

Informatika 1

Spolupráca objektov



Pojmy zavedené v 5. prednáške (1)

- typy chýb
 - syntaktické
 - behové
 - logické
- techniky boja s chybami
 - testovanie
 - ladenie
 - písanie čitateľného kódu

Pojmy zavedené v 5. prednáške (2)

- testovanie – rôzne pohľady
 - testovanie komponentov/integračné testovanie/systémové testovanie/akceptačné testovanie
 - biela a čierna skrinka
 - pozitívne a negatívne
 - manuálne a automatizované
- manuálne testovanie jednotiek
 - prechádzanie zdrojového kódu
 - priama komunikácia s objektom – BlueJ

Pojmy zavedené v 5. prednáške (3)

- automatizované testovanie
 - testy regresie
 - testovacie triedy – JUnit
 - správa assertEquals, assertNotEquals, assertTrue, assertFalse
 - prípravky – fixtures

Pojmy zavedené v 5. prednáške (4)

- ladenie
 - manuálne prechádzanie kódu
 - ladiace výpisy
 - debugger

Pojmy zavedené v 5. prednáške (5)

- dokumentácia objektu
 - forma rozhrania
 - dokumentačné komentáre
 - javadoc – jazyk Java
 - tagy @author, @version, @param, @return

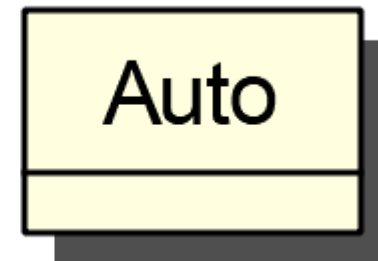
Cieľ prednášky

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

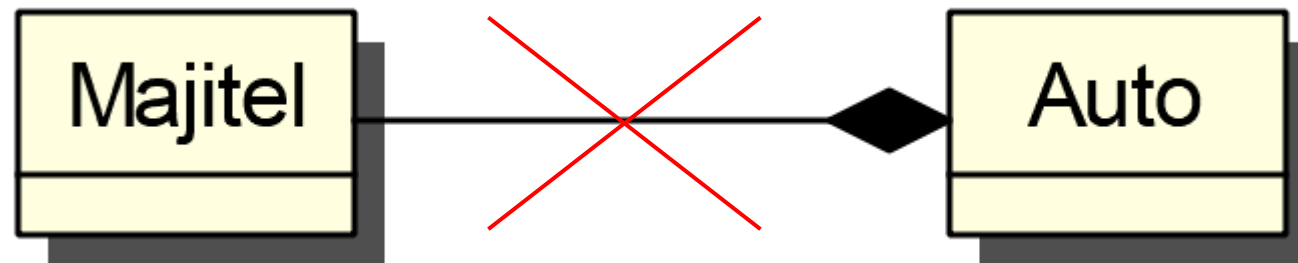
Asociácia – spolupráca objektov

- asociácia – ľubovoľná spolupráca dvoch objektov
 - príklady: klient a banka, učiteľ a študent
- zvyčajne nezávislé životné cykly oboch objektov
- spolupráca v konkrétnych situáciách
- (kompozícia = špeciálny typ asociácie)

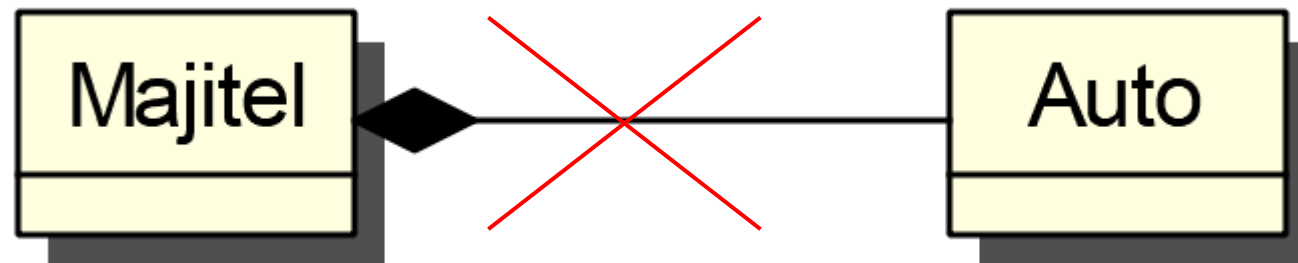
Príklad: Majiteľ – Auto



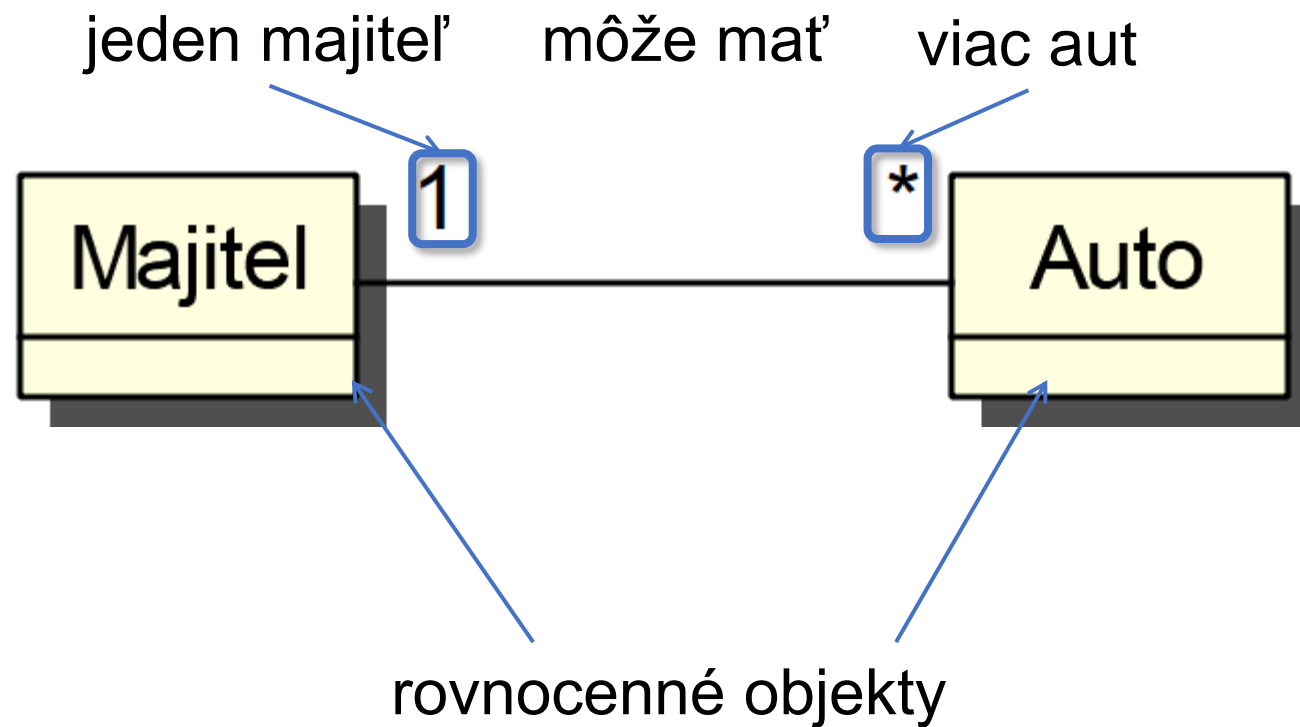
Príklad: Majiteľ – Auto cez kompozíciu (1)



Príklad: Majiteľ – Auto cez kompozíciu (2)

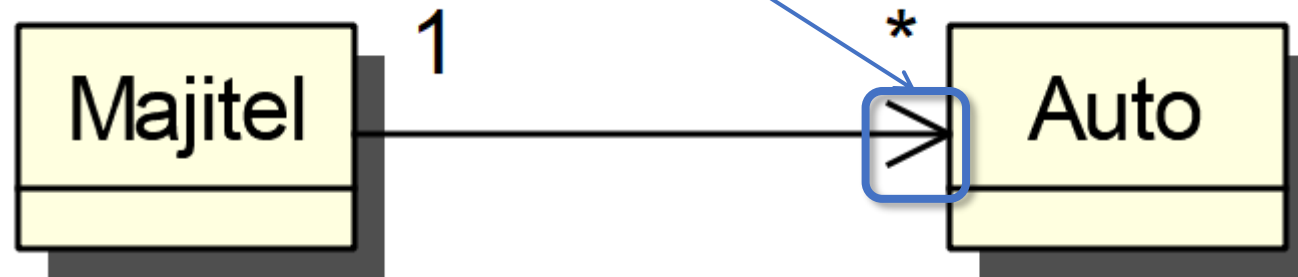


Príklad: Majiteľ – Auto cez asociáciu (1)



Príklad: Majiteľ – Auto cez asociáciu (2)

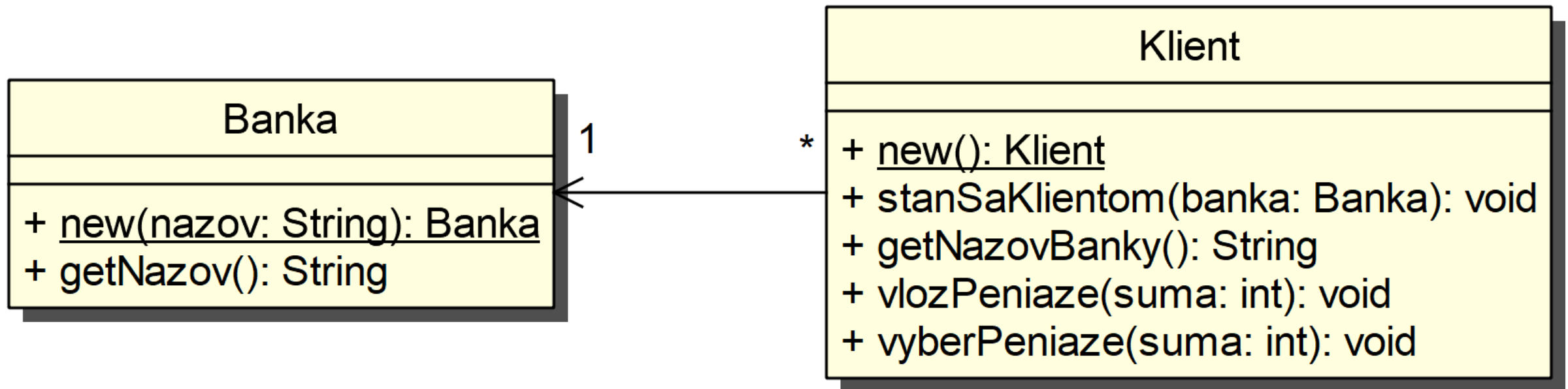
majiteľ pracuje s autami, ale opačne nie



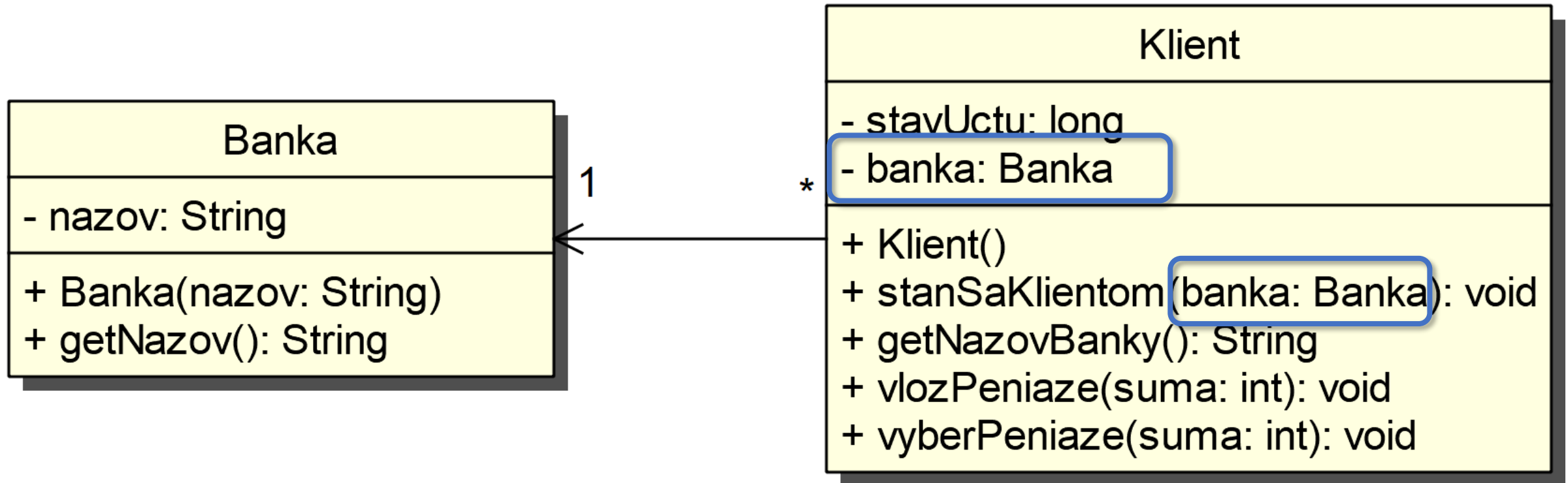
Projekt banka

- zaznamenávanie stavu účtu klienta
- klient vie, v ktorej banke má účet
- banka nie je časť klienta
- klient nie je časť banky
- asociácia

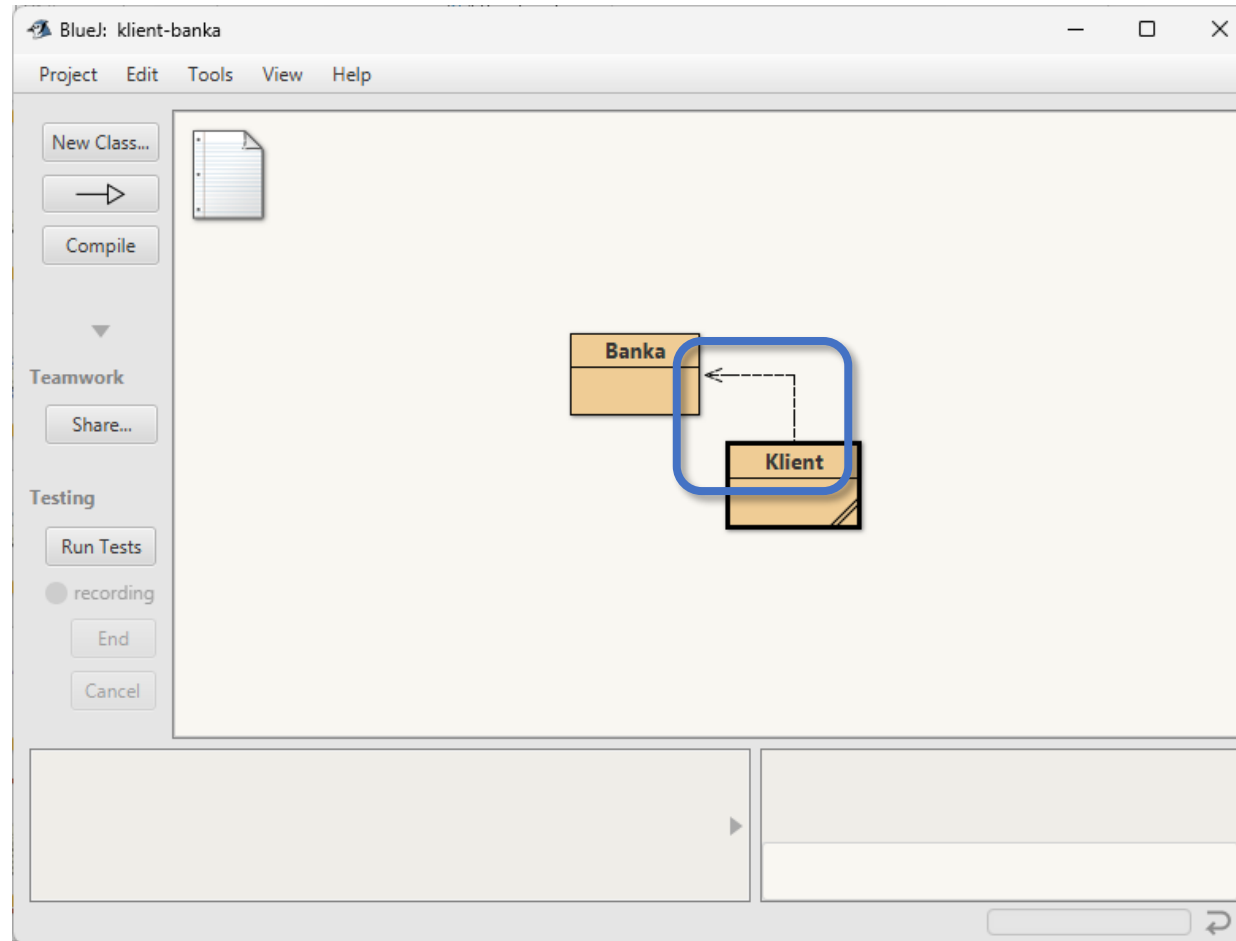
UML diagram projektu – vonkajší pohľad



UML diagram projektu – vnútorný pohľad



Projekt banka – BlueJ



Trieda klient – konštruktor a inicializácia atribútov

```
public class Klient {  
    public long stavUctu;  
    public Banka banka;  
  
    public Klient() {  
        this.stavUctu = 0;  
        this.banka = null;  
    }  
  
    ...  
}
```

Stav klienta po vytvorení inštancie

gwb : Banka

- nazov = "Gringotts Wizarding Bank"

harryPotter : Klient

- stavUctu = 50625

- banka = null

Hodnota null

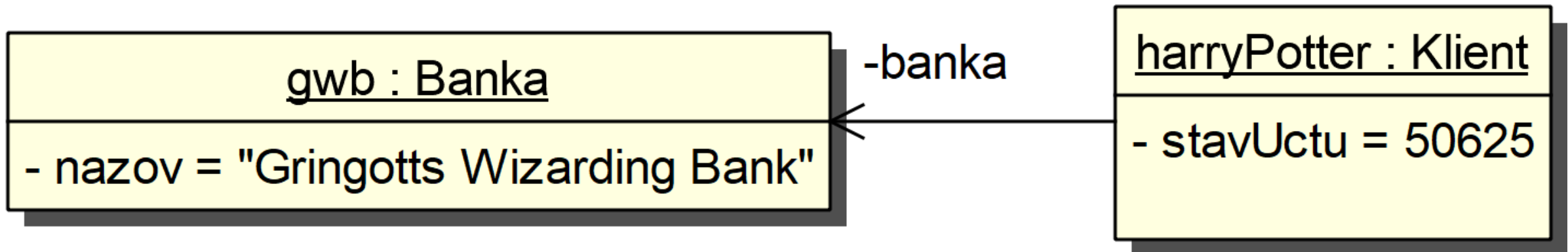
- null – objektový literál
- null = referencia neodkazuje na žiadny objekt
- pre referenciu na inštanciu ľubovoľnej triedy
 - typovo kompatibilná hodnota s ľubovoľným objektovým typom.

Metóda stanSaKlientom v triede klient

```
public void stanSaKlientom(Banka banka) {  
    this.banka = banka;  
}
```

Stav klienta po priradení banky

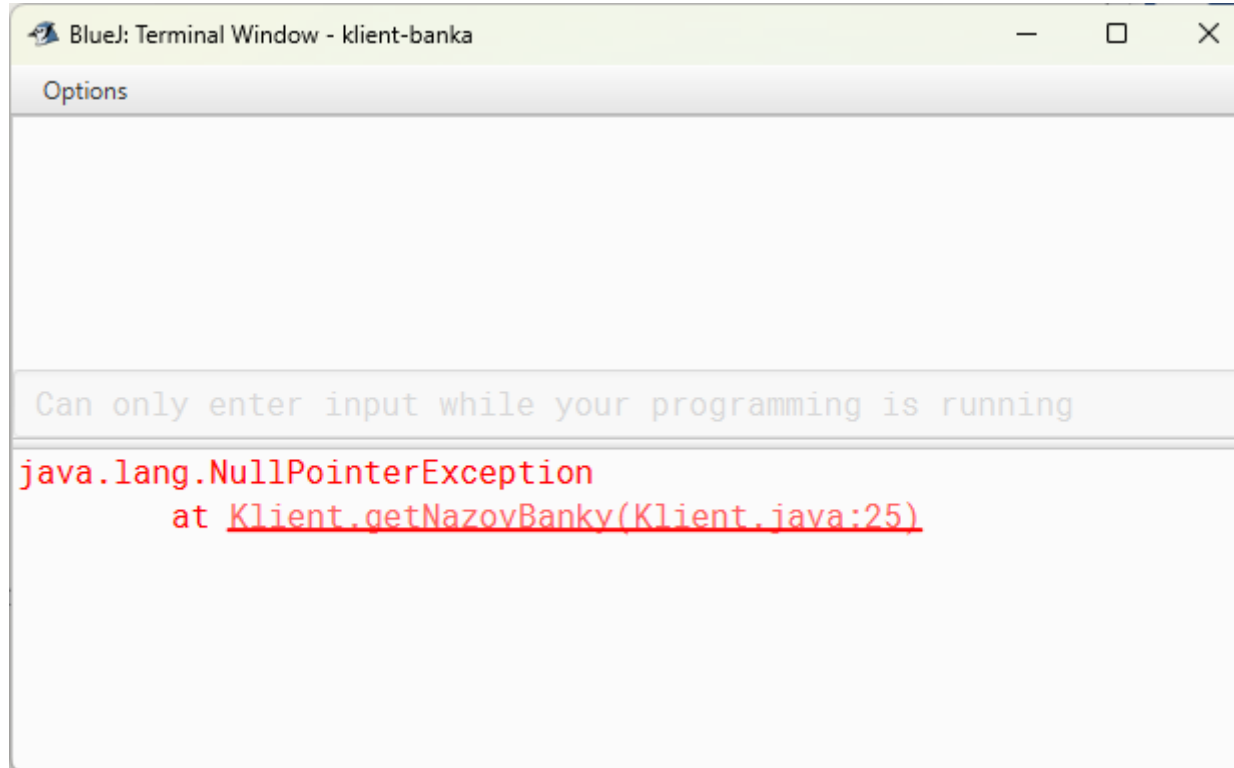
```
harryPotter.stanSaKlientom(gwb) ;
```



Metóda getNazovBanky v triede klient

```
public String getNazovBanky() {  
    return this.banka.getNazov();  
}
```

Chyba pri poslaní správy getNazovBanky



The image shows a screenshot of a BlueJ Terminal Window titled "BlueJ: Terminal Window - klient-banka". The window has a standard macOS-style title bar with minimize, maximize, and close buttons. Below the title bar is a tab labeled "Options". The main area of the terminal displays the message "Can only enter input while your programming is running" in a light gray font. Below this, a red error message is shown: "java.lang.NullPointerException" followed by "at Klient.getNazovBanky(Klient.java:25)".

```
BlueJ: Terminal Window - klient-banka
Options
Can only enter input while your programming is running
java.lang.NullPointerException
    at Klient.getNazovBanky(Klient.java:25)
```


Poslanie správy pri asociácii

- v niektorých prípadoch môže byť v objektovej premennej null
- poslanie správy spadne na behovú chybu
- je nutné kontrolovať

Metóda getNazovBanky v triede klient (1)

```
public String getNazovBanky() {  
    if (this.banka == null) {  
        return ???;  
    } else {  
        return this.banka.getNazov();  
    }  
}
```

Metóda getNazovBanky v triede klient (2)

```
public String getNazovBanky() {  
    if (this.banka == null) {  
        return null;  
    } else {  
        return this.banka.getNazov();  
    }  
}
```

Relačné výrazy s objektmi (1)

- relačné operátory pre čísla
 - <, <=, >, >=, ==, !=
- relačné operátory pre objekty
 - ==, !=
- porovnanie referencií na objekty

```
referencia1 == referencia2
```

Relačné výrazy s objektmi (2)

- == porovnanie dvoch referencií (napr. v dvoch premenných)
 - true – dve referencie na ten istý objekt
 - false – dve referencie na dva rôzne objekty
- != opak operátora ==

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

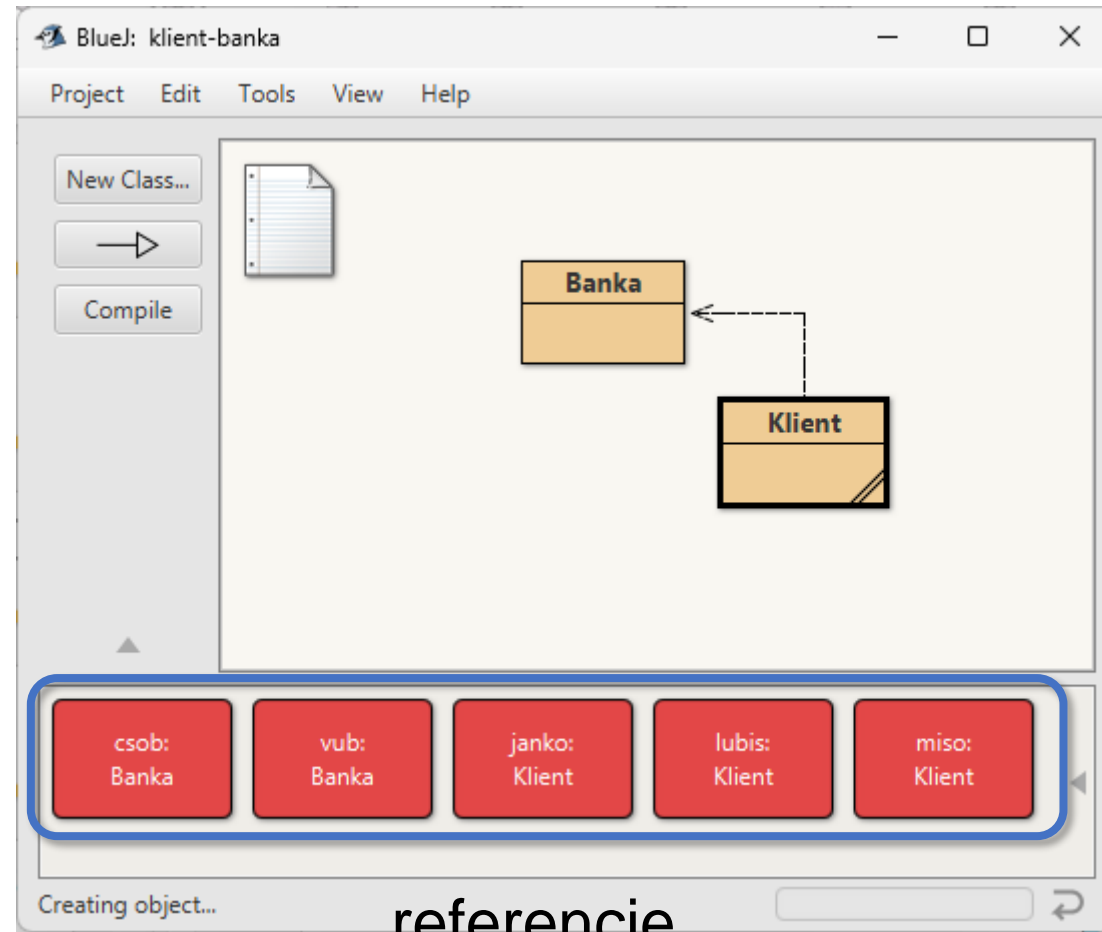
Zánik inštancie

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

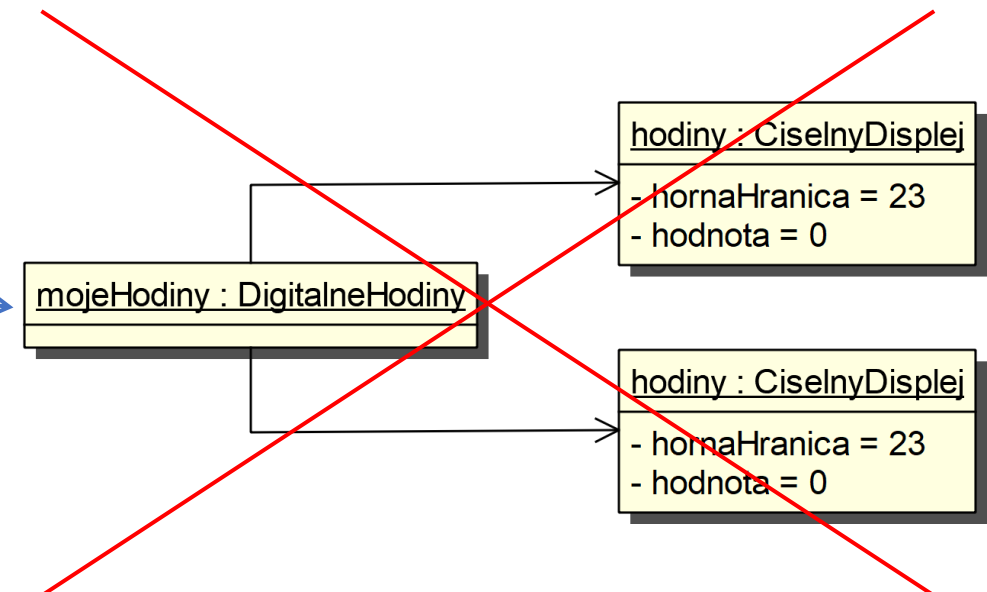
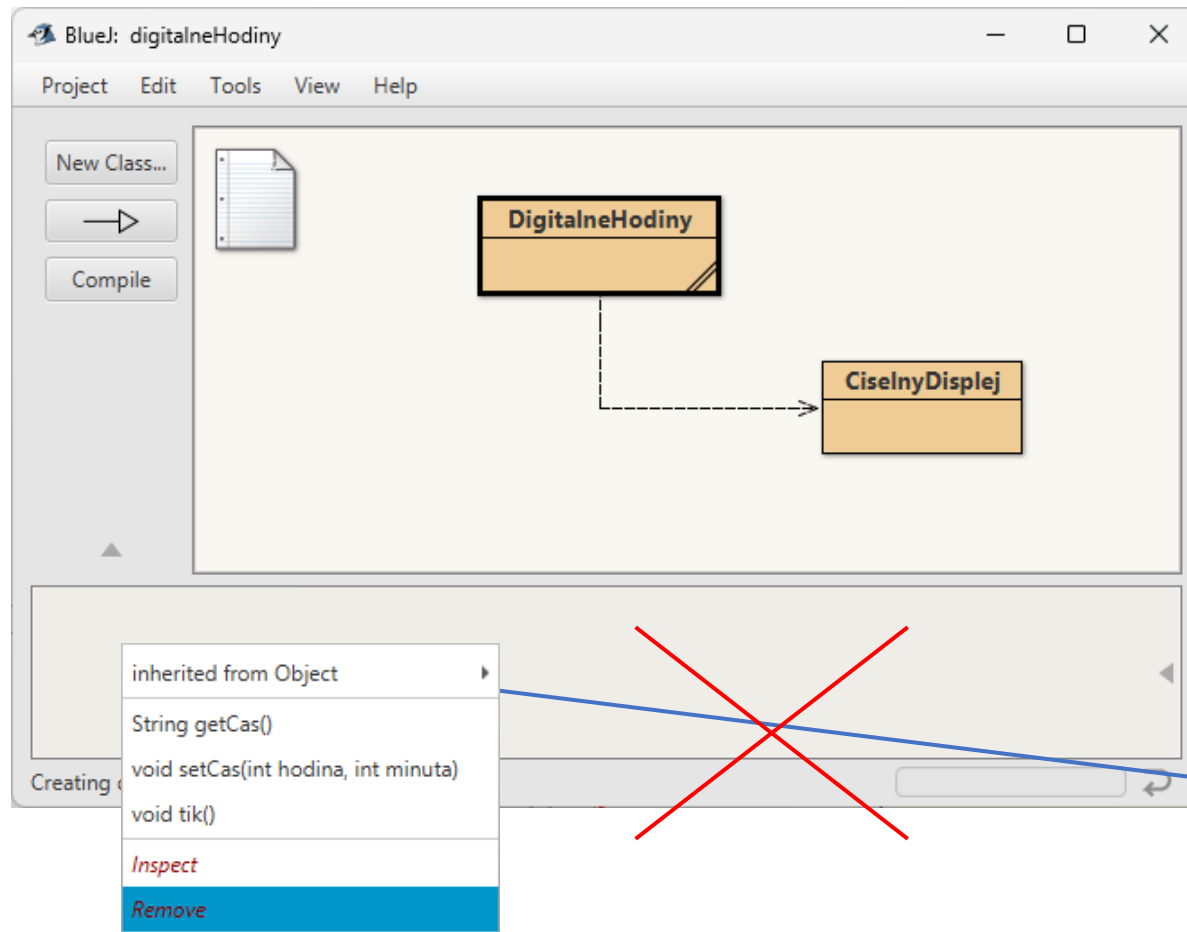
Zánik inštalácie

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

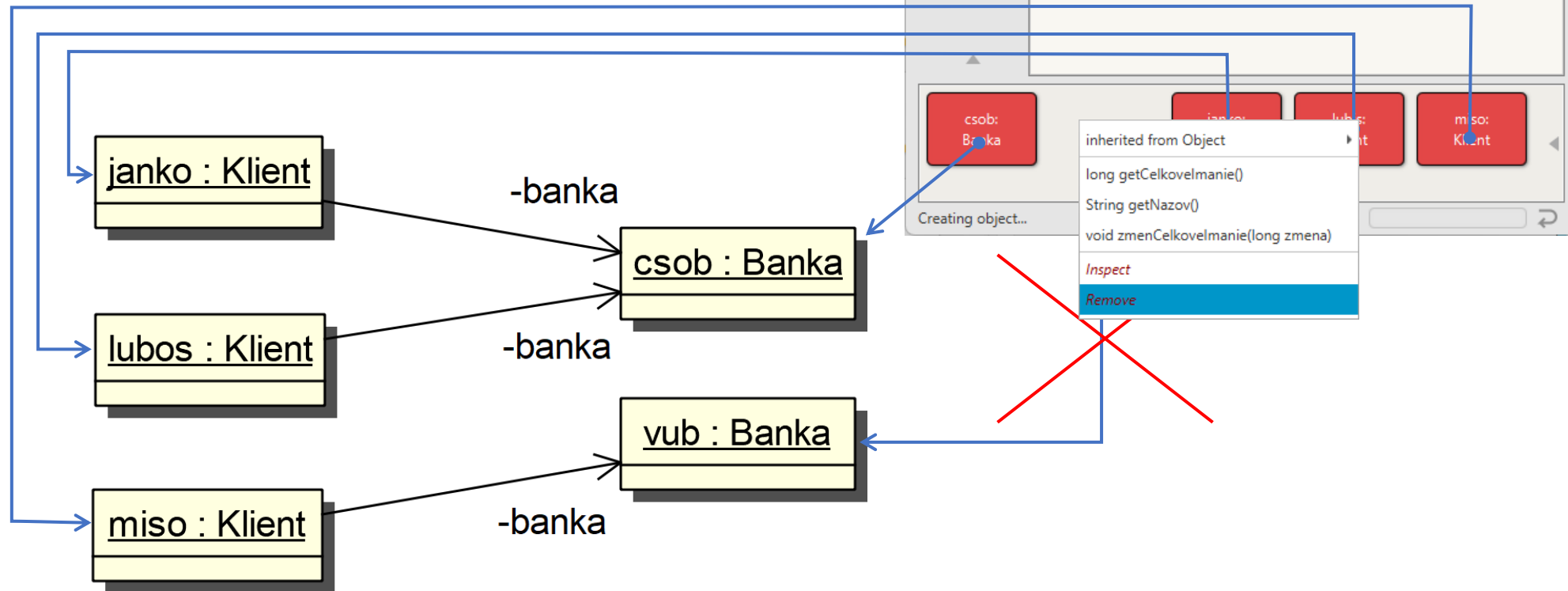
Referencie v BlueJ



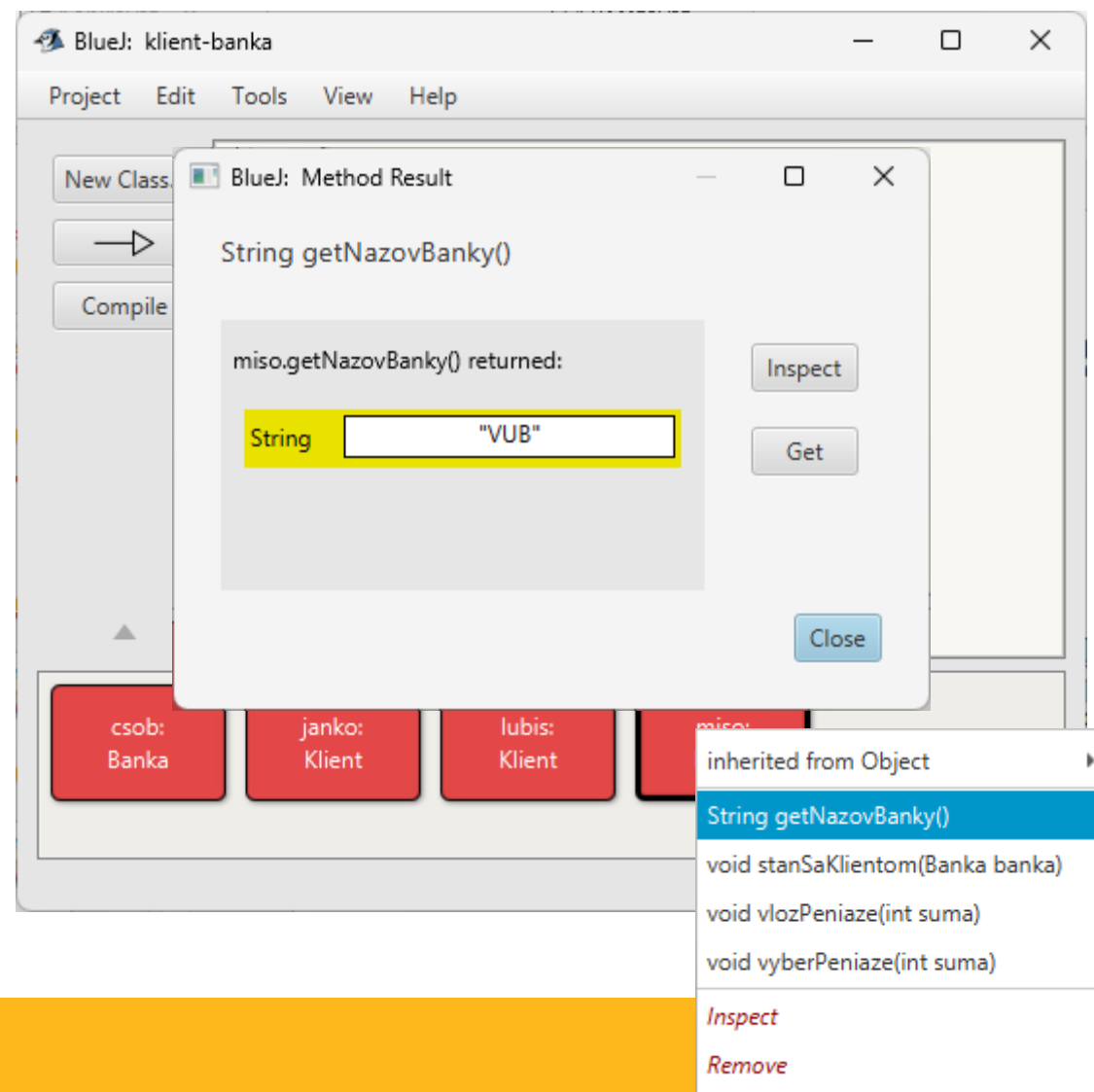
Zánik objektov pri kompozícii



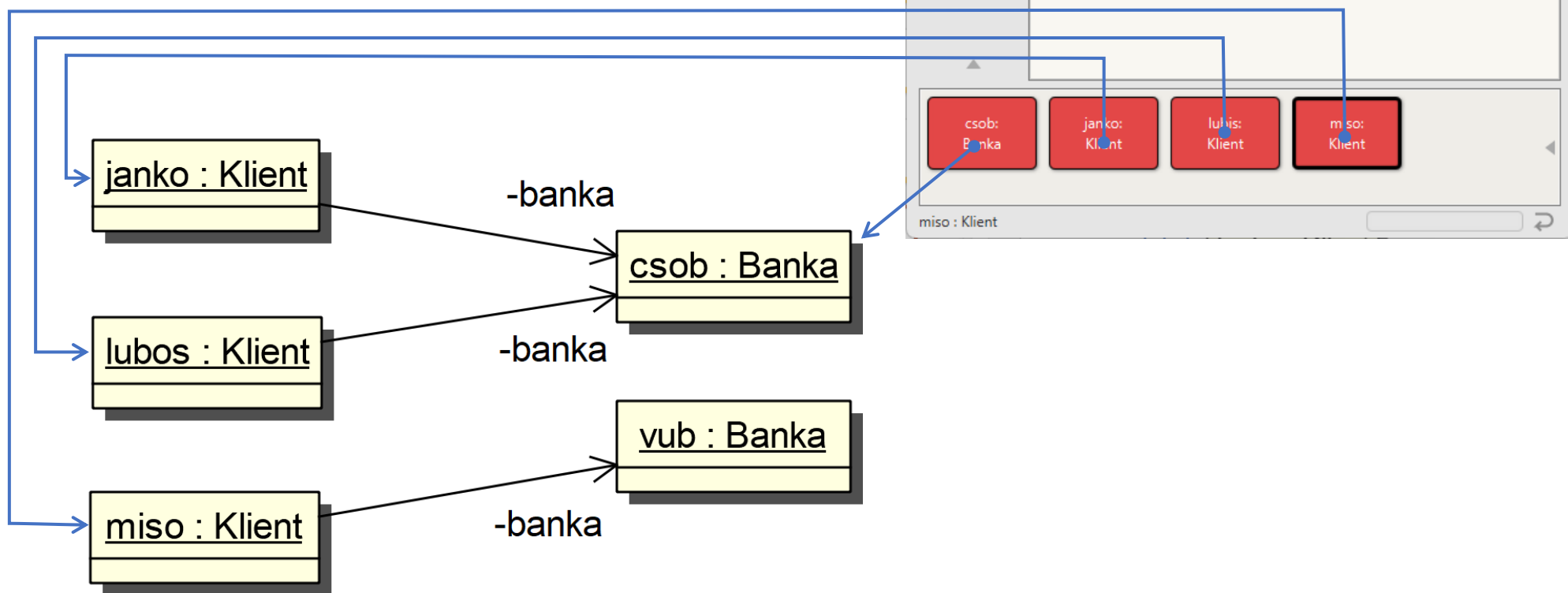
Zánik objektov pri asociácii (1)



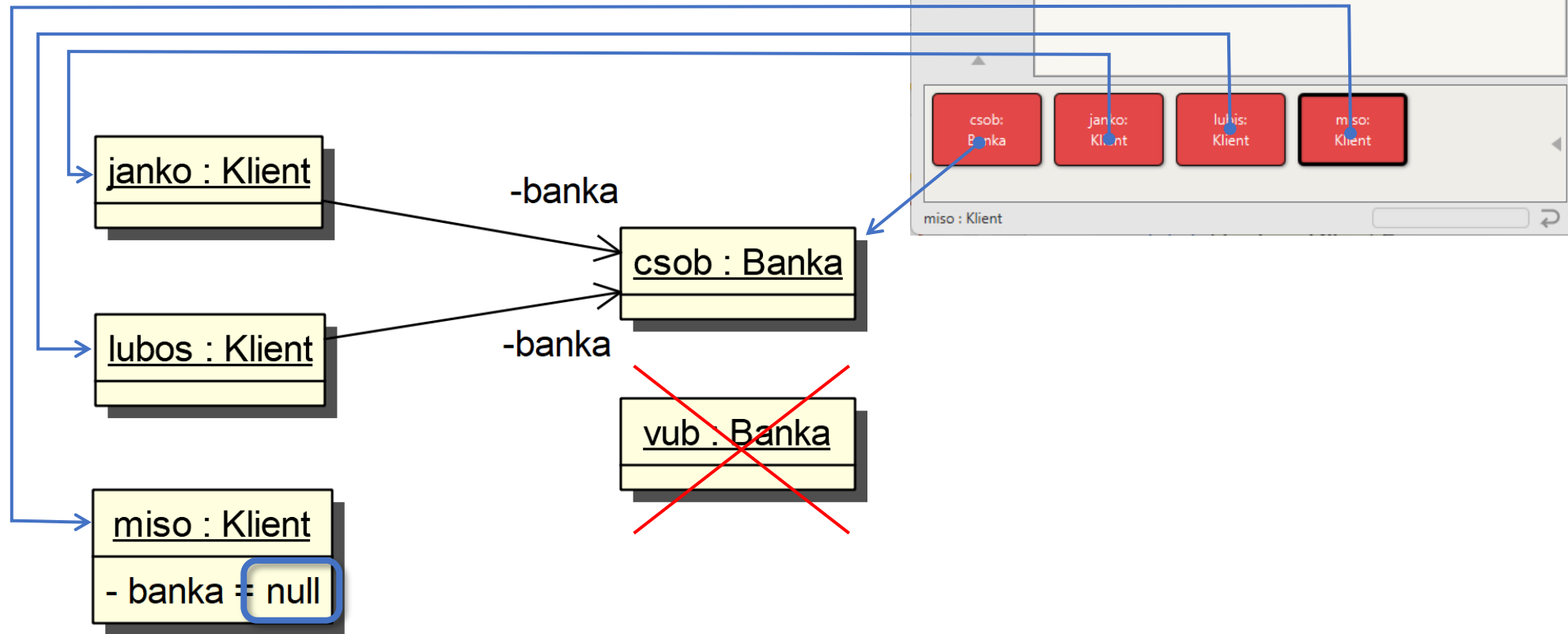
Zánik objektov pri asociácii (2)



Zánik objektov pri asociácii (3)



Zánik objektov pri asociácii (4)



Hodnota null – úlohy (1)

- null = referencia neodkazuje na žiadny objekt
- inicializácia objektovej premennej
- zrušenie referencie (popr. zánik inštancie)
 - priradovací príkaz
 - premenna = null;

Hodnota null – úlohy (2)

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

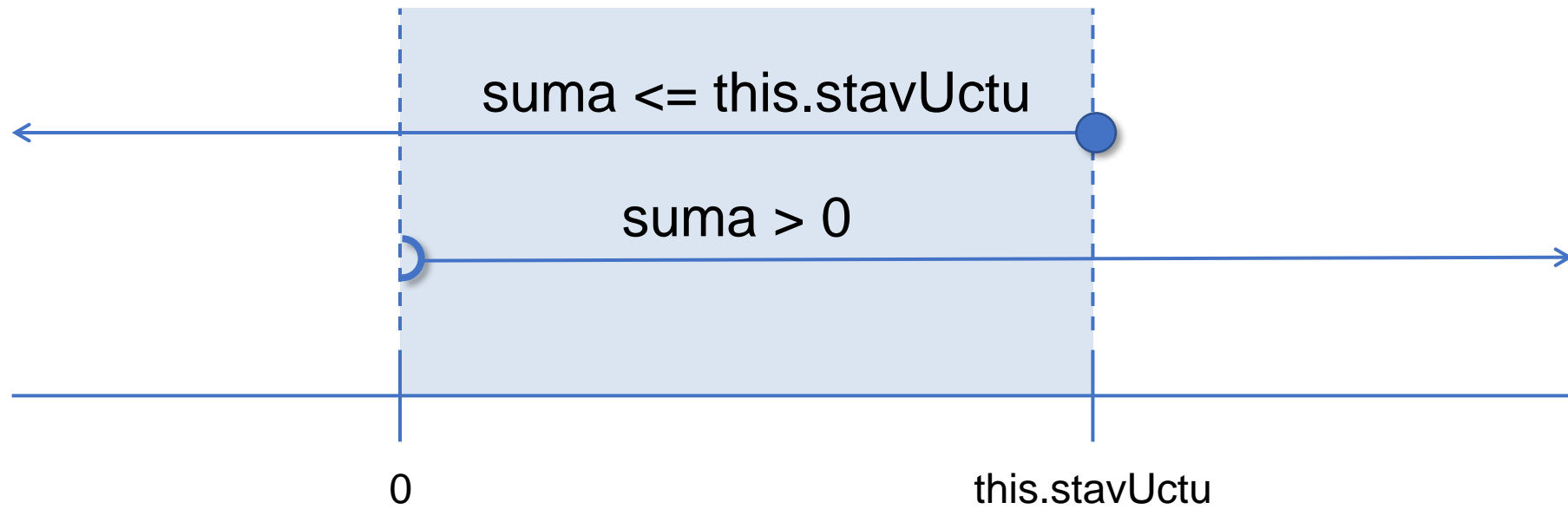
Metóda vyberPeniaze v klientovi (1)

```
public void vyberPeniaze(int suma) {  
    // Chýbajú kontroly  
    this.stavUctu = this.stavUctu - suma;  
}
```

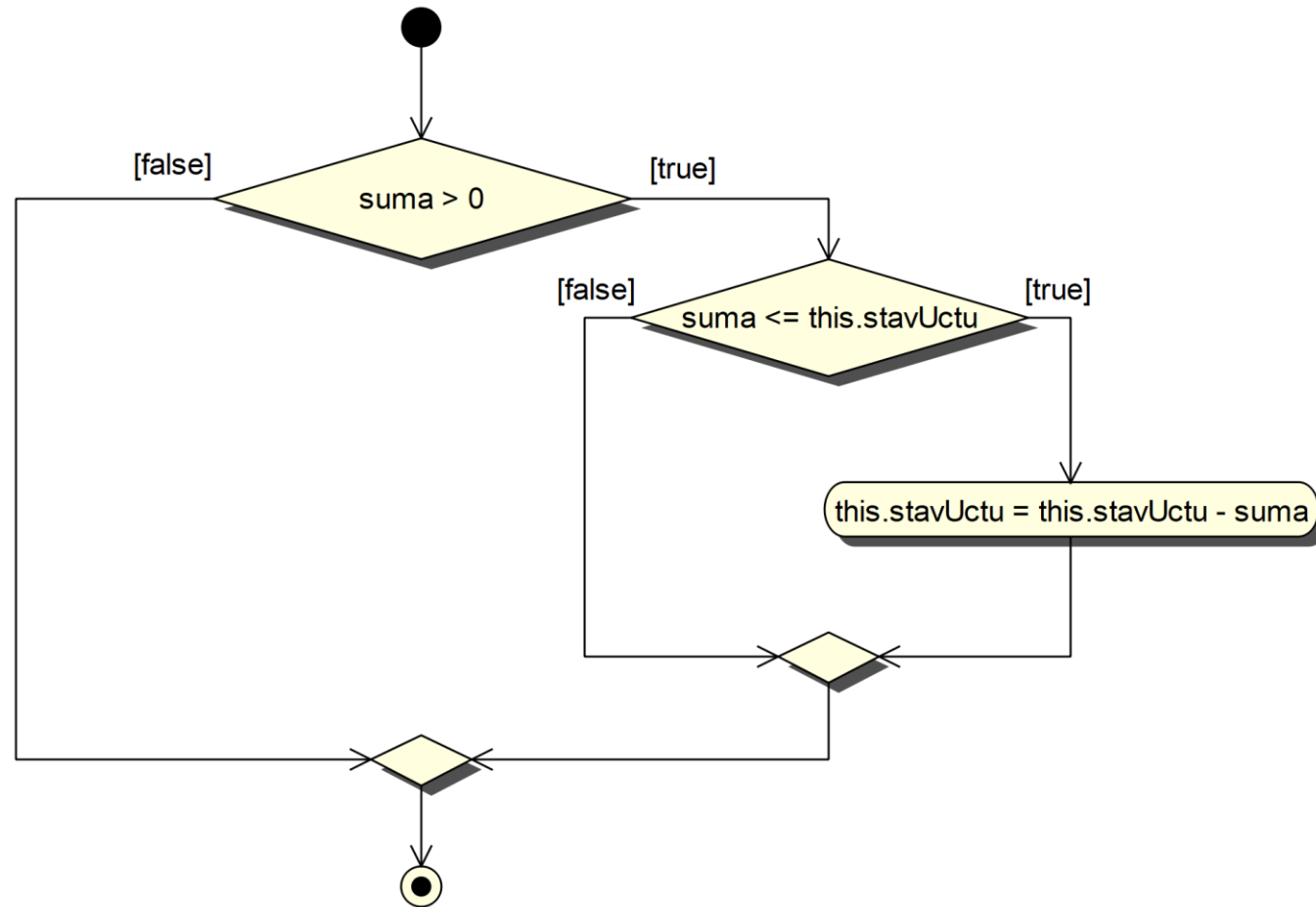
Metóda vyberPeniaze v klientovi (2)

- zmení stav účtu
- nesmie dovoliť sumu mimo rozsah
 - min 0 (vyjme)
 - max this.stavUctu

Zložená podmienka (1)



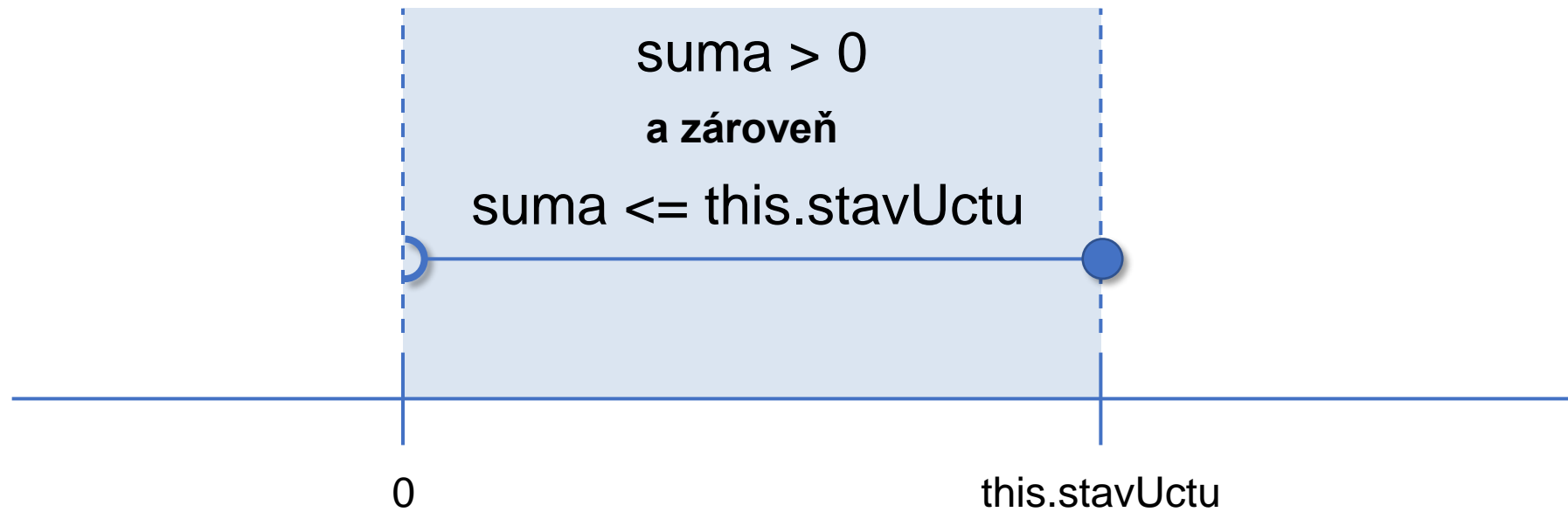
Zložená podmienka (2)



Metóda vyberPeniaze v klientovi

```
public void vyberPeniaze(int suma) {  
    if (suma > 0) {  
        if (suma <= this.stavUctu) {  
            this.stavUctu = this.stavUctu - suma;  
        }  
    }  
}
```

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 <= x && x <= b
```

Metóda vyberPeniaze v klientovi

```
public void vyberPeniaze(int suma) {  
    if (suma > 0 && suma <= this.stavUctu) {  
        this.stavUctu = this.stavUctu - suma;  
    }  
}
```

& &

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 +, -, !
	*, /, %
	binárne +, -
	<, <=, >, >=
	==, !=
	&&
najnižšia	

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

Skrátené vyhodnocovanie

- `false` `&&` čokoľvek = `false`
- `true` `||` čokoľvek = `true`
- „čokoľvek“ sa nevyhodnocuje
- príklad:

```
this.banka != null && this.banka.jeOtvorena()
```

- správa sa nepoše, ak je `this.banka` `null`

Zmeny na účte – výber z účtu

```
public void vyberPeniaze(int suma) {  
    if (suma > 0) {  
        if (suma <= this.stavUctu) {  
            this.stavUctu = this.stavUctu - suma;  
        }  
    }  
}
```

Zmeny na účte – vklad na účet

```
public void vložPeniaze(int suma) {  
    if (suma > 0) {  
        this.stavUctu = this.stavUctu + suma;  
    }  
}
```

Zmeny na účte – problém

- zmena stavu sa vykonáva na viac miestach
- vadí?
 - pravdepodobne nie
 - ale:
 - v budúcnosti viac operácií (zarátanie poplatkov, spracovanie výpisu, uchovanie informácií v súbore, ...)
 - v budúcnosti na viac miestach (zarátanie úrokov, prevody medzi účtami, ...)
- riešenie:
 - jedna metóda so zmenou stavu účtu
 - ostatné metódy ju využívajú

Posielanie správ sám sebe

- kompozícia – objekt celok posiela správy častiam
- asociácia – objekt posiela správy spolupracujúcim objektom
- objekt posiela správy sám sebe

- formát správy
 - adresát.selektor(parametre)
- objekt sám seba označuje this

Poslanie správy sám sebe (1)

```
public void vyberPeniaze(int suma) {  
    if (suma > 0) {  
        if (suma <= this.stavUctu) {  
            this.zmenStavUctu(-suma);  
        }  
    }  
}
```

Poslanie správy sám sebe (2)

```
public void vložPeniaze(int suma) {  
    if (suma > 0) {  
        this.zmenStavUctu(suma);  
    }  
}
```

zmenStavUctu

```
public void zmenStavUctu(int zmena) {  
    this.zmenStavUctu = this.stavUctu + zmena;  
    // zarátanie poplatkov  
    // spracovanie výpisu  
    // uchovanie informácií v súbore  
    // ...  
}
```

Sekcie rozhrania

- verejné – obsahuje správy, ktoré môže poslať ľubovoľný objekt
 - definícia triedy obsahuje metódy public
- súkromné – obsahuje správy, ktoré si môže poslať len objekt sám
 - definícia triedy obsahuje metódy private

Súkromné rozhranie

```
private void zmenStavUctu(int zmena) {  
    this.zmenStavUctu = this.stavUctu + zmena;  
}
```

Súkromné rozhranie v UML

