2. Основни езикови конструкции на императивните програми

Лекционен курс “Програмиране на Java”
проф. д-р Станимир Стоянов

Структура на лекцията

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

Пример: преобразуване на температура

```
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");
   }
}
```

Императивна програма

```
% javac Temperature.java
% java Temperature

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

Пример: като програма на Pascal (опростен)

```
PROGRAM Temperature;
   {Convert temperature
    from Fahrenheit to Centigrade (Celsius)}

VAR tempFahr: real;
    tempCels: real;

BEGIN
   writeln("Temperature (deg F): ");
   readln(tempFahr);

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

   write(tempFahr);
   write(" deg F is ");
   write(tempCels);
   writeln(" deg C");
END.
```

Императивно програмиране: общо

- **Алгоритъм:**
  - Подход за изчисляване на търсени 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 Pascal

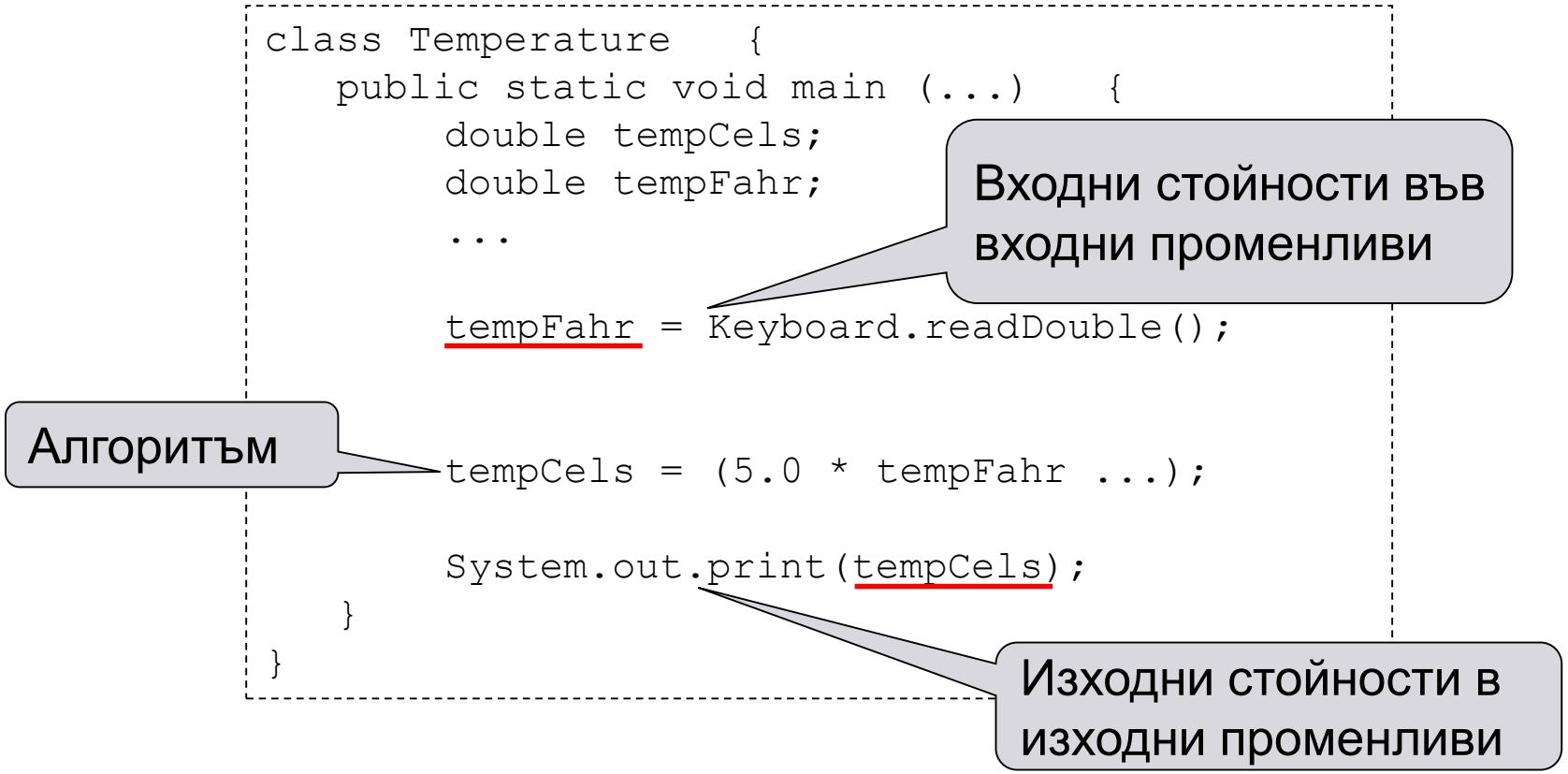
```
PROGRAM Temperature ;

VAR tempFahr : real ;
    tempCels : real ;


BEGIN
   ...
   readln (tempFahr);

   tempCels := (5.0 * tempFahr ...);

   write (tempCels);

END.
```

Входни стойности във входни променливи

Алгоритъм

Изходни стойности в изходни променливи

Императивна програма на Java



Императивно програмиране: детайли (1)

- Базови концепции:
  - Променлива, Оператор
- 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)
- Процедури (функции, методи):
  - Частични алгоритми: с оператори на езика

Императивно програмиране: детайли (2)

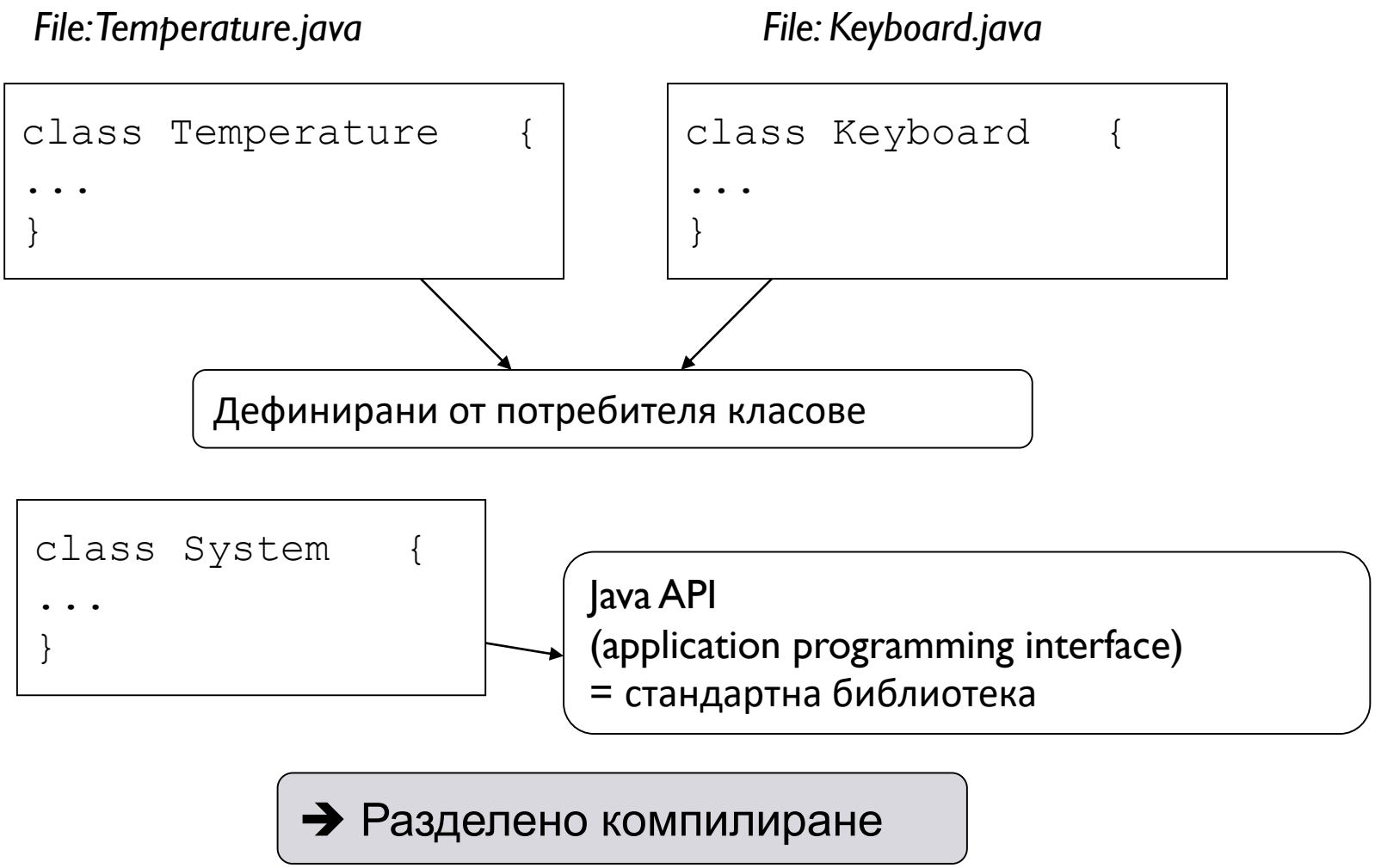
- Данни (променливи) и обработващи алгоритми (процедура, функция, метод) са разделени структури в императивното програмиране
- За сравнение с обектно-ориентираното:
  - Клас = единство от данни и обработващи алгоритми

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

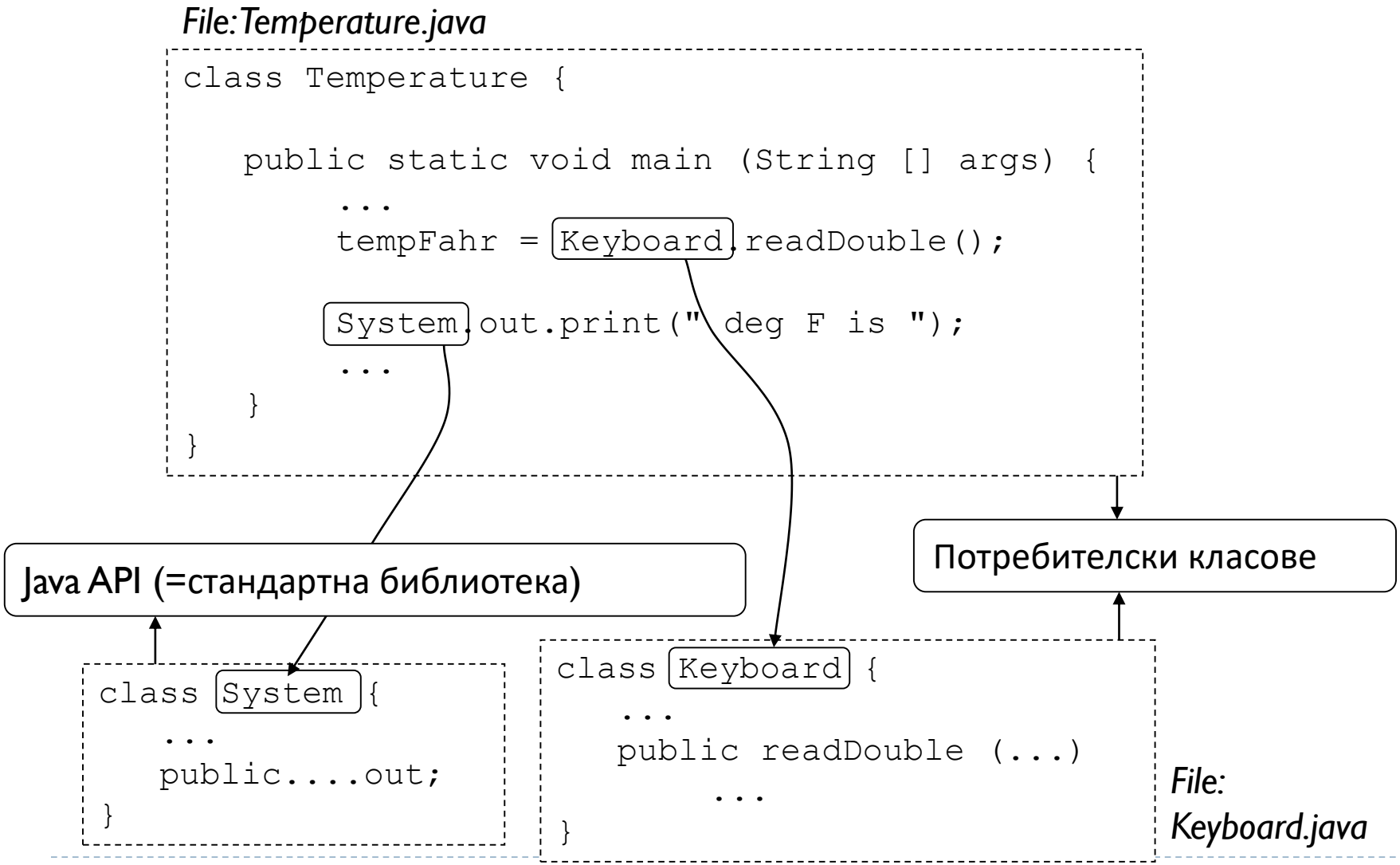
От колко компоненти се състои следната Java-програма?

```
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");
   }
}
```

Програмата се състои от три компонента



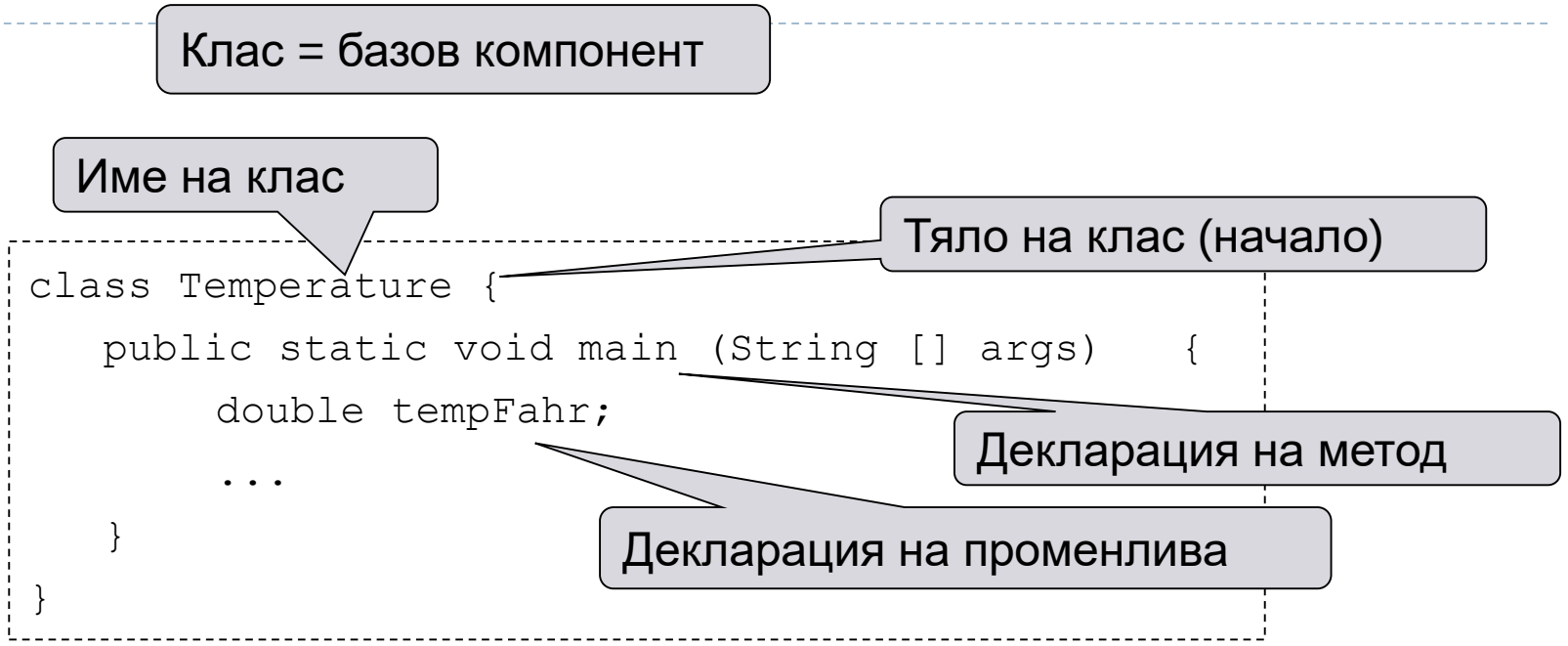
Интерфейси между компонентите



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

Базова структура на Java-програми

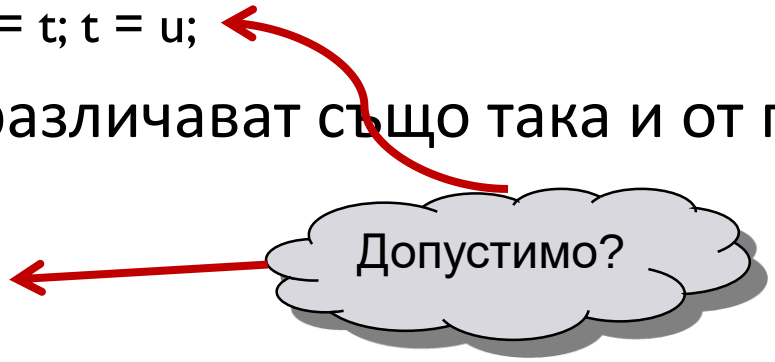


Клас: 'Множество' от
- Декларации на променливи
- Декларации на методи
  (метод = алгоритъм, процедура, функция)

За общата програма: *един метод m a i n ( ) → там започва обработката ! (както главна програма в Pascal)*

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
    - Напр. D t; u = t; t = u;
- Обектите се различават също така и от простите типове
  - Напр. u = 10



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 instance methods
- Някои методи могат да се извикват и посредством обикновения синтаксис
- Съществуват също така и 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'

```
import java.io.*;

class Keyboard {

// Author: M. Dennis Mickunas,
// June 9, 1997 Primitive Keyboard
// input of integers, reals,
// strings, and characters.

static boolean iseof = false;
static char c;
static int i;
static double d;
static String s;

/* WARNING: THE BUFFER VALUE IS SET
TO 1 HERE TO OVERCOME ** A KNOWN BUG
IN WIN95 (WITH JDK 1.1.3 ONWARDS)*/

static BufferedReader input
   = new BufferedReader(
new InputStreamReader(System.in),1);
```

```
public static int readInt() {
   if (iseof) return 0;
   System.out.flush();
   try {
         s = input.readLine();
   }
   catch (IOException e) {
         System.exit(-1);
   }
   if (s==null) {
         iseof=true;
         return 0;
   }
   i = new Integer(s.trim()).intValue();
   return i;
}

public static char readChar() {
   if (iseof) return (char)0;
   System.out.flush();
   ...
```

Недостатък: непрегледно представяне за потребителите на класа

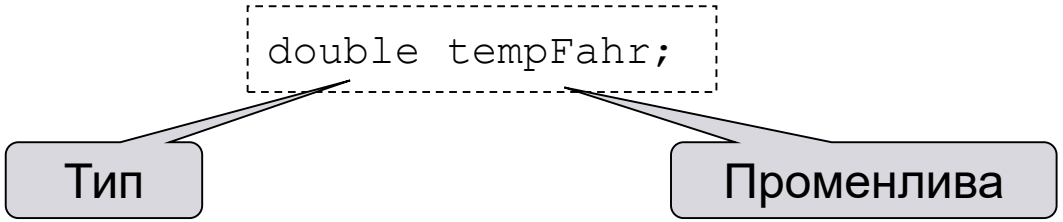
Клас Keyboard: една абстракция

Само това е важно за използването му

```
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



Действие:

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

Прости 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 (прод.)

- 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 имена 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 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

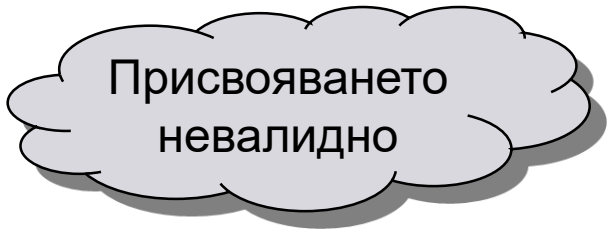
Изрази

- Могат да се образуват изрази, които съдържат:
  - Литерали
  - Променливи
  - Символни константи
  - Операции
- Съществуват правила за приоритети на операциите
  - Аналогично като в математиката
- Изразите могат да съдържат извиквания на методи
  - Могат да се появяват в дясната част на операторите за присвояване
    - С актуални параметри, които могат да бъдат изрази
  - Подобно на изразите те произвеждат стойности когато се обработват

Изрази

- Java винаги знае какъв тип на стойността имат изразите
  - **Много внимателно**: те да отговарят на типовете на променливите, в които се записват стойностите
- Пример:

```
int i;
double x;

i = 10.3*x;
```



Изрази (прод.)

- Съществуват изключения
  - Например: x = i + 10;
- Внимание с конвертирането на изрази, съдържащи различни типове данни
  - Възможно е автоматично конвертиране
    - Когато не се губи информация
  - При загуба на информация
    - Автоматично конвертиране не се извършва
- Можем да предизвикаме конвертиране
  - Оператор cast
  - Напр.: i = (int) (10.3 * x);

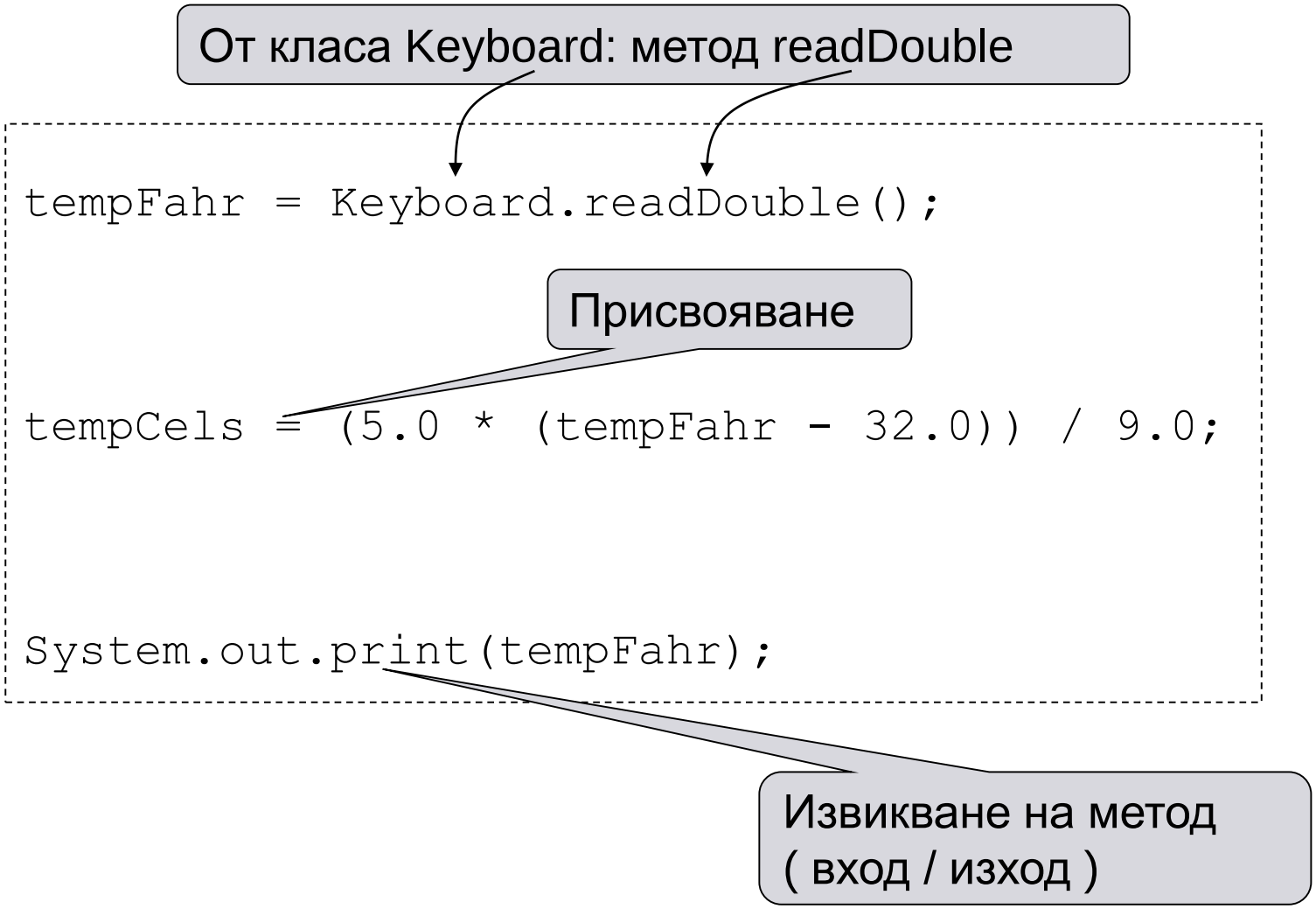
Коментари

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

      double tempFahr; // Fahrenheit

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

```
// до края на реда
```

```
/* между */
```

Прости оператори: присвояване и извикване на метод



Благодаря за вниманието!

Край лекция 2. “Основни езикови конструкции на императивните програми”