

Лекция 3.1

Аритметични операции и въведение в
управление на логиката на
изпълнение.

Графичен вход и изход на данни с
диалогов прозорец.

Форматиране на String.

Основни теми

- Аритметични операции
- Въведение в управление на логиката на изпълнение- оператор *if* и оператори за сравнение
- Вход и изход на данни в графичен режим
- Преобразуване на низ от цифри в числа
- Форматиране на низ
- Обобщение
- Задачи

- 3.1** Аритметични операции
- 3.2** Управление на логиката при изпълнение на команди
- 3.3** Извеждане на текст в Dialog Box
- 3.4** Въвеждане на текст с Dialog Box
- 3.5** Въвеждане на цифрови данни с графичен интерфейс Dialog Box
- 3.6** Обобщение
- 3.7** Задачи

Литература:

Java How to Program, Sixth Edition, Chapter 2 & 3

3.1 Аритметика

- Аритметични операции

- Означения наредени по приоритет на изпълнение

- $*$ умножение
 - $/$ делене
 - $\%$ делене по модул
 - $+$, $-$ събиране и изваждане

- Важно: *целочисленото делене отрязва остътъка от делене*

$7 / 5$ се пресмята равно на 1

Обаче, при *делене с плаваща запетая*

$7.0 / 5$ се пресмята равно на 1.4

- Остатъкът от делене $\%$ се получава като *делене по модул*

$7 \% 5$ се пресмята равно на 2

Java operation	Arithmetic operator	Algebraic expression	Java expression
Addition	+	$f + 7$	<code>f + 7</code>
Subtraction	-	$p - c$	<code>p - c</code>
Multiplication	*	Bm	<code>b * m</code>
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	<code>x / y</code>

Fig. 2. 11 | Аритметични оператори.

3.1 Аритметика ...

- **Приоритет на операциите**

- Някои операции се изпълняват преди другите (т.е. Умножение преди събиране и пр.)

- **ИЗПОЛЗВАЙТЕ КРЪГЛИ СКОБИ**

- Пример: Намерете средната стойност на a , b и c

- **ГРЕШНО:** $a + b + c / 3$

- **ПРАВИЛНО:** $(a + b + c) / 3$

Operator(s)	Operation(s)	Order of evaluation (precedence)
*	Multiplication	Evaluated first. If there are several operators of this type, they are evaluated from left to right.
/	Division	
%	Remainder	
+	Addition	Evaluated next. If there are several operators of this type, they are evaluated from left to right.
-	Subtraction	

Fig. 2.12 | Приоритет на аритметичните операции.

Правила за добро програмиране 2.14

Използването на скоби при сложни аритметични изрази, прави тези изрази лесни за разчитане и проверка.

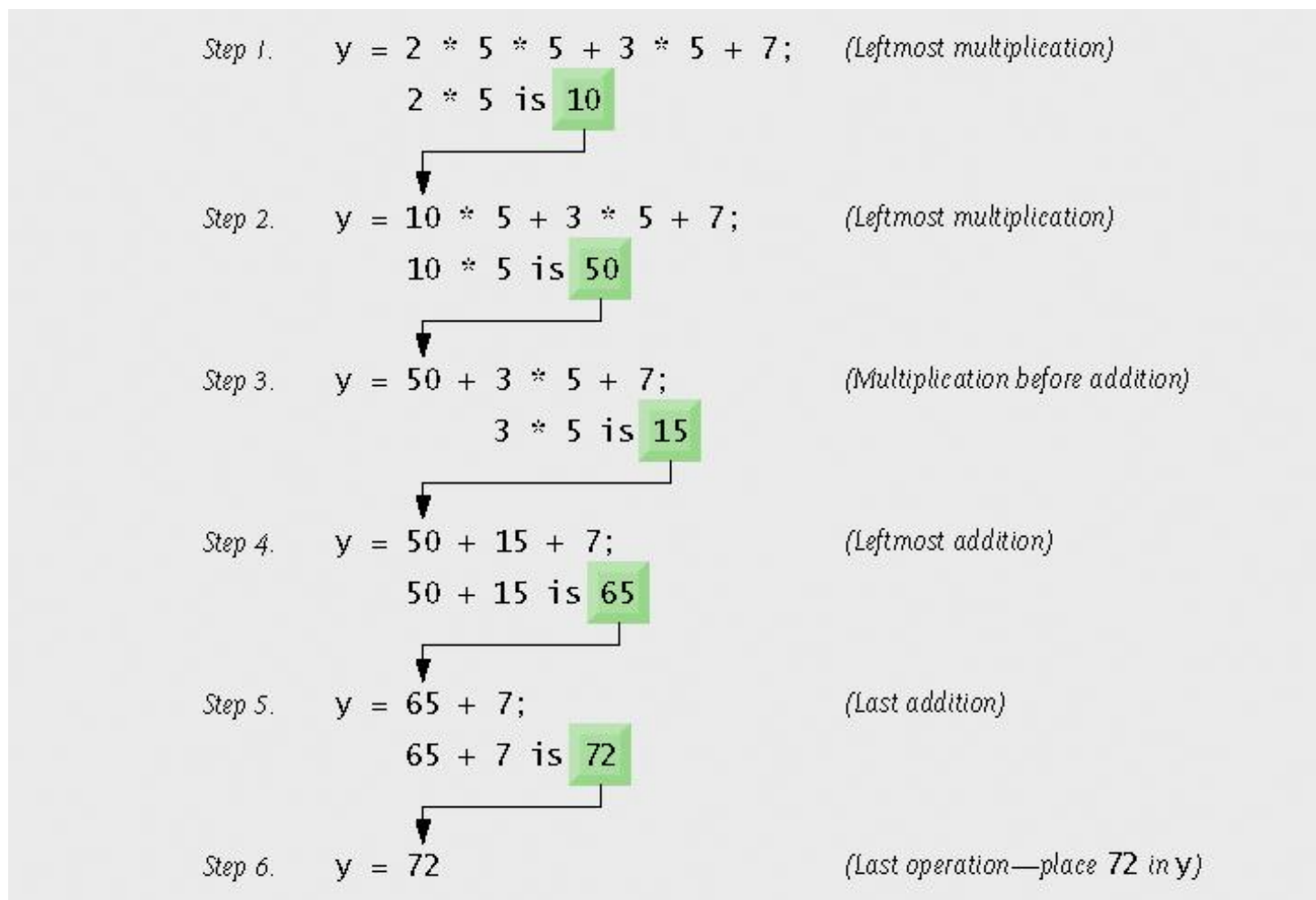


Fig. 2.13 | Порядък, в който се пресмята полином от втора степен.

Обичайна грешка при програмиране 2.10

Когато **двата операнда** на аритметична операция са от **целочислен тип данни**, то и резултата от аритметичната операция е от **целочислен тип**.

Когато **поне единият** от двата операнда на аритметичната операция е от **тип данни с плаваща запетая**, то и резултата от аритметичната операция е от **плаваща запетая**. Например,

```
double countInt  = 3 / 4;    // countInt = 0.0  
double countDb1  = 3.0 /4;   // countDb1 = 0.75
```

3.2 Управление на логиката при изпълнение на команди

- Условие за преход

- Включва пресмятане на израз, чиито резултат е **true** или **false**

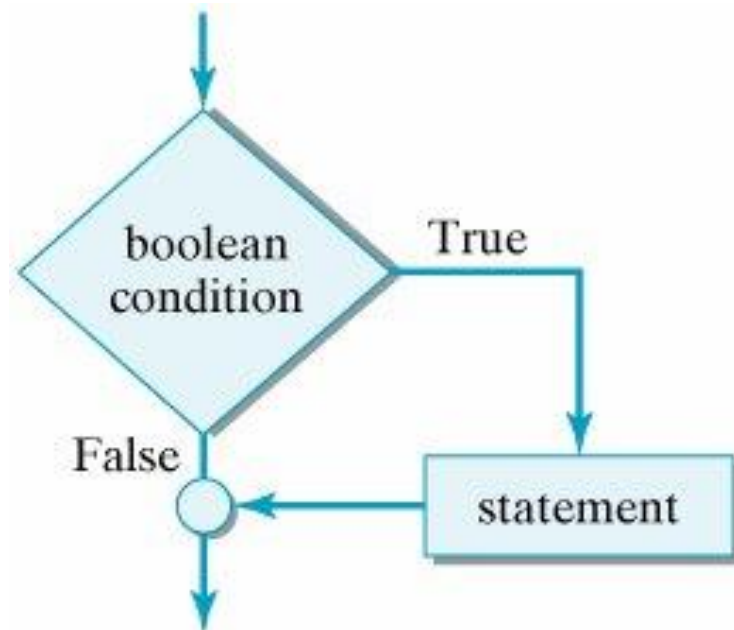
- **if** инструкция

- Тук е разгледана в **най-опростен вариант**

- Ако условието за преход е **true**, тогава се изпълнява тялото на **if** инструкцията (**boolean statement**)

- Управлението на програмата винаги продължава в края на **if** инструкцията

- Условието за преход **if** инструкциите се формират от операции за сравнения или логически изрази (next slide)



Standard algebraic equality or relational operator	Java equality or relational operator	Sample Java condition	Meaning of Java condition
<i>Equality operators</i>			
=	==	x == y	x е равно на y
≠	!=	x != y	x не е равно на y
<i>Relational operators</i>			
>	>	x > y	x е по- голямо от y
<	<	x < y	x е по- малко от y
≥	>=	x >= y	x е по-голямо или равно на y
≤	<=	x <= y	x е по-малко или равно на y

Fig. 2.14 | Оператори за равенство и сравнения.

Outline

Comparison.java

(1 of 2)

1. Class Comparison

1.1 main

1.2 Declarations

1.3 Input data (nextInt)

1.4 Compare two inputs using if statements

```

1 // Fig. 2.15: Comparison.java
2 // Compare integers using if statements, relational operators
3 // and equality operators.
4 import java.util.Scanner; // program uses class Scanner
5
6 public class Comparison
7 {
8     // main method begins execution of Java application
9     public static void main( String args[] )
10    {
11        // create Scanner to obtain input from command window
12        Scanner input = new Scanner( System.in );
13
14        int number1; // first number to compare
15        int number2; // second number to compare
16
17        System.out.print( "Enter first integer: " ); // prompt
18        number1 = input.nextInt(); // read first number from user
19
20        System.out.print( "Enter second integer: " ); // prompt
21        number2 = input.nextInt(); // read second number from user
22
23        if ( number1 == number2 )
24            System.out.printf( "%d == %d\n", number1, number2 );
25
26        if ( number1 != number2 )
27            System.out.printf( "%d != %d\n", number1, number2 );
28
29        if ( number1 < number2 )
30            System.out.printf( "%d < %d\n", number1, number2 );

```

Test за равенство, извежда
резултата с printf.

%d Форматен спецификатор за
целочислени данни

Сравнява две числа с
оператор <.



Outline

Comparison.java

(2 of 2)

Compares two numbers using relational operator >, <= and >=.

```
31 if ( number1 > number2 )
32     System.out.printf( "%d > %d\n", number1, number2 );
33
34
35 if ( number1 <= number2 )
36     System.out.printf( "%d <= %d\n", number1, number2 );
37
38 if ( number1 >= number2 )
39     System.out.printf( "%d >= %d\n", number1, number2 );
40
41 } // end method main
42
43 } // end class Comparison
```

Program output

```
Enter first integer: 777
Enter second integer: 777
777 == 777
777 <= 777
777 >= 777
```

```
Enter first integer: 1000
Enter second integer: 2000
1000 != 2000
1000 < 2000
1000 <= 2000
```

```
Enter first integer: 2000
Enter second integer: 1000
2000 != 1000
2000 > 1000
2000 >= 1000
```



3.2 Управление на логиката ...

- Line 6: започва декларацията на class *Comparison*
- Line 12: декларира променлива *input* реферираща *Scanner* и ѝ присвоява *Scanner* обект, който въвежда данни от Стандартен вход
- *input* е пример за променлива от референтен тип
- Lines 14-15: декларира *int* променливи (прост тип данни)
- Забележете: Методът *nextInt()* на *Scanner* обект служи за въвеждане на следващо цяло число от Стандартен вход
- Lines 17-18: подканя потребителя със съобщение (prompt) да въведе първото цяло число и присвоява въведеното число на *number1*
- Lines 20-21: подканя потребителя със съобщение (prompt) да въведе първото цяло число и присвоява въведеното число на *number2*

3.2 Управление на логиката ...

```
23     if ( number1 == number2 )
24         System.out.printf( "%d == %d\n", number1, number2 );
```

- Ако променливите са равни (условието за преход е `true`)
 - **Line 24** се изпълнява
- Ако променливите **не** са равни, **Line 24** се пропуска
- Няма точка и запетая в края на **if** инструкцията
- **Празна** инструкция (команда)
 - Нищо не се изпълнява в при празна инструкция
- **Lines 26-27, 29-30, 32-33, 35-36 и 38-39**
 - сравняваме **number1** и **number2** , съответно посредством операторите **!=**, **<**, **>**, **<=** и **>=**
- **Запомнете:**
 - **%d** е форматен спецификатор за целочислени данни
 - **%s** е форматен спецификатор за String (текстови) данни
 - **Използват се с `printf()` и `String.format()`** (ще научим след малко)

Обичайна грешка при програмиране 2.9

Пропускането на някоя от фигурните скоби за условието за преход на `if` инструкцията е синтактична грешка- тези скоби не трябва да се пропускат!

Обичайна грешка при програмиране 2.10

Смесването на оператора за равенство, `==`, с оператора за присвояване, `=`, води до логическа и синтактична грешка.

Операторът за равенство се чете като “.. е равно на ..,” докато операторът за присвояване се чете като “*взима*” или “*получава стойността на.*”

Обичайна грешка при програмиране 2.11

Синтактична грешка е ако операторите **==**, **!=**, **>=** и **<=** съдържат шпация(празен символ) между тях като например, **= =**, **! =**, **>** и **< =**, което е **ГРЕШНО!**

Обичайна грешка при програмиране 2.12

Пренареждането на символите в операторите \neq , \geq и \leq , като например $=!$, $=>$ и $=<$, е синтактична грешка.

Правила за добро програмиране 2.15

Подравнявайте тялото на `if` инструкцията за да си личи къде започва и свършва. Това допринася съществено за проверка на програмата за логически и синтактични грешки.

Правила за добро програмиране 2.16

Пишете по една инструкция/ команда на ред.

Това допринася за добра читаемост на програмата

Обичайна грешка при програмиране 2.13

Поставянето на **;** веднага след условието за преход на **if** е логическа грешка.

Правила за добро програмиране 2.17

Пренасяйте дълги команди на няколко реда.

Ако трябва пренасяне намерете подходящи точки за прекъсване на инструкцията като например, запетая на списък от аргументи или оператор в дълъг аритметичен израз. В такъв случай също подравнете в дясно частите на всеки отделен ред.

Operators				Associativity	Type
*	/	%		left to right	multiplicative
+	-			left to right	additive
<	<=	>	>=	left to right	relational
==	!=			left to right	equality
=				right to left	assignment

Fig. 2.16 | Приоритет и асциативност на операторите.

3.3 Извеждане на текст в Dialog Box

Графичен изход на данни

Повечето *Java* приложения използват windows или dialog box за **графично** извеждане на данни

Текстово извеждане на Command prompt window данни само на Стандартен изход ()

class JOptionPane осигурява използване на диалогови прозорци за вход и изход на данни

Библиотеки на *Java* - packages

Съвкупност от предефинирани класове за приложения

Класове , логически обособени в групи се наричат **packages**

Съвкупността от всички packages се наричат Java class library или Java Applications programming Interface (Java **API**)

JOptionPane е от ***javax.swing*** package (библиотека)

Тази библиотека (package) съдържа класове за работа с графичен потребителски интерфейс **Graphical User Interfaces (GUIs)**

Outline

Dialog1.java

```
1 // Fig. 3.17: Dialog1.java
2 // Printing multiple lines in dialog box.
3 import javax.swing.JOptionPane; // import class JOptionPane
4
5 public class Dialog1
6 {
7     public static void main( String args[] )
8     {
9         // display a dialog with the message
10        JOptionPane.showMessageDialog( null, "Welcome\nto\nJava" );
11        System.exit( 0 ); // terminate application with window
12    } // end main
13 } // end class Dialog1
```

Import class JOptionPane

Покажи този низ в диалогов прозорец



3.3 Извеждане на текст в Dialog Box

Lines 1-2: коментари

Две групи библиотеки (packages) на Java API

Базови библиотеки (core packages)

Имената им започват с java. Например, `java.util`, `java.lang`

Включени в Java 2 Software Development Kit

Разширения на базовите библиотеки (extension packages)

Например, започват с `javax`

Нови Java библиотеки

```
3 import javax.swing.JOptionPane; // program uses JOptionPane
```

`import` декларации

Използват се от компилатора, за да се означат класовете **външните за приложението класове** и да се **намерят техните дефиниции** в съответната Java библиотека

В този пример, указват на компилатора да зареди `class JOptionPane` от `javax.swing` библиотеката

3.3 Извеждане на текст в Dialog Box

Line 6-11: празен ред , започва *class Dialog1* and *main*

```
10      JOptionPane.showMessageDialog(null, "welcome\nto\nJava\nProgramming!" );
```

Изпълнява метод *showMessageDialog* от *class JOptionPane*

Метода изисква минимум 2 аргумента

Аргументите се разделят със запетая (,)

Засега, приемаме първият аргумент да е винаги *null*

Вторият аргумент представя низа, който искаме да изобразим в диалоговия прозорец

showMessageDialog е *static* метод на *class JOptionPane*

static методи се изпълняват като използваме *class* името, точка (.) следвана от името на метода

static методите **принадлежат на класа**, който реферират по име

3.4 Извеждане на текст в Dialog Box

Напомняне: Всички команди, инструкции на Java завършват с ;

Отделна команда може да се пренася на няколко реда

Не е позволено прекъсване на команда по средата на идентификатор
или низ от символи

Executing line 10 displays the dialog box



Автоматично се включва **OK button**

OK button позволява скриване на **dialog box**

По подразбиране, заглавието на диалоговия прозорец е низа *Message*

3.3 Извеждане на текст в Dialog Box

```
11      system.exit( 0 ); // terminate application with window
```

Изпълнява *static* метода *exit()* на *class System*

Спира изпълнението на приложението

Да се използва със всяко приложение, използващо GUI

Понеже метода е *static*, има нужда от рефериране на *class* име и разделител точка (.)

Напомняне: идентификатори, започващи с главна буква означават име на клас

Използване на аргумент **0** в *exit()* означава, че приложението е приключило без грешка- нормално

Ненулева стойност за аргумент се използва да се обозначи код на грешка при изпълнението на програмата- UNIX стандарт

class System е част от библиотеката *java.lang*

Напомняне: Няма нужда от import декларация за *java.lang*

java.lang автоматично се добавя към всяка Java програма

Lines 12-13: Скобки за край на *main()* и *class Dialog1*

3.4 Въвеждане на текст в Dialog Box

Диалогов прозорец за въвеждане на текст

Позволява на потребителя да въведе данни по графичен начин

Използва метода *showInputDialog* на *Class JOptionPane*


```

1 // Input dialog box
2 // Basic input with a dialog box.
3 import javax.swing.JOptionPane;
4
5 public class NameDialog
6 {
7     public static void main( String args[] )
8     {
9         // prompt user to enter name
10        String name =
11            JOptionPane.showInputDialog( "What is your name?" );
12
13        // create the message
14        String message =
15            String.format( "Welcome, %s, to Java Programming!", name );
16        // display the message to welcome the user by name
17        JOptionPane.showMessageDialog( null, message );
18        System.exit(0); // terminate the application
19    } // end main
20 } // end class NameDialog

```

Покажи диалогов прозорец за вход и
съобщение до потребителя

NameDialog.java

Форматиране на String
преди показването му в
диалоговия прозорец



3.4 Въвеждане на текст в Dialog Box

Lines 10-11:

Използват *showInputDialog* да покажат **съобщение и текстово поле**, където да се въведат данните

Метод *showInputDialog* връща *String* съдържащ въведеното от потребителя при натискане на OK

Метод *showInputDialog* връща *null* при натискане на Cancel

Lines 14-15:

Изпълнява *static* метода *format()* на *class String*

Синтаксисът на метода *String.format()* е идентичен на *System.out.printf()*

За разлика от *System.out.printf()* метода *format()* връща **форматирания стринг** вместо да го отпечата на Стандартния изход
Често използван способ за получаване на форматиран низ от символи

3.5 Въвеждане на цифрови данни с графичен интерфейс

- Програма за събиране на числа
 - Използва диалогов прозорец за въвеждане на числата
 - Използва преобразуване на текст в цифрови данни
 - Използва диалогов прозорец да изобрази сумата на две числа

Outline

Addition.java

1. import

2. class Addition

2.1 Declare variables (name and type)

3. showInputDialog

4. parseInt

5. Add numbers, put result in sum



```

1  // Fig. 2.9: Addition.java
2  // Addition program that displays the sum of two numbers.
3
4  // Java packages
5  import javax.swing.JOptionPane; // program uses JOptionPane
6
7  public class Addition {
8
9      // main method begins execution
10     public static void main( String[] args )
11     {
12         String firstNumber; // first string entered by user
13         String secondNumber; // second string entered by user
14
15         int number1; // first number to add
16         int number2; // second number to add
17         int sum; // sum of the two numbers
18
19         // read in first number from user as a String
20         firstNumber = JOptionPane.showInputDialog( "Enter first integer" );
21
22         // read in second number from user as a String
23         secondNumber =
24             JOptionPane.showInputDialog( "Enter second integer" );
25
26         // convert numbers from type String to type int
27         number1 = Integer.parseInt( firstNumber );
28         number2 = Integer.parseInt( secondNumber );
29
30         // add numbers
31         sum = number1 + number2;
32     }

```

Declare variables: name and type.

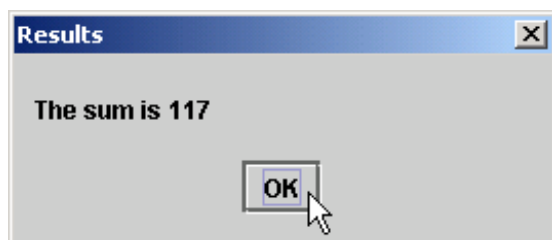
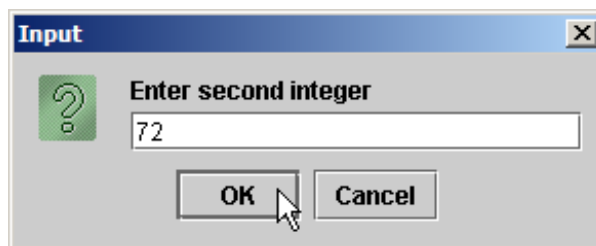
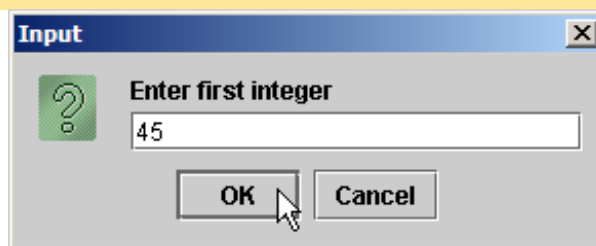
Input first integer as a String, assign to firstNumber.

Convert strings to integers.

Add, place result in sum.

```
33 // display result
34 JOptionPane.showMessageDialog( null, "The sum is " + sum,
35     "Results", JOptionPane.PLAIN_MESSAGE );
36
37     System.exit( 0 ); // terminate application with window
38
39 } // end method main
40
41 } // end class Addition
```

Program output



3.5 Въвеждане на цифрови данни с графичен интерфейс

```
5 import javax.swing.JOptionPane; // програм използва JOptionPane
```

- Указва в коя библиотека е ***class JOptionPane***

```
7 public class Addition {
```

- Започва декларацията на ***public class Addition***
 - Напомняне: Файлт с тази програма **трябва** да има име `Addition.java`
- Lines 10-11: ***main()***

```
12     String firstNumber; // first string entered by user
13     String secondNumber; // second string entered by user
```

- Декларации на променливи
 - ***firstNumber*** и ***secondNumber*** са променливи от тип ***String***

3.5 Въвеждане на цифрови данни с графичен интерфейс

```
12      String firstNumber;    // first string entered by user
13      String secondNumber;  // second string entered by user
```

– Променливи

- Място в паметта, което съхранява стойност от даден тип
 - Трябва да се декларира по име и тип преди да се използва
- ***firstNumber*** и ***secondNumber*** са от тип ***String*** (package ***java.lang***)
 - Тези променливи служат за съхраняване на ***String*** (низове)
- **Името на променливата**: произволен идентификатор
- **Декларациите свършват с ;**

```
String firstNumber, secondNumber;
```

- Може да се декларират повече от една променлива наведнъж от същия тип
 - Използва се запетая за разделител
- Може да се добавят коментари за описване типа на променливите

3.5 Въвеждане на цифрови данни с графичен интерфейс

```
15      int number1;           // first number to add
16      int number2;           // second number to add
17      int sum;                // sum of number1 and number2
```

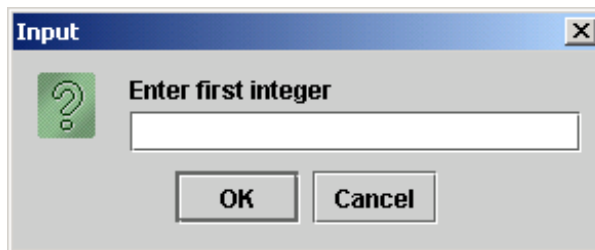
- Декларира променливи *number1*, *number2*, и *sum* от тип *int* (цяло число 32 бита)
 - *int* служи за съхраняване на **целочислени стойности**: т.е. *0*, *-4*, *97*
 - Типовете данни *float* и *double* съхраняват **числа с плаваща запетая** т.е. *0.0*, *-4.55*, *97.01*
 - Типът *char* служи за съхраняване на **отделен символ**: т.е. *'x'*, *'\$'*, *'\n'*, *'7'*
 - **Примитивни типове** данни- примери

3.5 Въвеждане на цифрови данни с графичен интерфейс

20

```
firstNumber = JOptionPane.showInputDialog( "Enter first integer" );
```

- Прочета ***String*** от потребителя, представляващо първото число за събиране
 - метод ***JOptionPane.showInputDialog()*** изобразява следното:



- Съобщението се нарича **промт**(prompt) – указва какво действие да изпълни потребителя
- Аргументът на метода ***JOptionPane.showInputDialog()*** задава текста на промта
- При въвеждане на **погрешен тип данни** (различен от цяло число) или натискане на ***Cancel***, възниква грешка-**преобразуване на данните е погрешно**

3.5 Въвеждане на цифрови данни с графичен интерфейс

20

```
firstNumber = JOptionPane.showInputDialog( "Enter first integer" );
```

- Присвояване на резултата от изпълнението на ***showInputDialog()*** се присвоява на ***firstNumber*** с оператора = в **String** формат!
 - Команда за присвояване на стойност
 - **Трябва да има съответствие между типовете на данни от лявата и дясната страна на оператора!**
 - = е бинарен оператор , използва два **операнда**
 - Изразът от дясно се изчислява и после се присвоява на променливата отлява на оператора
 - Чете се : ***firstNumber*** получава стойността на ***JOptionPane.showInputDialog("Enter first integer")***



3.5 Въвеждане на цифрови данни с графичен интерфейс

```
23      secondNumber =  
24      JOptionPane.showInputDialog( "Enter second integer" );
```

- Аналогично на предишната команда
 - Инициализира ***secondNumber*** на второто въведено число в ***String*** формат!

```
27      number1 = Integer.parseInt( firstNumber );  
28      number2 = Integer.parseInt( secondNumber );
```

- Метод ***Integer.parseInt()***
 - Преобразува ***String*** аргумент в целочислена данна (type ***int***)
 - ***class Integer*** от библиотека ***java.lang***
 - Цялото число върнато от ***Integer.parseInt()*** се присвоява на променливата ***number1*** (line 27)
 - **Спомнете си**, че ***number1*** беше декларирана като ***int***
 - **Line 28** е аналогична

3.5 Въвеждане на цифрови данни с графичен интерфейс

31

```
sum = number1 + number2;
```

– Команда за присвояване

- Пресмята сбора на ***number1*** и ***number2***
- операторът = присвоява резултата на променливата ***sum***
- Чете се: ***sum*** получава стойността на
number1 + number2

където ***number1*** и ***number2*** са операнди на ***+*** оператора

3.5 Въвеждане на цифрови данни с графичен интерфейс

```
34 JOptionPane.showMessageDialog( null, "The sum is " + sum,
35     "Results", JOptionPane.PLAIN_MESSAGE );
```

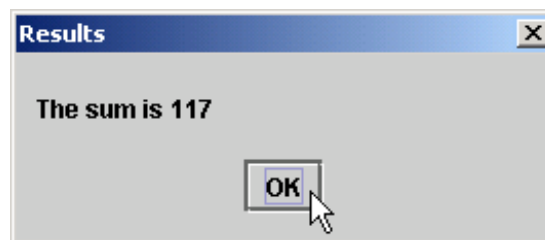
- Използва *showMessageDialog()* да изведе резултата *"The sum is " + sum*
 - Използва оператора *+* да “**събере**” низа *"The sum is"* и целочислената променлива *sum*
 - Събиране на *String* и данни от **друг примитивен тип**
 - Резултатът е **нов** низ (заделя се ново място в паметта)
 - променлива *sum* първо се преобразува до *String* и после се събира със низа отляво
 - **Забележка**: *sum + "The sum is "* е грешка
- Типът на изразът се определя от аргумента **отляво на +**
- Ако *sum* съдържа *117*, тогава *"The sum is " + sum* дава следния нов низ *"The sum is 117"*
- Забележете празния символ в края на низа *"The sum is "*



3.5 Въвеждане на цифрови данни с графичен интерфейс

```
34 JOptionPane.showMessageDialog( null, "The sum is " + sum,  
35 "Results", JOptionPane.PLAIN_MESSAGE );
```

- Друга версия на ***showMessageDialog()***
 - Изисква **4 аргумента** (вместо 2 както по-рано)
 - **Първият** аргумент е ***null*** засега
 - **Вторият**: низът, който искаме да изведем графично
 - **Третият**: низът, в заглавието на диалоговия прозорец
 - **Четвъртият**: типът на диалоговия прозорец и иконка
 - **Line 35** не се използва иконка, то задава се константата ***JOptionPane.PLAIN_MESSAGE***



3.5 Въвеждане на цифрови данни с графичен интерфейс


Тип на диалогов прозорец	иконка	Описание
<code>JOptionPane.ERROR_MESSAGE</code>		Изобразява диалог за означаване на грешка на потребителя.
<code>JOptionPane.INFORMATION_MESSAGE</code>		Изобразява диалог с информация до потребителя. Потребителят може да игнорира диалога.
<code>JOptionPane.WARNING_MESSAGE</code>		Изобразява диалог предупреждаващ за евентуален проблем.
<code>JOptionPane.QUESTION_MESSAGE</code>		Изобразява диалог с въпрос към потребителя. Изисква отговор от потребителя с натискане на Yes или No бутон.
<code>JOptionPane.PLAIN_MESSAGE</code>	no icon	Изобразява диалог със съобщение, но без иконка- обикновен диалогов прозорец.

Fig. 2.12 `JOptionPane` константи за типовете диалогови прозорци.



Comparison.java

1. import

2. Class Comparison

2.1 main

2.2 Declarations

2.3 Input data (showInputDialog)

2.4 parseInt

2.5 Initialize result



```

1 // Fig. 2.20: Comparison.java
2 // Compare integers using if statements, relational operators
3 // and equality operators.
4
5 // Java packages
6 import javax.swing.JOptionPane;
7
8 public class Comparison {
9
10     // main method begins execution of Java application
11     public static void main( String args[] )
12     {
13         String firstNumber;    // first string entered by user
14         String secondNumber;   // second string entered by user
15         String result;         // a string containing the output
16
17         int number1;           // first number to compare
18         int number2;           // second number to compare
19
20         // read first number from user as a string
21         firstNumber = JOptionPane.showInputDialog( "Enter first integer:" );
22
23         // read second number from user as a string
24         secondNumber =
25             JOptionPane.showInputDialog( "Enter second integer:" );
26
27         // convert numbers from type String to type int
28         number1 = Integer.parseInt( firstNumber );
29         number2 = Integer.parseInt( secondNumber );
30
31         // initialize result to empty String
32         result = "";
33     }

```

Променлива от референтен тип като String трябва да се инициализира, **преди** да се използва!



Test for equality, create new string, assign to `result`.

Comparison.java

3. if statements

4. showMessageDialog

```
34  if ( number1 == number2 )
35      result = result + number1 + " == " + number2;
36
37  if ( number1 != number2 )
38      result = result + number1 + " != " + number2;
39
40  if ( number1 < number2 )
41      result = result + "\n" + number1 + " < " + number2;
42
43  if ( number1 > number2 )
44      result = result + "\n" + number1 + " > " + number2;
45
46  if ( number1 <= number2 )
47      result = result + "\n" + number1 + " <= " + number2;
48
49  if ( number1 >= number2 )
50      result = result + "\n" + number1 + " >= " + number2;
51
52  // Display results
53  JOptionPane.showMessageDialog( null, result, "Comparison Results",
54      JOptionPane.INFORMATION_MESSAGE );
55
56  System.exit( 0 ); // terminate application
57
58  } // end method main
59
60  } // end class Comparison
```

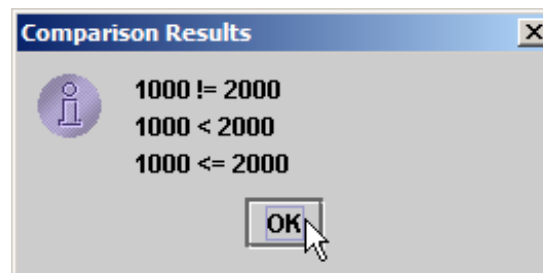
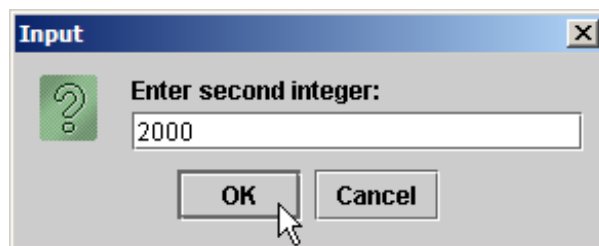
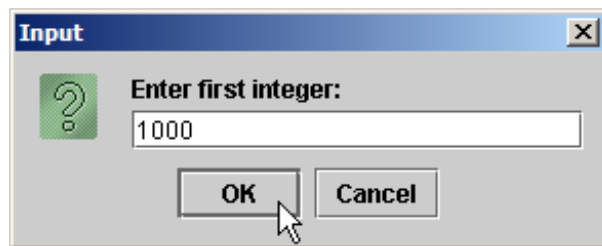
Notice use of
`JOptionPane.INFORMATION_MESSAGE`





Outline

Резултат от
работата на
програмата



3.5 Въвеждане на цифрови данни с графичен интерфейс

- Lines 1-12: коментари, *import JOptionPane*, начало на *class Comparison* и *main()*
- Lines 13-18: деклариране на променливи
 - Деклариране на списък от променливи, разделени със запетая:

```
13      String firstNumber,  
14          secondNumber,  
15          result;
```

- Lines 21-30: въвеждане на числа в текстов формат и преобразуването им в целочислени променливи

```
20      // read first number from user as a string  
21      firstNumber = JOptionPane.showInputDialog( "Enter first integer:" );  
22  
23      // read second number from user as a string  
24      secondNumber =  
25          JOptionPane.showInputDialog( "Enter second integer:" );  
26  
27      // convert numbers from type String to type int  
28      number1 = Integer.parseInt( firstNumber );  
29      number2 = Integer.parseInt( secondNumber );
```

3.5 Въвеждане на цифрови данни с графичен интерфейс

```
32      result = "";
```

- Инициализира **result** с *празен низ* от символи
- **result** е променлива от референтен тип и трябва да се инициализира преди да се използва

```
34      if ( number1 == number2 )  
35          result = result + number1 + " == " + number2;
```

- **if** команда за проверка на равенство (==)
 - Ако променливите са равни (условие **true**)
 - **result** се събира с оператора +
result = result + other strings
 - Дясната страна се пресмята първа, полученият string се присвоява на **result**
 - Ако променливите не са равни, това присвояване се пропуска



3.5 Въвеждане на цифрови данни с графичен интерфейс

- **Lines 37-50:** други *if* команди за проверка на **по-малко от . . . , по-голямо от . . .** и пр.
 - ако ***number1 = 123*** и ***number2 = 123***
 - Line 34 дава ***true*** (if ***number1 == number 2***)
 - понеже ***number1*** е равно на ***number2***
 - Line 40 дава ***false*** (if ***number1 < number 2***)
 - понеже ***number1*** не е по-малко от ***number2***
 - Line 49 дава ***true*** (if ***number1 >= number2***)
 - понеже ***number1*** е по-голямо или равно на ***number2***
- **Lines 53-54:** ***result*** се извежда в диалогов прозорец посредством ***showMessageDialog()***



3.6 Обобщение

- Библиотеката *javax.swing* съдържа класове за създаване на графичен потребителски интерфейс на Java приложения.
- *class JOptionPane* е дефиниран в библиотеката *javax.swing* и позволява създаване на диалогови прозорци за вход и изход на данни.
- Програмистите трябва да използват *import* декларации, за да обявят всички класове, чиито дефиниции са външни за приложението.
- Всеки статичен (*static*) метод се реферира с името на класа, на който принадлежи, следвано от точка и името на съответния метод
- Методът *exit()* на *class System* принадлежи на библиотека *java.lang* и служи за прекратяване (изход) от приложението. При това се освобождават всички ресурси използвани от графичния контекст на приложението.
- Декларация на променлива завършва със точка и запетая (;)

3.6 Обобщение

- Примитивни типове данни са *int*, *byte*, *short*, *long*, *float*, *double*, *boolean* и *char*.
- *Промпт съобщенията* са необходими за указване на очакваното от потребителя действие.
- *Integer.parseInt()* служи за преобразуване на *String* аргумент до *int* стойност.
- *Double.parseDouble()* служи за преобразуване на *String* аргумент до *double* стойност.
- *Float.parseFloat()* служи за преобразуване на *String* аргумент до *float* стойност.
- **Java** използва оператора **+** за събиране на низове, както и за събиране на низ с друг тип данни. В този случай първият операнд задължително е от тип *String*
- Когато първият операнд на оператора **+** представя число, то и вторият операнд трябва да е от числен тип, т.е не може да е низ от символи (*String*)

3.6 Обобщение

- Аритметично делене, при което и двата операнда са от целочислен тип (*byte*, *short*, *int*, *long*) се нарича целочислено делене. В този случай резултатът също е целочислен е числата след десетичната точка се отрязват
- *if* командата на *Java* позволява на програмата да изпълни една или друга група от инструкции, в зависимост от формулирано условие за преход
- Условието за преход на *if* командата се формулира посредством оператори за сравнение и логически изрази
- Празен низ от символи е низ, който не съдържа символи и може да се използва за първоначална инициализация на *String* променлива
- *String* променлива получава стойност *null* от компилатора по подразбиране при декларацията ѝ. Това НЕ инициализира променливата.
- Стойността *null* на *String* променлива не е еквивалентна на стойността зададена с празен низ и стойността *null* не се счита за инициализация на променлива от референтен тип, какъвто е *String*
- Всяка променлива от референтен тип трябва първо да се инициализира и после да се използва.

3.7 Задачи

1. Напишете програма (конзолно приложение), което

- въвежда цяло петцифрено десетично число.
- Отделя всяка от цифрите на числото и ги оппечатва на един ред,
 - В същата последователност, разделени със запетая и празен символ
 - В обратна последователност, разделени със запетая и празен символ

Предполагаме, че потребителят ще въвежда само петцифрено десетично число. Проверете какво ще стане с вашата програма, ако все пак потребителят въведе десетично число с повече или по-малко от 5 цифри

3.7 Задачи

2. Напишете програма (конзолно приложение), което прочете последователно 5 цели числа от клавиатурата и извежда колко от тези числа са били:

а) положителни

б) отрицателни

в) нула

Забележка: Да се използват *само средства на езика, предадени до този момент на лекции т.е без цикли. Използвайте printf() подходящи форматиращи спецификатори и кратка if инструкция*

3.7 Задачи

3. Напишете програма (конзолно приложение), което да преобразува от стойности от *Fahrenheit temperature* в *Celsius* еквивалентни стойност като използва диалогов прозорец (*class JOptionPane*) за ВХОД и ИЗХОД на данните

- чете от диалогов прозорец цяло число, което е *Fahrenheit temperature* – използвайте подходящо съобщение към потребителя
- пресмята *Celsius* еквивалентни стойност по следната формула
$$Celsius = 5.0 / 9.0 * (Fahrenheit - 32)$$
- изобразява пресметнатата *Celsius* еквивалентна стойност в графичен прозорец

3.7 Задачи

4. Напишете програма (конзолно приложение), което да пресметне колко ще имате по депозитната си сметка след 1 и след 3 месеца, ако са дадени :
- годишната лихва по сметката (*rate*)
 - депозираната сума в началото на периода (*deposit*)
 - Използвайте *class JOptionPane* методи за въвеждане (данните *a- b*) и извеждане на данни- паричната сума (*total*) в депозита в края на депозитния период

Използвайте следната формула за пресмятане на паричната сума в депозита в края на депозитния период , където *months* е 1 или 3, съответно с изискванията на задачата

$$\text{total} = \text{deposit} * (1 + \text{rate}/12)^{\text{months}}$$

3.7 Задачи

5. Изпълнете програмите, дадени на лекции с примерен код Fig. 2.19 и Fig 2.20.

- **Компилирайте и изпълнете тези две програми.**
- **Тествайте изпълнението на тези програми и установете при какви условия те прекъсват изпълнението с грешка.**
- **Предложете промяна в кода на тези програми, за да се избегне тази грешка.**

3.7 Задачи

6. Да се напише програма за пресмятане на израза

$$(x/y + 5) / 4$$

като се използват диалогови прозорци за вход и изход на данни.

Предайте проекта на NetBeans, архивиран в единичен файл