

5

Spolupráca objektov

Pojmy zavedené v 4. prednáške₍₁₎

- modularizácia a abstrakcia
- kompozícia – skladanie objektov
 - spoločné životné cykly
 - UML

Pojmy zavedené v 4. prednáške₍₂₎

- objektové typy
- referencie

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

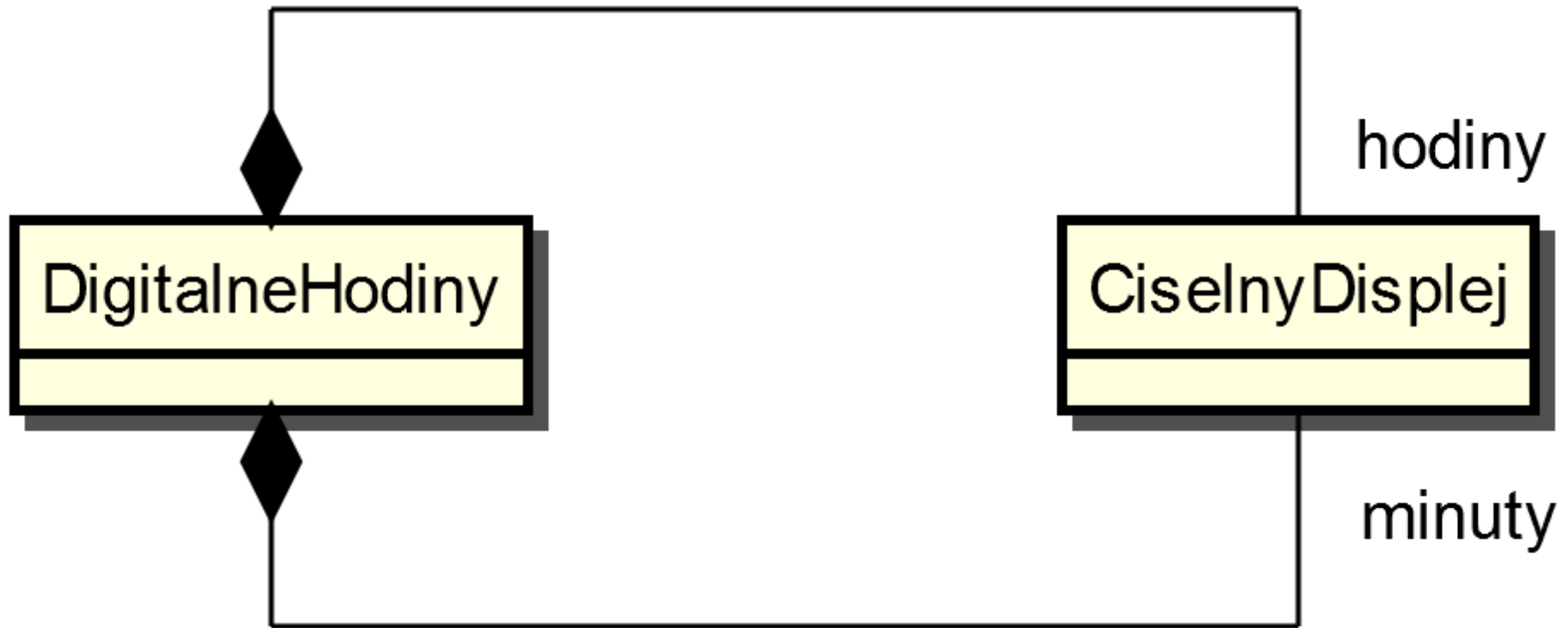
Pojmy zavedené v 4. prednáške₍₄₎

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

Cieľ prednášky

- logické výrazy
- relačné výrazy s objektmi
- spolupráca objektov – asociácia
- príklad: Digitálne hodiny

Digitálne hodiny – verzia 1



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
 - začiatkový čas: 00:00

DigitalneHodiny – vnútorný pohľad

DigitalneHodiny

- minuty: CiselyDisplej
- hodiny: CiselyDisplej

- + DigitalneHodiny()
- + tik(): void
- + setCas(hodiny: int, minuty: int): void
- + getCas(): String

CiselnyDisplej – vnútorný pohľad

CiselnyDisplej

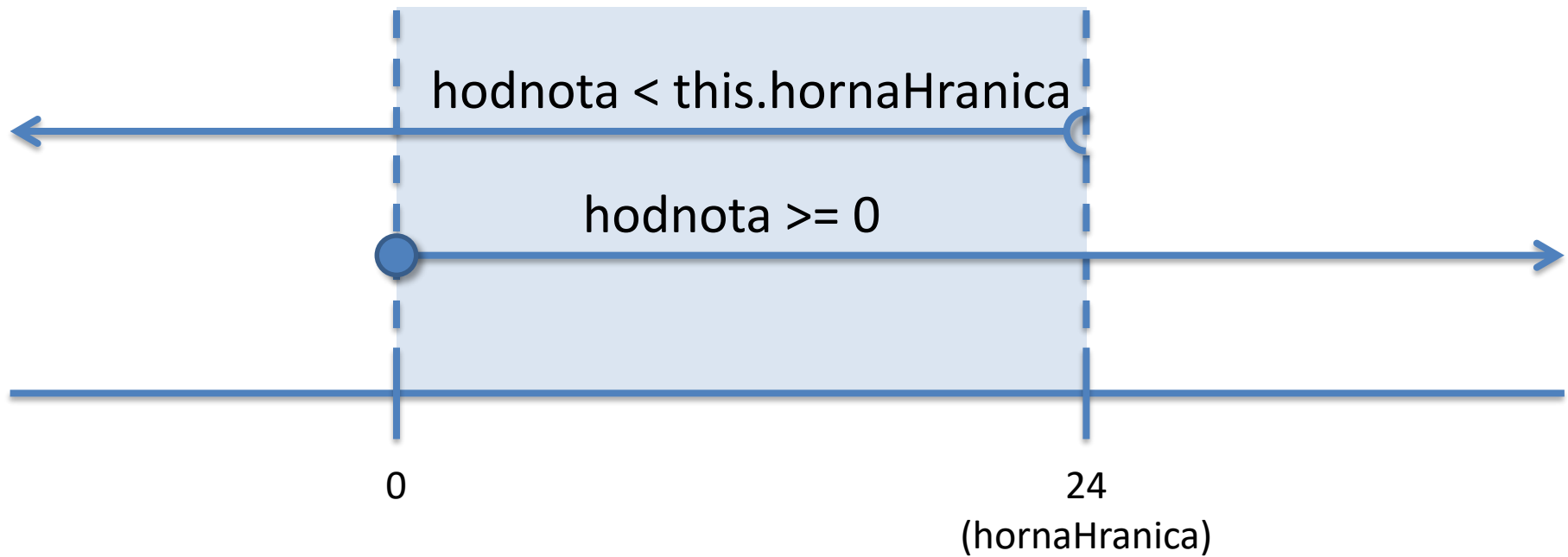
- hodnota: int
- hornaHranica: int

- + CiselnyDisplej(hornaHranica: int)
- + getHodnota(): int
- + setHodnota(hodnota: int): void
- + getHodnotaAkoRetazec(): String
- + krok(): void

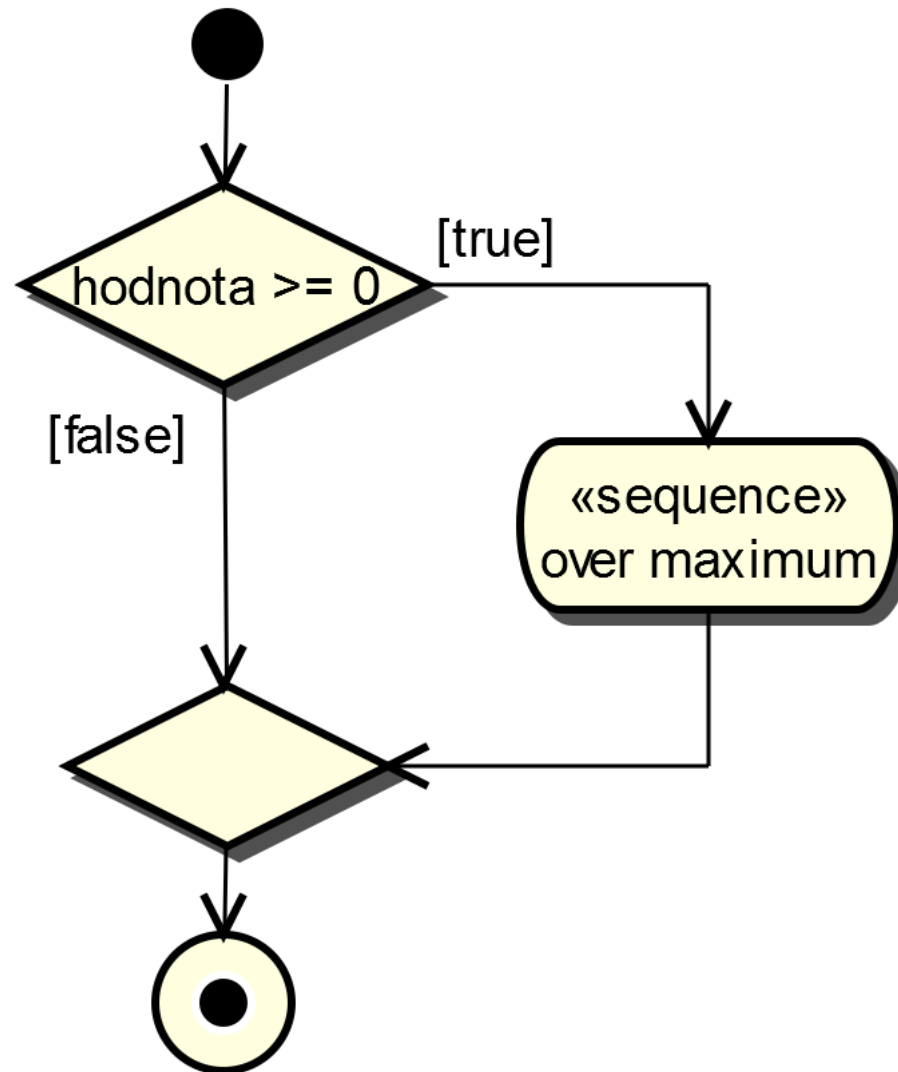
Metóda setHodnota

- zmení hodnotu na displeji
- nesmie nastaviť hodnotu mimo rozsah
 - min 0
 - max `this.hornaHranica`

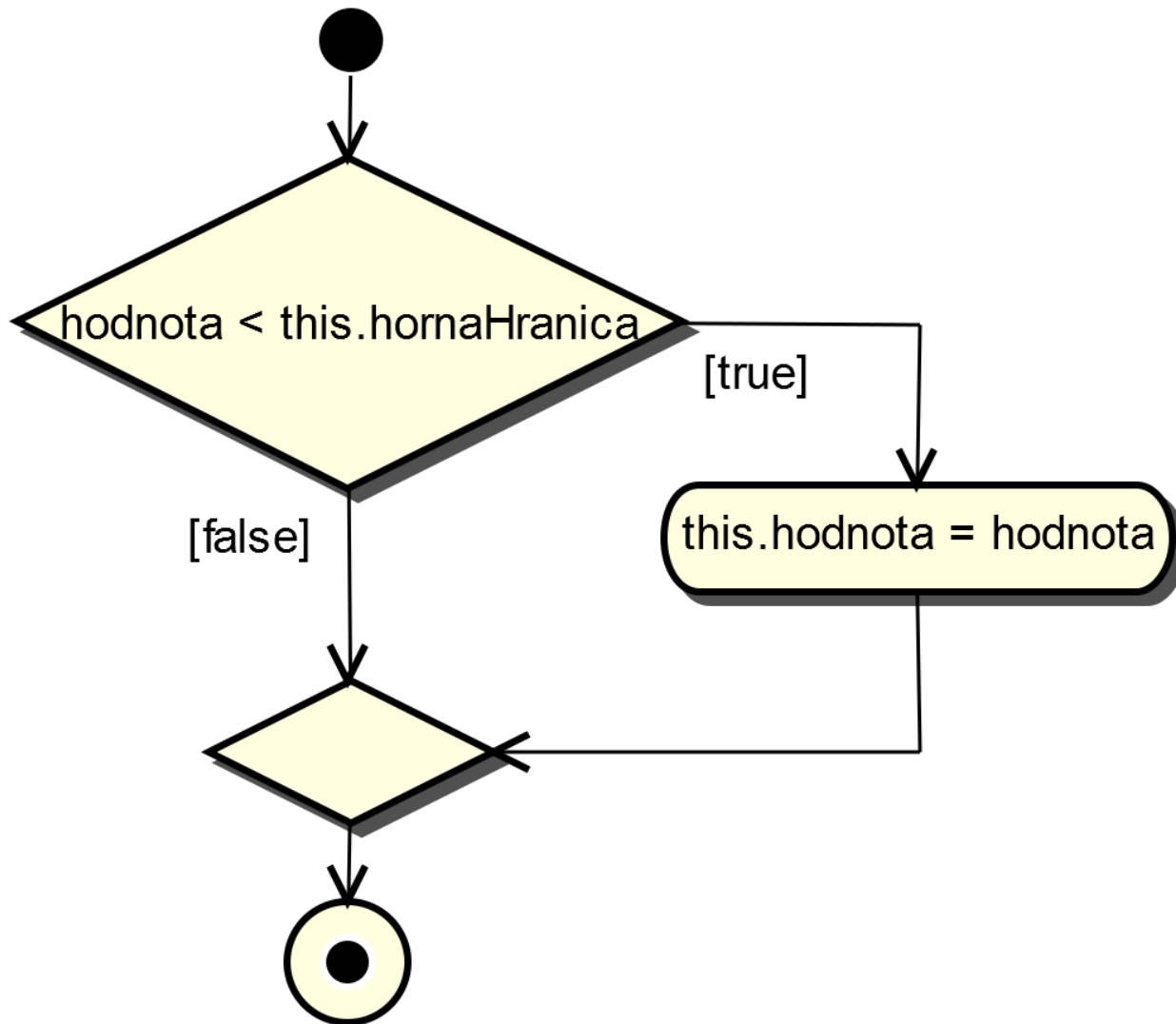
Zložená podmienka



Zložená podmienka – prvá časť



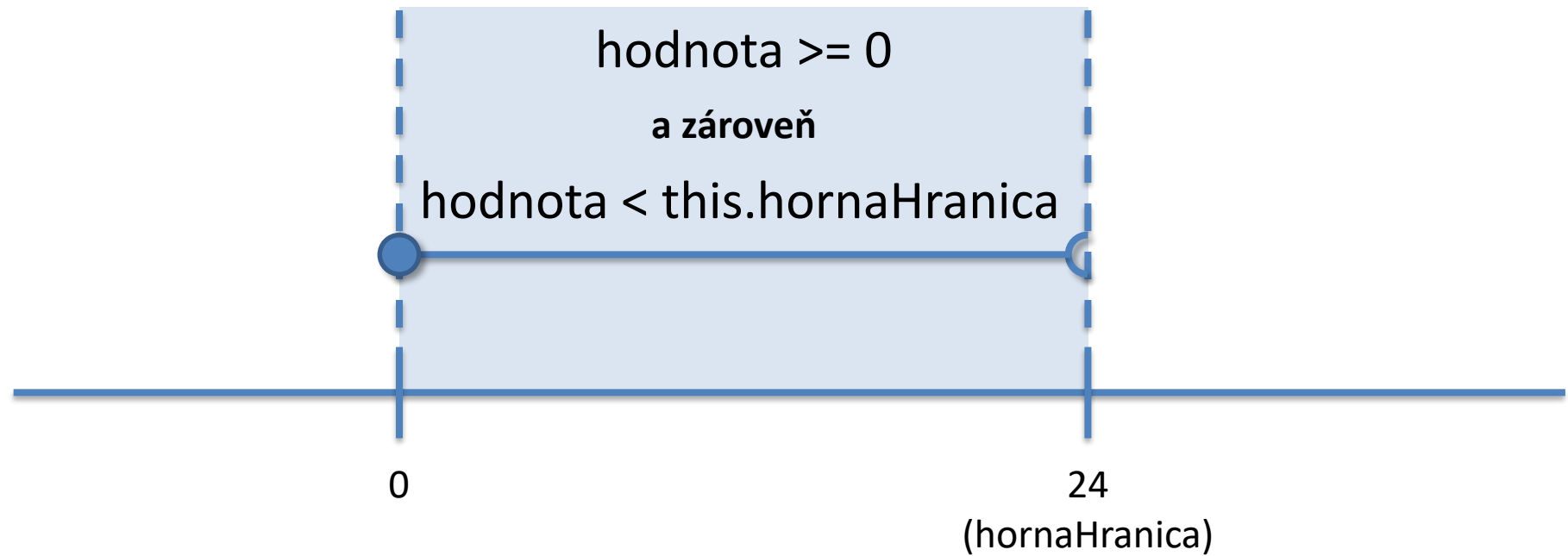
Zložená podmienka – druhá časť



Metóda setHodnota

```
public void setHodnota(int hodnota) {  
    if (hodnota >= 0) {  
        if (hodnota < this.hornaHranica) {  
            this.hodnota = hodnota;  
        }  
    }  
}
```

Zložená podmienka



Logické operátory₍₁₎

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

Logické operátory₍₂₎

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

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

Metóda setHodnota

```
public void setHodnota(int hodnota) {  
    if ((hodnota >= 0) && (hodnota < this.hornaHranica)) {  
        this.hodnota = hodnota;  
    }  
}
```

&&

Logické operátory

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

Priorita logických operátorov

priorita	operátory
najvyššia	unárne +, -, <u>!</u>
	*, /, %
	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

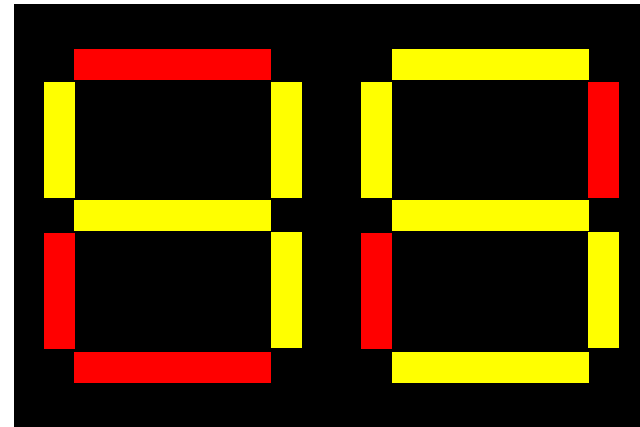
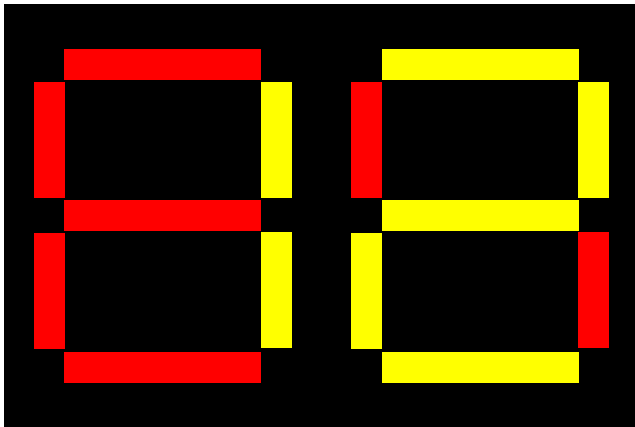
&& (and)	false	true
false	false	false
true	false	true

(or)	false	true
false	false	true
true	true	true

! (not)	false	true
	true	false

Úloha

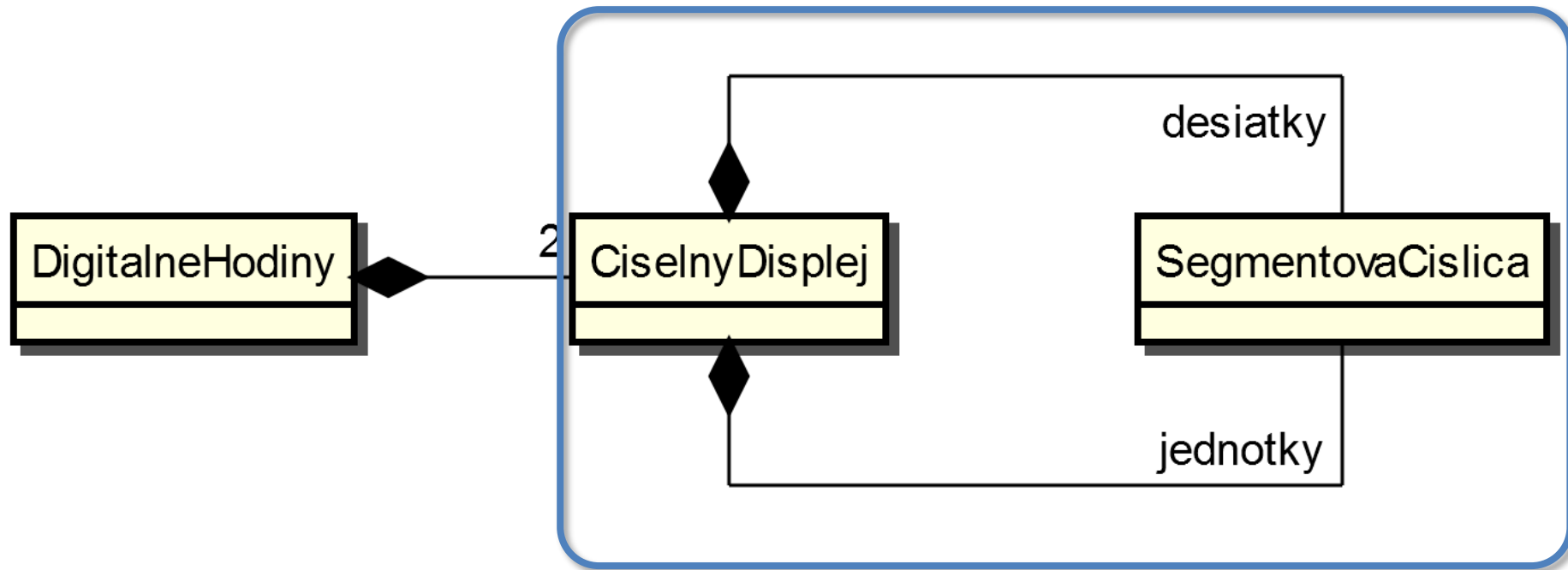
- zobrazit hodiny pomocou sedem segmentového displeja



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 – vnútorný pohľad

CiselnyDisplej

- hodnota: int
- hornaHranica: int
- desiatky: SegmentovaCislica
- jednotky: SegmentovaCislica

- + CiselnyDisplej(hornaHranica: int, x: int, y: int)
- + getHodnota(): int
- + setHodnota(hodnota: int): void
- + getHodnotaAkoRetazec(): String
- + krok(): void

CiselnyDisplej – konštruktor

```
public CiselnyDisplej(int hornaHranica, int x, int y) {  
    this.hornaHranica = hornaHranica;  
    this.hodnota = 0;  
    this.desiatky = new SegmentovaCislica(x, y);  
    this.jednotky = new SegmentovaCislica(x + 40, y);  
}
```

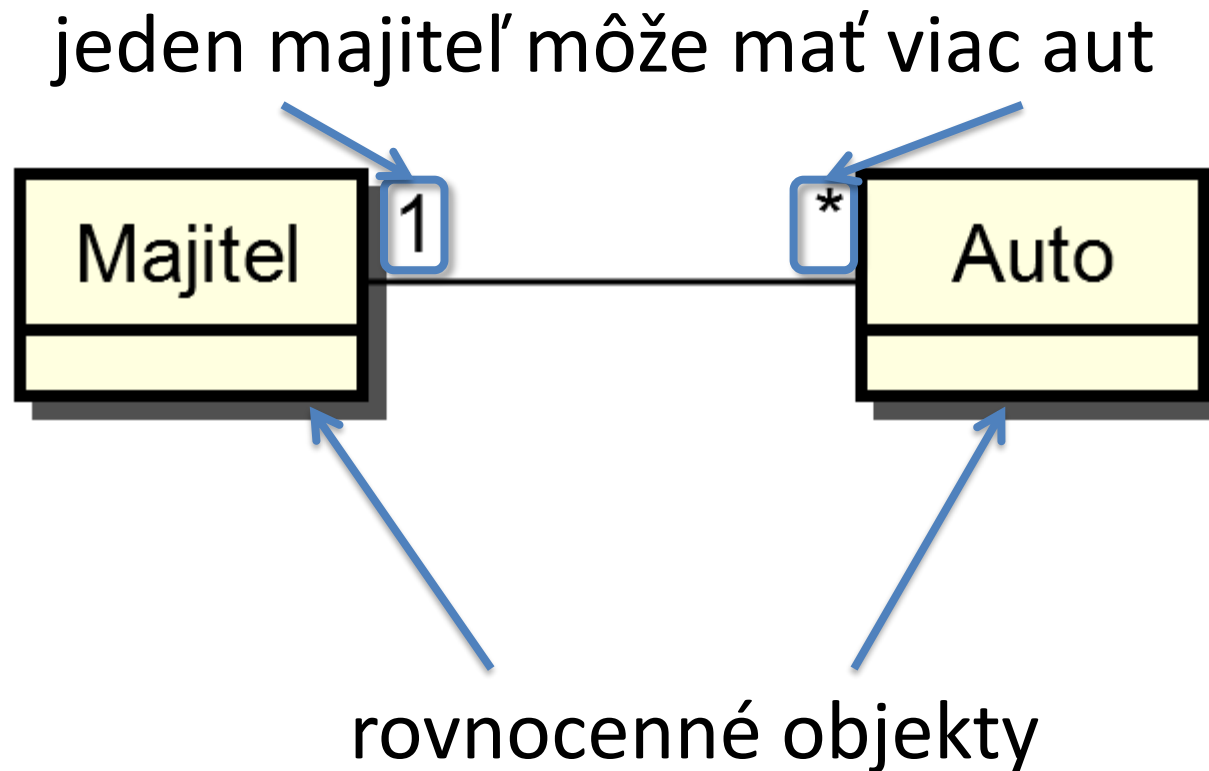
CiselnyDisplej – krok

```
public void krok() {  
    this.hodnota = (this.hodnota + 1)  
                    % this.hornaHranica;  
    this.desiatky.zobraz(this.hodnota / 10);  
    this.jednotky.zobraz(this.hodnota % 10);  
}
```

Asociácia – spolupráca objektov

- druhá možnosť implementácie
- asociácia – ľubovoľná spolupráca dvoch objektov
 - príklady: klient a banka, učiteľ a študent
- charakteristika asociácie – nezávislé životné cykly oboch objektov

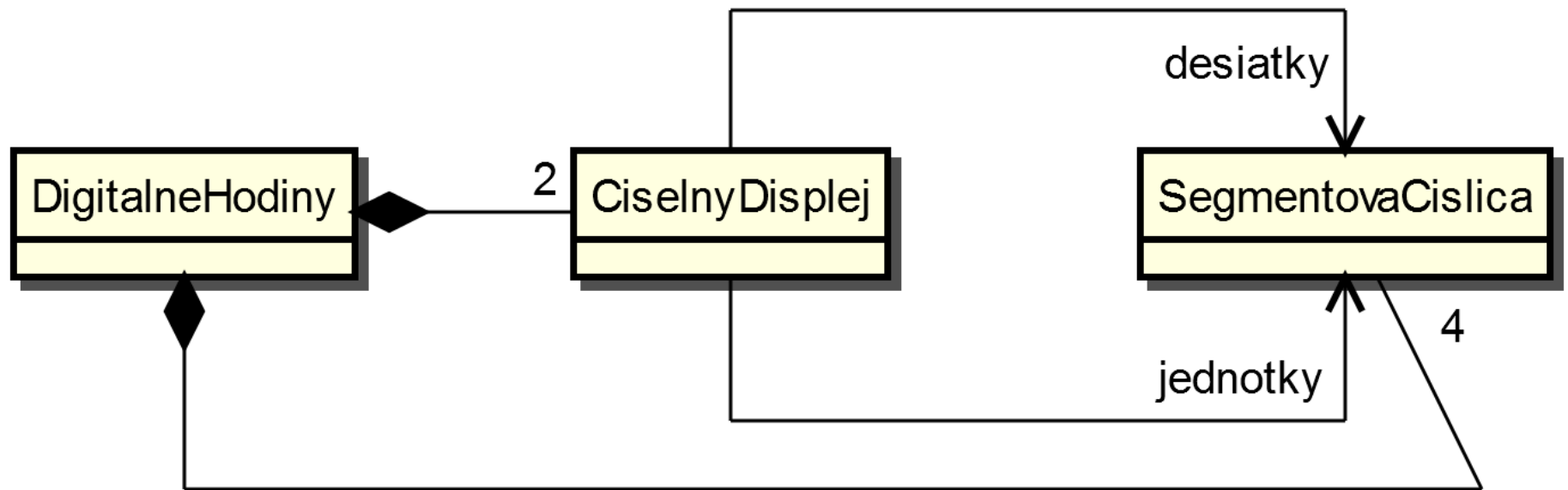
Asociácia v UML



Riešenie zobrazenia asociáciou

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

SegmentovaCislica – asociácia



CiselnyDisplej – vnútorný pohľad

CiselnyDisplej

- hodnota: int
- hornaHranica: int
- desiatky: SegmentovaCislica
- jednotky: SegmentovaCislica

- + CiselnyDisplej(hornaHranica: int, desiatky: SegmentovaCislica, jednotky: SegmentovaCislica)
- + getHodnota(): int
- + setHodnota(hodnota: int): void
- + getHodnotaAkoRetazec(): String
- + krok()

CiselnyDisplej – konstruktor

```
public CiselnyDisplej(int hornaHodnota,  
                    SegmentovaCislica desiatky,  
                    SegmentovaCislica jednotky) {  
    this.hornaHranica = hornaHranica;  
    this.hodnota = 0;  
    this.desiatky = desiatky;  
    this.jednotky = jednotky;  
}
```

CiselnyDisplej – krok

```
public void krok() {  
    this.hodnota = (this.hodnota + 1)  
                    % this.hornaHranica;  
    if (this.desiatky != null && this.jednotky != null) {  
        this.desiatky.zobraz(this.hodnota / 10);  
        this.jednotky.zobraz(this.hodnota % 10);  
    }  
}
```

Hodnota null

- null – objektový literál
- null = referencia 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
- POZOR: pre vlastné objekty nebude fungovať

Porovnávanie reťazcov – String

- správa equals
- testuje zhodnosť dvoch reťazcov
- hlavička metódy

```
public boolean equals(String retazec)
```

- 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

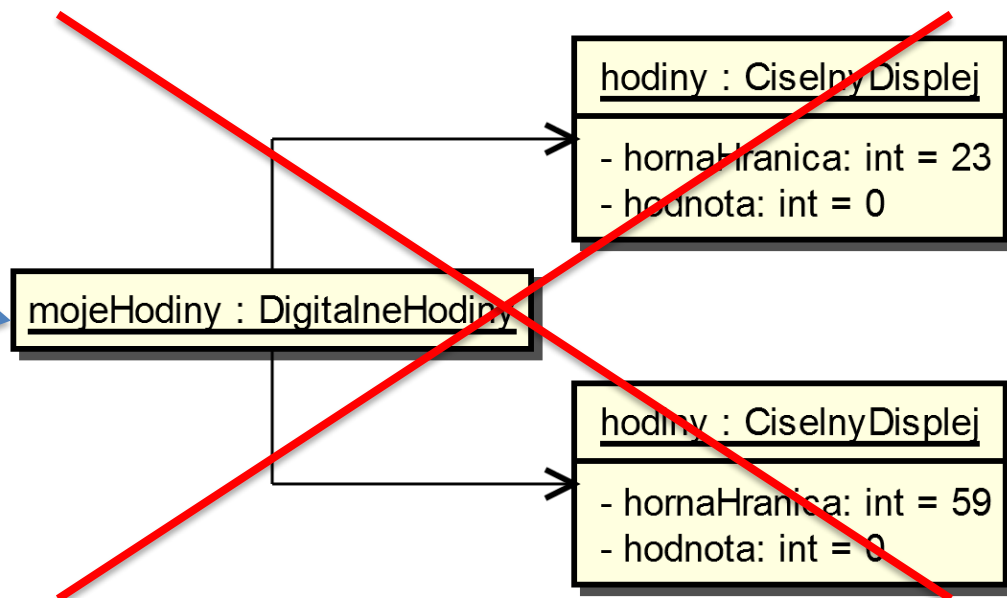
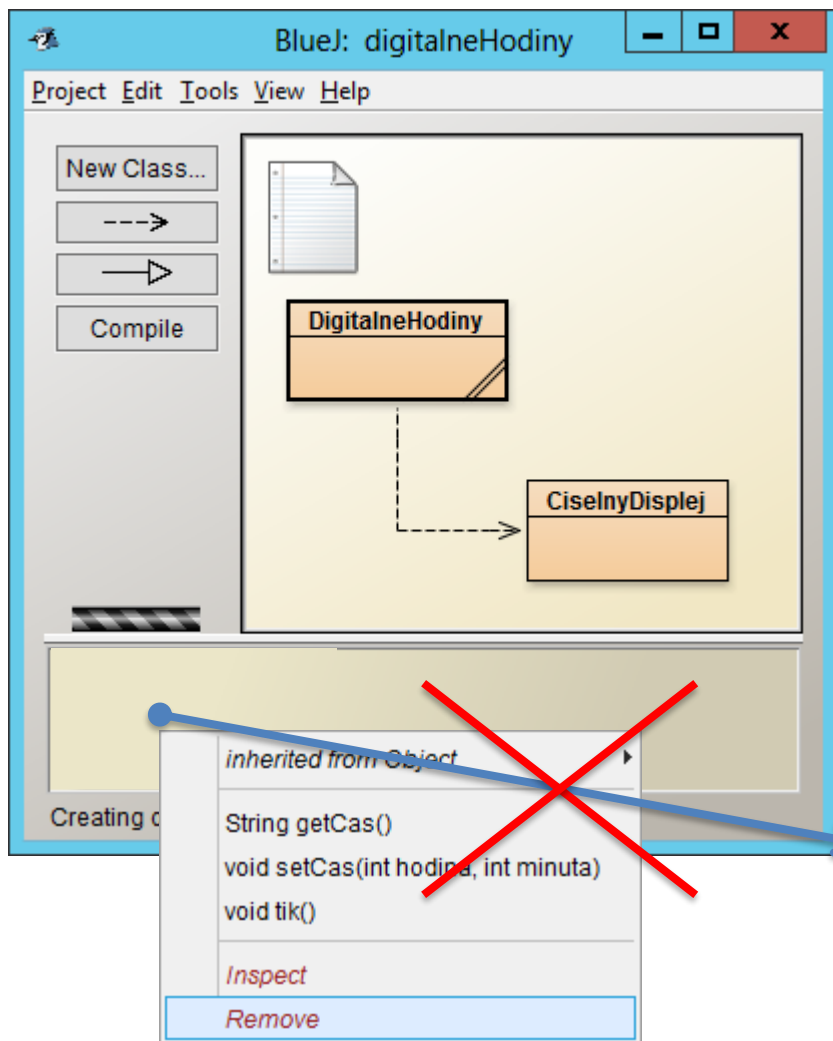
Zánik inštancie

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

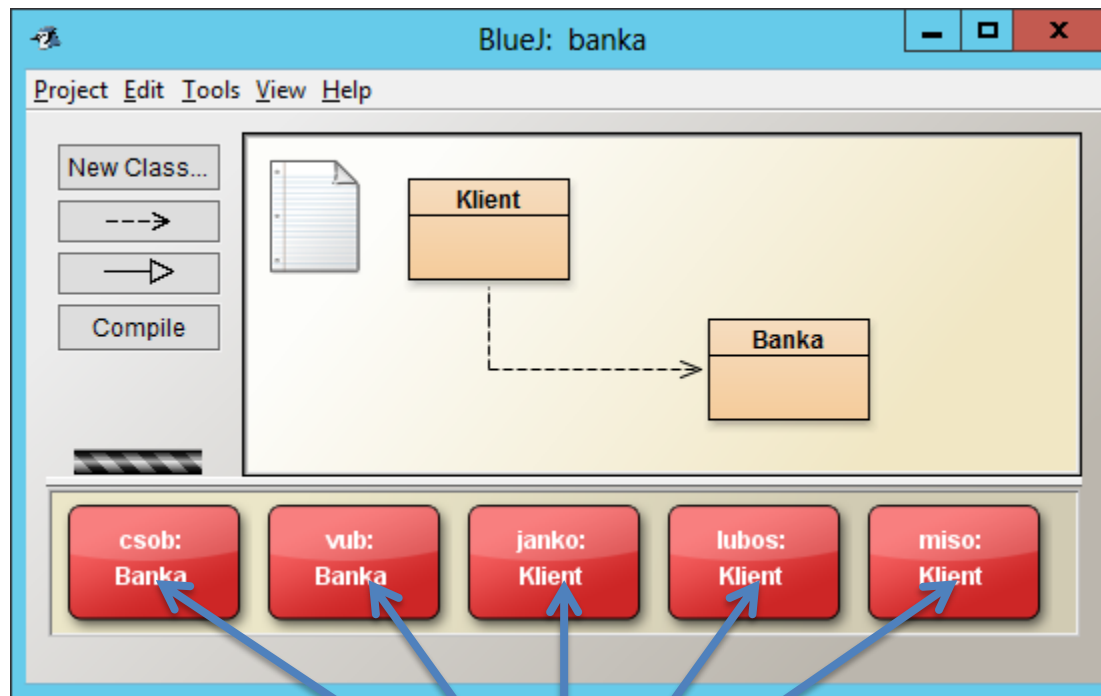
Zánik inštancie

- 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

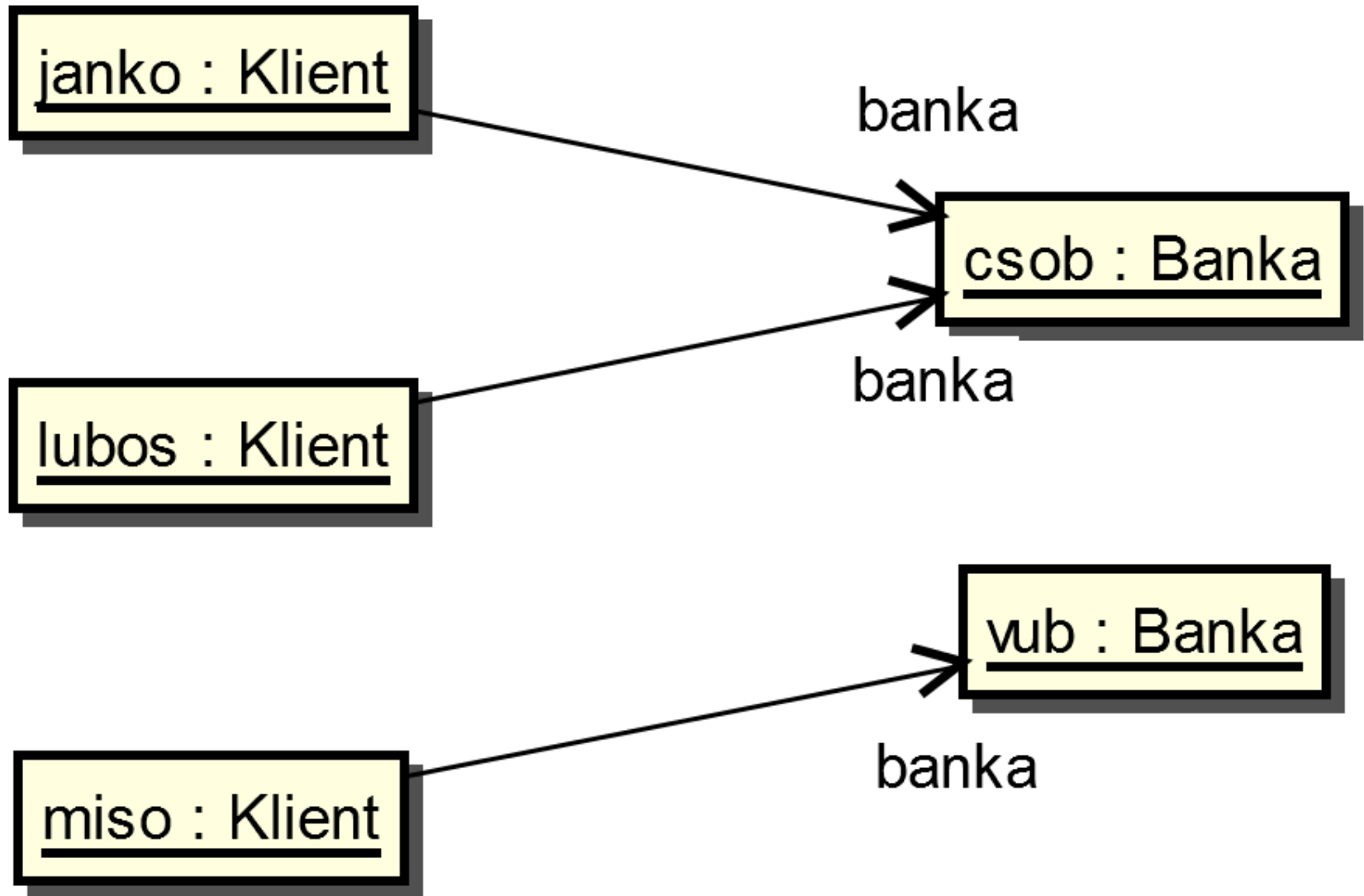


Referencie v BlueJ

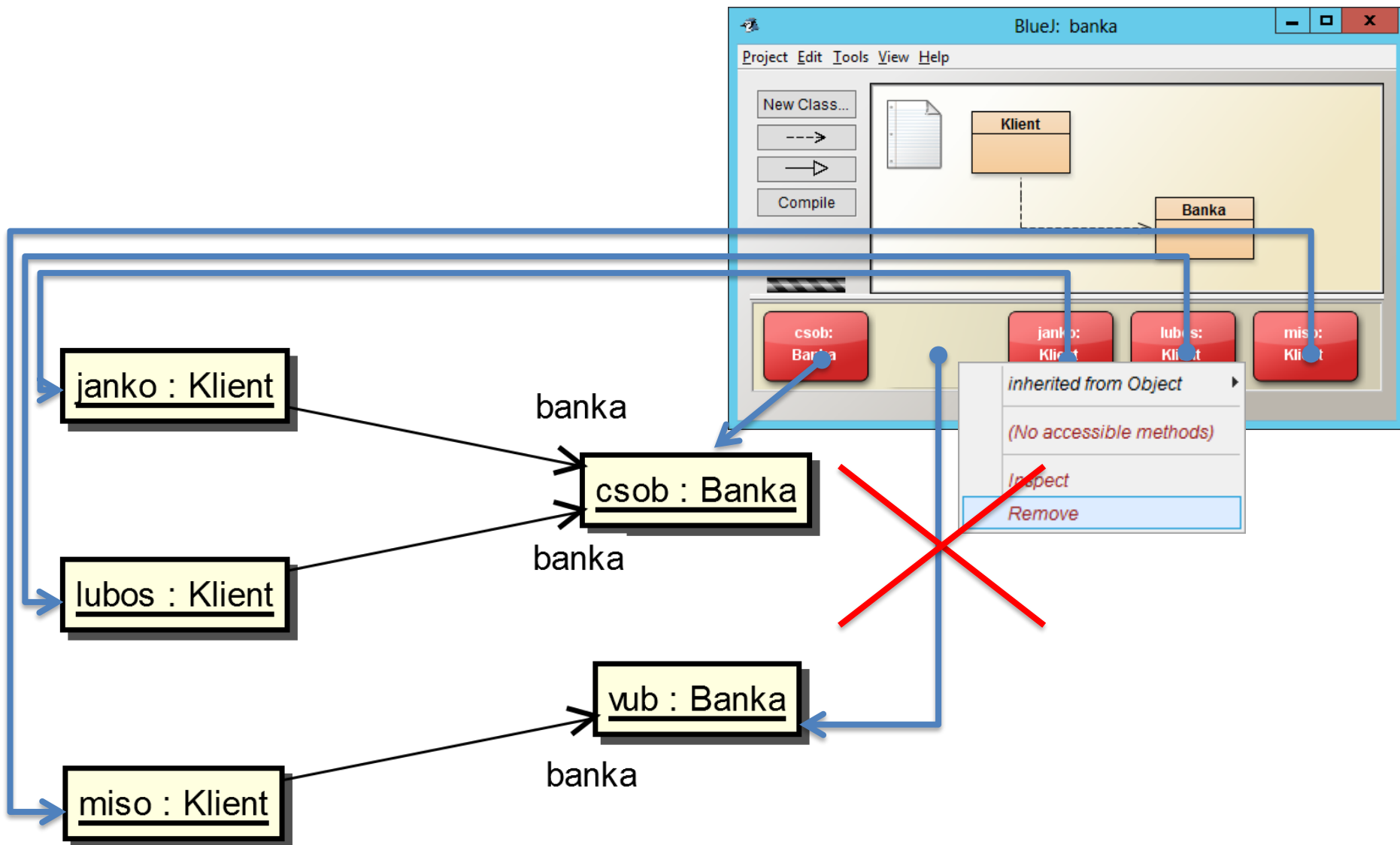


referencia

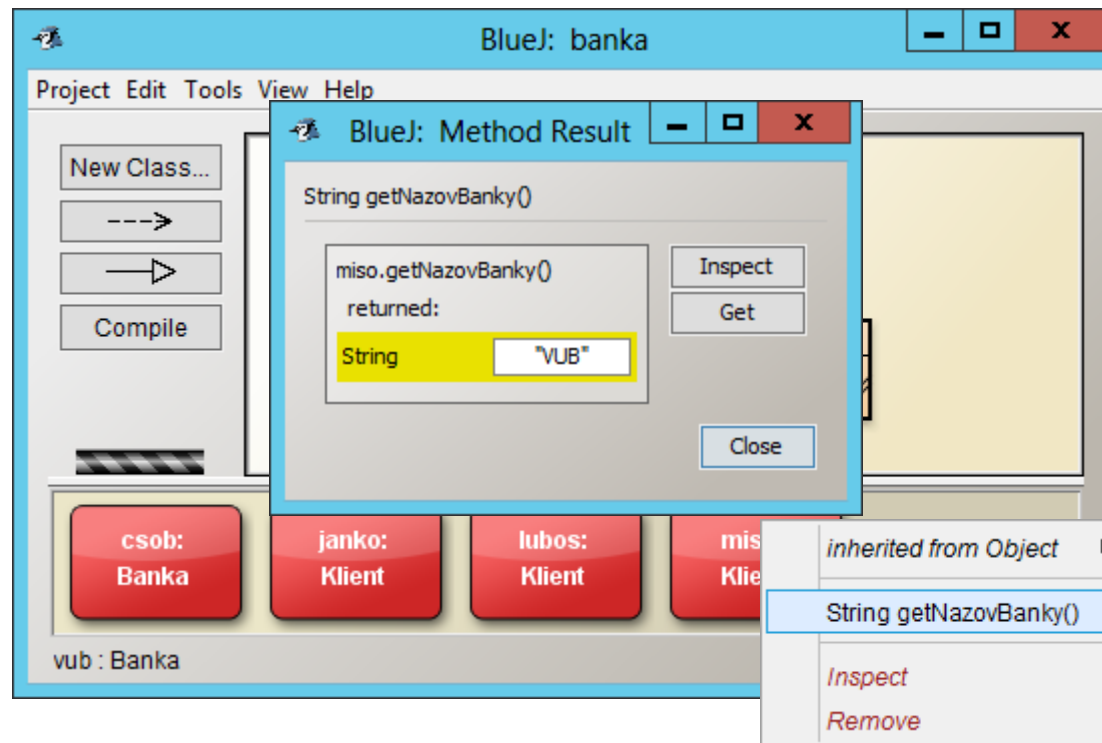
Objektový diagram príkladu



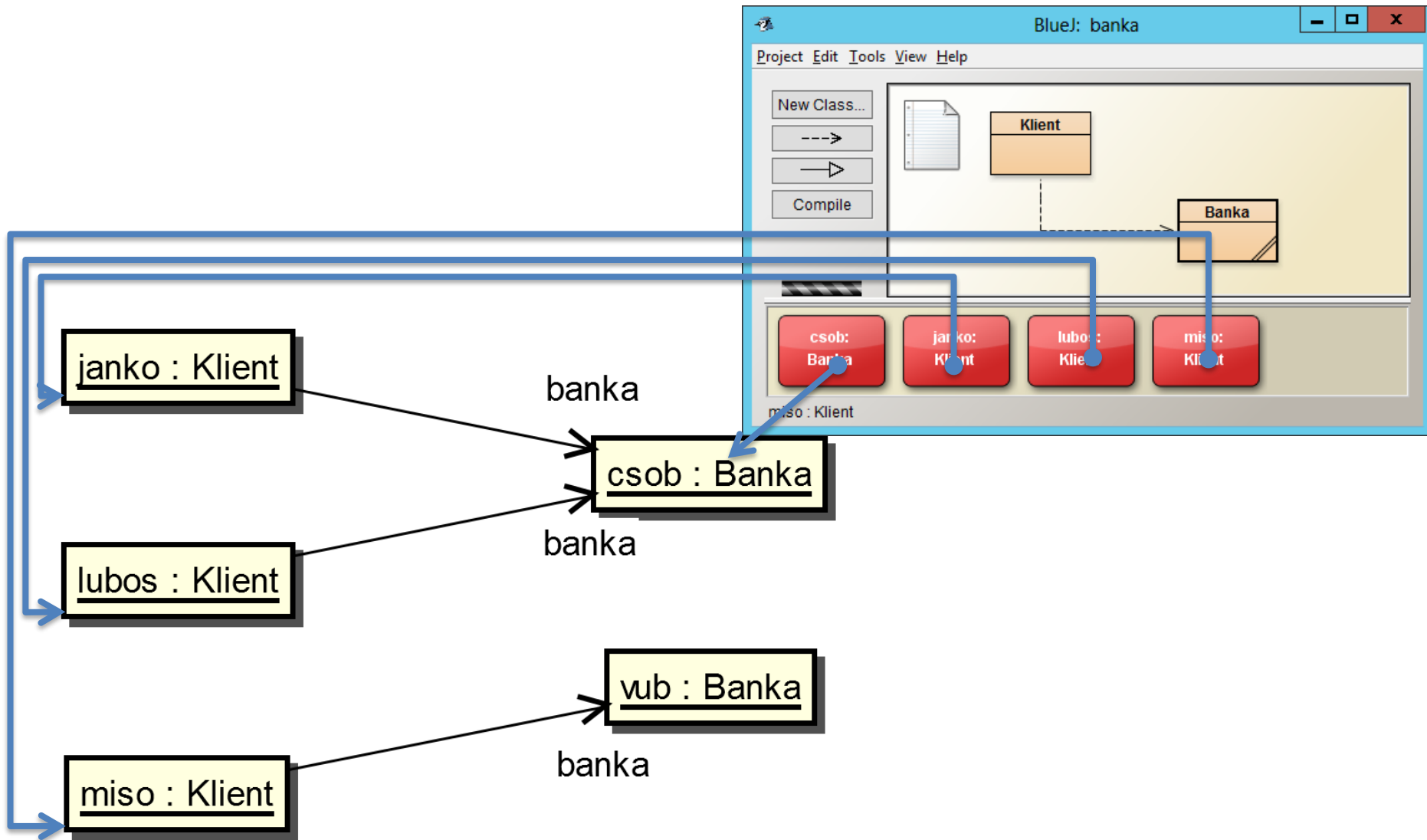
Zánik objektov pri asociácii⁽¹⁾



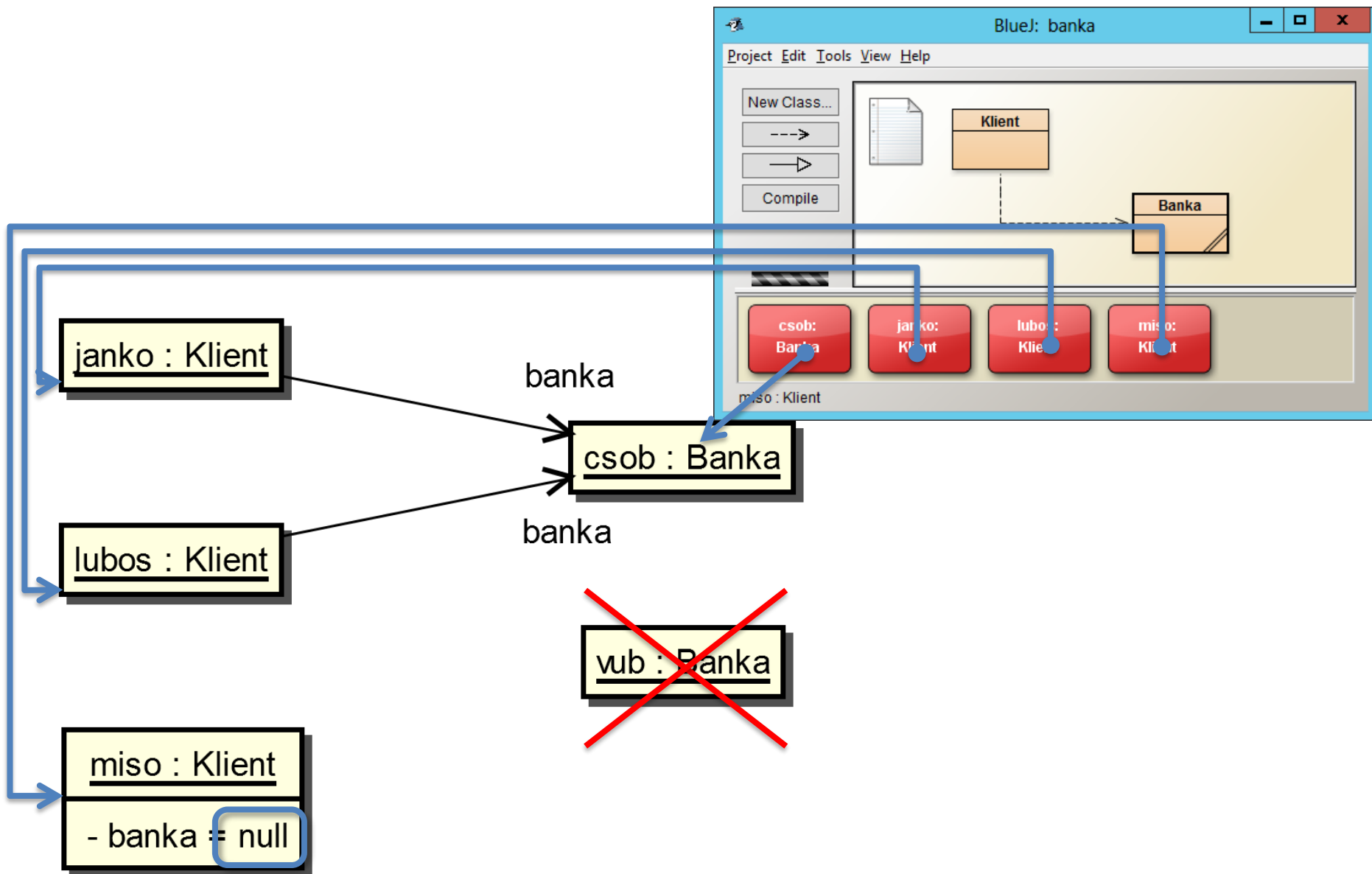
Zánik objektov pri asociácii₍₂₎



Zánik objektov pri asociácii₍₃₎



Zánik objektov pri asociácii₍₃₎



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