## Otázka 25

# Jazyk Java – řídící prvky programu

## Podmínky

V Javě se podmínky píší úplně stejně, jako ve všech CLike jazycích. Podmínky zapisujeme pomocí klíčového slova **if**, za kterým následuje logický výraz. Pokud je výraz pravdivý, provede se následující příkaz. Pokud ne, následující příkaz se přeskočí a pokračuje se až pod ním. Vyzkoušejme si to:

Spustit kód Klikni pro editaci

```
if (15 > 5)
   System.out.println("Pravda");
System.out.println("Program zde pokračuje dál");
```

Výstup programu:

```
Konzolová aplikace Pravda
Program zde pokračuje dál
```

Pokud podmínka platí (což zde ano), provede se příkaz vypisující do konzole text pravda. V obou případech program pokračuje dál. Součástí výrazu samozřejmě může být i proměnná:

Spustit kód Klikni pro editaci

```
System.out.println("Zadej nějaké číslo");
int a = Integer.parseInt(sc.nextLine());
if (a > 5)
   System.out.println("Zadal jsi číslo větší než 5!");
System.out.println("Děkuji za zadání");
```

Ukažme si nyní relační operátory, které můžeme ve výrazech používat:

| Operátor | C-like zápis |
|---|---|
| Rovnost | == |
| Je ostře větší | > |
| Je ostře menší | < |
| Je větší nebo rovno | >= |
| Je menší nebo rovno | <= |
| Nerovnost | != |
| Obecná negace | ! |

Rovnost zapisujeme dvěma == proto, aby se to nepletlo s běžným přiřazením do proměnné, které se dělá jen jedním =. Pokud chceme nějaký výraz znegovat, napíšeme ho do závorky a před něj vykřičník. Když budeme chtít vykonat více než jen jeden příkaz, musíme příkazy vložit do bloku ze složených závorek:

Spustit kód Klikni pro editaci

```
Scanner sc = new Scanner(System.in, "Windows-1250");
System.out.println("Zadej nějaké číslo, ze kterého spočítám odmocninu:");
int a = Integer.parseInt(sc.nextLine());
if (a > 0)
{
  System.out.println("Zadal jsi číslo větší než 0, to znamená, že ho mohu odmocnit!");
  double o = Math.sqrt(a);
  System.out.println("Odmocnina z čísla " + a + " je " + o);
}
System.out.println("Děkuji za zadání");
```

Konzolová aplikace Zadej nějaké číslo, ze kterého spočítám odmocninu:
144
Zadal jsi číslo větší než 0, to znamená, že ho mohu odmocnit!
Odmocnina z čísla 144 je 12.0
Děkuji za zadání

Často můžete vidět použití bloku i v případě, že je pod podmínkou jen jeden příkaz, mnohdy je to totiž přehlednější. Nezapomeňte si naimportovat java.util.Scanner, aby program znal třídu Scanner.

Program načte od uživatele číslo a pokud je větší než 0, vypočítá z něj druhou odmocninu. Mimo jiné jsme použili třídu Math, která na sobě obsahuje řadu užitečných matematických metod, někdy si ji blíže představíme. Sqrt() vrací hodnotu jako double. Bylo by hezké, kdyby nám program vyhuboval v případě, že zadáme záporné číslo. S dosavadními znalostmi bychom napsali něco jako:

Spustit kód Klikni pro editaci

```
Scanner sc = new Scanner(System.in, "Windows-1250");
System.out.println("Zadej nějaké číslo, ze kterého spočítám odmocninu:");
int a = Integer.parseInt(sc.nextLine());
if (a > 0)
{
  System.out.println("Zadal jsi číslo větší než 0, to znamená, že ho mohu odmocnit!");
  double o = Math.sqrt(a);
  System.out.println("Odmocnina z čísla " + a + " je " + o);
}
if (a <= 0)
{
  System.out.println("Odmocnina ze záporného čísla neexistuje!");
}
System.out.println("Děkuji za zadání");
```

Všimněte si, že musíme pokrýt i případ, kdy se a == 0, nejen když je menší. Kód však můžeme výrazně zjednodušit pomocí klíčového slova **else**, které vykoná následující příkaz nebo blok příkazů **v případě, že se podmínka neprovede**:

Spustit kód Klikni pro editaci

```
Scanner sc = new Scanner(System.in, "Windows-1250");
System.out.println("Zadej nějaké číslo, ze kterého spočítám odmocninu:");
int a = Integer.parseInt(sc.nextLine());
if (a > 0)
{
  System.out.println("Zadal jsi číslo větší než 0, to znamená, že ho mohu odmocnit!");
  double o = Math.sqrt(a);
  System.out.println("Odmocnina z čísla " + a + " je " + o);
}
else
{
  System.out.println("Odmocnina ze záporného čísla neexistuje!");
}
System.out.println("Děkuji za zadání");
```

Kód je mnohem přehlednější a nemusíme vymýšlet opačnou podmínku, což by v případě složené podmínky mohlo být někdy i velmi obtížné. V případě více příkazů by byl za else opět blok { }.

Else se také využívá v případě, kdy potřebujeme v příkazu manipulovat s proměnnou z podmínky a nemůžeme se na ni tedy ptát potom znovu. Program si sám pamatuje, že se podmínka nesplnila a přejde do sekce else. Ukažme si to na příkladu: Mějme číslo a, kde bude hodnota 0 nebo 1 a po nás se bude chtít, abychom hodnotu prohodili (pokud tam je 0, dáme tam 1, pokud 1, dáme tam 0). Naivně bychom mohli kód napsat takto:

Spustit kód Klikni pro editaci

```
int a = 0; // do a si přiřadíme na začátku 0

if (a == 0) // pokud je a 0, dáme do něj jedničku
{
  a = 1;
}
if (a == 1) // pokud je a 1, dáme do něj nulu
{
  a = 0;
}

System.out.println(a);
```

Nefunguje to, že? Pojďme si projet, co bude program dělat. Na začátku máme v *a* nulu, první podmínka se jistě splní a dosadí do *a* jedničku. No ale rázem se splní i ta druhá. Co s tím? Když podmínky otočíme, budeme mít ten samý problém s jedničkou. Jak z toho ven? Ano, použijeme else.

Spustit kód Klikni pro editaci

```
int a = 0; // do a si přiřadíme na začátku 0
```

```
        if (a == 0) // pokud je a 0, dáme do něj jedničku
        {
           a = 1;
        }
        else // pokud je a 1, dáme do něj nulu
        {
           a = 0;
        }

        System.out.println(a);
```

Podmínky je možné skládat a to pomocí dvou základních logických operátorů:

| Operátor | C-like Zápis |
|---|---|
| A zároveň | && |
| Nebo | \|\| |

Uveďme si příklad:

Spustit kód Klikni pro editaci

```
        Scanner sc = new Scanner(System.in, "Windows-1250");
        System.out.println("Zadejte číslo v rozmezí 10-20:");
        int a = Integer.parseInt(sc.nextLine());
        if ((a >= 10) && (a <= 20))
        {
           System.out.println("Zadal jsi správně");
        }
        else
        {
           System.out.println("Zadal jsi špatně");
        }
```

S tím si zatím vystačíme, operátory se pomocí závorek samozřejmě dají kombinovat.

Spustit kód Klikni pro editaci

```
        Scanner sc = new Scanner(System.in, "Windows-1250");
        System.out.println("Zadejte číslo v rozmezí 10-20 nebo 30-40:");
        int a = Integer.parseInt(sc.nextLine());
        if (((a >= 10) && (a <= 20)) || ((a >=30) && (a <= 40)))
        {
           System.out.println("Zadal jsi správně");
        }
        else
        {
           System.out.println("Zadal jsi špatně");
        }
```

### Switch

Switch je konstrukce, převzatá z jazyka C (jako většina gramatiky Javy). Umožňuje nám zjednodušit (relativně) zápis více podmínek pod sebou. Vzpomeňme si na naši kalkulačku v prvních lekcích, která načetla 2 čísla a vypočítala všechny 4 operace. Nyní si ale budeme chtít zvolit, kterou operaci chceme. Bez switche bychom napsali kód podobný tomuto:

Spustit kód Klikni pro editaci

```
        Scanner sc = new Scanner(System.in, "Windows-1250");
        System.out.println("Vítejte v kalkulačce");
        System.out.println("Zadejte první číslo:");
        float a = Float.parseFloat(sc.nextLine());
        System.out.println("Zadejte druhé číslo:");
        float b = Float.parseFloat(sc.nextLine());
        System.out.println("Zvolte si operaci:");
        System.out.println("1 - sčítání");
        System.out.println("2 - odčítání");
        System.out.println("3 - násobení");
        System.out.println("4 - dělení");
        int volba = Integer.parseInt(sc.nextLine());
        float vysledek = 0;
        if (volba == 1)
        {
            vysledek = a + b;
        }
        else if (volba == 2)
        {
            vysledek = a - b;
        }
        else if (volba == 3)
        {
            vysledek = a * b;
        }
        else if (volba == 4)
        {
            vysledek = a / b;
        }
        if ((volba > 0) && (volba < 5))
        {
            System.out.println("Výsledek: " + vysledek);
        }
        else
        {
            System.out.println("Neplatná volba");
        }
        System.out.println();
        System.out.println("Děkuji za použití kalkulačky.");
```

Konzolová aplikace Vítejte v kalkulačce
Zadejte první číslo:
3.14
Zadejte druhé číslo:
2.72
Zvolte si operaci:
1 - sčítání
2 - odčítání
3 - násobení
4 - dělení
2
Výsledek: 0.42
Děkuji za použití kalkulačky.

Všimněte si, že jsme proměnnou výsledek deklarovali na začátku, jen tak do ni můžeme potom přiřazovat. Kdybychom ji deklarovali u každého přiřazení, Java by kód nezkompilovala a vyhodila chybu redeklarace proměnné. Důležité je také přiřadit výsledku nějakou výchozí hodnotu, zde nula, jinak by nám Java vyhubovala, že se snažíme vypsat proměnnou, která nebyla jednoznačně inicializována. Proměnná může být deklarována (založena v paměti) vždy jen jednou. Další vychytávka je kontrola správnosti volby. Program by v tomto případě fungoval stejně i bez těch else, ale nač se dále ptát, když již máme výsledek.

Nyní si zkusíme napsat ten samý kód pomocí switche:

Spustit kód Klikni pro editaci

```
Scanner sc = new Scanner(System.in, "Windows-1250");
System.out.println("Vítejte v kalkulačce");
System.out.println("Zadejte první číslo:");
float a = Float.parseFloat(sc.nextLine());
System.out.print("Zadejte druhé číslo:");
float b = Float.parseFloat(sc.nextLine());
System.out.println("Zvolte si operaci:");
System.out.println("1 - sčítání");
System.out.println("2 - odčítání");
System.out.println("3 - násobení");
System.out.println("4 - dělení");
int volba = Integer.parseInt(sc.nextLine());
float vysledek = 0;
switch (volba)
{
  case 1:
    vysledek = a + b;
  break;
  case 2:
    vysledek = a - b;
  break;
  case 3:
    vysledek = a * b;
  break;
  case 4:
    vysledek = a / b;
  break;
}
```

```
        if ((volba > 0) && (volba < 5))
        {
          System.out.println("Výsledek: " + vysledek);
        }
        else
        {
          System.out.println("Neplatná volba");
        }
        System.out.println();
        System.out.println("Děkuji za použití kalkulačky.");


```

Vidíme, že kód je trochu přehlednější. Pokud bychom potřebovali v nějaké větvi switche spustit více příkazů, překvapivě je nebudeme psát do bloku, ale rovnou pod sebe. Blok {} nám zde nahrazuje příkaz break, který způsobí vyskočení z celého switche. Switch může místo case x: obsahovat ještě možnost default:, která se vykoná v případě, že nebude platit žádný case. Je jen na vás, jestli budete switch používat, obecně se vyplatí jen při větším množství příkazů a vždy jde nahradit sekvencí if a else. Nezapomínejte na breaky. Switch je v Javě podporován i pro hodnoty stringové proměnné a to od Javy 7.

# Cykly

Jak již slovo cyklus napoví, něco se bude opakovat. Když chceme v programu něco udělat 100x, jistě nebudeme psát pod sebe 100x ten samý kód, ale vložíme ho do cyklu. Cyklů máme několik druhů, vysvětlíme si, kdy který použít. Samozřejmě si ukážeme praktické příklady.

### FOR cyklus

Tento cyklus má stanovený **pevný počet opakování** a hlavně obsahuje tzv. řídící proměnnou (celočíselnou), ve které se postupně během běhu cyklu mění hodnoty. Syntaxe (zápis) cyklu for je následující:

```
for (promenna; podminka; prikaz)
```

- **promenna** je řídící proměnná cyklu, které nastavíme počáteční hodnotu (nejčastěji 0, protože v programování vše začíná od nuly, nikoli od jedničky). Např. tedy int i = 0. Samozřejmě si můžeme proměnnou i vytvořit někde nad tím a už nemusíme psát slovíčko int, bývá ale zvykem používat právě int i.
- **podminka** je podmínka vykonání dalšího kroku cyklu. Jakmile nebude platit, cyklus se ukončí. Podmínka může být např (i < 10).
- **prikaz** nám říká, co se má v každém kroku s řídící proměnnou stát. Tedy zda se má zvýšit nebo snížit. K tomu využijeme speciálních operátorů ++ a --, ty samozřejmě můžete používat i úplně běžně mimo cyklus, slouží ke zvýšení nebo snížení proměnné o 1.

Pojďme si udělat jednoduchý příklad, většina z nás jistě zná Sheldona z The Big Bang Theory. Pro ty co ne, budeme simulovat situaci, kdy klepe na dveře své sousedky. Vždy 3x zaklepe a poté zavolá: "Penny!". Náš kód by bez cyklů vypadal takto:

Spustit kód Klikni pro editaci

```
        System.out.println("Knock");
        System.out.println("Knock");
        System.out.println("Knock");
        System.out.println("Penny!");
```

My ale už nic nemusíme otrocky opisovat:

Spustit kód Klikni pro editaci

```
        for (int i=0; i < 3; i++)
        {
           System.out.println("Knock");
        }
        System.out.println("Penny!");
```

Konzolová aplikace Knock
Knock
Knock
Penny!

Cyklus proběhne 3x, zpočátku je v proměnné *i* nula, cyklus vypíše "Knock" a zvýší proměnnou *i* o jedna. Poté běží stejně s jedničkou a dvojkou. Jakmile je v *i* trojka, již nesouhlasí podmínka i < 3 a cyklus končí. O vynechávání složených závorek platí to samé, co u podmínek. V tomto případě tam nemusí být, protože cyklus spouští pouze jediný příkaz. Nyní můžeme místo trojky napsat do deklarace cyklu desítku. Příkaz se spustí 10x aniž bychom psali něco navíc. Určitě vidíte, že cykly jsou mocným nástrojem.

Zkusme si nyní využít toho, že se nám proměnná inkrementuje. Vypišme si čísla od jedné do deseti a za každým mezeru.

Spustit kód Klikni pro editaci

```
        for (int i = 1; i <= 10; i++)
        {
           System.out.printf("%d ", i);
        }
```

Vidíme, že řídící proměnná má opravdu v každé iteraci (průběhu) jinou hodnotu.

Pokud vás zmátlo použití printf(), můžeme místo ní použít pouze print(), která na rozdíl od println() po vypsání neodřádkuje:

Spustit kód Klikni pro editaci

```
        for (int i = 1; i <= 10; i++)
        {
           System.out.print(i + " ");
```

```
}
```

Nyní si vypíšeme malou násobilku (násobky čísel 1 až 10, vždy do deseti). Stačí nám udělat cyklus od 1 do 10 a proměnnou vždy násobit daným číslem. Mohlo by to vypadat asi takto:

Spustit kód Klikni pro editaci

```
for (int i = 1; i <= 10; i++)
{
   System.out.print(i + " ");
}
System.out.println();
for (int i = 1; i <= 10; i++)
{
   System.out.print((i * 2) + " ");
}
System.out.println();
for (int i = 1; i <= 10; i++)
{
   System.out.print((i * 3) + " ");
}
System.out.println();
for (int i = 1; i <= 10; i++)
{
   System.out.print((i * 4) + " ");
}
System.out.println();
for (int i = 1; i <= 10; i++)
{
   System.out.print((i * 5) + " ");
}
System.out.println();
for (int i = 1; i <= 10; i++)
{
   System.out.print((i * 6) + " ");
}
System.out.println();
for (int i = 1; i <= 10; i++)
{
   System.out.print((i * 7) + " ");
}
System.out.println();
for (int i = 1; i <= 10; i++)
{
   System.out.print((i * 8) + " ");
}
System.out.println();
for (int i = 1; i <= 10; i++)
{
   System.out.print((i * 9) + " ");
}
System.out.println();
```

```
        for (int i = 1; i <= 10; i++)
        {
            System.out.print((i * 10) + " ");
        }
```

Konzolová aplikace 1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100

Program funguje hezky, ale pořád jsme toho dost napsali. Pokud vás napadlo, že v podstatě děláme 10x to samé a pouze zvyšujeme číslo, kterým násobíme, máte pravdu. Nic nám nebrání vložit 2 cykly do sebe:

Spustit kód Klikni pro editaci

```
        System.out.println("Malá násobilka pomocí dvou cyklů:");
        for (int j = 1; j <= 10; j++)
        {
            for (int i = 1; i <= 10; i++)
            {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
```

Poměrně zásadní rozdíl, že? Pochopitelně nemůžeme použít u obou cyklů *i*, protože jsou vložené do sebe. Proměnná *j*nabývá ve vnějším cyklu hodnoty 1 až 10. V každé iteraci (rozumějte průběhu) cyklu je poté spuštěn další cyklus s proměnnou *i*. Ten je nám již známý, vypíše násobky, v tomto případě násobíme proměnnou *j*. Po každém běhu vnitřního cyklu je třeba odřádkovat, to vykoná System.out.println().

Udělejme si ještě jeden program, na kterém si ukážeme práci s vnější proměnnou. Aplikace bude umět spočítat libovolnou mocninu libovolného čísla:

Spustit kód Klikni pro editaci

```
        Scanner sc = new Scanner(System.in, "Windows-1250");
        System.out.println("Mocninátor");
        System.out.println("==========");
        System.out.println("Zadejte základ mocniny: ");
        int a = Integer.parseInt(sc.nextLine());
        System.out.println("Zadejte exponent: ");
```

```
        int n = Integer.parseInt(sc.nextLine());

        int vysledek = a;
        for (int i = 0; i < (n - 1); i++)
        {
           vysledek = vysledek * a;
        }

        System.out.println("Výsledek: " + vysledek);
        System.out.println("Děkuji za použití mocninátoru");


```

Asi všichni tušíme, jak funguje mocnina. Pro jistotu připomenu, že například $2^3 = 2 * 2 * 2$. Tedy $a^n$ spočítáme tak, že n-1 krát vynásobíme číslo *a* číslem *a*. Výsledek si samozřejmě musíme ukládat do proměnné. Zpočátku bude mít hodnotu *a*a postupně se bude v cyklu pronásobovat. Pokud jste to nestihli, máme tu samozřejmě článek s algoritmem výpočtu libovolné mocniny . Vidíme, že naše proměnná *vysledek* je v těle cyklu normálně přístupná. Pokud si však nějakou proměnnou založíme v těle cyklu, po skončení cyklu zanikne a již nebude přístupná.

```
Konzolová aplikace Mocninátor
==========
Zadejte základ mocniny:
2
Zadejte exponent:
3
Výsledek: 8
Děkuji za použití mocninátoru
```

Už tušíme, k čemu se for cyklus využívá. Zapamatujme si, že je **počet opakování pevně daný**. Do proměnné cyklu bychom neměli nijak zasahovat ani dosazovat, program by se mohl tzv. zacyklit, zkusme si ještě poslední, odstrašující příklad:

Spustit kód Klikni pro editaci

```
        // tento kód je špatně
        for (int i = 1; i <= 10; i++)
        {
           i = 1;
        }


```

Au, vidíme, že program se zasekl. Cyklus stále inkrementuje proměnnou *i*, ale ta se vždy sníží na 1. Nikdy tedy nedosáhne hodnoty > 10, cyklus nikdy neskončí. Program zastavíme tlačítkem Stop u okna konzole.

## While cyklus

While cyklus funguje jinak, jednoduše opakuje příkazy v bloku dokud platí podmínka. Syntaxe cyklu je následující:

```
while (podminka)
```

```
{
   // příkazy
}
```

Pokud vás napadá, že lze přes while cyklus udělat i FOR cyklus, máte pravdu FOR je vlastně speciální případ while cyklu. While se ale používá na trochu jiné věci, často máme v jeho podmínce např. metodu vracející logickou hodnotu true/false. Původní příklad z for cyklu bychom udělali následovně pomocí while:

Spustit kód Klikni pro editaci

```
int i = 1;
while (i <= 10)
{
   System.out.print(i + " ");
   i++;
}
```

To ale není ideální použití while cyklu. Vezmeme si naši kalkulačku z minulých lekcí a opět ji trochu vylepšíme, konkrétně o možnost zadat více příkladů. Program tedy hned neskončí, ale zeptá se uživatele, zda si přeje spočítat další příklad. Připomeňme si původní verzi kódu (je to ta verze se switchem, ale klidně použijte i tu bez něj, záleží na vás):

Spustit kód Klikni pro editaci

```
System.out.println("Vítejte v kalkulačce");
System.out.println("Zadejte první číslo:");
float a = Float.parseFloat(sc.nextLine());
System.out.println("Zadejte druhé číslo:");
float b = Float.parseFloat(sc.nextLine());
System.out.println("Zvolte si operaci:");
System.out.println("1 - sčítání");
System.out.println("2 - odčítání");
System.out.println("3 - násobení");
System.out.println("4 - dělení");
int volba = Integer.parseInt(sc.nextLine());
float vysledek = 0;
switch (volba)
{
   case 1:
      vysledek = a + b;
   break;
   case 2:
      vysledek = a - b;
   break;
   case 3:
      vysledek = a * b;
   break;
   case 4:
      vysledek = a / b;
   break;
}
```

```
        if ((volba > 0) && (volba < 5))
        {
            System.out.println("Výsledek: " + vysledek);
        }
        else
        {
            System.out.println("Neplatná volba");
        }
        System.out.println("Děkuji za použití kalkulačky.");


```

Nyní vložíme téměř celý kód do while cyklu. Naší podmínkou bude, že uživatel zadá "ano", budeme tedy kontrolovat obsah proměnné *pokracovat*. Zpočátku bude tato proměnná nastavena na "ano", aby se program vůbec spustil, poté do ní necháme načíst volbu uživatele:

```
Scanner sc = new Scanner(System.in, "Windows-1250");

System.out.println("Vítejte v kalkulačce");
String pokracovat = "ano";
while (pokracovat.equals("ano"))
{
  System.out.println("Zadejte první číslo:");
  float a = Float.parseFloat(sc.nextLine());
  System.out.println("Zadejte druhé číslo:");
  float b = Float.parseFloat(sc.nextLine());
  System.out.println("Zvolte si operaci:");
  System.out.println("1 - sčítání");
  System.out.println("2 - odčítání");
  System.out.println("3 - násobení");
  System.out.println("4 - dělení");
  int volba = Integer.parseInt(sc.nextLine());
  float vysledek = 0;
  switch (volba)
  {
    case 1:
        vysledek = a + b;
    break;
    case 2:
        vysledek = a - b;
    break;
    case 3:
        vysledek = a * b;
    break;
    case 4:
        vysledek = a / b;
    break;
  }
  if ((volba > 0) && (volba < 5))
  {
    System.out.println("Výsledek: " + vysledek);
  }
  else
  {
```

```
        System.out.println("Neplatná volba");
    }
    System.out.println("Přejete si zadat další příklad? [ano/ne]");
    pokracovat = sc.nextLine();
}
System.out.println("Děkuji za použití kalkulačky.");
```

Všimněte si, že **Stringy porovnáváme pomocí metody equals(), nikoli pomocí operátoru ==!** Je to dáno tím, že String je referenční datový typ. Podmínka ("Text" == "Text") je špatně, musíme psát ("Text".equals("Text")). V kapitole o objektově orientovaném programování pochopíme proč.

Konzolová aplikace Vítejte v kalkulačce
Zadejte první číslo:
12
Zadejte druhé číslo:
128
Zvolte si operaci:
1 - sčítání
2 - odčítání
3 - násobení
4 - dělení
1
Výsledek: 140
Přejete si zadat další příklad? [ano/ne]
ano
Zadejte první číslo:
-10.5
Zadejte druhé číslo: