БАЗОВИ ЕЗИКОВИ КОНСТРУКЦИИ

ЛЕКЦИОНЕН КУРС "ПРОГРАМИРАНЕ НА JAVA"

Distributed @learning center

СТРУКТУРА НА ЛЕКЦИЯТА

- Императивно програмиране
- Многокомпонентни програми
- Базови елементи на Java програми
- APIs

ВЪВЕЖДАЩ ПРИМЕР

1. Какво прави програмата?
2. Каква е програмата?
3. Какъв резултат?

Преобразуване на температура

Императивна програма

```
class Temperature {
   // Convert temperature
   // from Fahrenheit to Centigrade (Celsius)

   public static void main (String[] args) {
         double tempFahr; // Fahrenheit
         double tempCels; // Celsius

         System.out.print("Temperature (deg F): ");
         tempFahr = Keyboard.readDouble();

         tempCels = (5.0 * (tempFahr - 32.0)) / 9.0;

         System.out.print(tempFahr);
         System.out.print(" deg F is ");
         System.out.print(tempCels);
         System.out.println(" deg C");
   }
}
```

```
% javac Temperature.java
% java Temperature

Temperature (deg F): 10
10 deg F is -12.222222222222221 deg C
```

ИМПЕРАТИВНО ПРОГРАМИРАНЕ

- **Алгоритъм:**
  - Подход за изчисляване на търсене стойности от дадени такива, . . . който използва постъпково изпълнение на елементарни обработващи операции
- **Императивна програма:**
  - Алгоритми, описани посредством обръщения (достъп) към стойности на променливи
  - Промяна и четене на стойности

```
tempCels = (5.0 * (tempFahr - 32.0)) / 9.0;
```

Императивно програмиране е ориентирано към описание на алгоритми

ИМПЕРАТИВНА ПРОГРАМА НА JAVA

1 Основни части на програмата?

```
class Temperature   {
   public static void main (...)   {
         double tempCels;
         double tempFahr;
         ...

         tempFahr = Keyboard.readDouble();



         tempCels = (5.0 * tempFahr ...);



         System.out.print(tempCels);
   }
}
```

Входни стойности във входни променливи

Алгоритъм

Изходни стойности в изходни променливи

ИМПЕРАТИВНО ПРОГРАМИРАНЕ: ДЕТАЙЛИ (1)

- Базови концепции:
  - **Променлива:** притежава стойност, която се променя посредством оператори
  - **Оператор:** Служи за достъп до стойностите на променливите (четене и промяна на стойностите)
- Базов метод за структуриране на императивното програмиране:
  - **Процедури (функции, методи):** Частични алгоритми, представени като с оператори на езика

ИМПЕРАТИВНО ПРОГРАМИРАНЕ: ДЕТАЙЛИ (2)

- Данни (променливи) и обработващи алгоритми (процедура, функция, метод) са разделени структури в императивното програмиране
- За сравнение с обектно-ориентираното:
  - Клас = единство от данни и обработващи алгоритми

→ Java: обектно-оринетиран език за програмиране
→ Обаче: императивното програмиране е възможно и в Java

ПРИМЕР ЗА JAVA-ПРОГРАМА

1 От колко компонента се състои програмата? 3

```
class Temperature {
   // Convert temperature
   // from Fahrenheit to Centigrade (Celsius)

   public static void main (String[] args) {
         double tempFahr; // Fahrenheit
         double tempCels; // Celsius

         System.out.print("Temperature (deg F): ");
         tempFahr = Keyboard.readDouble();

         tempCels = (5.0 * (tempFahr - 32.0)) / 9.0;

         System.out.print(tempFahr);
         System.out.print(" deg F is ");
         System.out.print(tempCels);
         System.out.println(" deg C");
   }
}
```

КОМПОНЕНТИ НА ПРОГРАМАТА

1 Защо е възможно?

Разделено компилиране

```
class Temperature   {
...
}
```

File: Temperature.java

```
class Keyboard   {
...
}
```

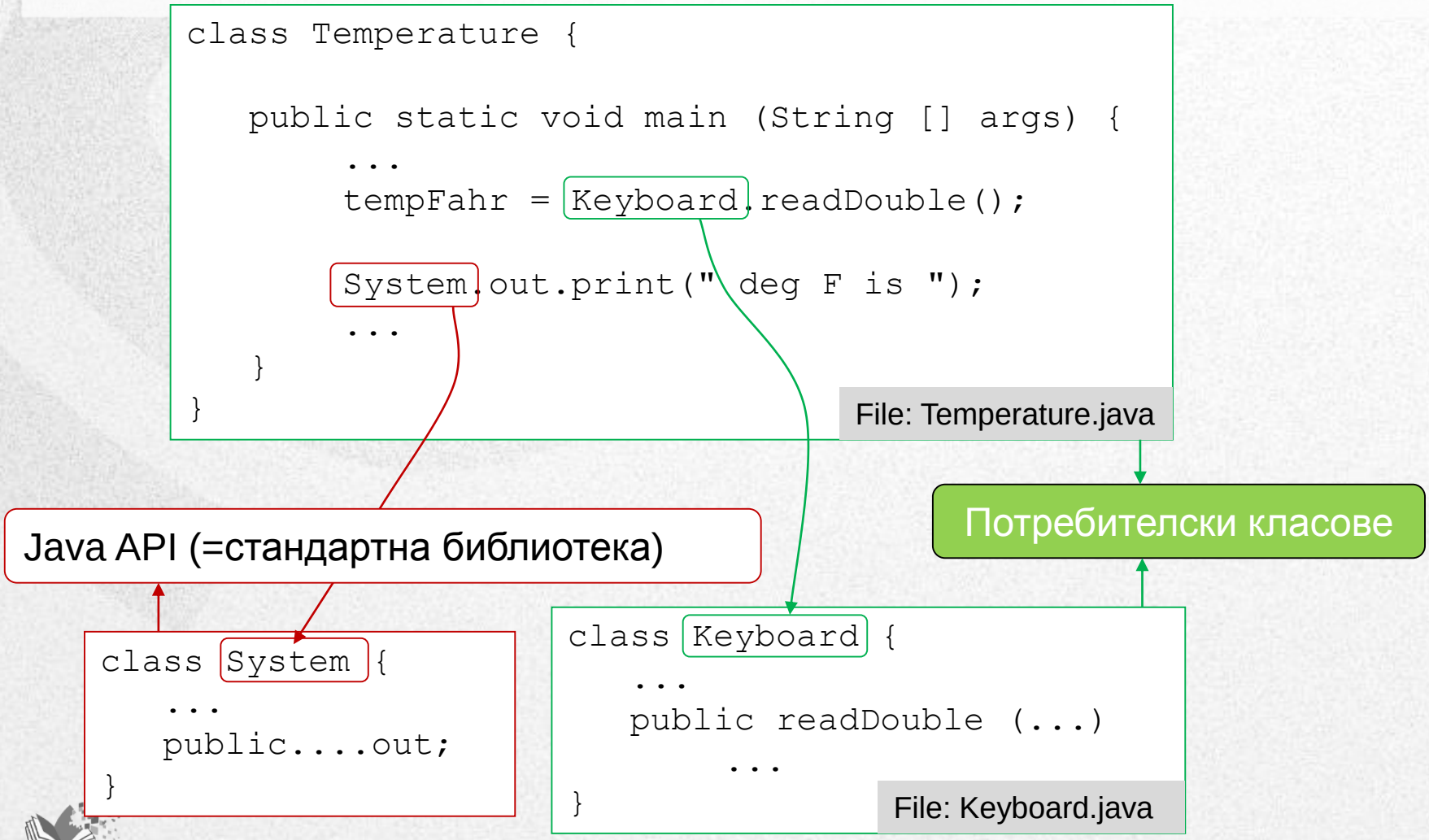
File: Keyboard.java

Дефинирани от потребителя класове

```
class System   {
...
}
```

Java API
(application programming interface)
= стандартна библиотека

ИНТЕРФЕЙСИ МЕЖДУ КОМПОНЕНТИТЕ

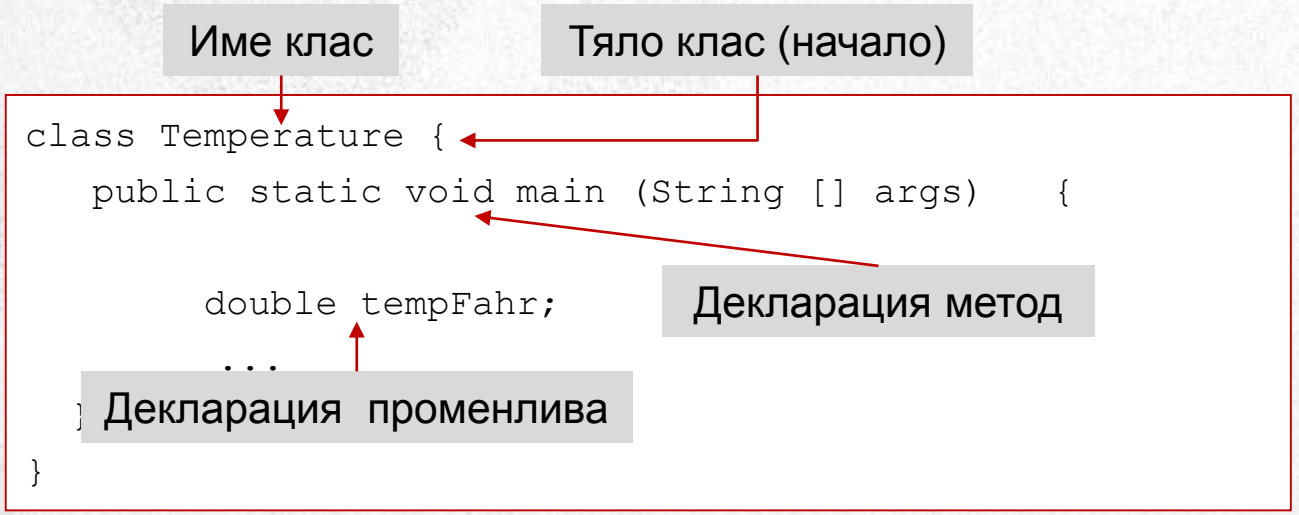


JAVA ПРОГРАМИ

- Принципно два вида Java програми:
  - Приложения
  - Аплети
- Могат да съдържат повече от един клас
- При приложенията точно един клас трябва да съдържа дефиниция на метод с име main
  - Активира се когато започне изпълнението на приложението

СТРУКТУРА НА JAVA-ПРОГРАМИ

Клас = базов компонент



Клас: 'Множество' от
- Декларации на променливи
- Декларации на методи
  (метод = алгоритъм, процедура, функция)

За общата програма: един метод m a i n ( )→ там започва обработката !

ОСНОВНИ АСПЕКТИ НА КЛАСОВЕТЕ

- Всеки клас дефинира множество от стойности
  - Наричат се обекти от този клас
- Декларираме променливи, които ще съдържат обектите
  - Както при простите типове
    - Напр. C u,v,w
  - Ако C и D са два различни класа техните обекти са от различни типове
- Обектите се различават също така и от простите типове

СЪЗДАВАНЕ НА ОБЕКТИ

- Когато е декларирана една променлива от тип клас, това създава един празен контейнер
  - Актуален обект може да се създаде посредством оператора new
    - Напр. u = new C (аргументи);
  - Операторът:
    - Създава нов обект
    - Извиква един метод на класа (конструктор) за инициализация на новия обект

ИЗПОЛЗВАНЕ НА МЕТОДИ

- Методите на обектите обикновено се използват (извикват) посредством dot-нотация
  - Напр.

    u.metod (…)
  - Наричат се методи на инстанции (instance methods)
- Някои методи могат да се извикват и посредством обикновения синтаксис
- Съществуват също така и методи на класове (class methods)

ПРОСТИ ВХОДНО-ИЗХОДНИ ОПЕРАТОРИ

- Output
  - Процес на показване на данни
  - Печат или дисплей
  - В Java най-проста възможност:
    - System.out.print()
    - System.out.println()
  - Показват символи върху екрана
    - Наричан stdout (standard output stream) или конзола (console)
- Input
  - Не толкова прост като изхода
  - За опростяване съществува клас ‘Keyboard’
  - Keyboard.readInt() - Чете цели числа

КЛАС KEYBOARD: ЕДНА АБСТРАКЦИЯ

```
class Keyboard {

   public static int readInt () ;

   public static char readChar () ;

   public static double readDouble () ;

   public static String readString () ;

   public static boolean eof () ;
}
```

Множество от полезни функции за въвеждане на цели числа, символи, реални числа, символни низове от входното устройство (клавиатура)

ДЕКЛАРАЦИИ НА ПРОМЕНЛИВИ

1. Какво съдържа една декларация?

```
double   tempFahr;
```

тип | променлива

2. Какво определя една декларация?

1. Област на стойностите на променливите
2. Определяне обема на паметта спрямо типа напр. double: 8 байта
3. Позволени операции

3. Стандартни типове?

Java-EBNF: 'стандартни типове'

boolean, char, byte, short, int, long, float, double

ПРОМЕНЛИВИ

- Съществен елемент в програмирането
  - Контейнер за стойности
  - Декларацията създава този контейнер
  - От това следва, че всяка променлива трябва да бъде декларирана
  - Веднага след декларацията една променлива не съдържа нищо
  - Стойности се записват (и променят) посредством оператор за присвояване
  - Една променлива съдържа винаги само една стойност

ПРОМЕНЛИВИ

- Повече променливи могат да се декларират заедно
- Всяка декларация дава типа на променливата
- Не всяко име е допустимо за променливите
  - Освен букви и цифри са допустими също така “_” и “$”
  - Чувствителност към малки и големи букви
- Имената на ключовите думи са резервирани
  - Коректните имана на променливи се наричат идентификатори (identifiers)
  - Използват се на различни места в Java програмите
- Понякога е полезно да имаме променливи, които никога не променят стойносите си
  - Удобни за документиране на програмите
  - Декларираме ги посредством final
  - Напр. final double PI = 3.14159

ТИПОВЕ ДАННИ

- Един от най-съществените елементи – и понякога объркващ
- Съществуват идентични с математическите, които имат обаче друго представяне
- Java предлага едно разнообразие от типове данни
- Всеки тип има:
  - Име
  - Множество от стойности (литерали)
- Два основни вида:
  - Прости
  - Обекти

ПРОСТИ ТИПОВЕ ДАННИ

- Простите типове данни, които се използват основно в Java:
  - int
    - Операции – (+,-,*,/,%)
    - (!) При деление резултатът също е int
    - Остатък при целочислено деление посредством %
  - double
    - Представят реални числа
      - E-нотация допустима
      - Напр. – 3.14159, -16.3e+002
    - Операции – (+,-,*,/)
  - boolean
  - char

КОМЕНТАРИ

1. Как задаваме коментари?
2. Каква е разликата?

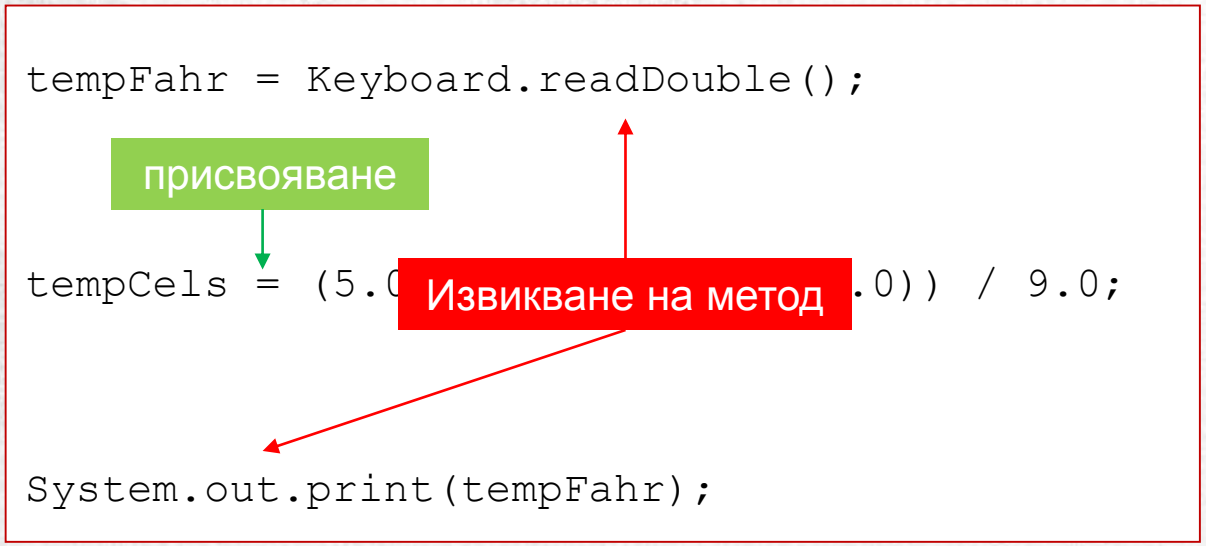
```
class Temperature {
        // Convert temperature
        // from ...

        double tempFahr; // Fahrenheit


        int /* only here */ temp;
```

ПРОСТИ ОПЕРАТОРИ

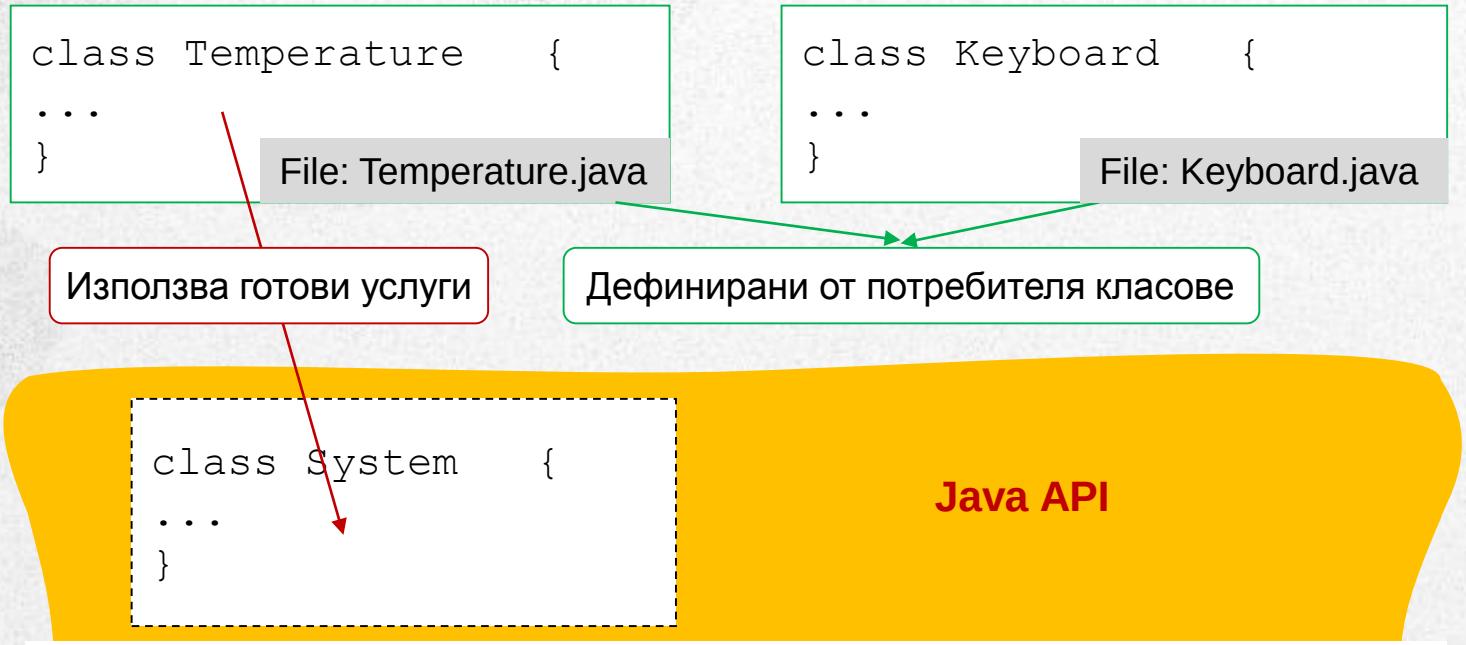
1 Прости оператори?



API: APPLICATION PROGRAMMING INTERFACE

1 Какво е API?

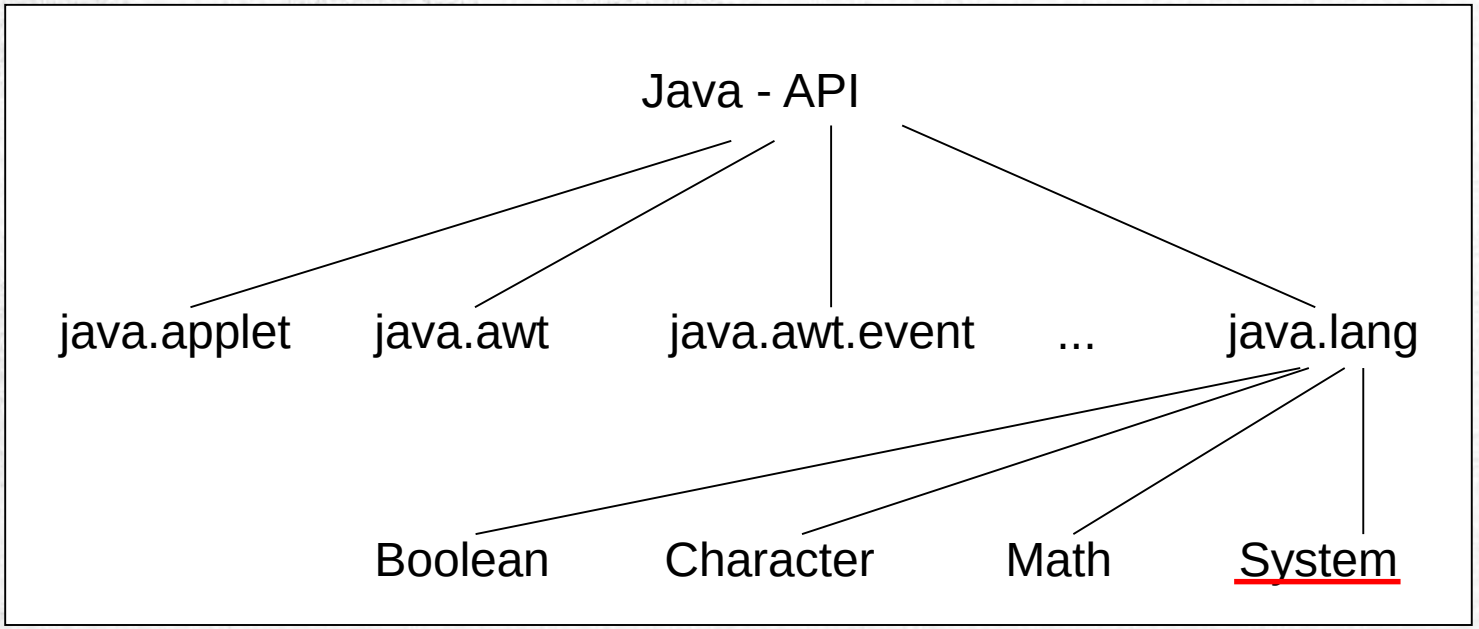
- Множество от предварително дефинирани компоненти, принадлежащи към всяка система (компилатор) на Java
- Стандартна библиотека



ОРГАНИЗАЦИЯ НА JAVA-API

Пакети

Класове



Пакети:
Сбирка от класове

Класове:
Софтуерни компоненти

Име на пакет (напр. java.awt.event) отразява Directory-Names:

→ /java/awt/event

Java™ 2 Platform Standard Edition 5.0
API Specification

This document is the API specification for the Java 2 Platform Standard Edition 5.0.

See:
Description

Java 2 Platform Packages	
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing *beans* -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.instrument	Provides services that allow Java programming language agents to instrument programs running on the JVM.
java.lang.management	Provides the management interface for monitoring and management of the Java virtual machine as well as the operating system on which the Java virtual machine is running.

Документ: 14 страници, 166 пакета

БЛАГОДАРЯ ЗА ВНИМАНИЕТО!

КРАЙ “БАЗОВИ ЕЗИКОВИ КОНСТРУКЦИИ”

Distributed @learning center