# УВОД ООП (ПОВТОРЕНИЕ)

## ЛЕКЦИОНЕН КУРС "ООП(JAVA)"

Distributed @learning center

проф. д-р Станимир Стоянов

# КОМПЛЕКСНОСТ НА СОФТУЕРА

- Софтуерните системи принадлежат към най-комплексните създания на човека:
  - Структурите и поведението на големите системи в общия случай не са обозрими;
  - Те не могат да бъдат напълно разбрани както в началото при развоя, така също и в края при тестването,експлоатацията и поддръжката.

- Решаващата характеристика на индустриално използвания софтуер е, че за отделния разработчик е много трудно (дори невъзможно) да рабере всички тънкости на развоя
  - Просто казано, комплексността на такива системи надхвърля възможностите на човешкия интелект

# ООП

- Основен проблем на разработване на софтуер: **комплексност**
- ООП възниква за решаване на този проблем

# ДЕКОМПОЗИРАНЕ

- Софтуерните системи не могат да бъдат напълно разбрани:
  - Както предварително при развоя
  - Така също и впоследствие, при използването им
- Първи принцип за овладяване комплексността на разработването на софтуер:
  - **декомпозиция**

# ДЕКОМПОЗИЦИЯ

- Разлагане на софтуера на модули
- Всеки модул: поединично обхващаем
- Модули:
  - Разработване независимо от останалата част на системата
  - Семантично:
    - Модулите съответстват на подзадачи

# АБСТРАКЦИЯ

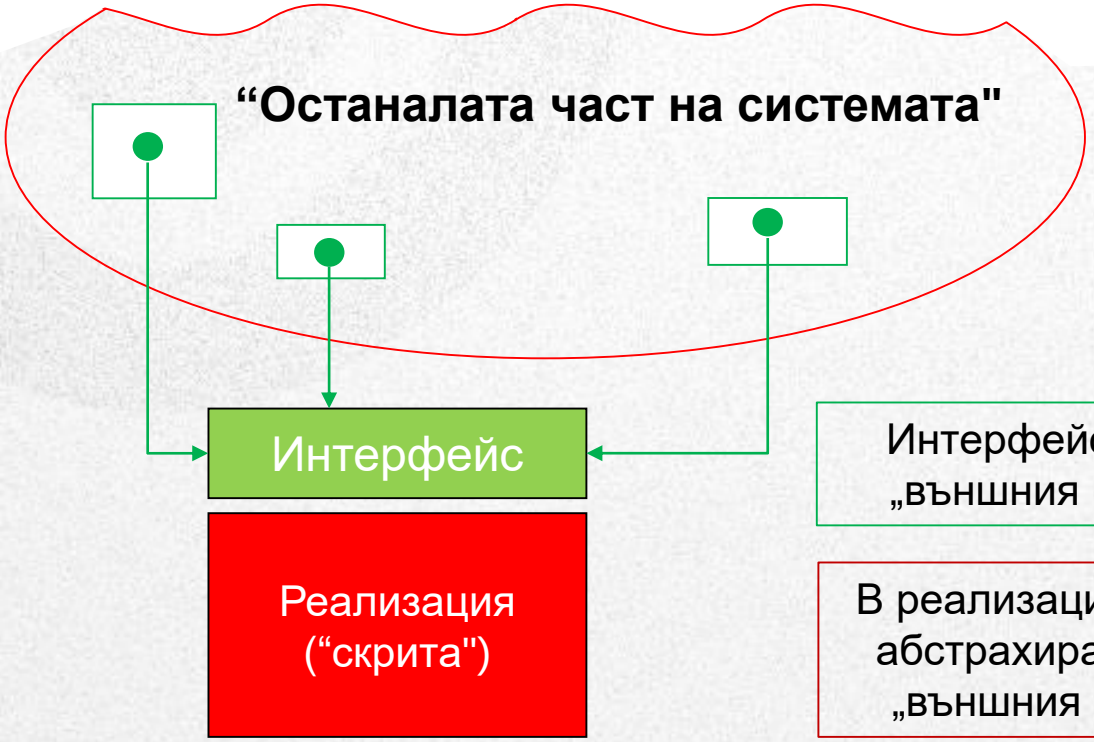
- Софтуерните системи не могат да бъдат напълно разбрани:
  - Както предварително при развоя
  - Така също и впоследствие, при използването им
- Втори принцип за овладяване комплексността на разработването на софтуер:
  - **абстракция**

# АБСТРАКЦИЯ

- Модулите са **абстракции**
- Останалата част от софтуерната система познава само:
  - Външното поведение на модулите
  - Не обаче детайли на реализацията

**Интерфейс на модула**

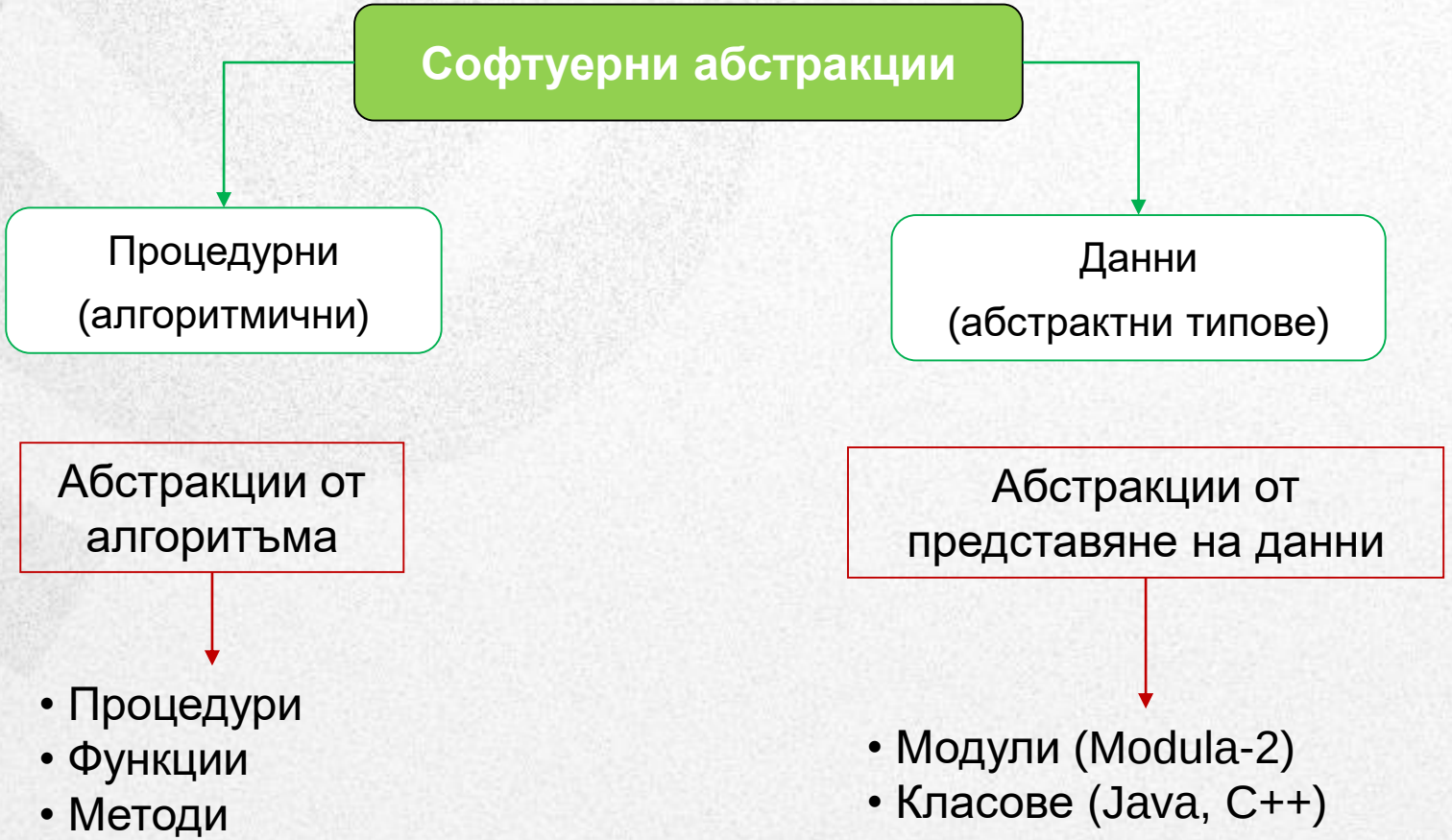
# МОДУЛИ




Интерфейс към „външния свят“

В реализацията се абстрахираме от „външния свят“

# СОФТУЕРНИ АБСТРАКЦИИ



# РАЗВИТИЕ НА АБСТРАКЦИЯТА

- Всички езици за програмиране предоставят абстракции
- Доколко сложността на проблемите, които решаваме, е директно свързана с вида и качеството на абстракцията?
  - Вид: това, което представя абстракцията
- Видове абстракция:
  - На компютъра (пространството на решението)
  - На задачата (пространство на задачата)

# ДВЕ ПРОСТРАНСТВА

- Асемблерните езици предоставят слаба абстракция на компютъра
- Много от императивните езици повишиха нивото на абстракция на асемблерните езици
  - Т.е., те са абстракция на асемблерните езици
  - Fortran, BASIC, C
  - Използвайки тези езици все още се изисква програмистите да мислят от гледна точка на структурата на компютъра
    - А не от аспекта на структурата на задачата
  - Програмистите трябва да установят асоциация между машинния модел (пространство на решението) и модела на задачата (пространство на задачата)
- Алтернативата на моделиране на машината е моделиране на задачата
  - Lisp – всички задачи в основата са си списъци
  - APL – всички задачи са алгоритмични
  - Prolog – задачите като верига от логически изводи

# ООП

- ООП отива една стъпка по-напред
- Предоставя инструменти на програмиста за представяне на елементи в пространството на задачата
- Това представяне е достатъчно общо, така че програмистът не е ограничен до определен тип задачи
- Тези елементи са „обекти“
  - Всеки обект изглежда като малък компютър
  - Има състояние и притежава операции, които може да изпълни
  - Същевременно не толкова лоша аналогия с обектите от реалния свят
    - Всички те притежават характеристики и поведения

# ХАРАКТЕРИСТИКИ НА ООП

- Следните характеристики:
  - Всичко е обект
    - Съхранява данни
    - Могат да се „правят заявки“ към обектите
    - Концептуалните елементи в една задача се представят като обекти
  - Една програма е съвкупност от обекти, взаимодействащи посредством изпращане на съобщения
    - Заявките към обектите се осъществяват чрез „изпращане на съобщения“
    - Едно съобщение е заявка за извикване на функция, която принадлежи към определен обект

# ХАРАКТЕРИСТИКИ НА ООП

- Обектите притежават собствена памет, съставена от други обекти
  - Нов вид обект съдържащ други обекти
  - По този начин можем да изграждаме сложност в една програма, скривайки я зад простотата на обектите
- Всеки обект има тип
  - Класовете представят (синоними са на) типове данни
  - Всеки обект е екземпляр на клас
  - Съществена характеристика на един клас - какви съобщения можем да изпращаме към него
- Всички обекти от определен тип могат да получават едни и същи съобщения

# КЛАСОВЕ

- Аристотел: начало на изучаване концепцията за типа
  - Класа на рибите, класа на птиците, …
- Идеята: всички обекти, макар и уникални, са също част от клас от обекти, които имат общи характеристики и поведение
- Ключовата дума “class” за първи път използвана в първия обектно-ориентиран език за програмиране Simula-67
  - За разработка на симулации
  - Обектите са идентични с изключение на състоянието им по време на изпълнение на програмата са групирани в „класове от обекти“
- Създаването на абстрактни типове данни (класове) е основна концепция в ООП

# АБСТРАКТНИ ТИПОВЕ ДАННИ

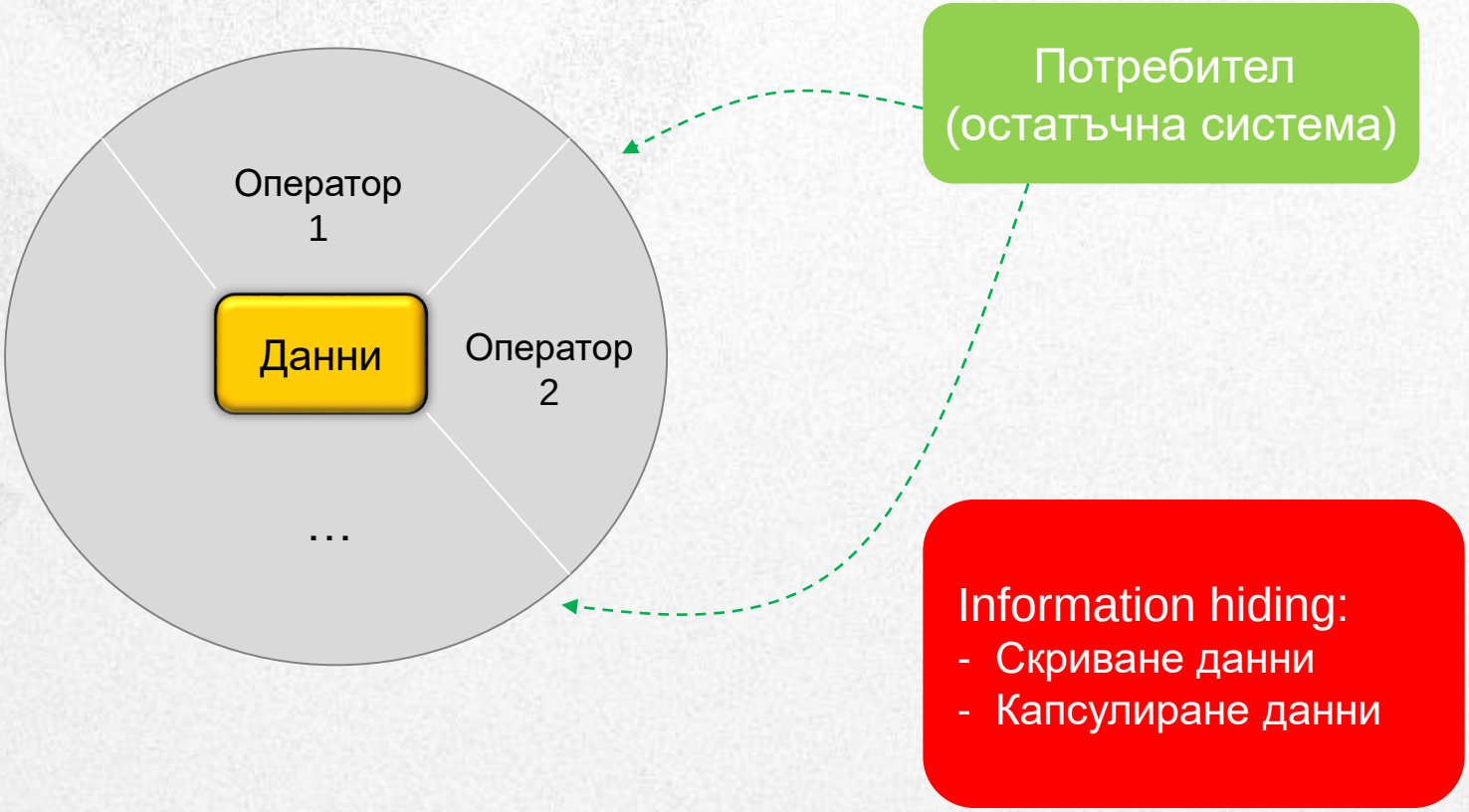
**Единица от данни и операции**

- **Операции:** служат за обработка на данни (инициализация, промяна, четене, изтриване)
- **Данни:** защитени/скрити от "външния свят"

Бележки:

- Основен принцип на разработването на софтуер: "information hiding"
- Сравнение с императивното програмиране: данни и алгоритми/операции са **разделени**

# АБСТРАКТНИ ТИПОВЕ ДАННИ



# ОСНОВНИ АСПЕКТИ НА ООП

- Основни характеристики на ООП:
  - Капсулиране
  - Многократна използваемост
    - Поведението на обектите е генетично: т.е. те могат да бъдат използвани в различни ситуации
  - Наследяване
    - Обектите са основа, от която водят началото си други обекти

# РАБОТА С КЛАСОВЕ

- Можем да създаваме променливи от типове, представени чрез класове
  - Обекти или екземпляри от класа
- Можем да обработваме тези променливи
  - Изпращане на съобщения или заявки
- Обектите на класа имат общи характеристики, но собствено състояние

# КЛАСОВЕ И СТАНДАРТНИ ТИПОВЕ ДАННИ

- Това, което правим в ООП е създаване на нови типове данни
- Едно число е също тип – има определени характеристики и поведение
  - Разлика: в ООП дефинираме класове, подходящи за решаване на дадена задача, вместо да ни бъде наложено да използваме съществуващ тип данни, който е бил проектиран да представя единица в машината
- Добавяйки нови типове данни, специфични за решаване на различни задачи, ние разширяваме езика за програмиране

# ДОСТАВЯНЕ НА ПРОСТИ РЕШЕНИЯ

- Използването на обектно-ориентирани техники може да редуцира голям брой задачи до едно просто решение
  - След като веднъж е създаден един клас, можем да създаваме произволен брой обекти
  - Тези обекти могат да се обработват като елементи от решението на дадена задача
  - Едно от предизвикателствата на ООП е създаване на съответствие между пространството на задачата и обектите от пространството на решението

# ЗАЯВКИ КЪМ ОБЕКТИ

- Как можем да накараме един обект да прави полезни за нас неща?
  - Трябва да съществува начин за правене заявки към обектите
- Всеки обект може да удовлетворява само определени заявки
  - Заявките, които можем да правим към един обект са дефинирани от неговия интерфейс
  - Типът е този, който определя интерфейса
- Интерфейсът установява какви заявки можем да правим за определен обект
  - Някъде обаче, трябва да съществува код, който да изпълнява заявката
  - Това, заедно със скритите данни включва имплементацията

# СЪЗДАВАНЕ И ИЗПОЛЗВАНЕ НА КЛАСОВЕ

- В ООП различаваме две основни роли:
- Създатели на класове
  - Целта им е създаване на нови класове, като разкриват само това, което е необходимо на клиентите и пази всичко останало
  - Скритите части не могат да се използват от клиентите-програмисти
  - Т.е., създателите могат да ги променят без да се тревожат за въздействието върху другите
- Клиенти на класове
  - Целта на клиент-програмистите е да създава сбирка от инструменти (класове), които да използва за бързо разработване на

# ПРАВА НА КЛИЕНТИТЕ

- Да декларират променливи от тип class
- Да създават обекти на класа, използвайки конструктури
- Да изпращат съобщения към обектите, използвайки методите на инстанции, дефинирани в класа
- Да познават публичния интерфейс на класа
  - Имената на методите на инстанции, броя и типа на параметрите, типа на резултатите
- Да знаят кои методи на инстанции променят обектите

# ПРАВА НА СЪЗДАТЕЛИТЕ НА КЛАСОВЕ

- Да дефинират публичния интерфейс на класа
- Да скриват от клиентите всички детайли на имплементацията
- Да защитават "вътрешните" данни от достъп на клиентите
- Да променят детайли на имплементацията по всяко време запазвайки публичния интерфейс
  - Ако се налага промена на интерфейса - съгласувано с клиентите

# КОНТРОЛ НА ДОСТЪП

- Взаимоотношенията между създателите и клиентите трябва да бъдат регулирани
- Ако всички елементи на един клас са достъпни за всички, тогава клиентите могат да правят всичко с класа и не съществува начин за налагане на правила
- За целта съществува контрол на достъп
  - Да се държат клиентите далече от частите, които не трябва да се пипат
    - Вътрешната обработка на данните, която не е част от интерфейса
    - В действителност, това е услуга за потребителите – лесно могат да се ориентират кое е важно за тях и кое могат да игнорират
  - Да се позволява на създателите да променят вътрешната структура на класа, без това да влияе на клиентите

# СПЕЦИФИКАТОРИ ЗА ДОСТЪП

- Java използва определени ключови думи за установяване границите на един клас
- Спецификатори за достъп:
  - public
    - Следващите спецификации достъпни за всички
  - private
    - Никой, освен създателят няма достъп до тези дефиниции в рамките на действието на спецификатора
    - Оперира като стена между създателя и клиента
    - При опит за установяване на достъп – грешка по време на компилиране
  - protected
    - Като private, с изключение на това, че наследяващият клас има достъп
  - „приятелски“ достъп (по подразбиране)
    - Ако не се използва някой от горепосочените
    - Достъп в същия пакет

# ОБОБЩЕНИЕ

- Ядро на ООП:
  - Типизиране на абстрактни данни
  - Многократно използване (композиция, наследяване)
  - Полиморфизъм
- В езика за програмиране Java съществуват допълнителни съществени концепции:
  - Създаване и премахваме на обекти
  - Еднобазова йерархия (Object)
  - Идентификация на типове и отражение (Class)
  - Вътрешни класове
  - Събития и изключения
  - Графични компоненти (Swing)
  - Входно-изходна система
  - Многонишковост
  - Персистентност
  - Генетично програмиране
  - Функционално програмиране (Java 8)
  - …

# НАПРАВЛЕНИЯ НА JAVA

- Java се разработва в няколко направления:
  - Java Standard Edition - базовата версия която се използва за разработка на клиентски приложения и т.нар. Java аплети
  - Java Enterprise Edition - базирана е върху Standard Edition, използва се най-често при сървърните приложения в интернет под формата на т.нар. Java сървлети и Java Server Pages (JSP), но освен това включва и голям брой други софтуерни технологии.
  - Java Micro Edition - самостоятелна олекотена версия за работа на устройства с ограничена изчислителна мощност като мобилни телефони, смартфони, PDA устройства и други.

# БЛАГОДАРЯ ЗА ВНИМАНИЕТО!

## КРАЙ “УВОД В ООП (ПОВТОРЕНИЕ)”