Объектно–ориентированное программирование

Лекция №1,

к.т.н., Александр Александрович Власов

Несколько слов о себе

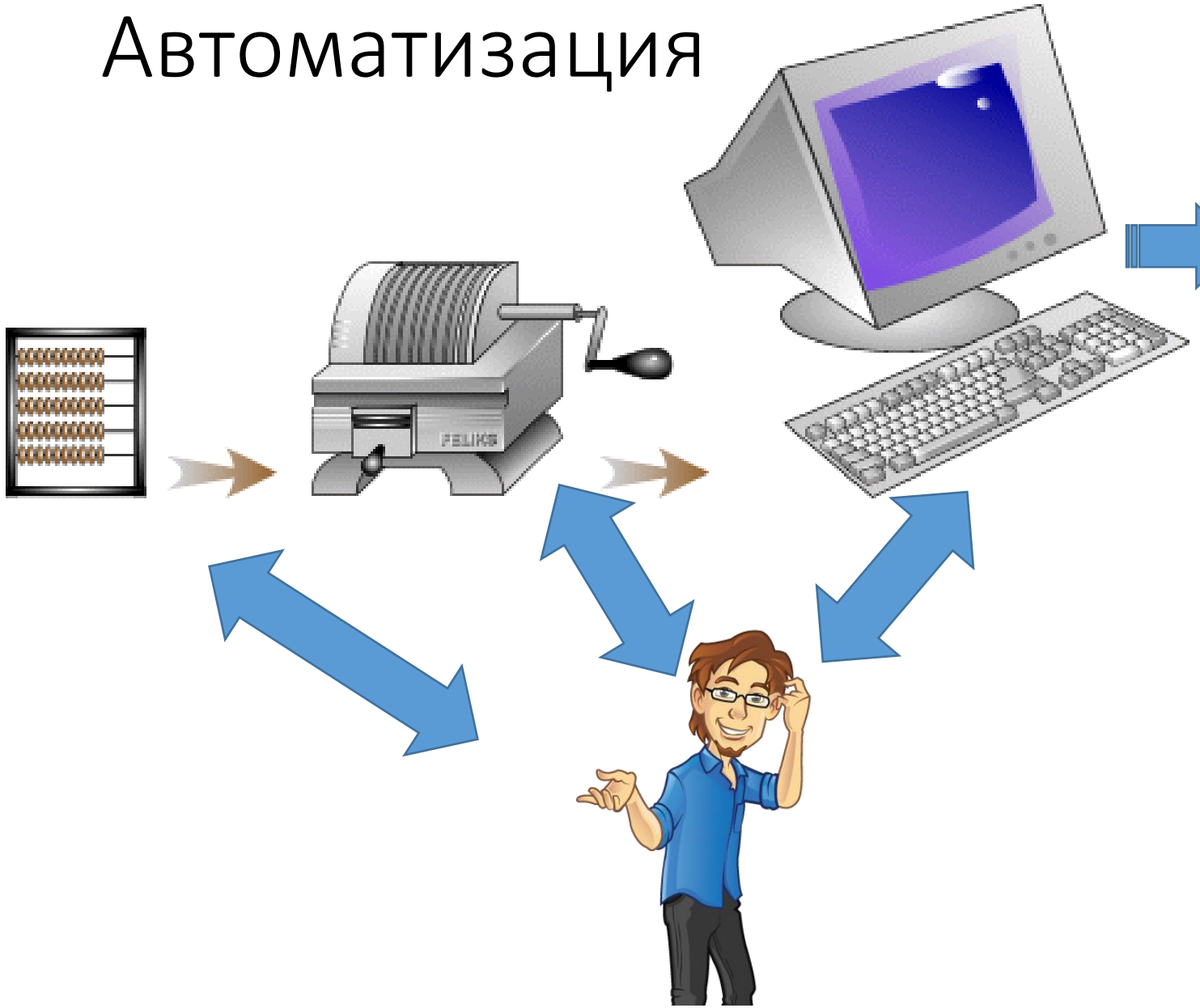
- Выпускник магистратуры ФИТ НГУ 2007 года,
  кафедра систем информатики
- В 2013 г. защитил диссертацию по специальности «25.00.10 Геофизика, геофизические методы поисков полезных ископаемых» на тему «Комплекс программ для обработки и интерпретации данных скважинной геоэлектрики на основе единой информационной модели»
- Разработка программно-аппаратных решений для строительства нефтегазовых скважины
- 34 года
- Женат, двое детей

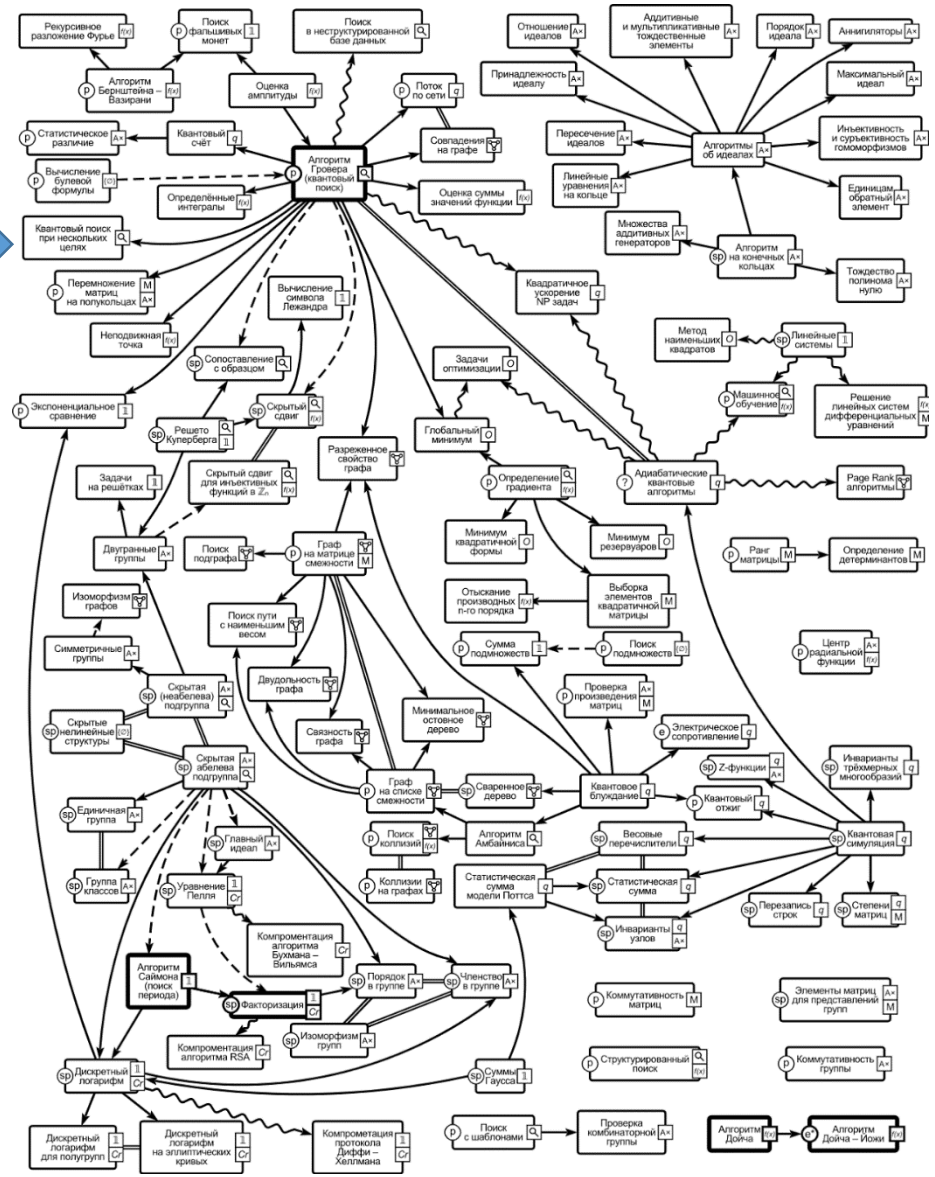
Современные тенденции

- Рост вычислительных ресурсов
- Рост объема информации
- Усложнение программных систем
- Усложнение инструментария разработки
- Широкое понятие «программист»
- Широкий спектр специалистов разных профессий участвуют в разработке ПО
- Изменение требований к ПО
- Изменяющиеся окружение исполнения

человек есть существо,
чьей самой лакомой пищей является информация

Автоматизация





Парадигма программирования

Парадигма (философия науки) – устоявшаяся система научных взглядов, в рамках которой ведутся исследования (Т. Кун)

Парадигма программирования – совокупность принципов, методов и понятий, определяющих способ конструирования программ.

Парадигма программирования

- Императивное (процедурное) программирование
  - Последовательное исполнение инструкций с изменением состояния
- Структурное программирование
  - Нет goto, блоки, [последовательность, ветвление, цикл]
- Функциональное программирование
  - данные=функция, состояния нет, только вычисление функций
- Логическое программирование
  - логический вывод информации на основе заданных фактов и правил вывода
- Объектно-ориентированное программирование
  - взаимодействие объектов

Языки программирования

Years	Number of new programming languages
Pre-1950	10
1950s	50
1960s	54
1970s	63
1980s	66
1990s	68
2000s	50
2010s	30

https://en.wikipedia.org/wiki/Timeline_of_programming_languages

Поколения языков

Поколение	Задачи	Языки
1GL	Пишем программу в кодах исполняющего устройства. Программа должна работать максимально эффективно!	Ассемблер, коды процессора.
2GL	Работу программиста можно облегчить путем использования привычных человеку языковых конструкций. Программа должна работать так же быстро, как ее аналог реализованный в машинных кодах!	Fortran, Algol , LISP, Cobol …
3GL	Алгоритм это математическое понятие. Программы реализующие алгоритмы не должны зависеть от устройства на котором они будут исполняться.	PL/1, Prolog, C, Pascal, Ada, Basic…
4GL	Очень сложные системы. Одного языка мало, нужны целые платформы ориентированные на решение прикладных задач. Алгоритмы должны реализовываться в терминах предметной области. Для управления самолетом я не хочу знать как образуется подъёмная сила, дайте мне две команды: взлет и посадка!	C++, Java, C#, Python, Ruby, ….
5GL	Дайте средство программирования каждому. Почему нельзя программировать речью, картинами, движениями и другими средствами самовыражения доступных человеку. Задай ограничения и машина сама решит оптимально вашу задачу. Научить программировать летчика проще, чем программиста летать!	….

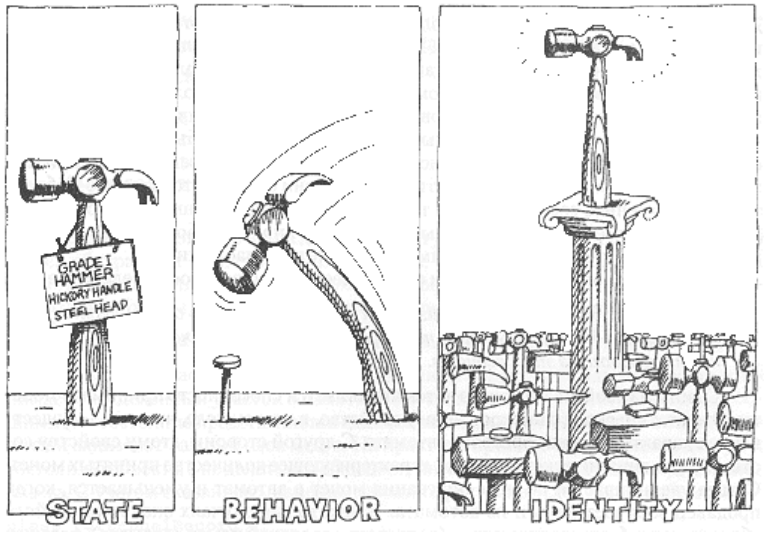
Чистая функция (pure function)

- является детерминированной
- не обладает побочными эффектами

К побочным эффектам относятся:

- изменение переменных вне контекста функции
- изменение параметров
- ввод-вывод

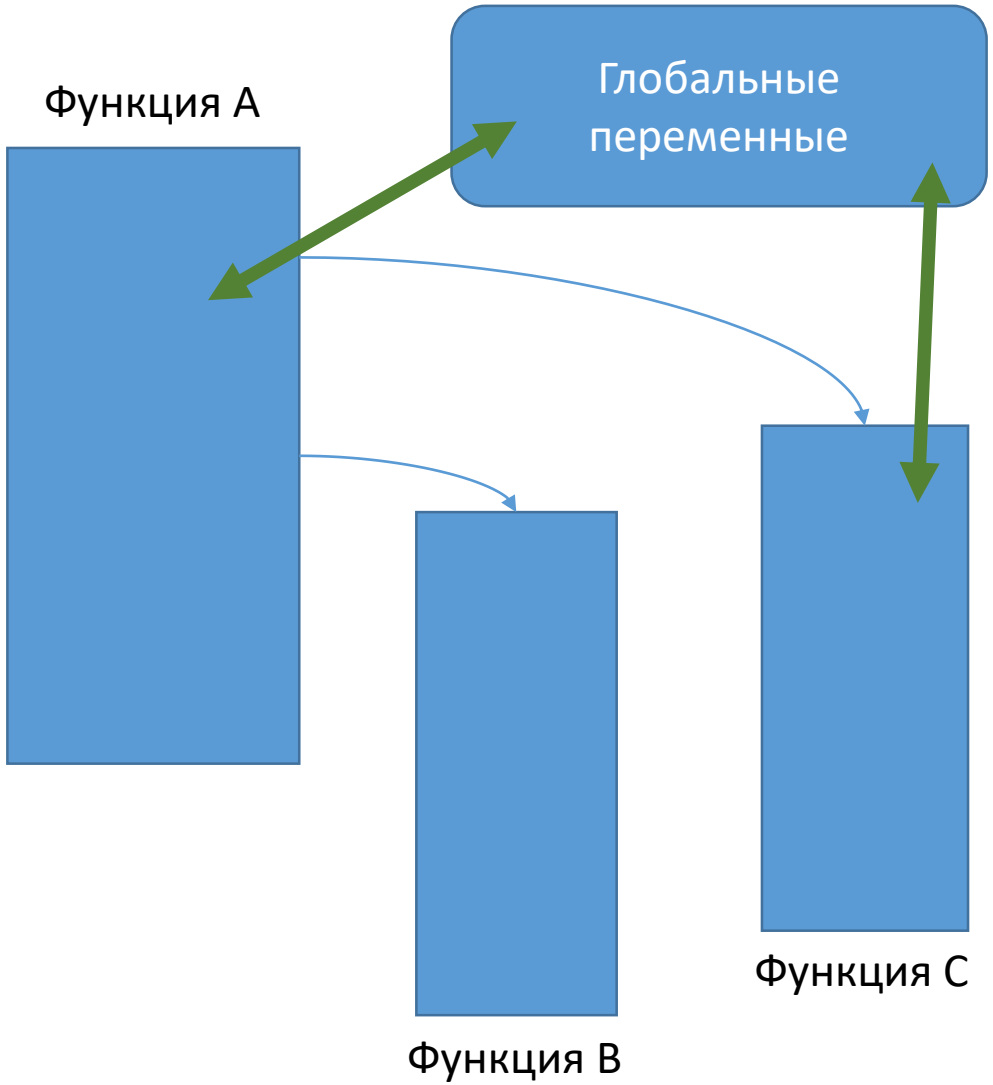
Объект

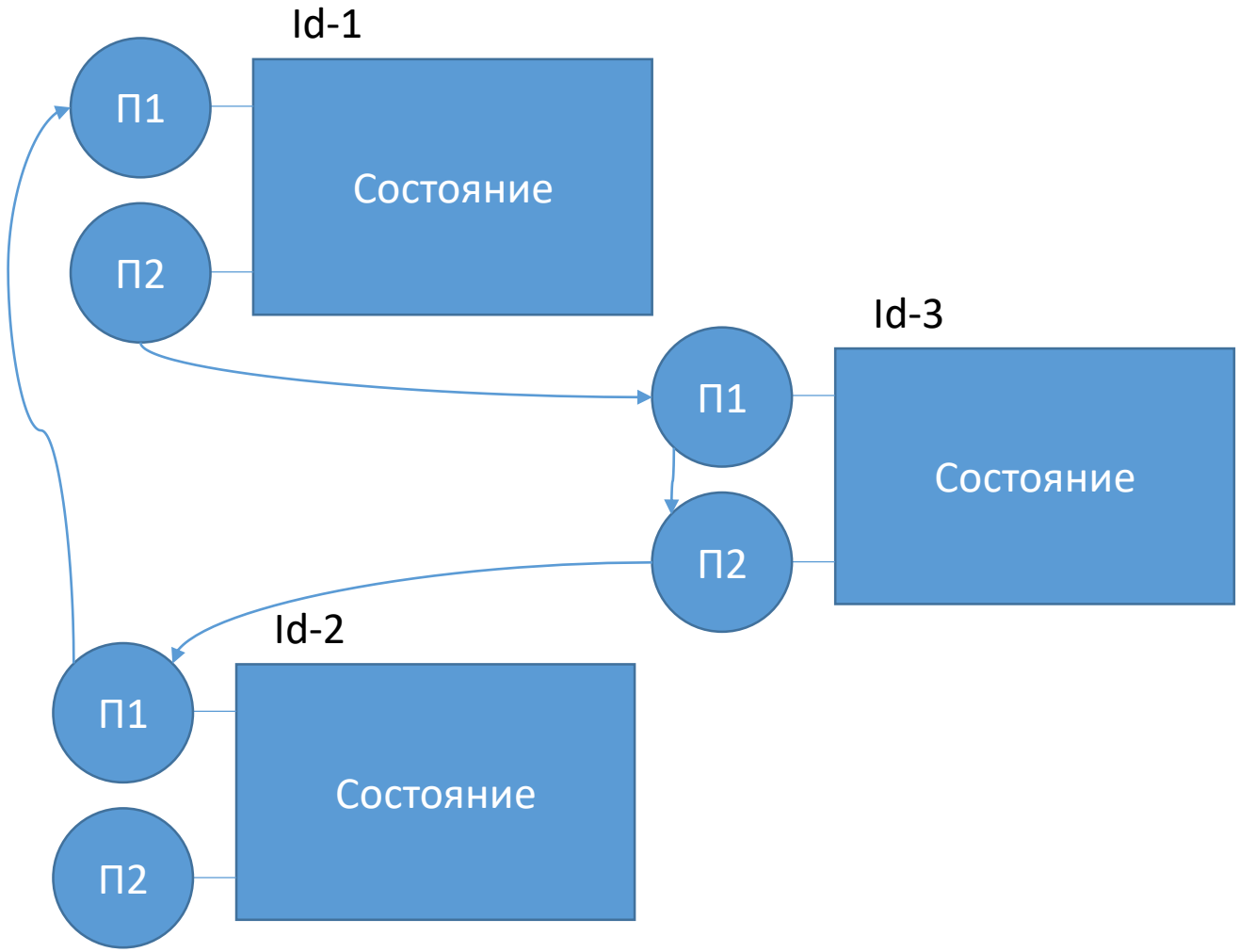


- Состояние – характеризуется перечнем (обычно статическим) всех свойств данного объекта и текущими (обычно динамическими) значениями каждого из этих свойств
- Поведение – реакция на сообщение, которая зависит от текущего состояния, и выражается в изменении состояния объекта, а также посылки сообщений другим объектам
- Уникальностью –позволяет отличать один объект от другого

Объект - конечный автомат?

Схема объектной программы



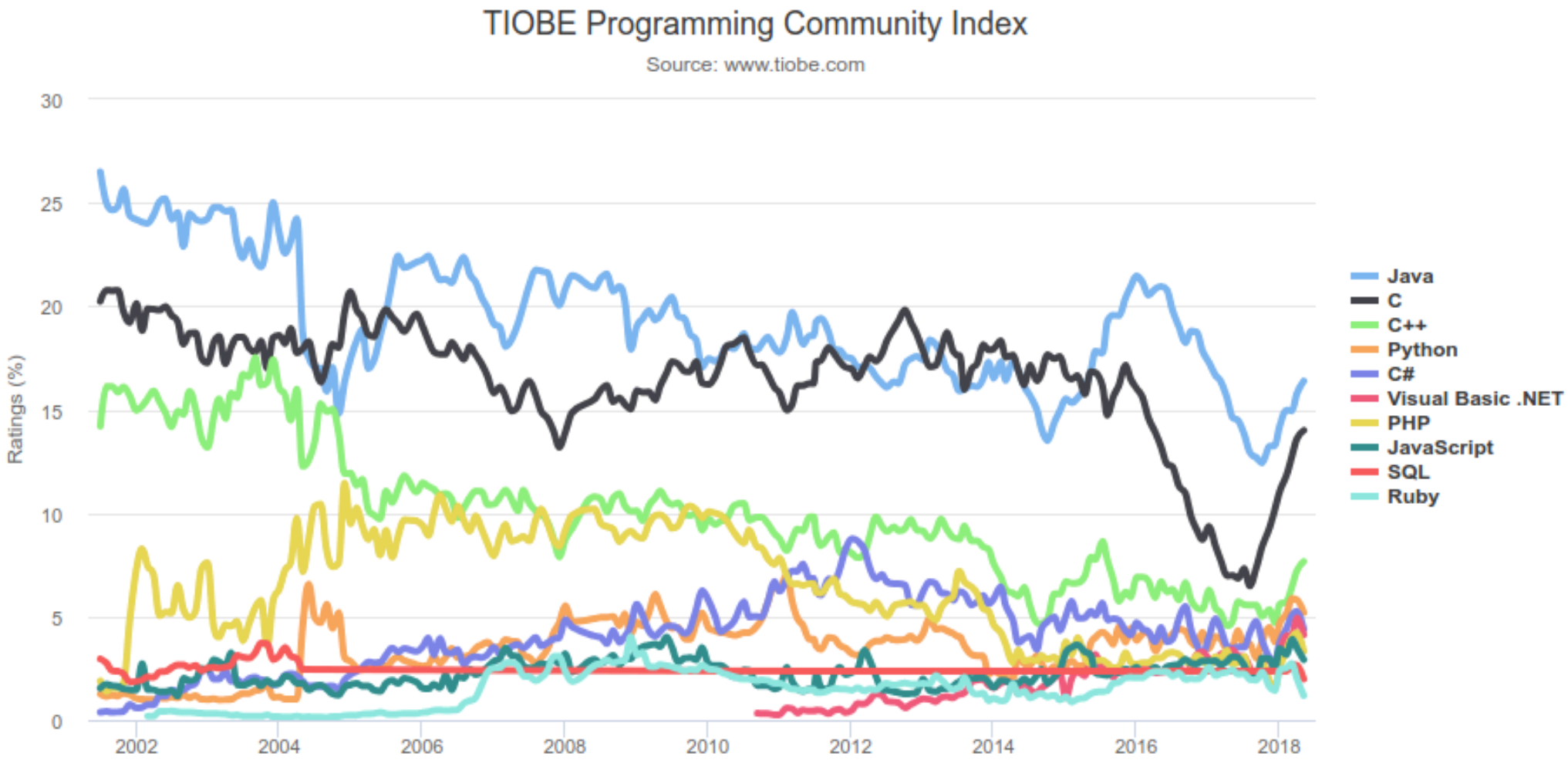


Предпосылки к появлению Java

- Количество разнотипных исполняемых устройств растет лавинообразно
- Для запуска приложения на новой платформе необходимо программный код скомпилировать повторно
- Старт написания любой программы начинается с подключения минимального набора типовых библиотек для решения стандартных задач
- Рост сложности программных систем привел к появлению большого количества ошибок работы с памятью, такие ошибки в программных системах живут и развиваются годами
- Требования к квалификации разработчиков программных систем растет

Simula (Си́мула - "объектный Алгол") - 1967 г., SmallTalk (Смо́лток) - 1980 г., C++ - 1980-е г.г.

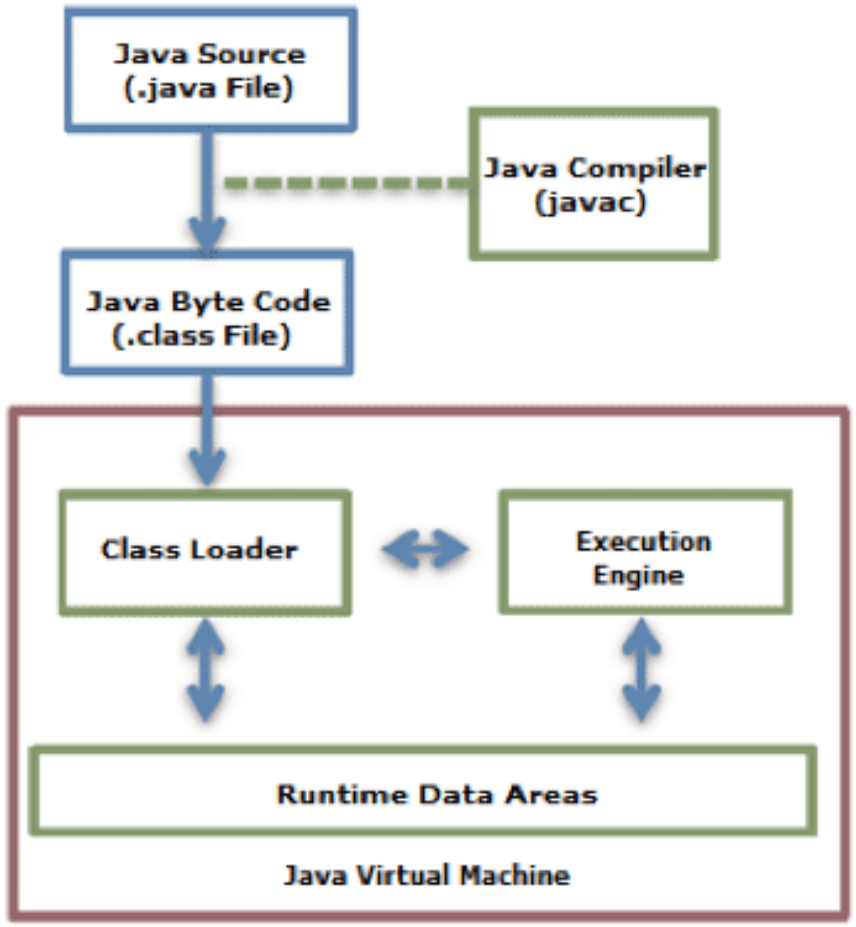
Почему Java? Какой язык лучше?



Язык Java

- Основан на синтаксисе C
- Развитая система типов («абстракция», «инкапсуляция», «типизация»)
- Одиночное наследование классов и множественноенаследование интерфейсов («иерархия»)
- Развитая система пакетов («модульность»)
- Обработка исключений
- Автоматическая сборка мусора
- Обеспечение конкурентного доступа к данным при многопоточности («параллелизм»)
- Доступ к метаинформации (reflection api)
- Отсутствие низкоуровнего управления памятью
- Развитая библиотека (с поддержкой «сохраняемости»)

JVM, JRE, JDK


Java Source
(.java File)
Java Compiler
(javac)
Java Byte Code
(.class File)
Class Loader
Execution
Engine
Runtime Data Areas
Java Virtual Machine


JDK
javac, jar, debugging tools,
javap
JRE
java, javaw, libraries,
rt.jar
JVM
Just In Time
Compiler (JIT)

Ahead-of-Time (AOT) компилятор

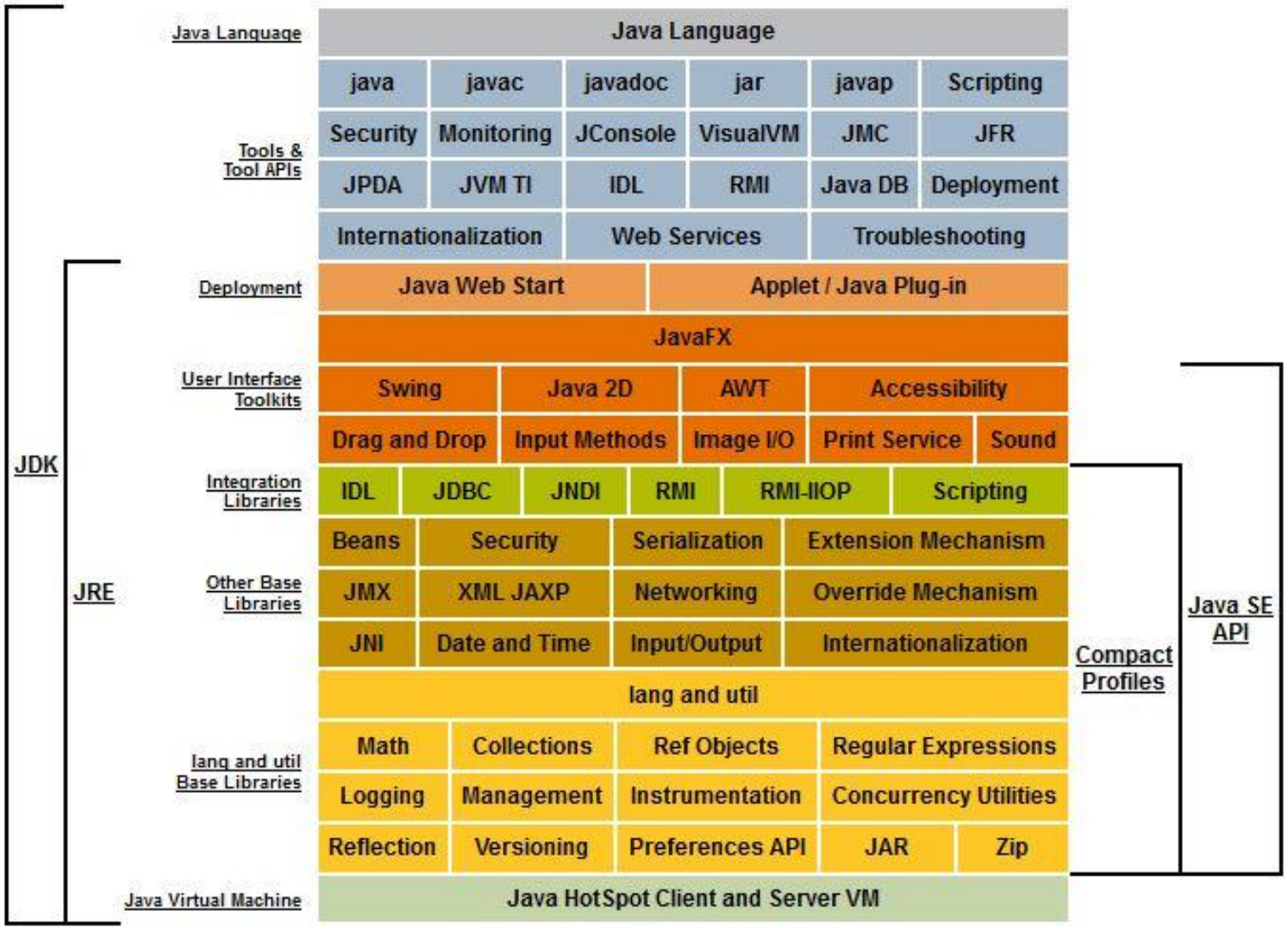
- Процесс трансляции в команды исполняющего устройства полностью выполняется перед выполнением программы
  - Excelsior JET
  - ART для Android

Oak

- Язык программирования Oak (Дуб) (1991)
  - Платформенная независимость, переносимость, виртуальная машина
  - Первоначальный акцент на «бытовую электронику», микроконтроллеры
  - Рождение Internet, сетевое взаимодействие

Java SE (Standard Edition)

- стандартная версия платформы Java 2, предназначенная для создания и исполнения десктоп приложений, рассчитанных на индивидуальное пользование



Java ME

- Подмножество платформы Java для устройств, ограниченных в ресурсах
- Ограничение набора допустимых инструкций
- Ограниченный набор базовых классов платформы
- Профили
  - CLDC (Connected Limited Device Configuration — конфигурация устройства с ограниченными ресурсами и коммуникационными возможностями)
  - CDC (Connected Device Configuration — конфигурация устройства с нормальными ресурсами и коммуникационными возможностями)
- Байт-код для ОС Android (*.dex) несовместим с классическим Java

Java Card

- Java-платформа для устройств с крайне ограниченными вычислительными ресурсами
- подмножество более полных платформ Java: все языковые конструкции Java Card присутствуют в Java и ведут себя так же
- не поддерживаются многие ключевые слова, в том числе: *char*, *double*, *float* и *long.*
- Байткод, выполняемый виртуальной машиной Java Card, функционально является подмножеством байткода Java

Hello World

- javac HelloWorld.java
- java HelloWorld
- Комментарии
- Стартовая функция
- Область видимости

```java
/* This is a simple Java program.
   FileName : "HelloWorld.java". */
class HelloWorld
{
    // Your program begins with a call to main().
    // Prints "Hello, World" to the terminal window.
    public static void main(String args[])
    {
        System.out.println("Hello, World");
    }
}
```

```
Hello, World
```

https://www.geeksforgeeks.org/beginning-java-programming-with-hello-world-example/

Сборка мусора (Garbage Collector)

```
public class TestGarbage{
 public void finalize(){System.out.println("object is garbage collected");}
 public static void main(String args[]){
  TestGarbage obj1=new TestGarbage();
  TestGarbage obj2=new TestGarbage();
  s1=null;
  System.gc();
  System.out.println(“point 1");
  s2=null;
  System.gc();
  }
}
```

```
object is garbage collector
point 1
object is garbage collector
```

Таблица умножения

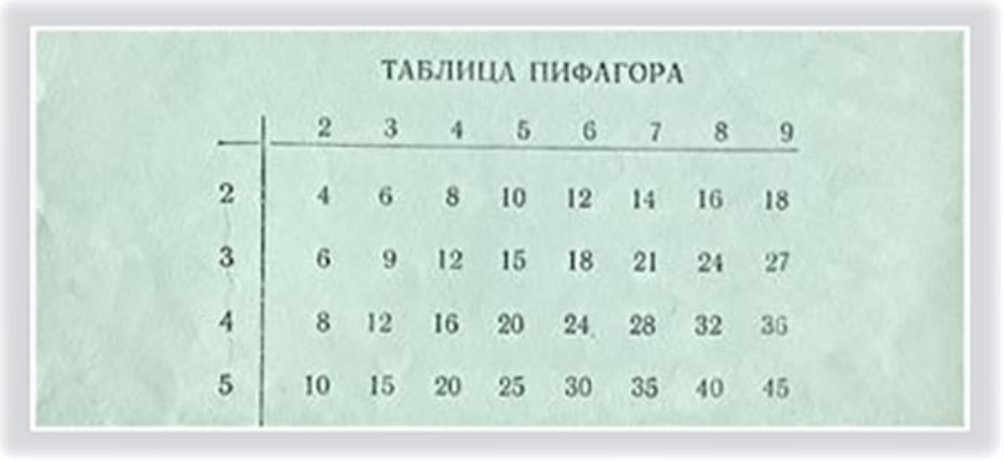


ТАБЛИЦА ПИФАГОРА

	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45

```java
public static void main(String[] args) {
    // Выводим значения второго множителя в строке
    System.out.println("    2  3  4  5  6  7  8  9");
    int i = 2;       // первый множитель, присваиваем стартовое значение
    while(i<10) {    // Первый цикл, выполняем пока первый множитель меньше 10
        System.out.print(i + " | ");// Выводим первый множитель в начало строки
        int j = 2;                  // второй множитель, стартовое значение
        while (j<10) { // Второй цикл, выполняем пока второй множитель меньше 10
            int mul=i*j; // Считаем произведение множителей
            if (mul<10)  // Если содержит одну цифру-после него выводим два пробела
                System.out.print(mul + "  ");
            else   // иначе выводим произведение и после него - один пробел
                System.out.print(mul + " ");
            j++;     // Увеличиваем на единицу второй множитель,
        }            // Переходим к началу второго цикла (while (j<10 )....  )
        System.out.println(); // Перейдем на следующую строку вывода
        i++;                  // Увеличиваем на единицу первый множитель,
    }                         // Перейдем к началу первого цикла (while ( i<10 ) ....
}
```

```
    2  3  4  5  6  7  8  9
2 | 4  6  8  10 12 14 16 18
3 | 6  9  12 15 18 21 24 27
4 | 8  12 16 20 24 28 32 36
5 | 10 15 20 25 30 35 40 45
6 | 12 18 24 30 36 42 48 54
7 | 14 21 28 35 42 49 56 63
8 | 16 24 32 40 48 56 64 72
9 | 18 27 36 45 54 63 72 81
```

Выполнение семинарских заданий

- В каком программном средстве мне редактировать исходные коды программы?
  - IntelliJ IDEA, Java SE
- Как мне передать выполненные задания для проверки преподавателю?
  - Пользуемся системой контроля версий Git, на общедоступном ресурсе github.com
- Как запустить для отладки мою реализацию алгоритма пирамидальной сортировки?
  - Для отладки и проверки работы семестровых заданий пишем тесты. Набор тестов является неотъемлемой частью выполненного задания.