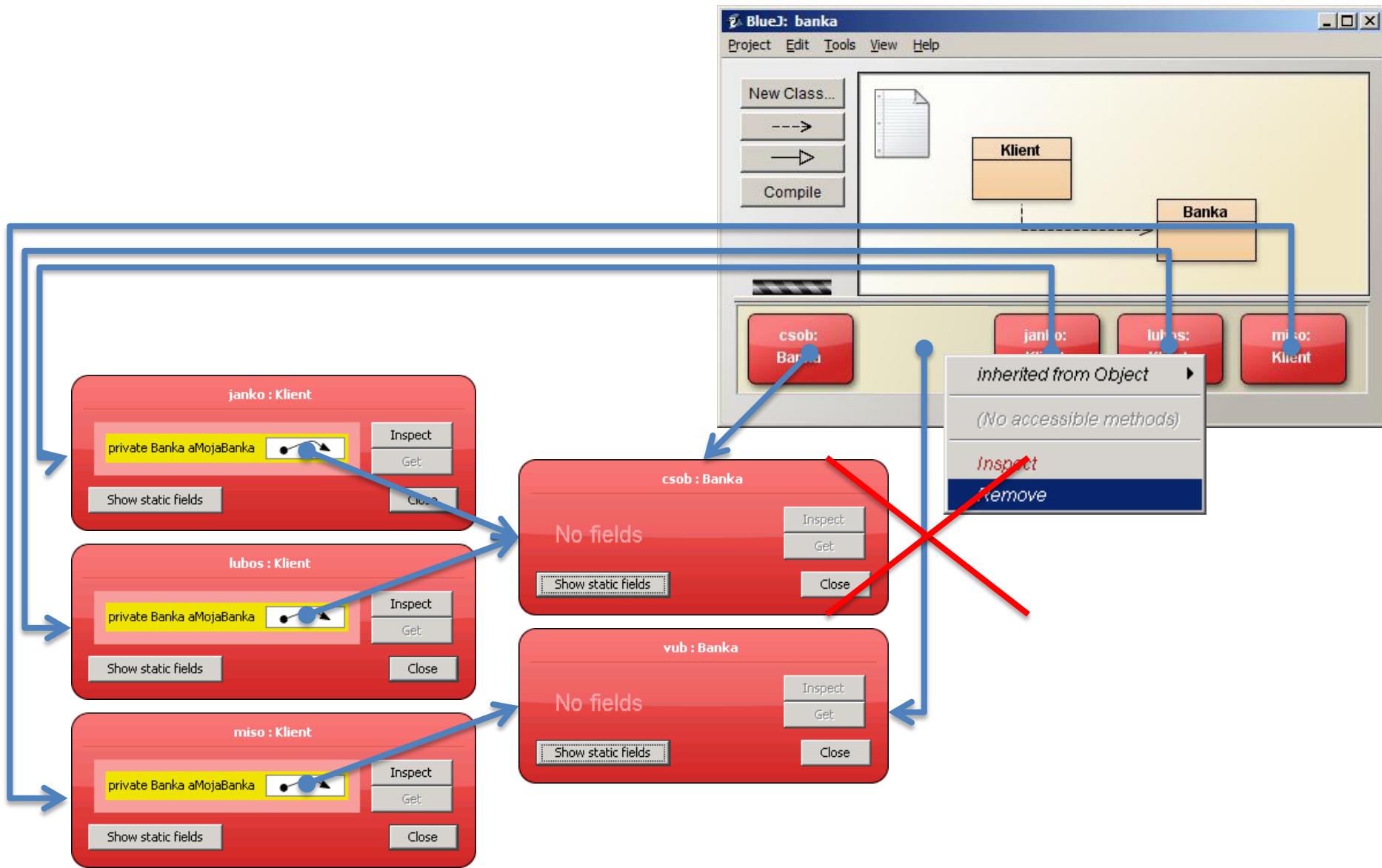


referencia

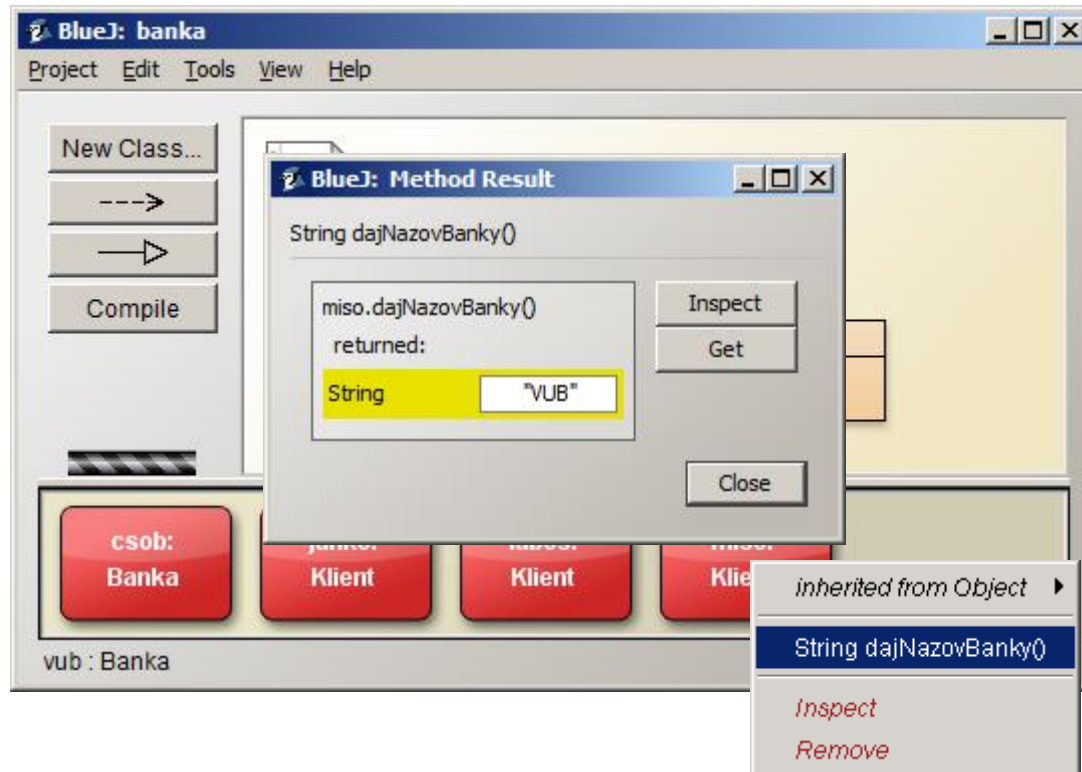
Objektový diagram príkladu



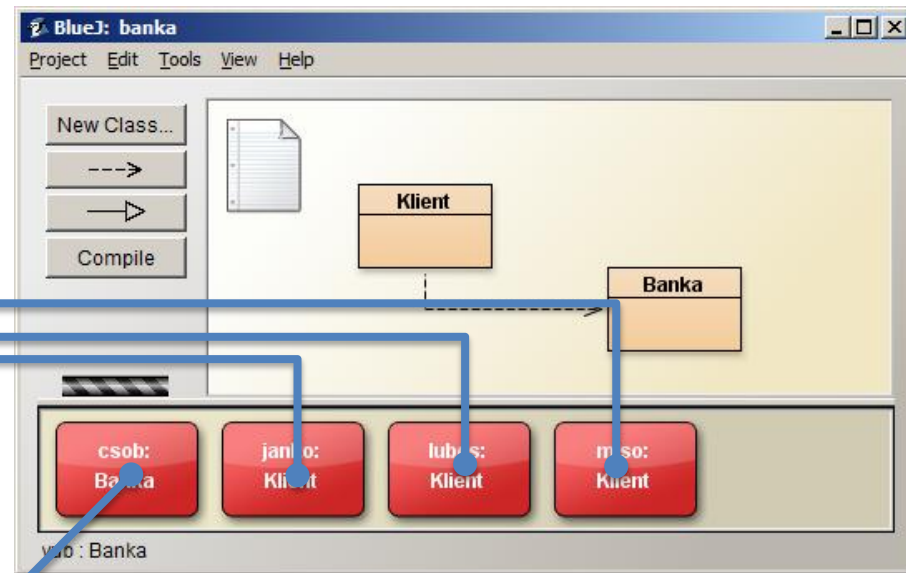
Zánik objektov pri asociácii (1)



Zánik objektov pri asociácii⁽²⁾



Zánik objektov pri asociácii (2)



janko : Klient

private Banka aMojaBanka

Inspect Get Close

Show static fields

lubos : Klient

private Banka aMojaBanka

Inspect Get Close

Show static fields

miso : Klient

private Banka aMojaBanka

Inspect Get Close

Show static fields

csob : Banka

No fields

Inspect Get Close

Show static fields

vub : Banka

No fields

Inspect Get Close

Show static fields

hodnota null

Hodnota null – úlohy₍₁₎

- null = referencia neodkazuje na žiadny objekt
- inicializácia objektovej premennej
- zánik inštancie
 - priradovací príkaz
 - premenna = null;

Hodnota null – úlohy₍₂₎

- adresát nie je určený
 - podmienka pre poslanie správy
 - premenna != null

Vďaka za pozornosť