КОМПОНЕНТИ

ЛЕКЦИОНЕН КУРС "ПРОГРАМИРАНЕ НА JAVA"

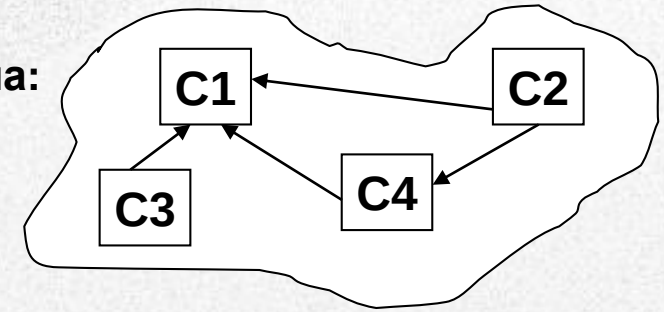
Distributed @learning center

ПРОБЛЕМ

1 Какви видове компоненти са възможни (смислени)?

Клас = компонент на една обектно-ориентирана софтуерна система
(в Java, C++, Smalltalk, Simula 67, Eiffel ...)

Софтуерна система:



Концепцията за класове в Java (също: C++ и др.) отразява само недостатъчно значимите видове софтуерни компоненти.

КЛАСОВЕ И КОМПОНЕНТИ

Клас: синтаксис

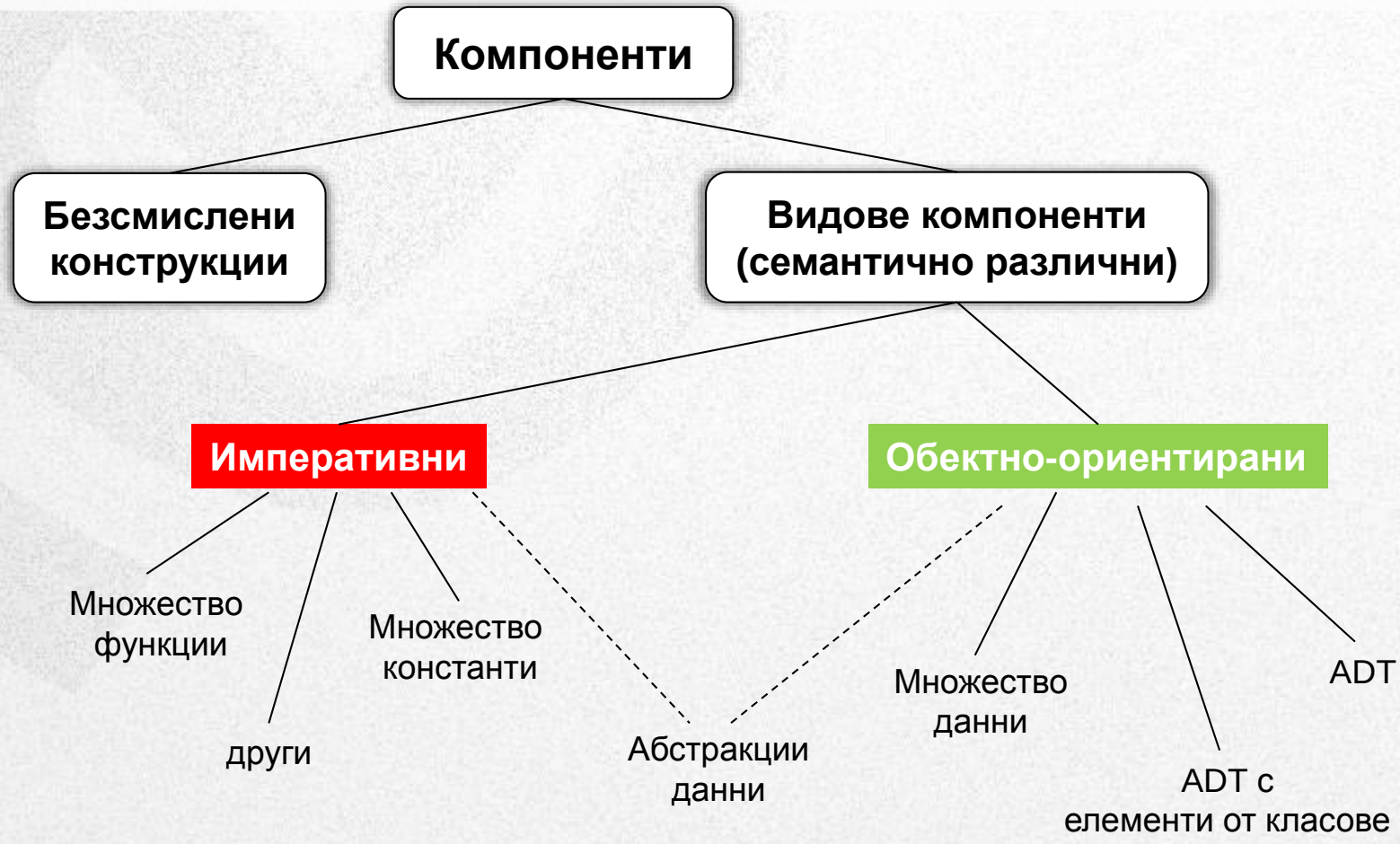
Множество от: **променливи** и **методи**

static/non-static public/private

Различни **видове** компоненти с най-различни свойства могат да бъдат създавани (напр. ADT, императивни компоненти …).

Цел на лекцията:
→ Въвеждане на наредба (класификация)
→ Ориентация и методическа помощ

КЛАСОВЕ И КОМПОНЕНТИ



ПРОБЛЕМИ: РАЗНООБРАЗИЕ ОТ ВИДОВЕ КОМПОНЕНТИ

- **Проблем на разпознаване**: четене на чужд софтуер

```
class ...
   private static int number;
   public void m() ...
   ...
```

напр. в Java-API

разпознаване: кой вид компонент?
(напр. Множество от функции, ADT, ...)
→ изводи: начин на използване

- **Проблем на разработването на нов софтуер**

Преднамерен вид на компонента (напр. ADT - базов)

Развитие на синактическата структура в Java

РАЗПОЗНАВАНЕ ВИДА НА КОМПОНЕНТА: ПО КЛЮЧОВИ ДУМИ

Абстрактни типове данни (ADT)

В примера: променливи и методи без static

```
class Stack   {
   private char[] stackElements;
        ...
   public Stack(int n)    {
        stackElements = new char[n];
        top = -1;
   }
   public void push(char x)    {
        ...
   }
}
```

- Променливи, методи: не 'static'
  → Променливи и методи на инстанции
- Променливи: 'private' → ‘скрит'
- Методи: 'public' → ‘навън'

Използване:
Създаване инстанции

ВЪПРОСИ

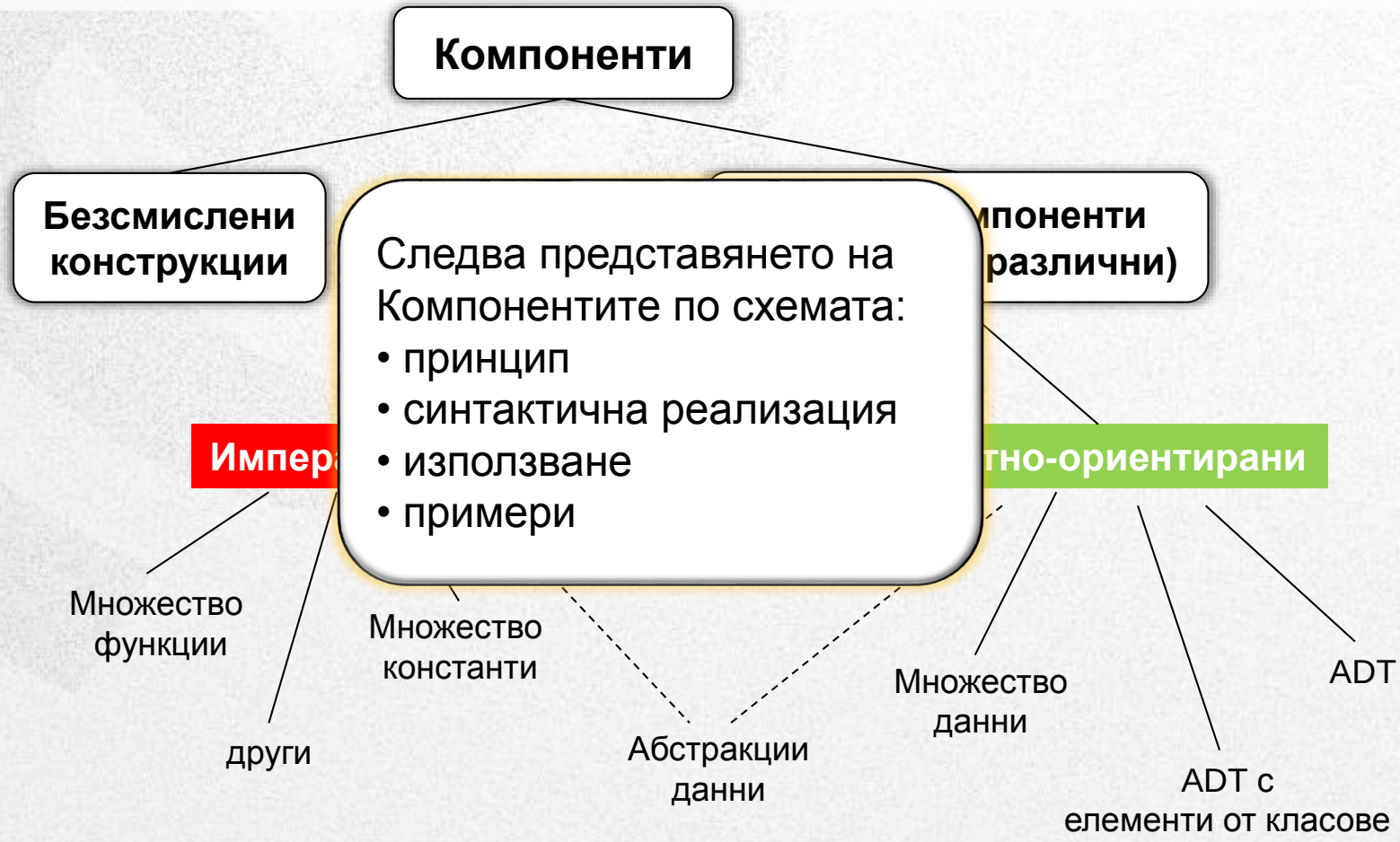
1. Кои видове компоненти са съществени за разработване на софтуер?
2. Как се реализират те в Java?

Разработване на подходящи компоненти е един от най-важните и най-тежките проблеми на разработването на софтуер (фаза: проектиране/разработване)

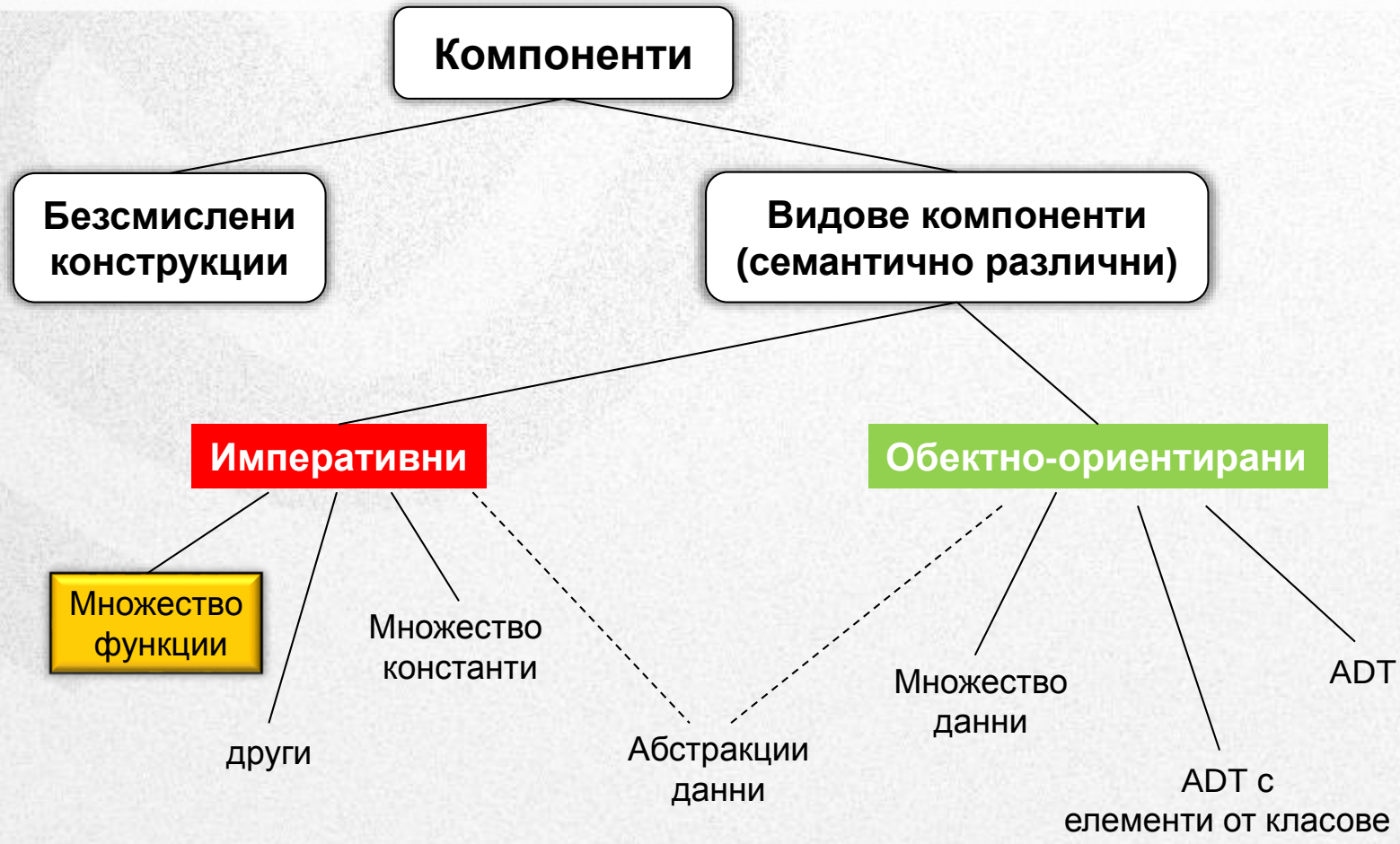
(специална лекция “Софтуерни технологии”)

Първично: Какви видове компоненти трябва да бъдат моделирани?
Изведено: Синтактична интерпретация на класа?

КЛАСОВЕ И КОМПОНЕНТИ



КЛАСОВЕ И КОМПОНЕНТИ



МНОЖЕСТВО ФУНКЦИИ

Множество от свързани функции без общи данни (ев. някои константи)

→ Данните не могат да служат за обмен на информация между функциите (сравни Stack)
→ Така: множество от независими алгоритми

Java-клас: само 'public' 'static'-методи (+ някои 'final' 'static' - променливи)

Отклонение: private-методи (помощни функции)

Използване: създаване на инстанции безсмислено

Пример: математически функции в Java-API

API-Klasse java.lang.Math

java.lang
Class Math

java.lang.Object
└ java.lang.Math

public final class Math
extends Object

The class Math contains metho ...s.

Unlike some of the numeric me ...same
results. This relaxation permits ...

By default many of the Math m ...n-
specific native libraries or micr ...nce
implementations still must conf ...

The quality of implementation s ...oint
Math methods is measured in t ...n the
two floating-point values brack ... ulps
cited is for the worst-case erro ...he
exact result; such a method is ...al for
many floating-point methods to be correctly rounded. Instead, for the Math class, a larger error bound of 1 or 2 ulps is allowed for certain methods. Informally, with a 1 ulp error bound, when the exact result is a representable number, the exact result should be returned as the computed result; otherwise, either of the two floating-point values which bracket the exact result may be returned. For exact results large in magnitude, one of the endpoints of the bracket may be infinite. Besides accuracy at individual arguments, maintaining proper relations between the method at different arguments is also important. Therefore, most methods with more than 0.5 ulp errors are required to be *semi-monotonic*: whenever the mathematical function is non-decreasing, so is the floating-point approximation, likewise, whenever the mathematical function is non-increasing, so is the floating-point approximation. Not all approximations that have 1 ulp accuracy will automatically meet the monotonicity requirements.

Since:
JDK1.0

Field Summary	
static double	E The ...
static double	PI The ...

public final class java.lang.Math

The class Math contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

Две константи:

static double E
static double PI

Method Summary

static double	**abs**(double a) Returns the absolute value of a double value.
static float	**abs**(float a) Returns the absolu
static int	**abs**(int a) Returns the absolu
static double	**acos**(double a) Returns the arc co
static double	**asin**(double a) Returns the arc sin
static double	**atan**(double a) Returns the arc tan
static double	**ceil**(double a) Returns the smalle integer.
static double	**cos**(double a) Returns the trigono
static double	**exp**(double a) Returns Euler's nu
static double	**floor**(double a) Returns the largest
static double	**log**(double a) Returns the natural
static double	**max**(double a, doubl Returns the greater
static float	**max**(float a, float Returns the greater
static int	**max**(int a, int b) Returns the greater of two int values.
static double	**min**(double a, double b) Returns the smaller of two double values.
static float	**min**(float a, float b) Returns the smaller of two float values.
static int	**min**(int a, int b) Returns the smaller of two int values.
static double	**pow**(double a, double b) Returns the value of the first argument raised to the power of the second argument.

Java-API:

public final class Java.lang.Math

Избор на методи:

static double abs(double a)
static double asin(double a)
static double cos(double a)
static double log(double a) // Basis e
static double pow(double a, double b)

public static final double E

The double that is closer than any other to e, the base of the natural logarithm

public static final double PI

The double that is closer than any other to pi,

…

public static double asin(double a)

Returns the trigonometric sine of an angle.
Special cases: …

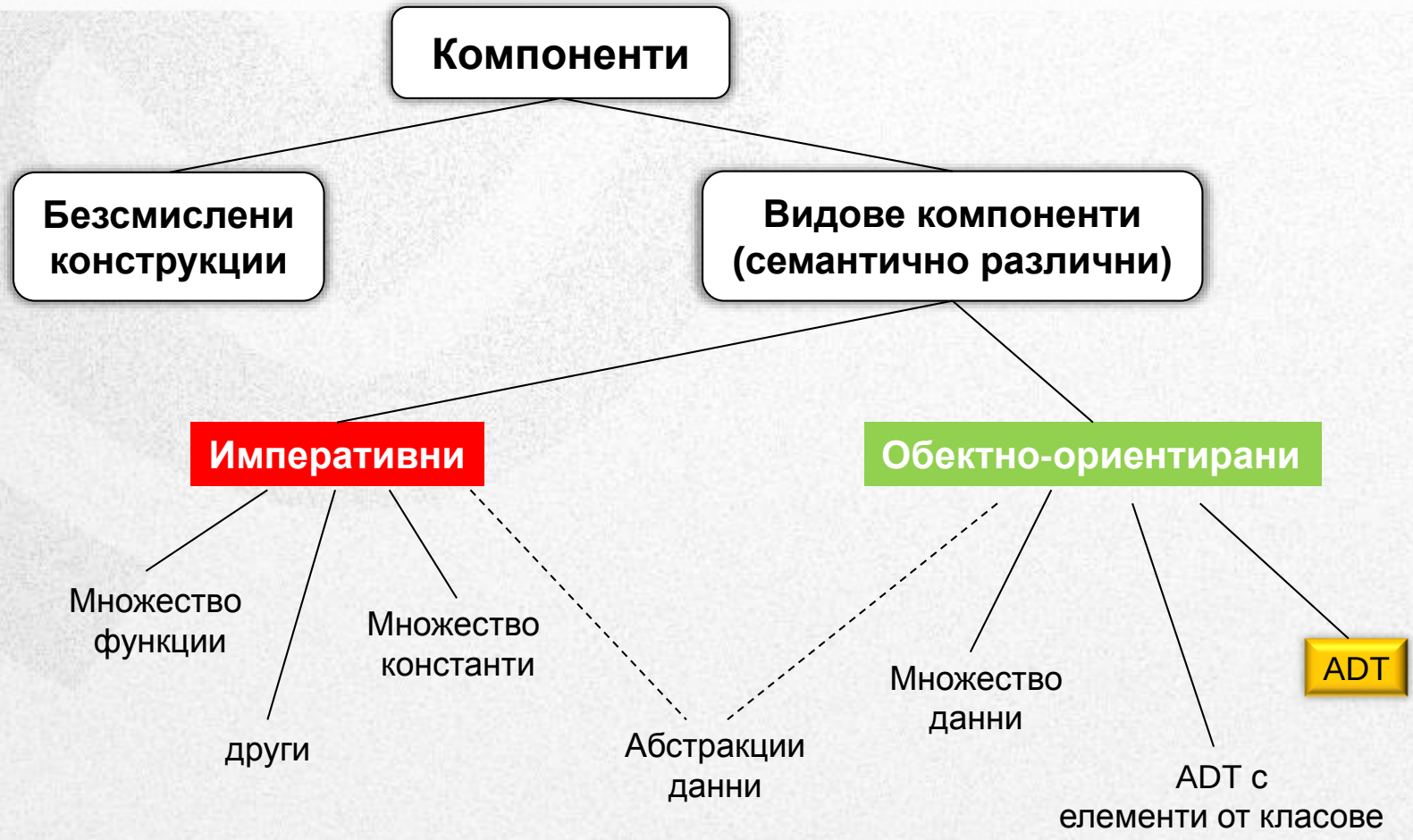
Parameters:

a – an angle, in radians

Използване: без създаване на инстанции

Math.sin(1.2); Math.PI;

КЛАСОВЕ И КОМПОНЕНТИ



ADT

Множество от свързани функции с общи скрити данни
Нов тип, дефиниран от потребителя

Java-клас:
'private' non-static - променливи
'public' non-static - методи

Използване: произволно много инстанции (обекти) от типа могат да бъдат създадени

Примери: Stack, Time, HashTable

Отклонения:
- public-променливи (по-бърз достъп)
- private-методи (помощни функции)

Опасно!

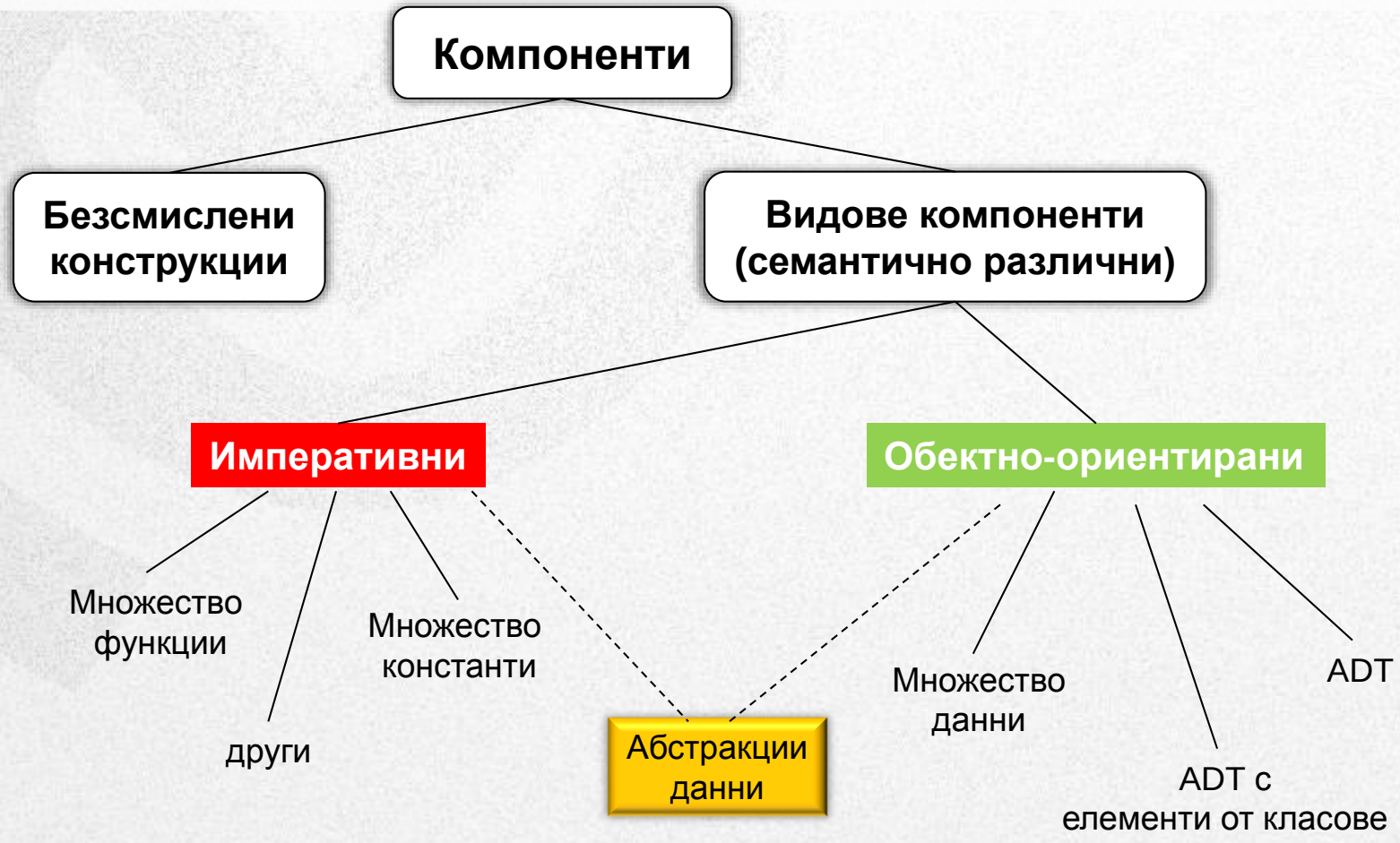
ПРИМЕР ЗА ADT

```
class Time   {
   // all non-static
   private int hour, minute;
   public Time() ...
   public addMinutes ...
   public printMinutes ...
   public timeInMinutes ...
   public printTimeInMinutes ...
}
```

```
class Schedule   {
   public main ...
      Time t1 = new Time(8,30);
      Time t2 = new Time();
      Time t3, t4;
  ...
}
```

Използване: Създаване на инстанции

КЛАСОВЕ И КОМПОНЕНТИ



АБСТРАКЦИЯ ДАННИ

Множество на свързани функции с общи скрити данни
Без възможност за създаване на инстанции
(необходим само един екземпляр)

Специален случай на ADT

Java-клас:
’private’ ’static’ – променливи
’public’ ’static’ - методи

Използване: няма създаване на инстанции

Пример: Keyboard
(също: Stack, Time могат да бъдат така реализирани)

СТЕК КАТО АБСТРАКЦИЯ ДАННИ

```
class Stack {
   private char [] stackElements;
   private int top;

   public void Stack(int n) {
         stackElements = new char [n];
         top = -1;
   }
   public boolean isempty() {
         return top == -1;
   }
   public void push(char x) {
         top++;
         stackElements[top] = x;
   }
   public char top() {
         if (isempty()) {
            System.out.println("Stack empty");
            return ' ';
         }
         else return stackElements [top];
   }
   public void pop() {
         if (isempty())     System.out.println("Stack empty");
         else               top--;
   }
}
```

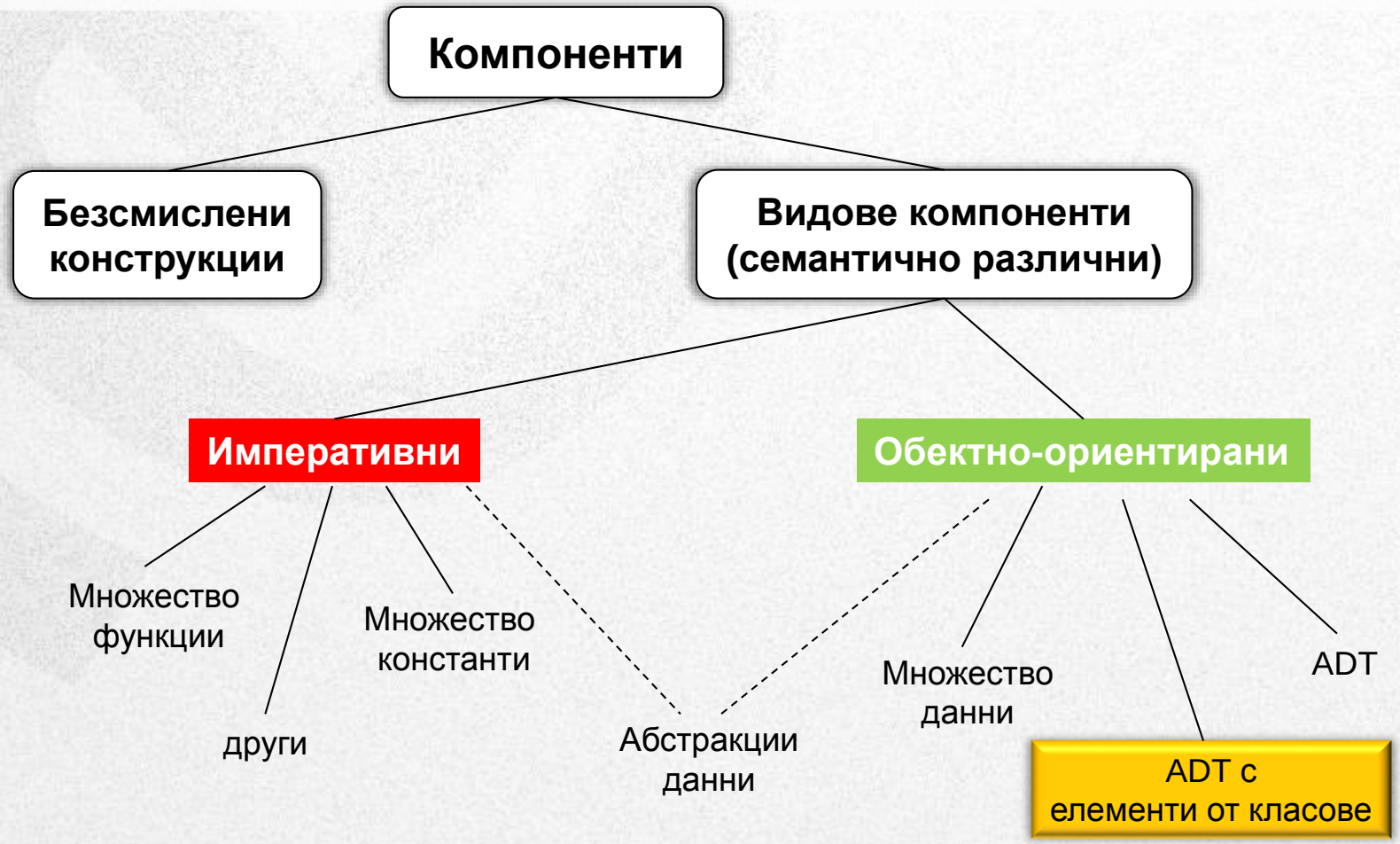
Тук: ADT Stack

1. Какво трябва да се промени?

СТЕК КАТО АБСТРАКЦИЯ ДАННИ

```
class Stack {
   private static char [] stackElements;
   private static int top;

   public static void init(int n) {
         stackElements = new char [n];
         top = -1;
   }
   public static boolean isempty() {
         return top == -1;
   }
   public static void push(char x) {
         top++;
         stackElements[top] = x;
   }
   public static char top() {
         if (isempty()) {
            System.out.println("Stack empty");
            return ' ';
         }
         else     return stackElements [top];
   }
   public static void pop() {
         if (isempty())     System.out.println("Stack empty);
         else               top--;
   }
}
```

КЛАСОВЕ И КОМПОНЕНТИ



ADT С ЕЛЕМЕНТИ НА КЛАСОВЕ

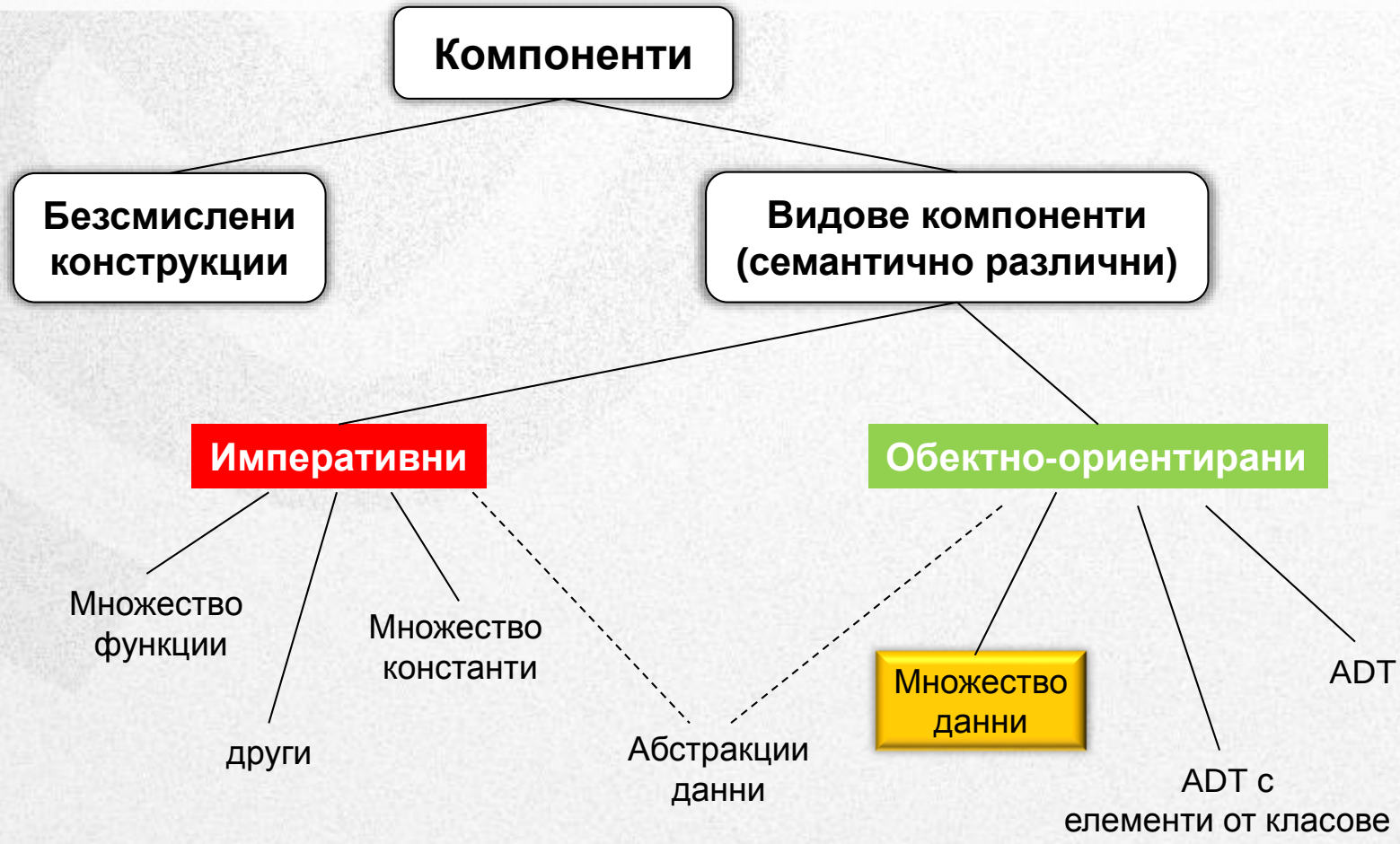
Множество от свързани функции с общи скрити данни
- Някои променливи/методи само прости (без създаване на инстанции)
- Произволно много инстанции (обекти/променливи) от типа

Променливи като обща памет на всички инстанции от типа

Java-клас:
- 'private' non-static или static – променливи
- 'public' non-static или static – методи

Пример: Time с променливи на клас 'noonHour' … и методи на клас switchTimeFormat() → TimeC

КЛАСОВЕ И КОМПОНЕНТИ



МНОЖЕСТВО ДАННИ (КЛАСОВЕ ДАННИ)

Обобщение на видими данни в един тип

Java-клас:
няма методи
само не-static public данни

Използване: създаване на инстанции

Пример: Point3D
→ както Pascal-Record (съотв. C-Struct)

ПРИМЕР: МНОЖЕСТВО ДАННИ

1 Коментар?

Точки в триразмерното пространство се състоят от 3 стойности: x-, y-, z- координати

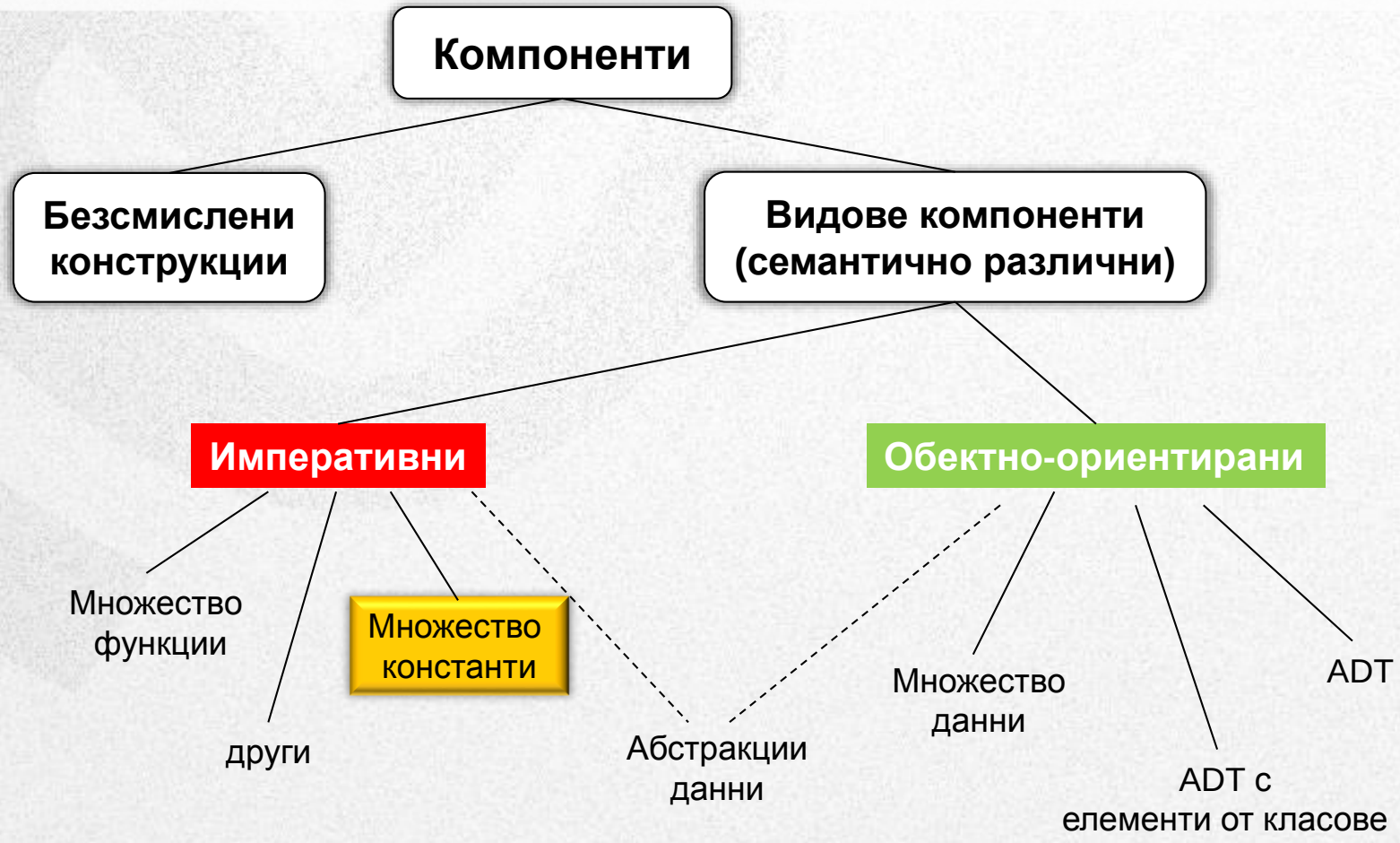
```
class Point3D   {
   double x, y, z;
}
```

Видима в актуалния пакет

```
class Point3D   {
   public double x, y, z;
}
```

Видима във всички пакети

КЛАСОВЕ И КОМПОНЕНТИ



МНОЖЕСТВО КОНСТАНТИ

Обобщение на видими константи

Java-клас:
- няма методи
- само public final static променливи

Използване: няма създаване на инстанции

Пример: управляващи печата символи като константи

ПРИМЕРИ: МНОЖЕСТВО КОНСТАНТИ

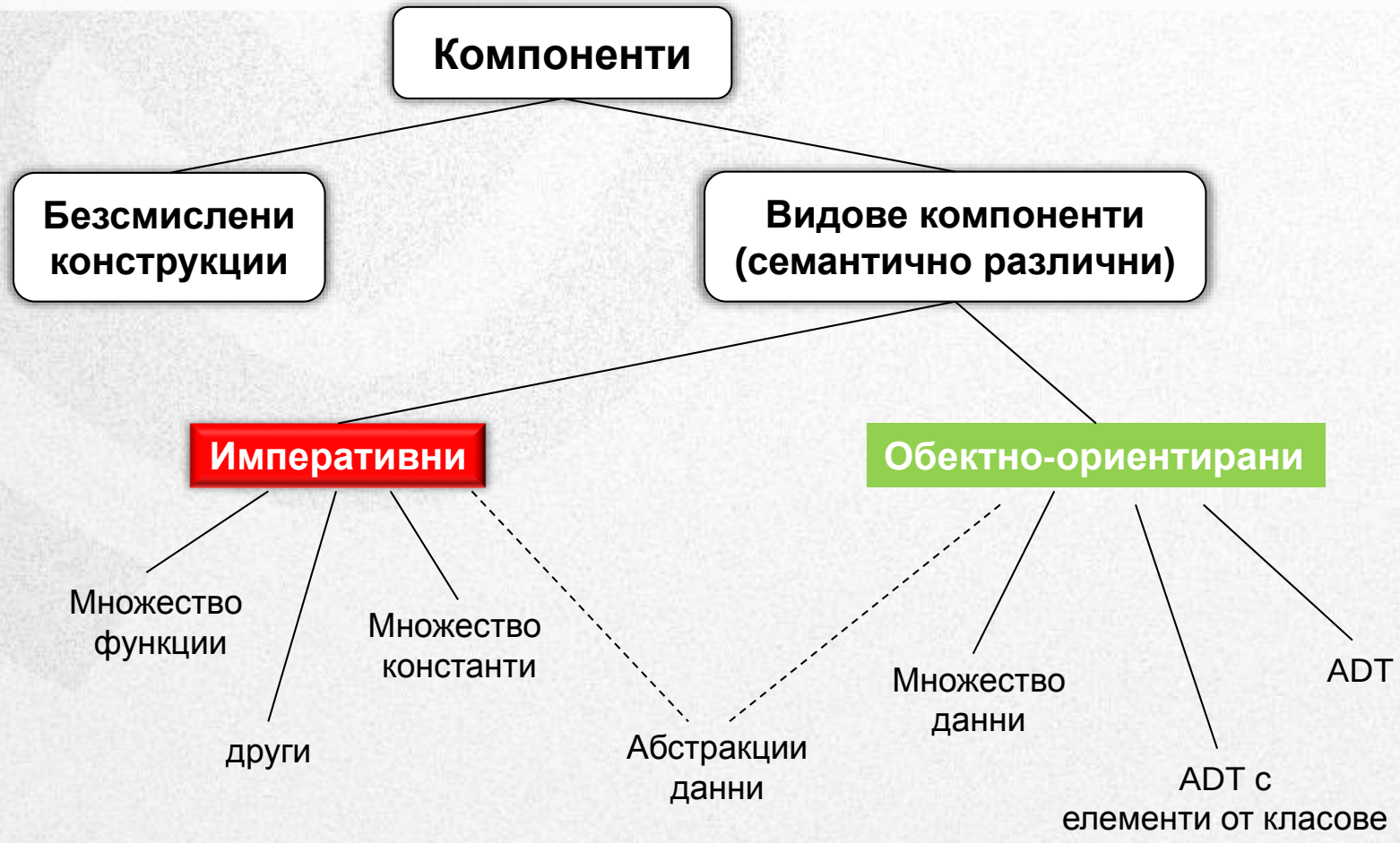
```
class IO_Constants {

   public final static char LF = '\n',
                            FF = '\f',
                            CR = '\r';

   public final static int DRUCK_WIDTH = 80;

}
```

C, C++: .h-Files:

```
#define LF '\n'
#define FF '\f'
#define CR '\r'
#define DRUCK_WIDTH 80
```

В C, C++: имената несвързани с типа → чиста замяна на текст

КЛАСОВЕ И КОМПОНЕНТИ



ИМПЕРАТИВНИ КОМПОНЕНТИ

Компоненти, за които създаването на инстанции не е смислено

Java-клас:
данни и методи само 'static'

Използване: няма създаван на инстанции от класове

Примери: засяга алгоритмите

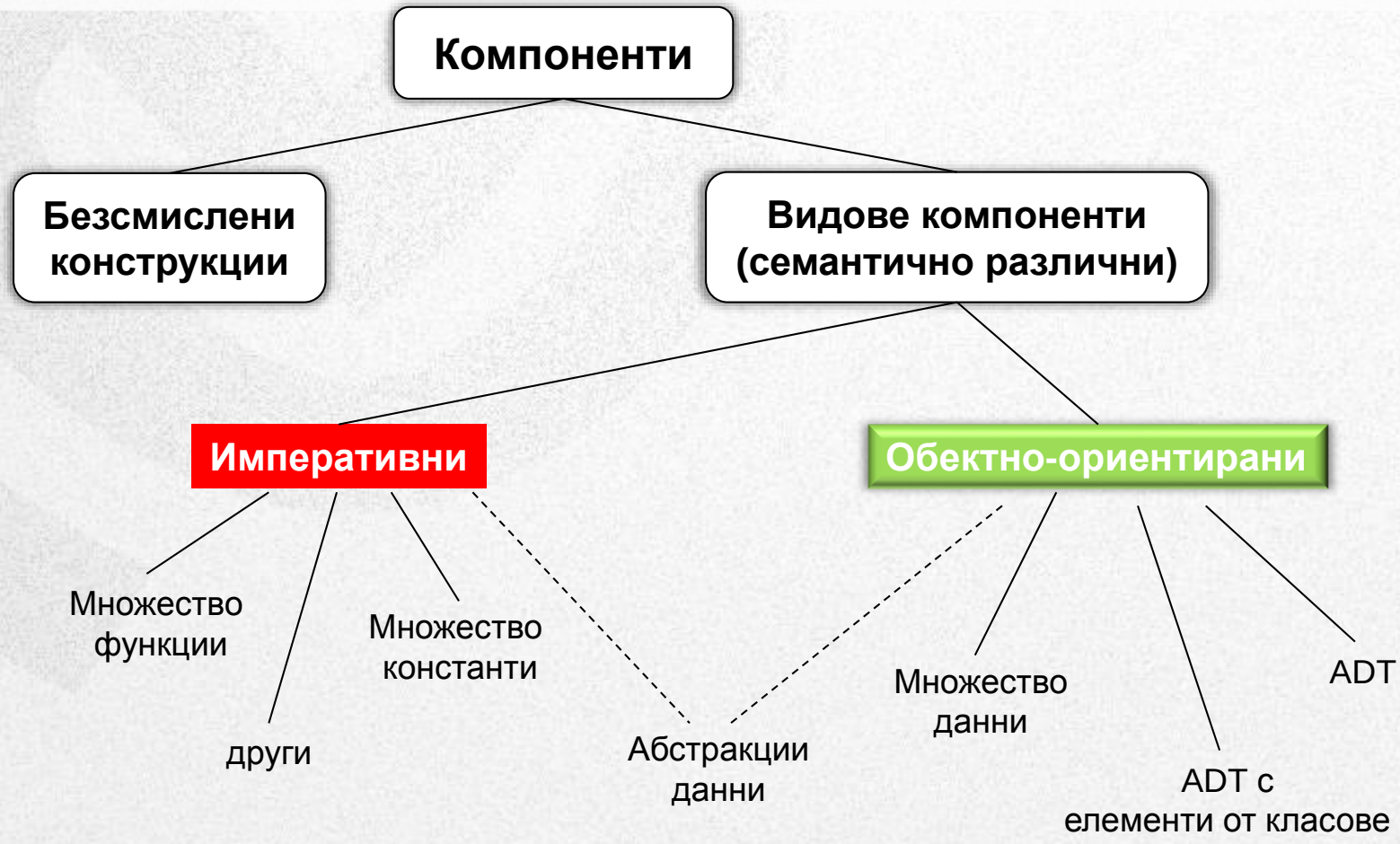
- Методи за сортиране
- Hanoi.java
- Множества от функции
- Main-клас: съдържа main()
  → Никога създаване на инстанции
  → Начало на алгоритъма

ИНСТАНЦИИРАНЕ НА ЕДИН MAIN-КЛАС

1 Коментар?

```
class Welcome {

    Welcome()    {
         System.out.println("Welcome!");
    }

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

КЛАСОВЕ И КОМПОНЕНТИ



ОБЕКТНО-ОРИЕНТИРАН КОМПОНЕНТ

Компоненти, за които създаването на инстанции е смислено (необходимо)

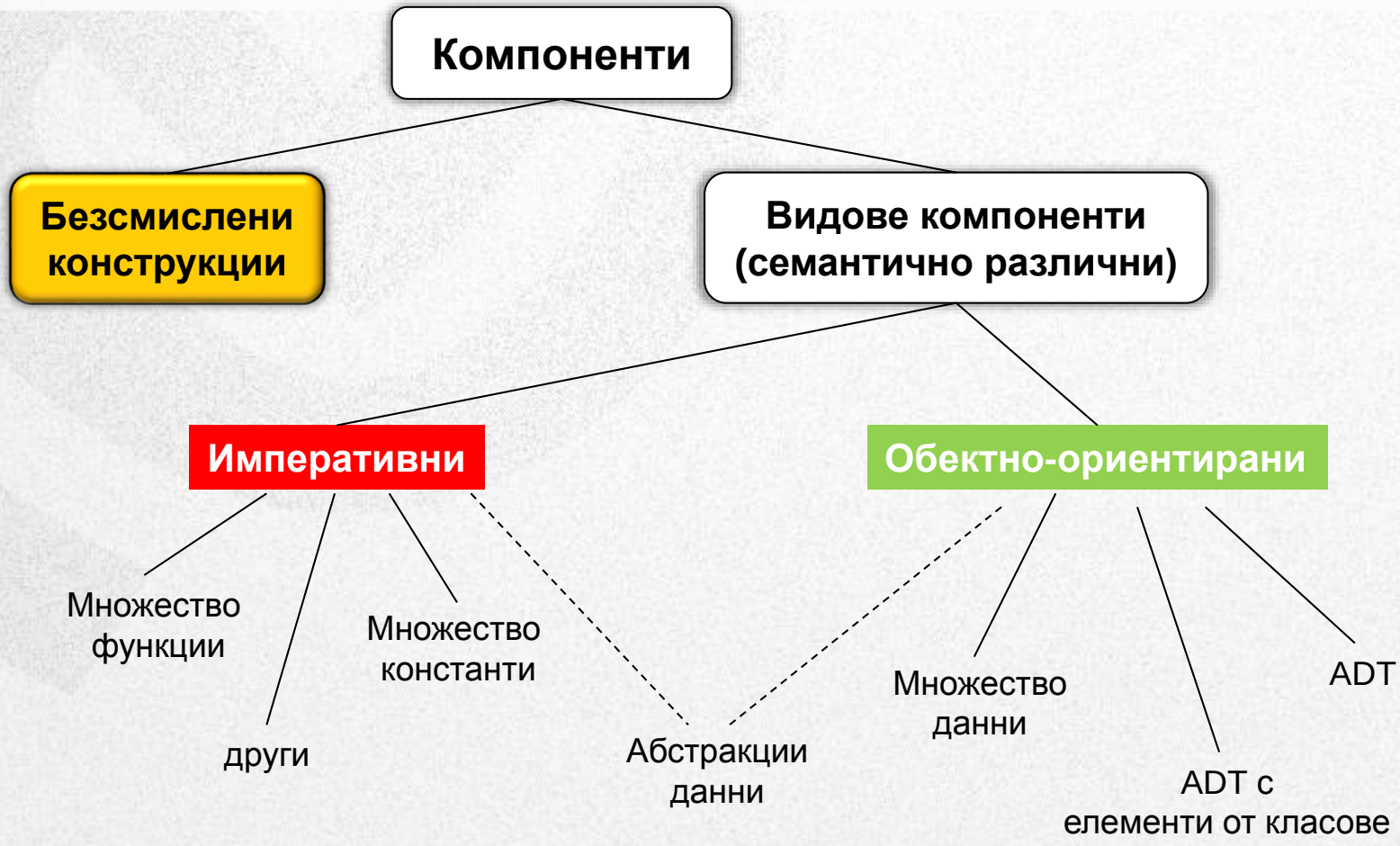
Java-клас:
съдържа non-static елементи (данни или методи)

Използване: Създаване на инстанции необходимо, за да се създадат променливи (памет) съответно методите да се направят използваеми

Примери:

- ADT
- ADT с елементи на компоненти
- Множества данни

КЛАСОВЕ И КОМПОНЕНТИ



БЕЗСМИСЛЕНИ КОНСТРУКЦИИ НА КЛАСОВЕ

Въпреки синтактичната коректност

- Всички елементи (данни, методи): 'private'
- Всички данни: 'public'; всички методи: 'private'
- Всички елементи на класа C: 'static, + създаване на инстанции new C()
- Променливи: 'private', non-static; всички методи: 'static'
- Всички променливи: 'static'; всички методи: non-static
- Други ?

ПРИМЕР: ВСИЧКИ ЕЛЕМЕНТИ 'PRIVATE'

1. Какво следва?

Никога достъпни

```
class Time {

   private int hour, minute;

   private final int noonHour = 12;

   private addMinutes (int m)    {
        ...
   }

}
```

ПРИМЕР: (ВСИЧКИ) ДАННИ 'PUBLIC, ВСИЧКИ МЕТОДИ 'PRIVATE'

1. Какво следва?

Безполезни методи

```
class Time {

   public int hour, minute;

   public final int noonHour = 12;

   private addMinutes (int m)    {

        hour = ...
   }

}
```

ПРИМЕР: ВСИЧКИ ЕЛЕМЕНТИ 'STATIC, + СЪЗДАВАНЕ НА ИНСТАНЦИИ

1 Какво следва?

Празни инстанции

```
class Time {

   public static int hour, minute;
   public final static int noonHour = 12;

   public static addMinutes (int m)    {
        hour = ...
   }
}
```

```
class Apply {

   Time t1, t2;
   public static void main(...)  {
        t1 = new Time ();
        t2 = new Time ();
   }
}
```

ПРИМЕР: ПРОМЕНЛИВИ 'PRIVATE', NON-STATIC ВСИЧКИ МЕТОДИ 'STATIC'

1. Какво следва?

Не е възможна обработката на променливи на инстанции

```
class Time {

   private int hour, minute;

   private final int noonHour = 12;

   public static Time() {
        ...
   }

   public static addMinutes (int m)    {

        hour = ...
   }

}
```

Методите на класове (static) не могат да обработват променливи на инстанции

ПРИМЕР : ПРОМЕНЛИВИ 'STATIC, ВСИЧКИ МЕТОДИ NON-STATIC

1 Какво следва?

Инстанции без данни

Методите на всички инстанции идентични

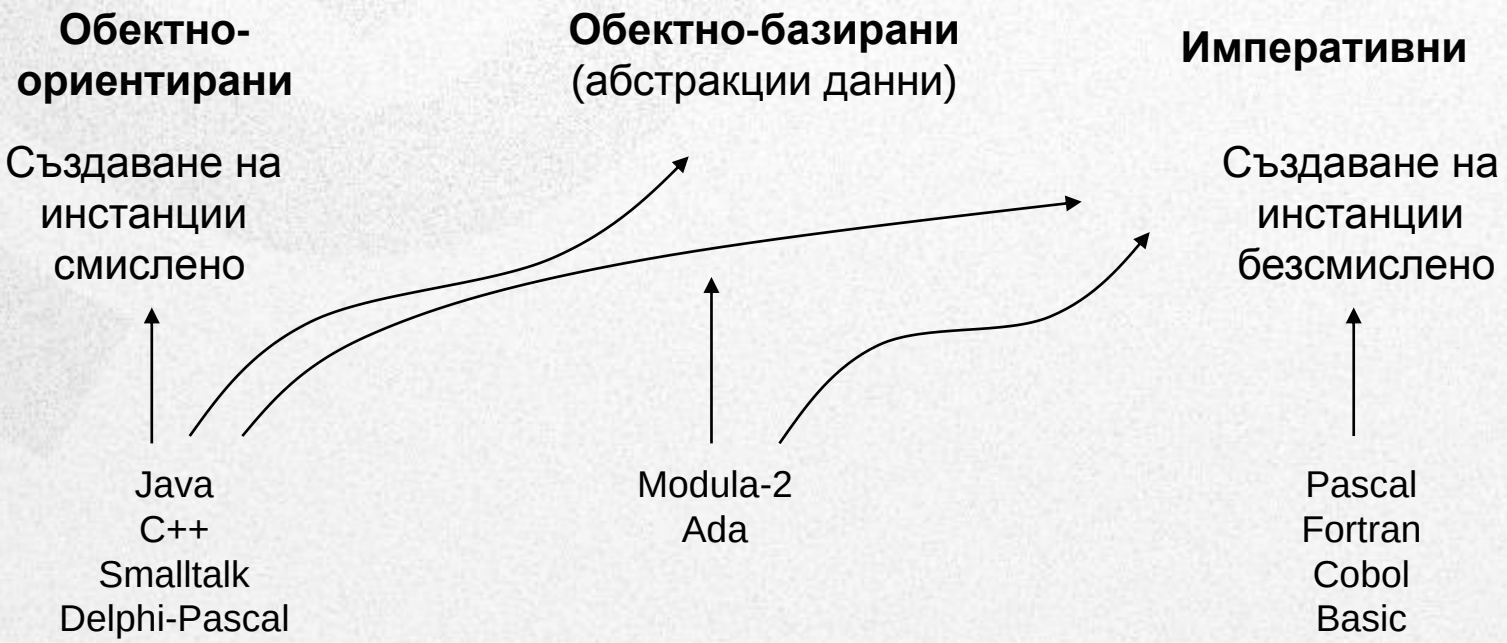
```
class Time {

   private static int hour, minute;
   private final static int noonHour = 12;

   public addMinutes (int m)    {
        hour = ...
   }
}
```

```
Time t1, t2;
t1 = new Time ();
t2 = new Time ();
t1.addMinutes(20);
t2.addMinutes(10);
```

ВИДОВЕ КОМПОНЕНТИ И ЕЗИЦИ ЗА ПРОГРАМИРАНЕ

Проблем: кой език за програмиране кои видове компоненти може да поддържа?

Всички видове компоненти



БЛАГОДАРЯ ЗА ВНИМАНИЕТО!

КРАЙ “КОМПОНЕНТИ”

Distributed
@learning center