15.Projekt Kalkulačka

15.1. Základní popis, zadání úkolu

Pracujeme na projektu Kalkulačka, který je ke stažení na `java.vse.cz`. Po otevření v BlueJ vytvoříme instanci třídy `Kalkulacka` a zavoláme metodu `show()`. Výsledkem je spuštění grafické aplikace představující jednoduchou kalkulačku viz obrázek15.1.

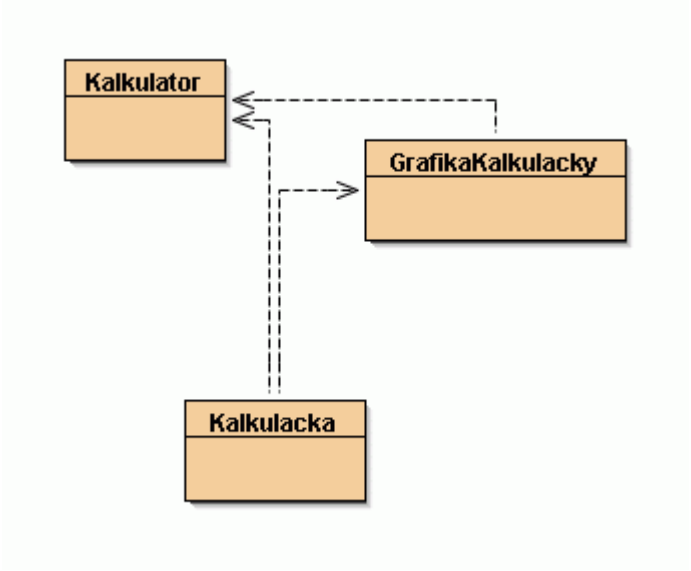
Obrázek 15.1 Vzhled kalkulačky po spuštění

Naším úkolem bude „naučit kalkulačku počítat“, zatím neumí nic. Konkrétně to znamená doplnit datové atributy a dokončit předpřipravené metody.
Tento projekt má následující cíle:

- ukázat jednu z možností oddělení grafického rozhraní od věcné třídy,
- navrhnout datové atributy pro třídu Kalkulátor,
- procvičit základní operace s primitivními datovými typy (čísla, znaky, logická hodnota),
- naučit se příkaz `if` a vytváření podmínek.

15.2. Struktura tříd

Projekt Kalkulačka se skládá z následujících tříd (obrázek z BlueJ):

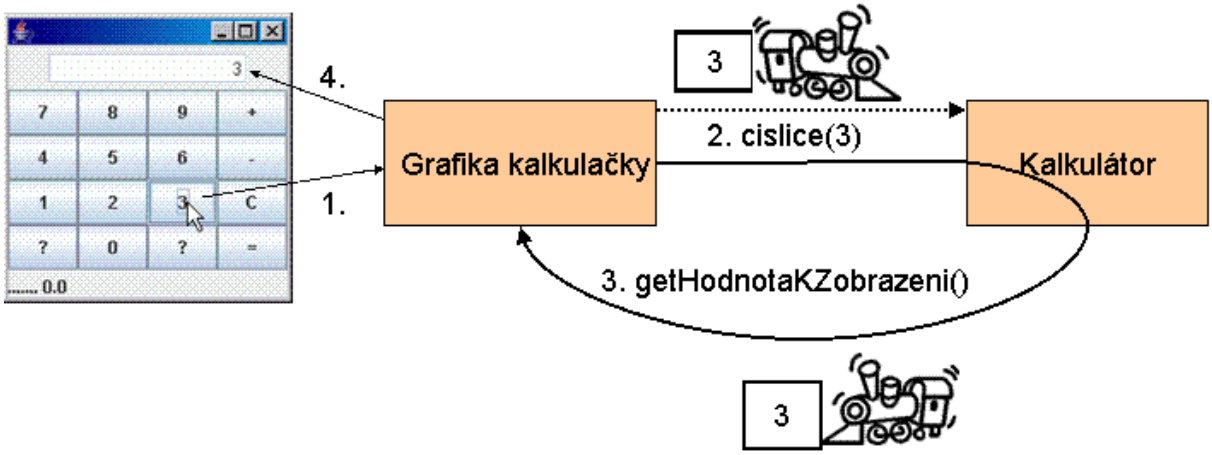


Obrázek 15.2 Diagram tříd projektu Kalkulačka, převzato z BlueJ

15.3. Popis komunikace mezi objekty

V konstruktoru třídy `Kalkulacka` se vytváří instance třídy `Kalkulator` a instance třídy `GrafikaKalkulacky`, která jako parametr při vzniku získá odkaz na vytvořenou instanci třídy `Kalkulator`. V metodě `show()` se zobrazí grafické rozhraní kalkulačky a předá se mu řízení komunikace.
Komunikace mezi uživatelem a instancemi tříd `GrafikaKalkulacky` a `Kalkulator` probíhá v této posloupnosti:

1. Uživatel stiskne některou klávesu (na obrázku je to stisknutí tlačítka s hodnotou 3).
2. Instance třídy `GrafikaKalkulacky` vyhodnotí, které tlačítko to bylo a zavolá odpovídající metodu ze třídy `Kalkulator`. Na našem obrázku zavolá metodu `cislice()` a jako parametr jí předá číslo 3, které odpovídá stisknuté klávese. Tato metoda zpracuje poslaný údaj.
3. Instance třídy `GrafikaKalkulacky` zavolá metodu `getHodnotaKZobrazeni()` instance třídy `Kalkulator`. Tato metoda vrátí hodnotu, která má být zobrazena na displeji.
4. Instance třídy `GrafikaKalkulacky` zobrazí na displeji obdrženou hodnotu.



Obrázek 15.3 Znázornění komunikace mezi jednotlivými instancemi při stisknutí číslice 3

Třída `GrafikaKalkulacky` zajišťuje obsluhu grafického rozhraní a řízení aplikace, instance třídy `Kalkulator` provádí vlastní výpočty (vždy vypočte hodnotu, která se má zobrazit na displeji). Třída `Kalkulator` obsahuje pouze hlavičky metod, naším **úkolem je doplnit potřebné datové atributy a obsah metod**.

15.4. Popis kódu třídy Kalkulator

```
 1  /**
 2   * Tato třída provádí vlastní výpočty kalkulátoru.
 3   * Třída má tři skupiny metod:
 4   * a) pomocí metody getHodnotaKZobrazeni() grafická
 5   *      třída zjišťuje, co má zobrazit na displeji,
 6   * b) metody cislice(), plus(), minus(), rovnaSe() a vymaz() se
 7   *      volají při stisknutí příslušné klávesy na kalkulačce,
 8   * c) metody getAutor() a getVerze() jsou informační.
 9   * Třída není dokončena
10   *     - úkolem studentů je tuto třídu dokončit, tj. navrhnout
11   *       datové atributy instancí třídy a dokončit obsah metod.
12   * @author     Luboš Pavlíček
13   * @version    1.0 (31 July 2004)
```

```
14   */
15  public class Kalkulator {
16
17    /**
18     *  konstruktor třídy
19     */
20    public Kalkulator () {
21       // DOPLNIT TUTO METODU
22    }
23
24    /**
25     * Metoda vrací hodnotu, která se má zobrazit na displeji
26     * kalkulačky. Tato metoda se obvykle volá po zavolání metody
27     * odpovídající stisku tlačítka.
28     *
29     * @return   hodnota k zobrazení
30     */
31    public int getHodnotaKZobrazeni() {
32       // DOPLNIT TUTO METODU
33       return 1;
34    }
35
36    /**
37     * metoda se volá při stisknutí tlačítka s číslicí na
38     * kalkulačce. Parametrem je hodnota na stisknuté klávese.
39     *
40     * @param hodnota    hodnota na stisknutém tlačítku,
41     *    hodnota je v rozsahu od 0 do 9
42     */
43    public void cislice(int hodnota) {
44       // DOPLNIT TUTO METODU
45    }
46
47     /**
48     * metoda se volá při stisknutí tlačítka "+" (plus)
49     * na kalkulačce
50     */
51    public void plus() {
52       // DOPLNIT TUTO METODU
53    }
54
55    /**
56     * metoda se volá při stisknutí tlačítka "-" (minus)
57     * na kalkulačce
58     */
59    public void minus() {
60       // DOPLNIT TUTO METODU
61    }
62
63    /**
64     * metoda se volá při stisknutí tlačítka "=" (rovná se)
65     * na kalkulačce
66     */
67    public void rovnaSe() {
68       // DOPLNIT TUTO METODU
69    }
70
```

```
71    /**
72     * metoda se volá při stisknutí tlačítka "C" (clear)
73     * na kalkulačce
74     */
75    public void vymaz() {
76       // DOPLNIT TUTO METODU
77    }
78
79    /**
80     * metoda vrací jméno autora, např. "autor: Jan Novák"
81     *
82     * @return   řetězec se jménem autora
83     */
84    public String getAutor() {
85       // UPRAVIT TUTO METODU
86       return ".......";
87    }
88
89    /**
90     * metoda vrací označení verze, např. "verze 1.0.3"
91     *
92     * @return   řetězec s verzí programu
93     */
94    public String getVerze() {
95       // UPRAVIT TUTO METODU
96       return "0.0";
97    }
98 }
```

Jak je vidět ze zdrojového kódu, třída `Kalkulator` nic neumí. Ať je na kalkulačce stisknuto cokoli, vrací na displeji vždy hodnotu 1. Metoda `getAutor()` vrací "....." a metoda `getVerze()` vrací řetězec "0.0".

15.5. Postup řešení:

Řešení si rozdělíme do několika kroků:

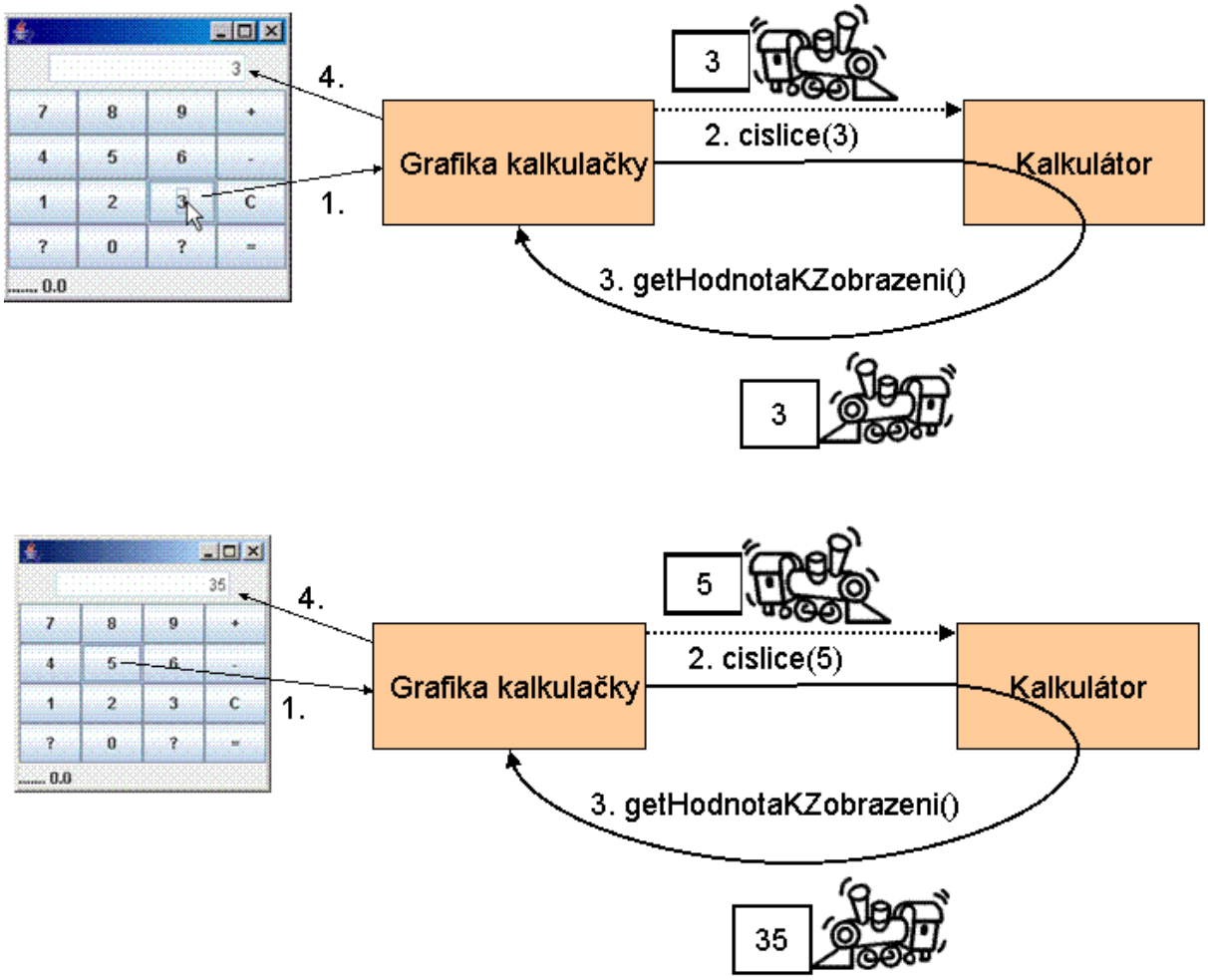
- doplnění autora a verze (toto sice nenapomůže počítání kalkulačky, ale zvýší sebevědomí programátora),
- vkládání čísla (číselné klávesy),
- výmaz čísla (klávesa C),
- sčítání dvou čísel (klávesy + a =),
- sčítání více čísel,
- odčítání.

15.5.1. Doplnění autora a verze

Nejjednodušší je doplnění metod `getAutor()` a `getVerze()`. Do kódu na řádcích 86 a 96 stačí pouze doplnit správné údaje. Ověřte si zobrazování autora a verze – údaje by se měly zobrazovat po stisknutí tlačítka

15.5.2. Vkládání prvního čísla

Druhým krokem bude „naučit“ kalkulačku správně zobrazovat první zadávané číslo. Po každém stisku tlačítka na kalkulačce je zavolána metoda `cislice()`. V parametru hodnota dostaneme číslici, kterou uživatel zadal.
Na následujícím obrázku vidíte, jak probíhá komunikace mezi instancemi tříd `GrafikaKalkulacky` a `Kalkulator` v situaci, kdy uživatel zadává číslo 35.



Obrázek 15.4 Znázornění komunikace mezi instancemi při zadávání čísla 35

Mezi jednotlivými stisky tlačítek si musíme pamatovat, co již bylo zadáno. Budeme tedy potřebovat datový atribut pro uchovávání vkládaného čísla (a současně čísla, které se bude vracet metodou `getHodnotaKZobrazeni()` k zobrazení na displeji). Toto číslo bude typu `int` (kalkulačka umí pouze celá čísla) a jeho deklarace může vypadat následovně:

```
    private int hodnotaKZobrazeni;
```

V konstruktoru přiřadíme do datového atributu počáteční hodnotu[35] 0. Ve výše uvedeném příkladu by po stisknutí klávesy 3 (tj. na konci provádění metody `cislice()`) měl tento datový atribut obsahovat hodnotu 3, po následném stisknutí klávesy 5 hodnotu 35.

[35] Toto přiřazení není nutné – Java automaticky číselným datovým atributům přiděluje počáteční nulu. V případě lokálních proměnných to ale neplatí. I z tohoto důvodu je vhodné si zvyknout vždy přiřazovat počáteční hodnotu datovým atributům a lokálním proměnným.

Postupným vkládáním jednotlivých číslic se skládá hodnota čísla. Číslo je zadáváno zleva, při zadání další číslice se předchozí hodnota zvětší o jeden řád (z jednotek budou desítky, z desítek stovky atd.) a přičte se k ní hodnota naposledy vložené číslice. Kód metody `cislice()` by mohl vypadat následovně:

```
public void cislice(int hodnota) {
     this.hodnotaKZobrazeni = this.hodnotaKZobrazeni*10+hodnota;
}
```

Proměnná `hodnotaKZobrazeni` je uvozena klíčovým slovem `this`, které zdůrazňuje, že se jedná o datový atribut této instance. Pokud nedochází ke kolizi jména datového atributu s lokální proměnnou či jménem parametru, není potřeba v Javě toto klíčové slovo uvádět. Kód proto může vypadat následovně:

```
public void cislice(int hodnota) {
     hodnotaKZobrazeni = hodnotaKZobrazeni*10+hodnota;
}
```

Všimněte si též parametru metody `cislice()` – deklarace se skládá z typu (`int`) a identifikátoru (`hodnota`). Důležitý je datový typ – při volání metody se kontroluje pouze typ. Identifikátor slouží pro programátora metody – pomocí tohoto identifikátoru se programátor na parametr metody odkazuje. Není problém tento identifikátor změnit, aniž by bylo potřeba něco měnit ve třídách/metodách, které tuto metodu volají. Metoda by mohla vypadat následovně:

```
public void cislice(int cislo) {
     hodnotaKZobrazeni = hodnotaKZobrazeni*10+cislo;
}
```

Metoda `getHodnotaKZobrazeni()` vrací obsah datového atributu `hodnotaKZobrazeni`, její kód bude vypadat následovně:

```
public int getHodnotaKZobrazeni() {
     return hodnotaKZobrazeni;
}
```

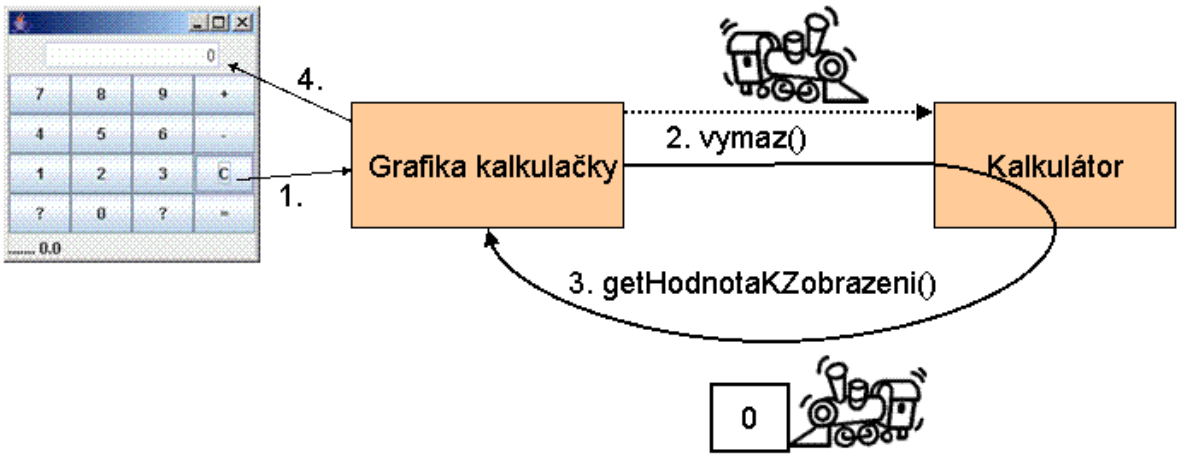
Přeložte aplikaci a vyzkoušejte kalkulačku. Nyní již můžete vkládat čísla do kalkulačky, narazíte však na dva problémy:

- pro zadání nového čísla je potřeba znovu spustit kalkulačku,
- při vložení většího počtu číslic dojde k přetečení rozsahu číselného typu `int` (přibližně 2 miliardy) – tento problém nebudeme řešit.

15.5.3. Operace výmaz (C)

Po stisknutí klávesy výmaz (C) se volá metoda `vymaz()` a měla by se vynulovat hodnota na displeji. Průběh komunikace vidíte na obrázku 15.5.
Řešení této situace je poměrně jednoduché – v metodě `vymaz()` by se měla vynulovat hodnota datového atributu `hodnotaKZobrazeni`. Kód metody `vymaz()` bude vypadat následovně:

```
public void vymaz() {
     hodnotaKZobrazeni = 0;
}
```



Obrázek 15.5 Znázornění komunikace mezi instancemi při stisknutí tlačítka C

15.5.4. Operace plus (+)

Nyní budeme řešit situaci, kdy uživatel vloží první číslo (např. 35) a stiskne tlačítko se znaménkem plus.

Po stisknutí tlačítka plus by měla na displeji zůstat zobrazena hodnota 35 nebo by se měla zobrazit nula. Varianta s nulou je jednodušší, proto s ní začneme.
Musíme si zapamatovat první číslo momentálně uložené v datovém atributu *`hodnotaKZobrazeni`*. Při vkládání druhého čísla se obsah tohoto atributu přepíše, je tedy nutné uložit jeho obsah do dalšího datového atributu. Bude opět typu *`int`* a můžeme ho pojmenovat např. *`prvniOperand`*, deklarace bude vypadat takto:

```
    private int prvniOperand;
```

V konstruktoru mu přiřadíme jako počáteční hodnotu nulu. Metoda *`plus()`* vypadá nyní následovně:

```
public void plus() {
    prvniOperand = hodnotaKZobrazeni;
    hodnotaKZobrazeni = 0;
}
```

Po vložení dalšího čísla a stisknutí tlačítka rovná se (=) by se na displeji měl zobrazit výsledek.
V metodě *`rovnaSe()`* se sečte první operand s aktuálně vloženým číslem a výsledek se uloží do datového atributu *`hodnotaKZobrazeni`*:

```
public void rovnaSe() {
    hodnotaKZobrazeni = prvniOperand + hodnotaKZobrazeni;
}
```

15.5.5. Operace mínus (−)

Místo plus může uživatel stisknout klávesu minus:

Obdobně jako u sčítání se výsledek počítá v metodě `rovnaSe()`. Abychom odlišili sčítání od odčítání, musíme v metodách `plus()` a `minus()` uložit nejen první operand, ale i typ operace. Pro uložení požadované operace se nabízí několik možností:

- použijeme datový atribut typu `char`,
- použijeme datový atribut typu `String`,
- použijeme datový atribut typu `int` a nadefinujeme konstanty pro jednotlivé operace,
- použijeme výčtový datový typ pro označení operací.

Ukážeme si nejprve použití prvního řešení, potom i řešení dle třetí varianty. Druhá varianta je velmi podobná první, jen si musíme uvědomit, že `String` je referenční datový typ a pro porovnávání je třeba použít metodu `equals()` a ne operátory == či !=. Čtvrtou variantu si ukazovat nebudeme, použití výčtového typu bude ukázáno v projektu Trojúhelníky (viz kapitola 17 str. 168).
Datový atribut typu `char` pro uložení typu operace pojmenujeme `operator` a jeho deklarace bude vypadat takto:

```
     private char operator;
```

V konstruktoru budeme atribut `operator` inicializovat hodnotou mezera (nezapomeňte, že znaky se uvozují apostrofy, ne uvozovkami). Tato hodnota bude znamenat, že není požadována žádná operace. V metodě `plus()` musíme přidat uložení typu operace do proměnné `operator`, metoda `minus()` je velmi podobná metodě `plus()`:

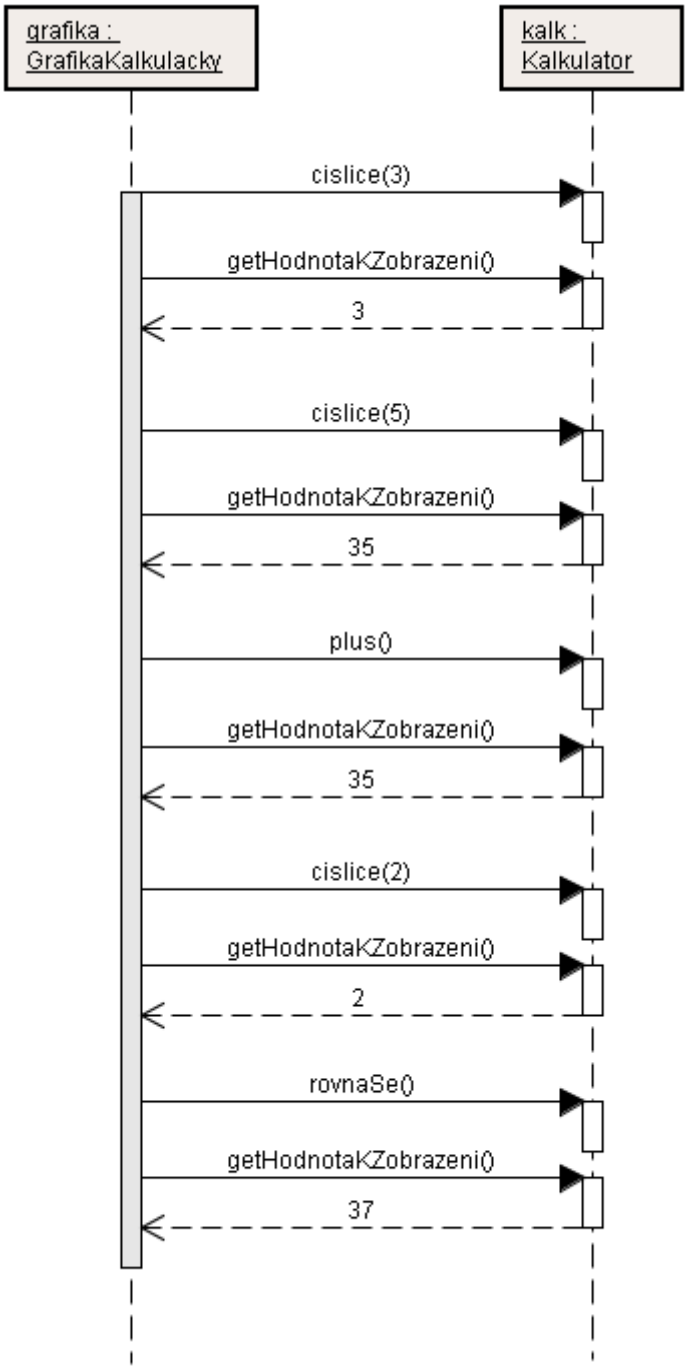
```
public void plus() {
     prvniOperand = hodnotaKZobrazeni;
     hodnotaKZobrazeni = 0;
     operator='+';
}
public void minus() {
     prvniOperand = hodnotaKZobrazeni;
     hodnotaKZobrazeni = 0;
     operator='-';
}
```

V kódu metody `rovnaSe()` je potřeba pomocí selekce `if` rozlišovat, zda uživatel stiskl tlačítko plus či minus. Po provedení výpočtu se do proměnné `operator` vloží mezera:

```
public void rovnaSe() {
     if (operator == '+') {
          hodnotaKZobrazeni = prvniOperand + hodnotaKZobrazeni;
     }
     else {
          if (operator == '-') {
               hodnotaKZobrazeni = prvniOperand - hodnotaKZobrazeni;
          }
     }
     operator=' ';
}
```

15.5.6. Komunikace mezi instancemi

Na obrázku 15.6 je sekvenční diagram zobrazující průběh komunikace mezi instancemi kalkulačky při zadávání výrazu 35 + 2 = 37. Na diagramu není znázorněn vstup od uživatele.



Obrázek 15.6 Diagram zobrazuje průběh komunikace mezi instancemi při zadávání 35+2=

15.5.7. Vkládání dalších čísel

Pokud po stisknutí tlačítka rovná se (=) a zobrazení výsledku uživatel stiskne další číslici, tak se nezačne zobrazovat nové číslo, ale číslice se připojí na konec spočítaného výsledku. Tj. po následující posloupnosti kláves

se na displeji zobrazí číslo 372. Pro ošetření této situace potřebujeme v metodě `cislice()` odlišit situaci, kdy se začne vkládat nová hodnota. Tento datový atribut můžeme pojmenovat např. `noveCislo` a bude typu `boolean`, tj. bude nabývat hodnot `true` (uživatel začíná vkládat nové číslo) nebo `false` (uživatel pokračuje vložením stávajícího čísla). V konstruktoru zinicializujeme tento datový atribut s hodnotou `true` (po spuštění kalkulačky začíná uživatel s vkládáním nového čísla). Na konci metody `rovnaSe()` se přiřadí hodnota `true` do proměnné `noveCislo`.

```
public void rovnaSe() {
     if (operator == '+') {
          hodnotaKZobrazeni = prvniOperand + hodnotaKZobrazeni;
     }
     else {
          if (operator == '-') {
               hodnotaKZobrazeni = prvniOperand - hodnotaKZobrazeni;
          }
     }
     operator=' ';
     noveCislo = true;
}
```

Musíme změnit i kód metody `cislice()`. Pokud je v atributu `noveCislo` hodnota `true`, musíme do atributu `hodnotaKZobrazeni` vložit obsah parametru metody a změnit obsah atributu `noveCislo` na `false`. Metoda `cislice()` bude vypadat takto:

```
public void cislice(int hodnota) {
     if (noveCislo) {
          hodnotaKZobrazeni = hodnota;
          noveCislo = false;
     }
     else {
          hodnotaKZobrazeni = hodnotaKZobrazeni*10 + hodnota;
     }
}
```

Stejnou proměnnou můžeme též využít v metodách `plus()` a `minus()` – po stisknutí klávesy plus zůstane zobrazena předchozí hodnota a po stisknutí další číslice se začne číslo vkládat od nuly. Metoda `plus()` by nyní vypadala takto:

```
public void plus() {
     prvniOperand = hodnotaKZobrazeni;
     noveCislo = true;
     operator='+';
}
```

Metoda `minus()` bude doplněna o přiřazení hodnoty `true` do proměnné `noveCislo`.
Existence nových datových atributů se musí projevit i v kódu metody `vymaz()`. Když uživatel stiskne tlačítko představující tuto operaci, musí se „vynulovat“ všechna nastavení. Kód metody bude vypadat takto:

```
public void vymaz() {
     prvniOperand = 0;
     operator = ' ';
     hodnotaKZobrazeni = 0;
     noveCislo = true;
}
```

15.5.8. Operace plus – pokračování

Zatím jsme uvažovali o situaci, že uživatel sčítal/odčítal jen dvě čísla. Uživatel může však sečíst více čísel:

Na začátku metody `plus()` musíme zjistit, zda uživatel v předchozím kroku již požadoval nějakou operaci nebo ne. Pokud ano, je třeba dříve zadanou operaci (může to být i minus) nejdříve provést, její výsledek zobrazit a též uložit jako `prvniOperand`. Kód metody `plus()` by mohl vypadat takto:

```
public void plus() {
     if (operator == '+') {
          prvniOperand = prvniOperand + hodnotaKZobrazeni;
     }
     else {
          if (operator == '-') {
               prvniOperand = prvniOperand - hodnotaKZobrazeni;
          }
          else {
               prvniOperand = hodnotaKZobrazeni;
          }
     }
     operator='+';
     hodnotaKZobrazeni = prvniOperand;
     noveCislo=true;
}
```

Metoda `minus()` bude z větší části duplicitní s metodou `plus()`. Duplicit v kódu je však ještě více, velká část kódu metody `rovnaSe()` je také stejná. Z důvodu lepší udržovatelnosti a rozšiřovatelnosti kódu je vhodné shodný kód umístit do jedné privátní metody. Pokud do kalkulačky přidáme ještě násobení a dělení, budeme výpočty řešit pouze v jedné metodě. Metodu, do které přesuneme společný kód, nazveme `vypocet()`. Je to pomocná metoda pro jiné metody ze třídy `Kalkulator` a není třeba (není to ani vhodné) ji volat z jiných tříd – proto bude označena modifikátorem `private`. Kód metody `vypocet()` a upravený kód metod `plus()` a `rovnaSe()` vidíme na následujícím výpise. Kód metody `minus()` bude analogický s kódem metody `plus()`.

```
/**
 * metoda se volá při stisknutí tlačítka "+" (plus) na kalkulačce
 */
public void plus() {
     vypocet();
     hodnotaKZobrazeni=prvniOperand;
     noveCislo=true;
     operator='+';
}
/**
* metoda se volá při stisknutí tlačítka "=" (rovná se) na kalkulačce
*/
public void rovnaSe() {
     vypocet();
     hodnotaKZobrazeni = prvniOperand;
     noveCislo=true;
     operator=' ';
}
```

```
/**
 * metoda spočítá mezivýsledek a uloží ho do proměnné prvniOperand
 */
private void vypocet () {
     if (operator == '+') {
          prvniOperand = prvniOperand + hodnotaKZobrazeni;
     }
     else {
          if (operator == '-') {
               prvniOperand = prvniOperand - hodnotaKZobrazeni;
          }
          else {
               prvniOperand = hodnotaKZobrazeni;
          }
     }
}
```

15.5.9. Řešení s konstantami

Řešení, ve kterém jsou pro označení operací místo znaků (typu `char`) použity pojmenované konstanty, se bude od předchozího lišit v několika drobnostech. Konstanty pro jednotlivé operace deklarujeme jako datové atributy s modifikátorem `final`:

```
     private final int ZADNA_OPERACE = 0;
     private final int OPERACE_PLUS = 1;
     private final int OPERACE_MINUS = 2;
```

Datový atribut `operator` bude typu `int` a v konstruktoru nastavíme jeho počáteční hodnotu na `ZADNA_OPERACE`. V metodách `plus()`, `minus()` a `rovnaSe()` budeme místo znaků přiřazovat do tohoto datového atributu odpovídající konstantu. Metoda `vypocet()` se změní takto:

```
/**
 * metoda spočítá mezivýsledek a uloží ho do proměnné prvniOperand
 */
private void vypocet () {
     if (operator == OPERACE_PLUS) {
          prvniOperand = prvniOperand + hodnotaKZobrazeni;
     }
     else {
          if (operator == OPERACE_MINUS) {
               prvniOperand = prvniOperand - hodnotaKZobrazeni;
          }
          else {
               prvniOperand = hodnotaKZobrazeni;
          }
     }
}
```

15.6. Domácí úkoly

1. Zjistěte, zda se po následující kombinaci kláves zobrazí výsledek 2

   tj. že se neprovedla opět operace plus. Pokud ne, tak opravte kód.

2. Kalkulačka ve Windows v následující situaci vypočte 70 (první operand použije i jako druhý operand). Upravte tak kód třídy `Kalkulator`.

3. Upravte kód třídy `Kalkulator` tak, aby se při opakovaném stisku klávesy rovná se zopakovala poslední operace. Např. v následující kombinaci by se měla zobrazit hodnota 39:

4. Zkuste ošetřit přetečení rozsahu čísla `int`, tj. aby po dosažení velikosti čísla `int` nebylo možno vkládat další číslice.

5. Předělejte vnořené příkazy `if` na příkaz `switch` v metodě `vypocet()`.