гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Компютърно образователен център
Progress
Палитра от възможности!

# Лекция 6.1

## Използване на методи в Java – 2-ра част

- **Особености в използването на методи в Java**
- **Статични методи и клас данни**
- **Използване на библиотечни статични методи в class Math**
- **Предаване на данни между методи**
- **Генериране на случайни стойности**
- **Задачи**

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net



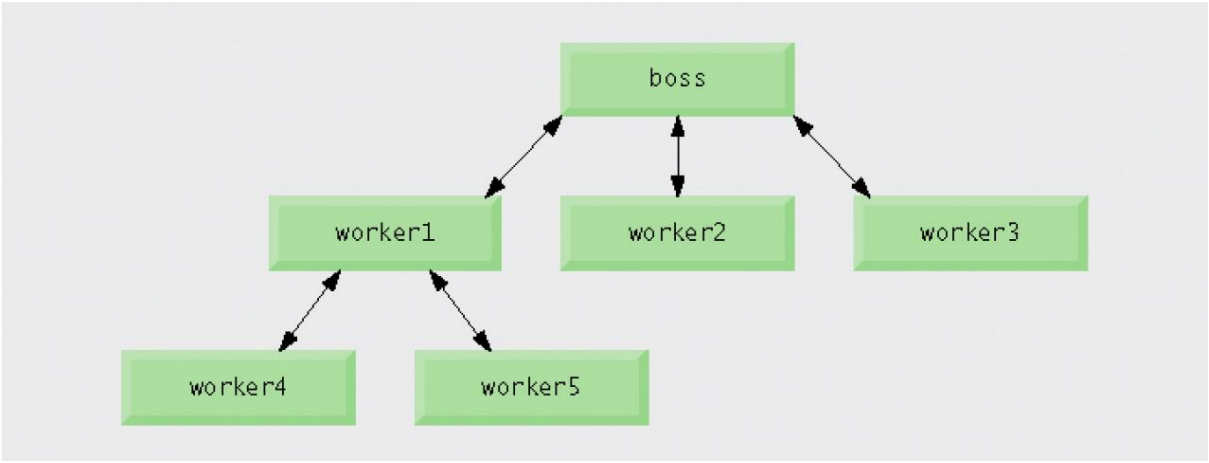
## Въведение

- Приложните програми на практика са много по-сложни от разглежданите тук примери
- За по-лесна разработка и поддържане те се конструират от по-малки части, наречени модули
- Програмните модули се обособяват с прилагане на техника „Разделяй и Владей”
  - Разбиват една голяма задача на по-малки части(**modules**)
  - На най-ниско ниво тази техника се свежда подходящо дефиниране клас методи
  Примери : мутатор и аксесор методи

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net





гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net



## Въведение

- Тук ще научим как се дефинират аргументи на метод.Ще поясним как Java следи кой метод се изпълнява в момента,как се представят локалните данни в машинната памет и как един метод знае къде да предаде управлението на логиката след приключване на изпълнението си.
- За илюстрация ще разгледаме способите за генерация на случайни числа.Допълнително ще научим как се декларират данни, които не трябва да се менят в процеса на изпълнение на програмата – **константи**.
- Често се налага метод с едно и също име да се извиква с различен набор от аргументи(например,конструктор по подразбиране и конструктор за общо ползване) Тази техника е известна като overloading(допълнително дефиниране на метод) и се използва за изпълнение на сходни дейности, но с различен брой или тип данни на аргументите.

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net



# Програмни модули в Java

- Java поддържа 3 типа модули – методи, класове и библиотеки
- Java програми комбинират потребителски дефинирани класове и методи с библиотечни(предефинирани) класове и методи от Java API

- Известна е като Java клас библиотеки
- Съдържа предефинирани класове и методи
  - Логически свързани класове са обособени в пакети библиотеки(packages)
  - Пример : математическите методи, обработка на текст и символи, вход и изход на данни, обработка на файлове и др.

- Методи в Java
  - Наричат се функции или процедури в термините на други програмни езици

  - Структурират програмите в модули, чрез разделяне на задачите в по-малки и лесно изпълними подзадачи

  - Позволяват да се приложи техниката за „Разделяй и Владей”

  - Обхващат блок от команди, които се използват в многократно изпълнението на програмата

  -Избягва се повторно писане на едни и същи групи от команди

**Обичайна грешка** при програмиране

Един кратък метод за изпъленение на задача е по-лесно да се тества от един по-дълъг метод, който е проектиран да изпълнява множество от задачи.
Пишете методи не по-дълги от един екран.

- static метод ( метод общ за всички обекти от даден клас)
  - Принадлежи на класа като цяло, вместо на всеки обект поотделно
  - static метод не зависи от статуса(текущо състояние, стойностите на клас данните) на отделен обект на класа
  - Извикването на static Метод става по следната схема : ClassName.methodName( arguments)
  - Пример : всички методи на class **Math** са static
    - Math.sqrt( 900)
    - Math.pow( 2.0, 2.5)
    - Math.sin(3.1415)

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net



# Методи на class Math :

| Метод | Описание | Пример |
|---|---|---|
| abs( *x* ) | Абсолютна стойност на *x* | *abs(*23.7 ) е 23.7<br>abs( 0.0 ) е 0.0<br>abs( -23.7 ) е 23.7 |
| ceil( *x* ) | закръгля *x* до най- малкото цяло не по-малко от *x* | *ceil(*9.2 )е 10.0<br>ceil( -9.8 ) е -9.0 |
| cos( *x* ) | косинус *x* (*x* в радиани) | cos( 0.0 ) е 1.0 |
| exp( *x* ) | Експонента на *x* | *exp(*1.0 ) е 2.71828<br>exp( 2.0 ) е 7.38906 |
| floor( *x* ) | закръгля *x* до най- голямото цяло не по-голямо от *x* | *floor(*9.2 ) е 9.0<br>floor( -9.8 ) е -10.0 |
| log( *x* ) | Естествен логаритъм на *x* (основа *e*) | log( Math.E ) е 1.0<br>log( Math.E * Math.E ) е 2.0 |
| max( *x*, *y* ) | По- голямото между *x* и *y* | *max(*2.3, 12.7 ) е 12.7<br>max( -2.3, -12.7 ) е -2.3 |
| min( *x*, *y* ) | По- малкото между *x* и *y* | *min( 2.3, 12.7 )* е 2.3<br>min( -2.3, -12.7 ) е -12.7 |
| pow( *x*, *y* ) | *x* на степен *y* | pow( 2.0, 7.0 ) е 128.0<br>pow( 9.0, 0.5 ) е 3.0 |
| sin( *x* ) | Синус от *x* (*x* е в радиани) | sin( 0.0 ) е 0.0 |
| sqrt( *x* ) | Корен квадратен от *x* | *sqrt( 900.0 )* е 30.0 |
| tan( *x* ) | Тангенс от *x* (*x* е в радиани) | tan( 0.0 ) е 0.0 |

- Деклариране на static метод
  - Използва се ключовата дума static пред типа на връщаните данни
  - Пример :

```
public static void main(String[] args){}
```

Принадлежи на класа като цяло, вместо на всеки обект поотделно

✓ class Math принадлежи на java.lang пакета, който се импортира неявно, така че няма нужда да се импортира class Math за да се използват неговите методи

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net



# Константи

- Константи
  - Декларират се с ключовата дума final
  - Инициализират се на мястото на декларацията им
  - Не могат да се променят след инициализация
  - В общия случай всеки обект може да има своя стойност за дадена константа
- static клас данни( клас атрибути, полета и др)
  - Такива данни са общи за всички обекти от даден клас
  - Могат да се инициализират след като са декларирани
  - Могат да се променят в процеса на изпълнението

**Примери :**

- Math.PI (числото PI) и Math.E(числото e) са
- final static Клас данни на class Math
- Следователно, това са константи общи за всички обекти от този клас.

- Пример
  - main() е деклариран като
  public static void main( String args[])
  - При изпълнение на командата
  java ClassName argument1 argument2 …

  → JVM извиква main() на ClassName със списък от аргументи(от тип String)
  - По такъв начин изпълнението на ClassName може да зависи от определен набор параметри , задавани на командния ред( ще обясним как в следващата лекция)

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net



# Деклариране на методи с много аргументи

- Методи с повече от един аргумент се декларират в списък от декларации на данни, разделени със запетая.
- Тези данни се част от локалните данна на метода и се наричат още – формални аргументи

- При изпълнение на метод на всеки формален аргумент се съпоставя променлива, реферираща стойност или обект съответстваща по тип на формалния аргумент – в този случай данните се наричат още **реални аргументи**

- Пример : Fig 6.3 и Fig. 6.4 използва потребителски дефиниран метод **maximum()** за намиране и връщане на най-голямото от 3 числа(double) въведени от потребителя
- В началото на изпълнението, метода main() на class MaximumFinderTest(редове 7-11 от Fig. 6.4) създава обект от class MaximumFinder(ред 9) и извиква метода determineMaximum(ред 10) за извеждане на крайните резултати
- В class MaximumFinder(Fig. 6.3) редове 14-18 от метод determineMaximum() известяват потребителя да въведе 3 **double** числа и ги прочита от клавиатурата.
- Ред 21 извиква метод maximum(редове 28-41) за определяне на най-голямото от въведените числа.Когато метод maximum() връща резултата на ред 21, програмата присвоява крайния резултат от изпълнението на maximum() на локалната променлива result.
- Тогава ред 24 извежда крайния резултат.

# MaximumFinder.java

Outline

```
1  // Fig. 6.3: MaximumFinder.java
2  // Programmer-declared method maximum.
3  import java.util.Scanner;
4
5  public class MaximumFinder
6  {
7     // obtain three floating-point values and locate the maximum value
8     public void determineMaximum()
9     {
10       // create Scanner for input from command window
11       Scanner input = new Scanner( System.in );
12
13       // obtain user input
14       System.out.print(
15          "Enter three floating-point values separated by spaces: " );
16       double number1 = input.nextDouble(); // read first double
17       double number2 = input.nextDouble(); // read second double
18       double number3 = input.nextDouble(); // read third double
19
20       // determine the maximum value
21       double result = maximum( number1, number2, number3 );
22
23       // display maximum value
24       System.out.println( "Maximum is: " + result );
25    } // end method determineMaximum
26
27    // returns the maximum of its three double parameters
28    public double maximum( double x, double y, double z )
29    {
30       double maximumValue = x; // assume x is the largest to start
31
32       // determine whether y is greater than maximumValue
33       if ( y > maximumValue )
34          maximumValue = y;
35
36       // determine whether z is greater than maximumValue
37       if ( z > maximumValue )
38          maximumValue = z;
39
40       return maximumValue;
41    } // end method maximum
42 } // end class MaximumFinder
```

Съобщава на потребителя да въведе три double числа

Извиква метод maximum с три реални аргумента- по тип и брой съответстват на формалните

Съответства по тип на връщаната от метода данна

Извежда максималната стойност

Декларация на метода maximum с три формални аргумента

Сравнява y и maximumValue

Сравнява z и maximumValue

Връща намерената максимална стойност, съответства на декларирания тип за връщане на данни в заглавието на метода

MaximumFinderTest.java

```
1  // Fig. 6.4: MaximumFinderTest.java
2  // Application to test class MaximumFinder.
3
4  public class MaximumFinderTest
5  {
6     // application starting point
7     public static void main( String args[] )
8     {
9        MaximumFinder maximumFinder = new MaximumFinder();
10       maximumFinder.determineMaximum();
11    } // end main
12 } // end class MaximumFinderTest
```

Създава обект от клас `MaximumFinder`

Изпълнява метод `determineMaximum`

Outline

MaximumFinderTest.java

```
Enter three floating-point values separated by spaces: 9.35 2.74 5.1
Maximum is: 9.35
```

```
Enter three floating-point values separated by spaces: 5.8 12.45 8.32
Maximum is: 12.45
```

```
Enter three floating-point values separated by spaces: 6.46 4.12 10.54
Maximum is: 10.54
```

**Обичайна грешка** при програмиране :

Декларирането на аргументи на метод от един и същ тип като
float x,y
вместо
float x, float y
е синтактична грешка – деклариране на типа се изисква за всеки отделен аргумент в списъка с аргументи на метода

Метод с дълъг списък от аргументи вероятно изпълнява твърде много задачи.
Помислете за разделянето на такъв метод на няколко по-малки методи.
Стремете се списъка с аргументи да не надхвърля един ред.

- Многократно използване на методи(пример, метод Math.max)
  - Изразът **Math.max( x, Math.max(y , z ) )**
  пресмята максимума от y и z, а след това пресмята максимума от x и пресметната стойност
  - Така редове 20-21 от fig 6.3 може да се пренапишат като

  20 // determine the maximum value
  21 double result = **Math.max( x, Math.max(y , z ) );**

## Събиране на низове( String concatenation)

- Използването на оператора + с два String-а ги събира в нов низ(String)
- Използването на оператора + с един String и стойност от друг тип е сумарен String с текстово представяне на първия низ и String представянето на втория операнд на оператора +
- Когато вторият оперант е обект, то се извиква неговия toString() метод (ако е дефиниран) за генериране на String представянето на този обект
- Методът toString() трябва да е дефиниран като

**public String toString(){}**

**Обичайна грешка** при програмиране :

Синтактична грешка е да се разбие една String константа на няколко реда в програмата.

Когато един String не се събира на един ред, разбийте го на части и използвайте събиране на низове, за да представите зададения String, чрез неговите части.

**Объркването на + оператора, използван за събиране на низове и + оператора, използван за събиране на числа може да доведе до странни резултати.**

**Java пресмята операндите на оператор отляво надясно. Например, ако целочислената y е 5, то изразът**

```
"y + 2 = " + y + 2
```

**дава низа**

**"y + 2 = 52", а не**

**"y + 2 = 7",**

**понеже първо y (5) сесъбира с низа "y + 2 = ", а после 2 се събира с низа "y + 2 = 5".**

**Изразът "y + 2 = " + (y + 2) дава желаният резултат "y + 2 = 7".**

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net



# Обобщение на деклариране и използване на методи

- Има 3 начина за извикване на метод:
  - Използванесамо на името на метода(директно извикване)
  - Използване на променлива, реферираща обект и следвана от точка(.) и иметода на метода, принадлежащ на референтния метод
  - Използване на име на клас, следвано от точка(.) и име на метод за извикване на static метод от този клас

  Важно :
- static методите **не могат** да извикват **директно** не- static методи от същия клас
- static методите могат да извикват директно само други static методи от същия клас

- Има 3 начина за връщане на управлението на след изпълнение на метод:
  - Ако методът не връща данни(деклариран е от тип void), методът прекратява изпълнението си при :
    - Достигане на затваряща фигурна скоба за блока от команди на метода
    - Методът изпълнява командата return;
  - Ако методът връща данни(деклариран е от тип различен от void),като краен резултат :
    - Методът изпълнява командата return expression;
      Където expression се пресмята до тип данни, съответстващи на типа данни в заглавието на метода и после стойността на данните се връща на викащия метод

гр. София, пл. Славейков 11 ет.5  тел. : +359 897 91 93 96

office@progressbg.net  www.progressbg.net

гр. София, пл. Славейков 11 ет.5  тел. : +359 897 91 93 96

office@progressbg.net  www.progressbg.net