nexign

Java Fundamentals

Write once, run anywhere!

Александр Митясов,

специалист по технологическому обучению

Nexign Team

Александр Митясов

Старший специалист по технологическому обучению

Технологии JVM: Профиль Java/Kotlin
Курсы Spring framework, groovy, логирование и мониторинг высоконагруженных приложений

nexign

Давайте познакомимся

nexign

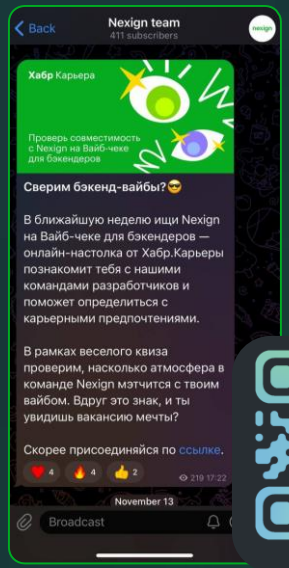
Разработчик софта B2B

2000 сотрудников	11 филиалов
25+ технологических продуктов	150+ реализованных проектов

nexign

Где можно ставить лайки?

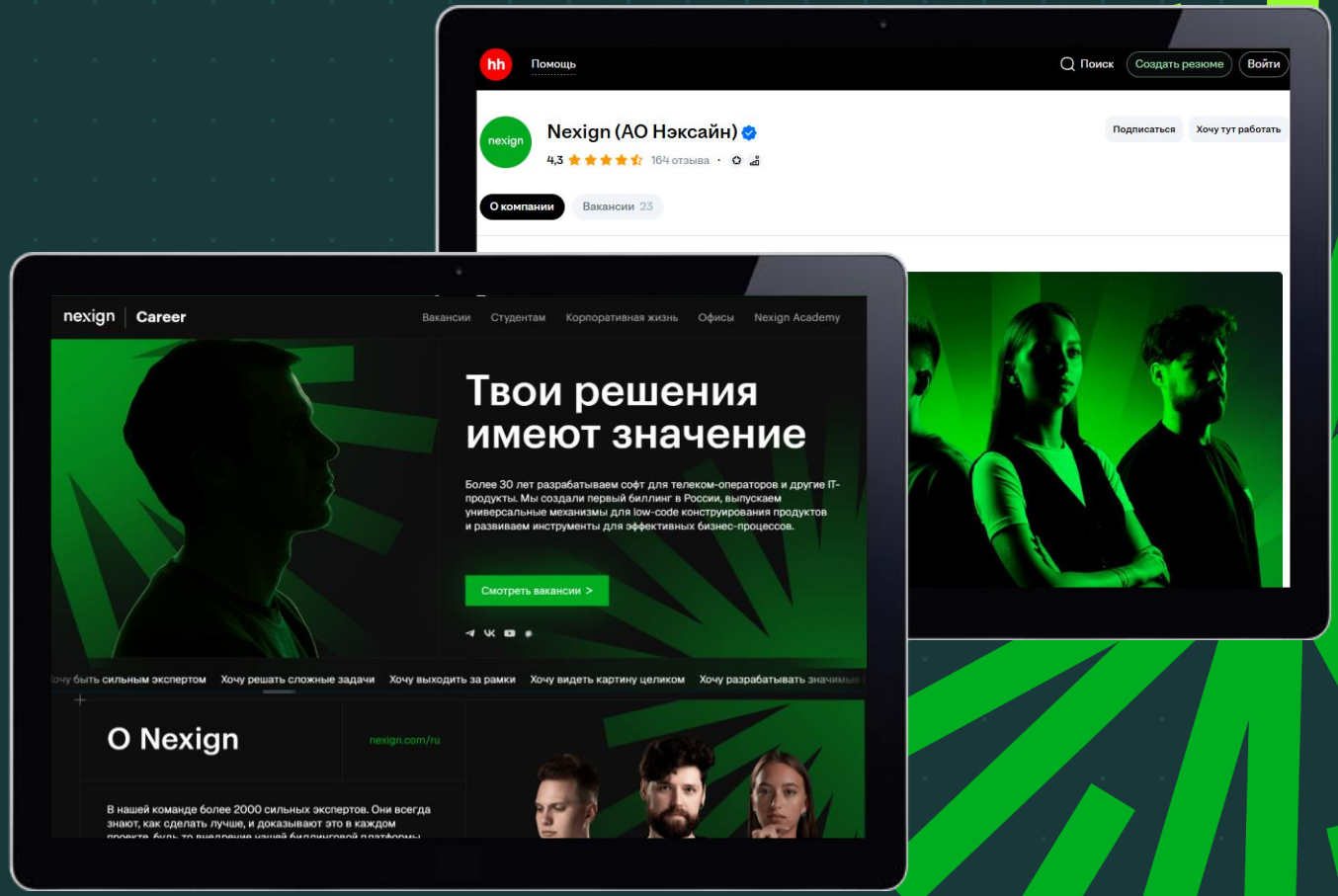
Телеграм-канал



Подкаст «Кто за кодом?»

Где искать вакансии

или карьерные возможности



План курса

Введение. Базовый синтаксис

01

- особенности языка Java
- структура программы
- типы данных
- управление выполнением программы

ООП

02

- классы и объекты
- поля и методы
- модификаторы доступа
- конструкторы
- принципы ООП

Обработка ошибок. Исключения

03

- классы исключений
- порядок обработки исключений
- создание собственных исключений

Многопоточность. Generics. Sreams

04

- Stream API
- Generics
- ввод/вывод (доступ к файловой системе)
- сетевое взаимодействие и многопоточность

nexign

Основы технологии Java

Понятия и термины

nexign

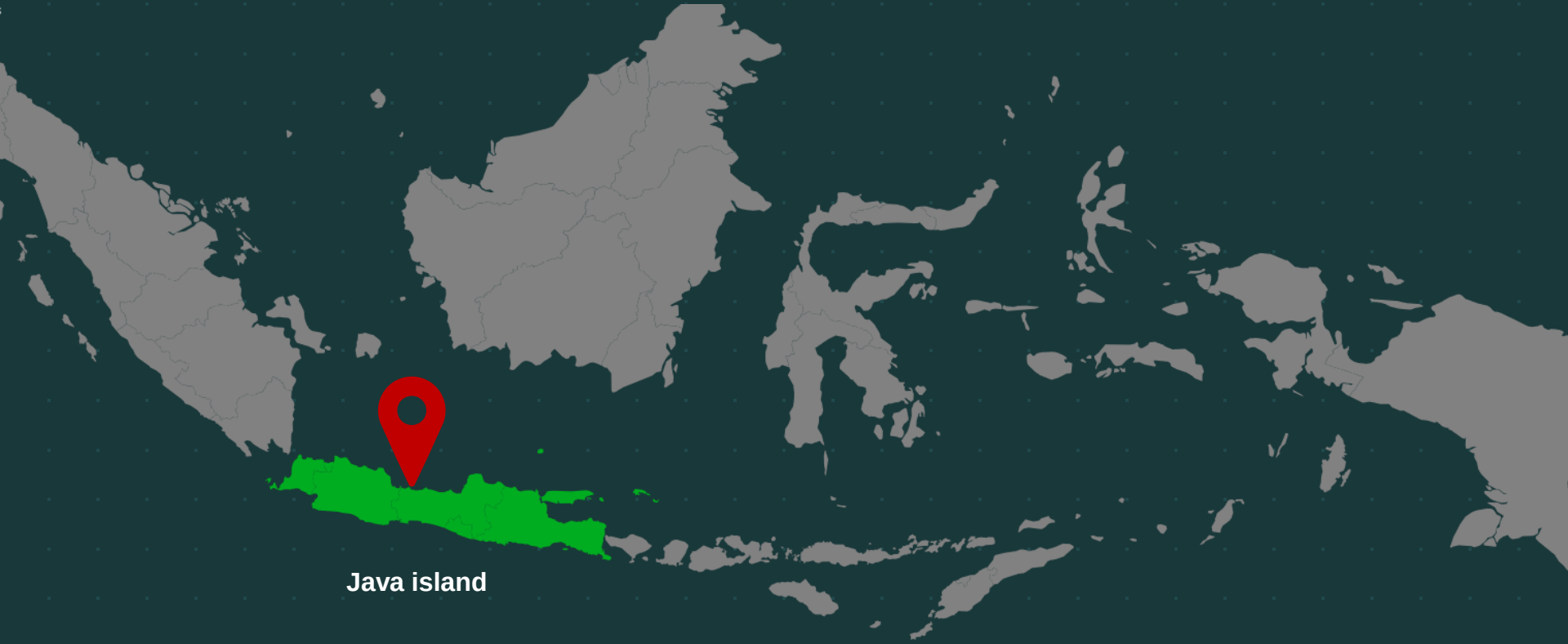
План урока

- Java – что это? История появления
- Особенности языка
- Инструменты для работы
- Настройка и установка необходимого ПО

nexign

Java. История и основные понятия

JAVA – что это?



nexign

Java

мультипарадигменный (преимущественно объектно - ориентированный) статически типизированный кроссплатформенный язык программирования общего назначения.

nexign

История появления

1991
внутренний проект Sun Microsystems по созданию платформы для разработки встраиваемых систем — Green Project;

1992
первое демонстрационное устройство на новой платформе — PDA Star7

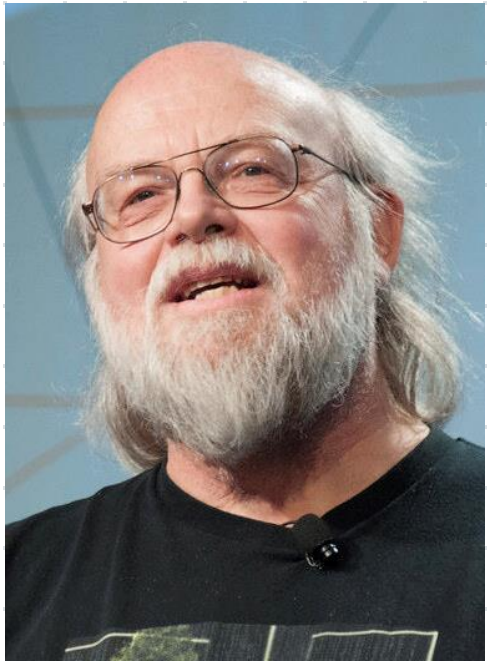
1994
фокус на разработке интерактивных приложений (апплетов) для веб-страниц;
Язык переименован в Java

1996
Java Development Kit 1.0

Маскот языка Duke

nexign

Джеймс Гослинг



- Переносимость
- Синтаксис близкий к C++
- Язык для бекенда и встаиваемых систем

Q&A with James Gosling, June 2020:

https://www.youtube.com/watch?v=XThhlihqTXI

nexign

История версий

2010 г. – компания Sun Microsystem поглощена компанией Oracle

LTS – Long Term Support

nexign

Редакции Java

- Standard Edition (SE)
- Enterprise Edition (EE)

  SE + дополнительные библиотеки и возможности
- Micro Edition (ME)

  подмножество SE + специфические библиотеки
- Java Card

  урезанная версия SE, изменения в виртуально машине

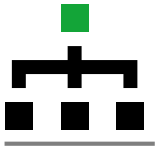
nexign

Особенности языка Java

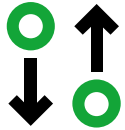
Особенности языка

Переносимый

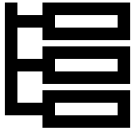
Безопасный



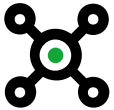
Объектно-ориентированный



С автоматическим управлением памятью



Многопоточный

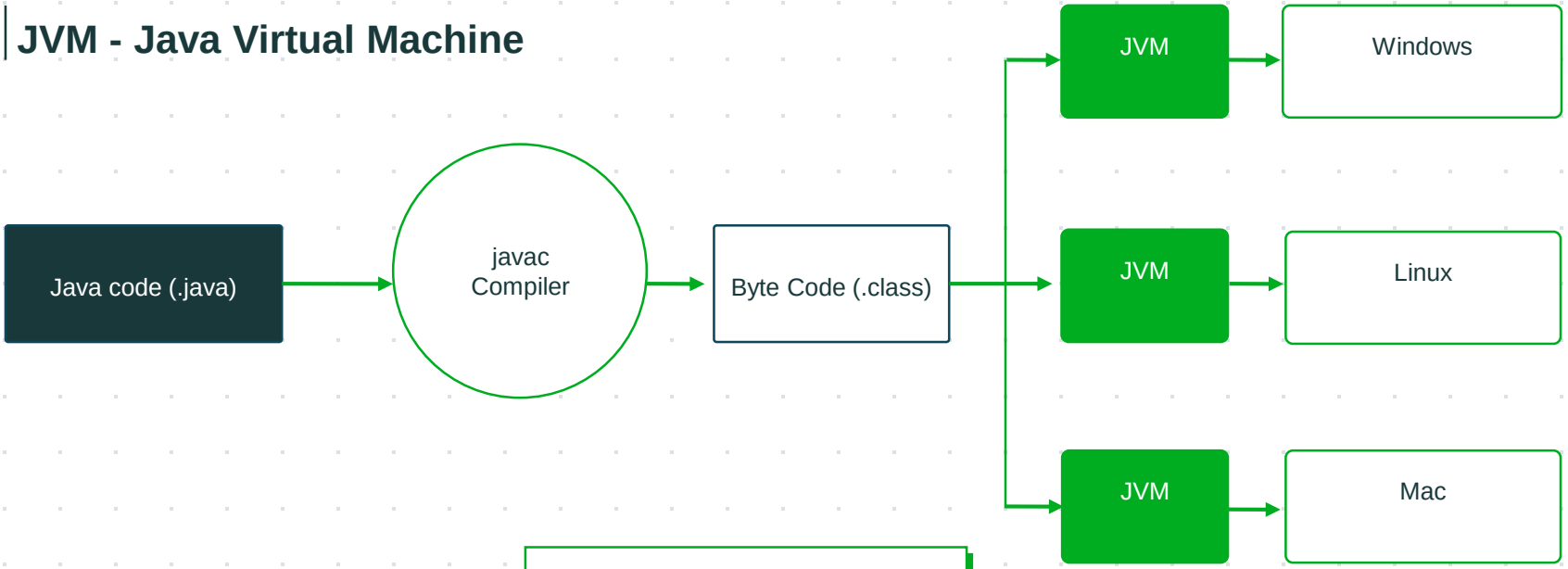


Распределённый

nexign

Переносимость

JVM - Java Virtual Machine



Write Once, Run Anywhere

nexign

JIT

JIT – Just In Time compiler

Участки кода, которые выполняются **часто**

компилятор может оптимизировать, «на лету»

преобразовывая фрагмент сразу в машинный код.

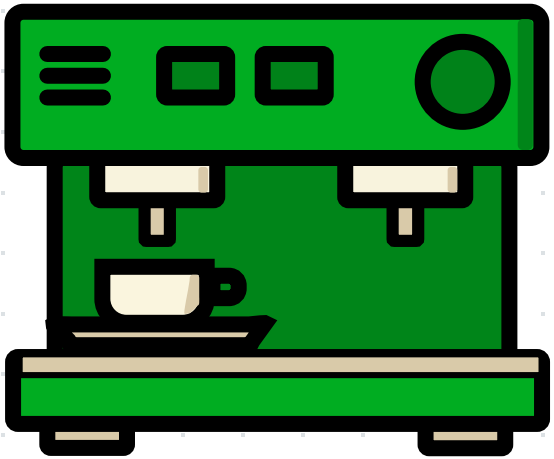
nexign

JVM

- **HotSpot (Reference Implementation) - Oracle Java**
- Azul Zing
- IBM I9 (Open J9)
- Amazon Coretto
- SAP JVM
- Liberica JDK
- …

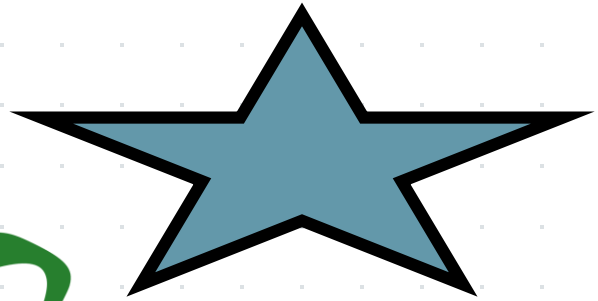
20+

Реализаций



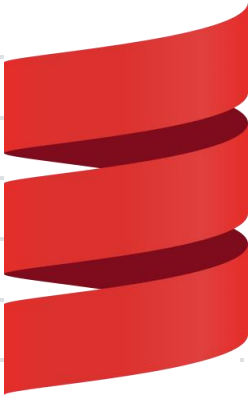
nexign

JVM языки



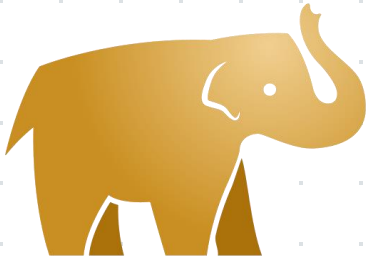
Groovy

Kotlin



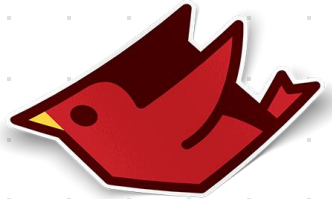
Scala

Jython



Ceylon

Clojure

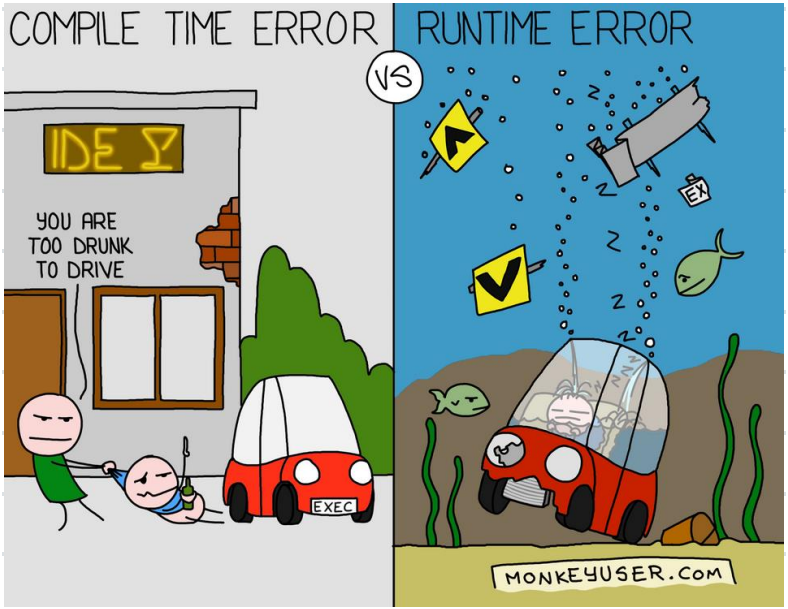


JRuby

nexign

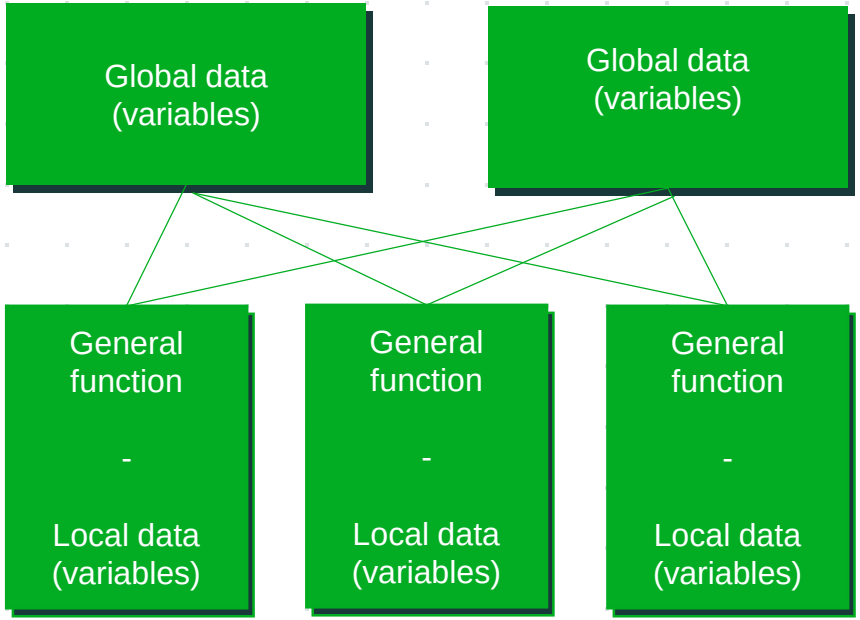
Безопасность

- Верификация байткода
- Автоматическое управление памятью
- Встроенный механизм управления правами

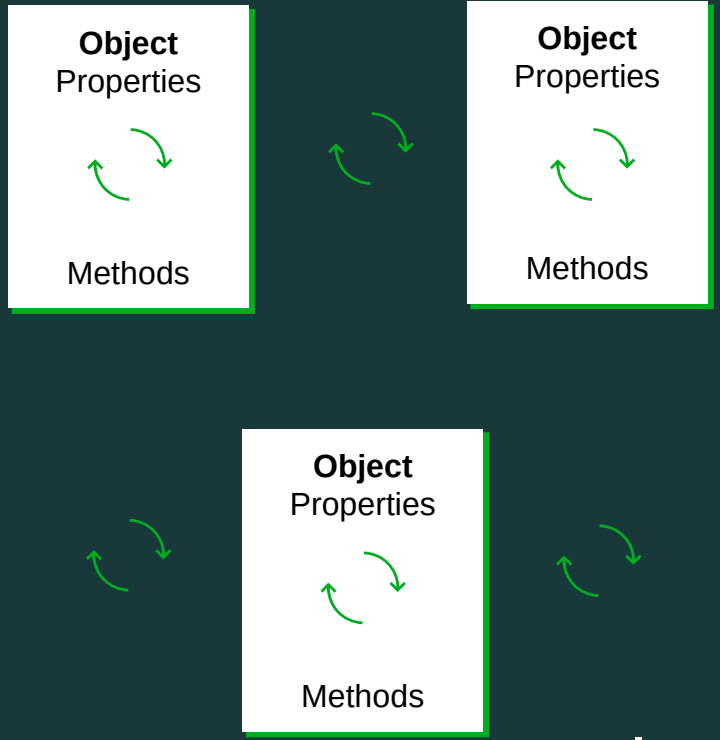


nexign

Процедурное



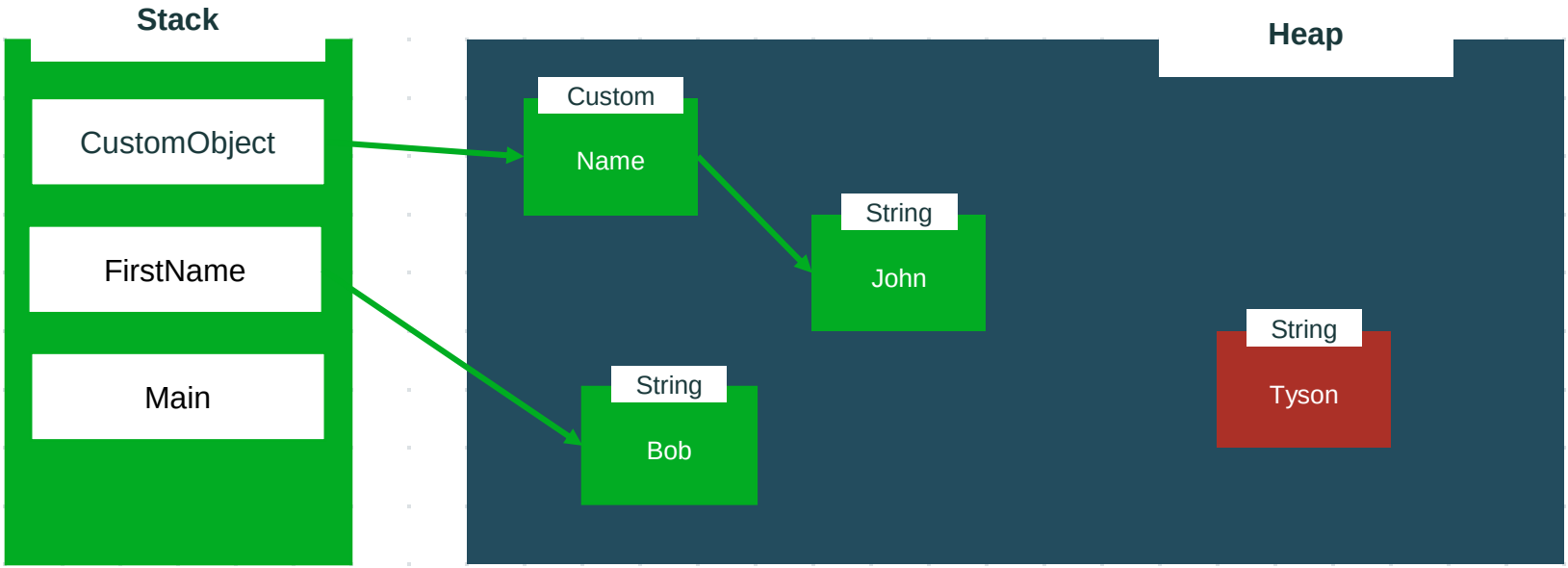
ООП



nexign

Управление памятью

GC – Garbage Collector



nexign

Garbage Collector

Достоинства:

- Меньше кода
- Не бывает некорректных ссылок (на разрушенный объект)
- Меньше утечек памяти
- Разработчику не нужно заботиться о высвобождении памяти

Недостатки:

- Не контролируем время работы GC
- STW-паузы (бывает, в неподходящий момент)

Многопоточность и распределенность

Многопоточность

- встроенная поддержка потоков
- богатая библиотека примитивов синхронизации

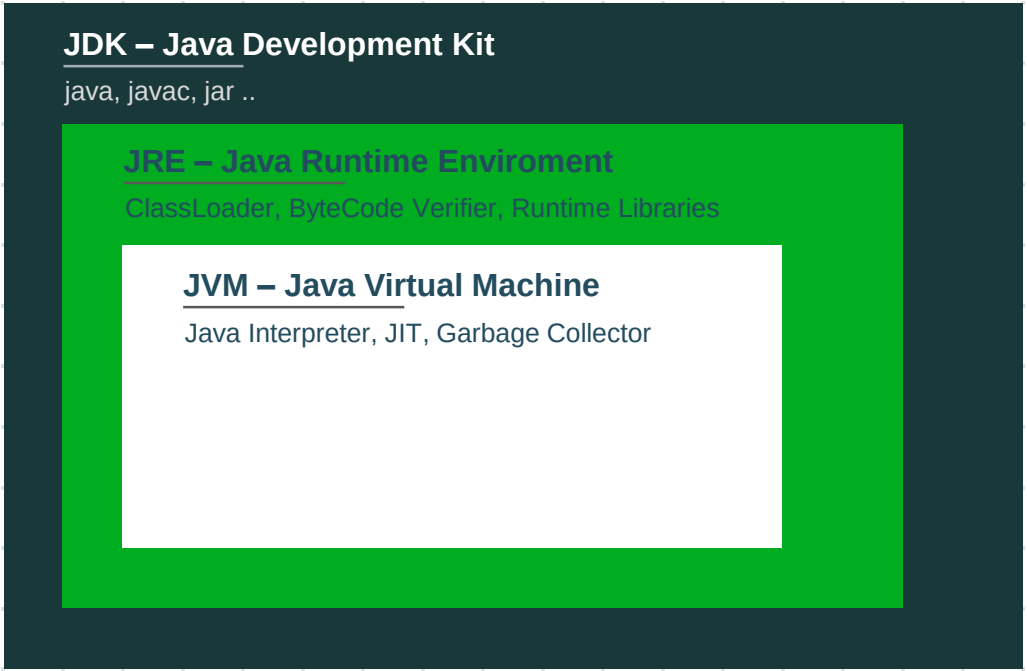
Распределенность

- встроенные сетевые возможности
- пересылка данных и объектов по сети
- работа с удаленными объектами (RMI)

nexign

Основные инструменты для работы

Инструменты



JDK

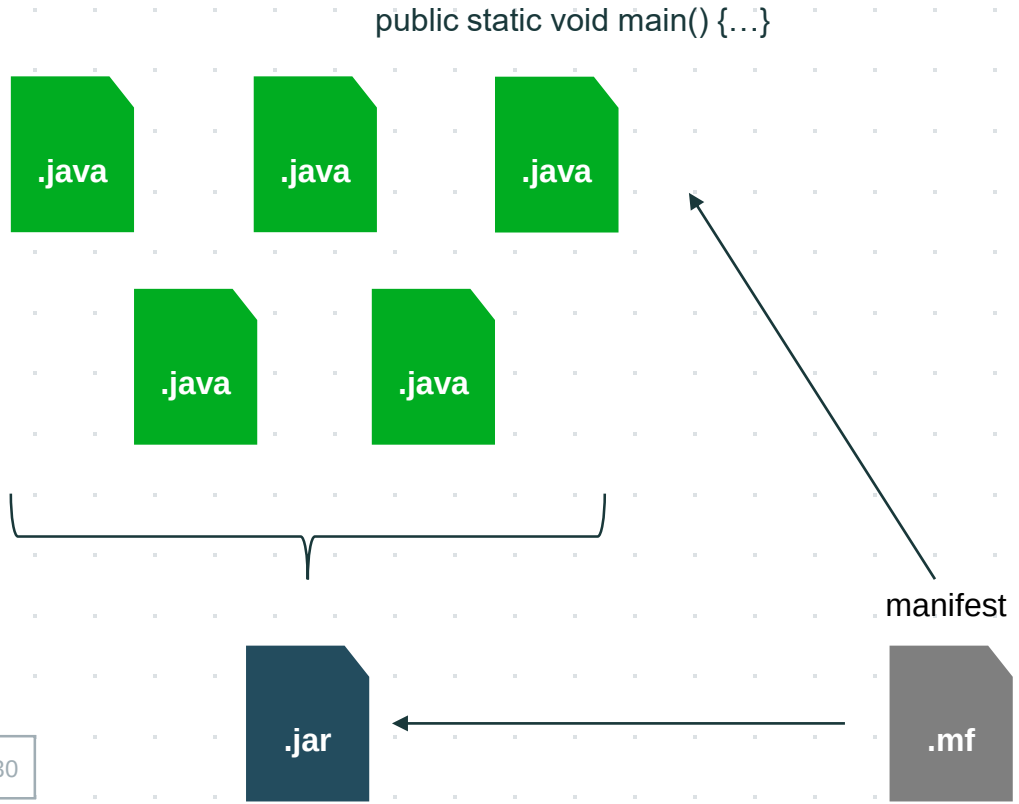
- **Oracle JDK**
- **OpenJDK Build от Oracle**
- AdoptOpenJDK
- Red Hat OpenJDK
- Azul Zulu
- Amazon Corretto

nexign

Утилиты JDK

- **javac** – компилирует исходный код (*.java) в байткод (*.class)
- **java** – Исполняет байткод
- **jdb** – отладчик
- **javadoc** – генератор документации
- **jar** – создание и управление архивами jar
- **javah** – генератор h-файлов C/C++ интерфейса JNI
- **javap** – дизассемблер классов
- **extcheck** – обнаружение конфликтов между файлами архивов jar

JAR



Программа состоит из множества файлов - классов.

Чтобы не путаться в том, какой именно файл запускать, классы объединяют в архивы с расширением .jar, с добавлением специального файла manifest, в котором есть указатель на файл, содержащий “главный” метод **main()**

nexign

Важнейшие элементы экосистемы

- Бибилиотеки и фреймворки (в т. ч. для тестирования)
- Среды разработки
- Системы сборки
- CI

Jenkins

nexign

Q&A

Ещё по этой теме

Книги:

- Cay S. Horstmann. Core Java vol 1 & 2
- Raoul-Gabriel Urma et al. Modern Java in Action
- Joshua Bloch Effective Java
- Brian Goetz et al. Java Concurrency in Practice
- Head First Java, 3nd Edition by Kathy Sierra, Bert Bates

nexign

Структура программы
Типы данных

nexign

План урока

- Структура программы
- Примитивные типы данных
- Ссылочные типы

nexign

Структура программы

Синтаксис

You **vs** **The guy she tells you not to worry about**

```java
public class Main {
    public static String reverseString(String str) {
        StringBuilder reverse = new StringBuilder();
        for (int idx = hello.length() - 1; idx >= 0; idx--) {
            reverse.append(hello.charAt(idx));
        }
        return reverse.toString();
    }

    public static void main(String[] args) {
        String hello = "Hello world!";
        System.out.println(reverseString(hello));
    }
}
```

```python
hello = 'Hello world!'
print(hello[::-1])
```

https://www.reddit.com/r/ProgrammerHumor/comments/66jj7f/java_vs_python/

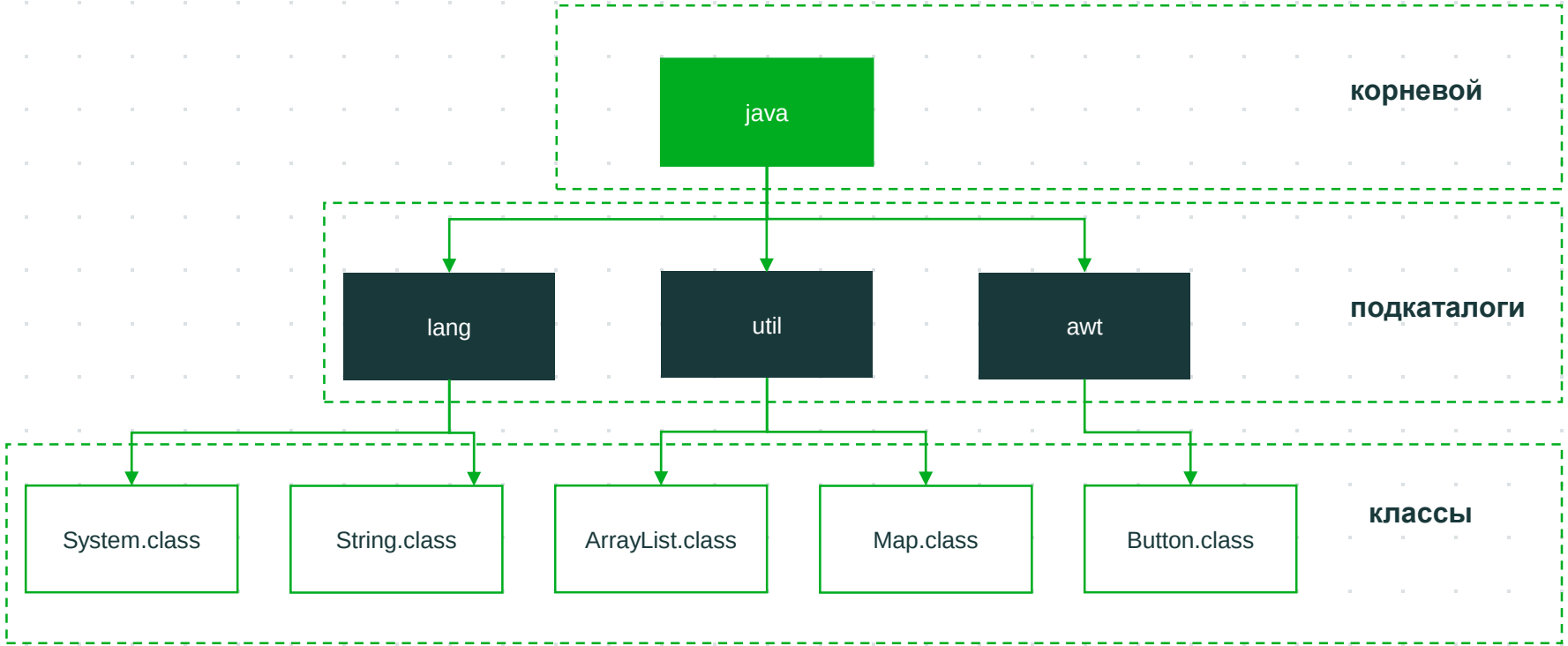
Структура программы

- Package statement
- Import statements
- Class declaration{
  - Variables
  - Constructors
  - Methods
  - Nested classes
  - Nested interfaces
  - Enum

}

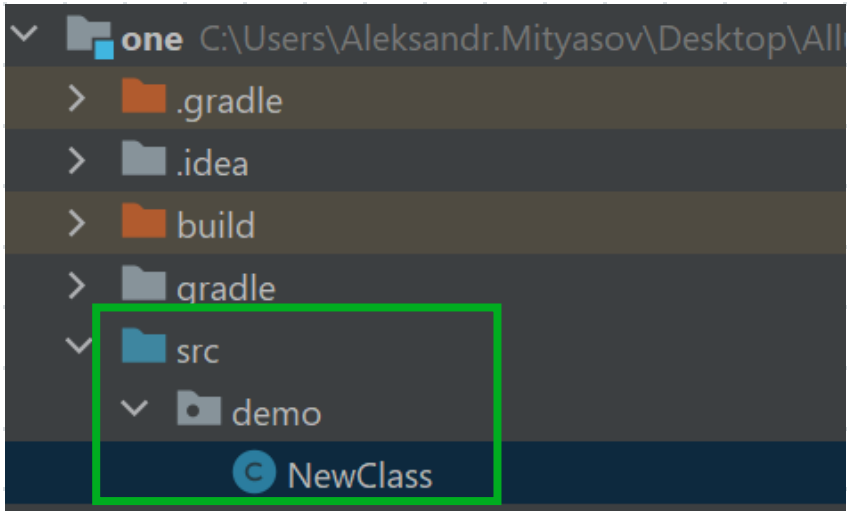
```java
package test;
import java.util.Scanner;
public class Test {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Введите число:");
        int number = sc.nextInt();
        System.out.println("Вы ввели " + number);
    }
}
```

Packages



nexign

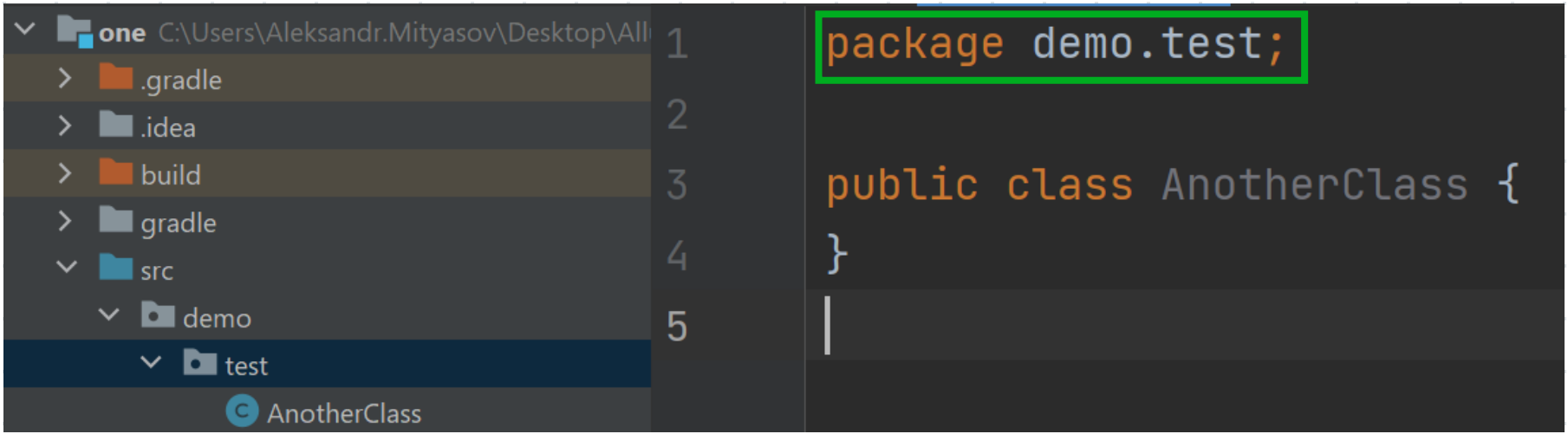
Packages




```
1  package demo;
2
3  public class NewClass {
4  }
5
```


nexign

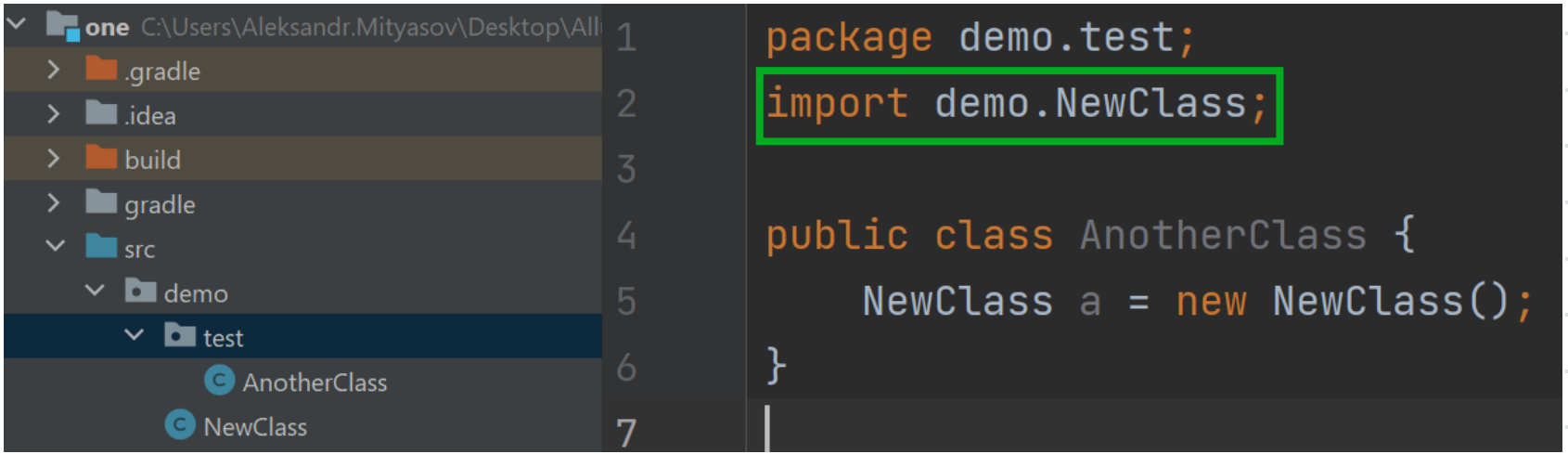
Packages



nexign

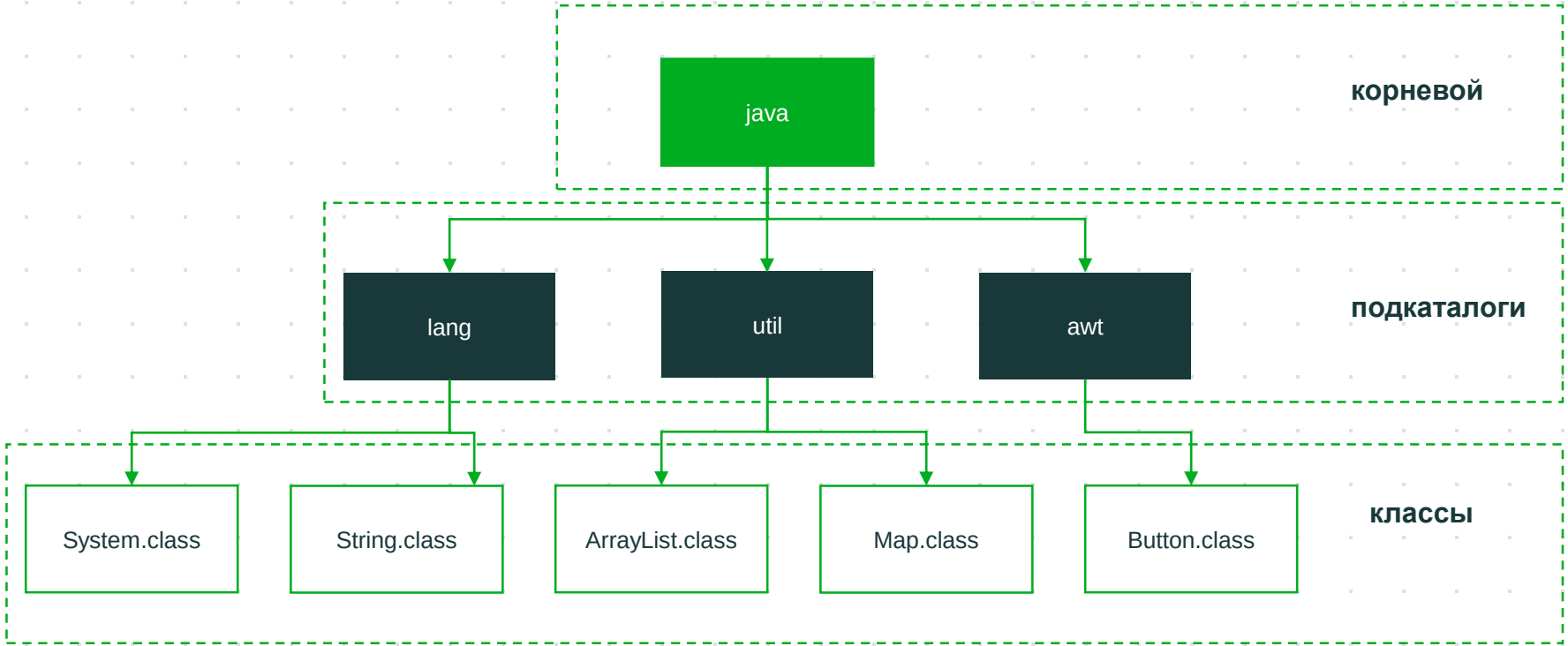
Import

Оператор **import** сообщает компилятору Java, где найти классы, на которые ссылается код.



nexign

Packages



nexign

Import

```
import java.util.*; //импорт всех классов из пакета
```

```
package demo.test;
import java.lang.Math;

public class AnotherClass {
   double a = Math.sqrt(4);
}
```

```
package demo.test;
import static java.lang.Math.sqrt;

public class AnotherClass {
   double a = sqrt(4);
}
```

Import

Оператор **import** сообщает компилятору Java, где найти классы, на которые ссылается код.

```
import java.util.*; // импорт всех классов из пакета java.util
import java.util.Scanner;
import static java.lang.Math.sqrt;
```

Class

Программа, которая печатает в консоль фразу «Hello World»

```
package test;
public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Class

```java
package test;
public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Имя класса = имя файла

Регистр важен

nexign

Class

```
package test;
public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Модификатор уровня доступа

***public** – нет ограничений*

на доступ к классу

nexign

Class

```
package test;
public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

static *– позволяет вызывать метод* ***main()*** *без получения экземпляра класса*

nexign

Class

```
package test;
public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

main() *– имя метода класса (член класса), ключевое слово* ***void*** *указывает на то, что метод ничего не возвращает*

nexign

Class

```
package test;
public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

args *– имя массива экземпляров класса* ***String*** *и принимает любые аргументы командной строки, присутствующие во время выполнения программы*

Тип_данных имяПеременной

nexign

Class

```
package test;
public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

В IntelliJ Idea,можно быстро создать метод при помощи сниппета ***«psvm»***

nexign

Class

```
package test;
public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

println() *отображает переданную ему символьную строку с переводом на новую строку,* ***System*** *обозначает предопределенный класс, предоставляющий доступ к системе, а* ***out*** *- поток вывода, связанный с консолью*

nexign

Class

```
package test;
public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

*Сниппет **«sout»***

nexign

nexign

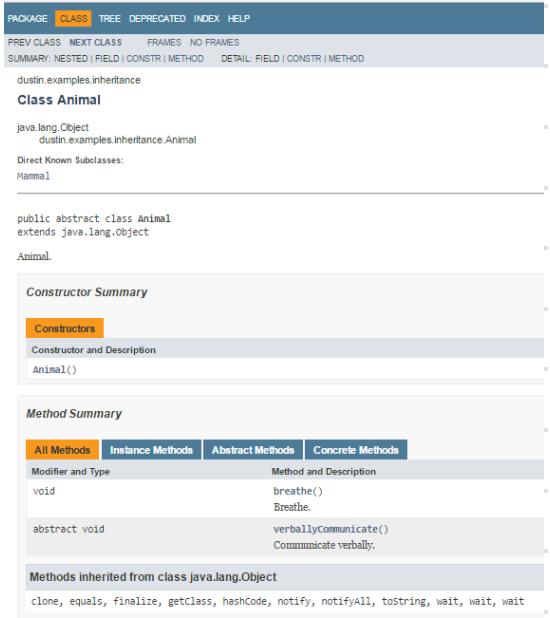
Рекомендации по стилю кода

Комментарии

- Однострочный // …
- Многострочный /* … */
- Многострочный комментарий документации (javadoc) /** … */

Для вывода документации на русском,
использовать команды в качестве аргументов при сборке:

-encoding UTF-8 -docencoding UTF-8 -charset UTF-8


PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

dustin.examples.inheritance

Class Animal

java.lang.Object
dustin.examples.inheritance.Animal

Direct Known Subclasses:
Mammal

public abstract class Animal
extends java.lang.Object

Animal.

Constructor Summary

Constructors

Constructor and Description
Animal()

Method Summary

All Methods | Instance Methods | Abstract Methods | Concrete Methods

Modifier and Type	Method and Description
void	breathe() Breathe.
abstract void	verballyCommunicate() Communicate verbally.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, tostring, wait, wait, wait

JavaDoc File

nexign

Именование объектов

- Классы
- Переменные
- Константы
- Методы

С большой буквы

Стиль CamelCase

Пример: MyFirstClass

С маленькой буквы

Использовать понятные по смыслу значения

Пример: userId, todayTemperature

Заглавные бкувы. Разделители «_»

Пример: MAX_VALUE

С маленькой буквы

Использовать глагольные выражения

Пример: getCount(), setTemperature()

Рекомендации по стилю кода

- В промышленной разработке важно, когда все файлы имеют согласованный стиль!
- Code Conventions for the Java ™ Programming Language (https://www.oracle.com/technetwork/java/codeconvtoc-136057.html)
- **Правила:**
    1. Комментарии/Javadoc: пишите их; используйте стандартный стиль.
    2. Короткие методы: не пишите объёмных (по коду) методов.
    3. Поля: должны быть вверху файла, или прямо перед методом, который их использует.

    ….

nexign

Типы данных.
Определение переменных

Определение переменных

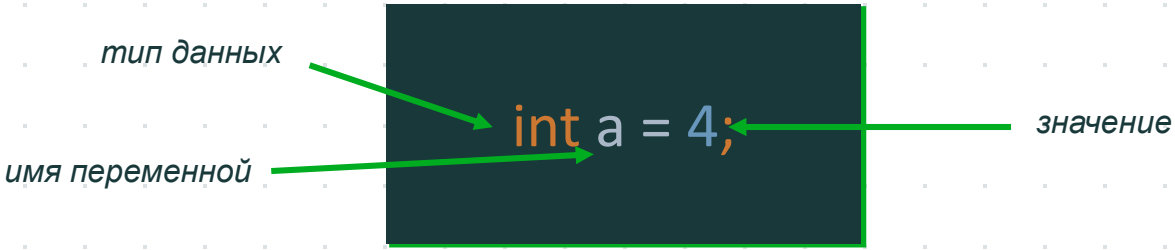
Декларация

```
int a;
```

Инициализация

```
a = 4;
```

Декларация + инициализация



nexign

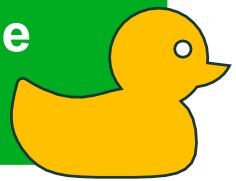
Статическая типизация

```
int a = 4;
a = "строка"; //ошибка нессответствия типов
//или
String a = "строка"; // тип изменить нельзя
```

```
int a;
System.out.println(a);
//ошибка - переменная не инициализированна
```

Типы данных

Примитивные



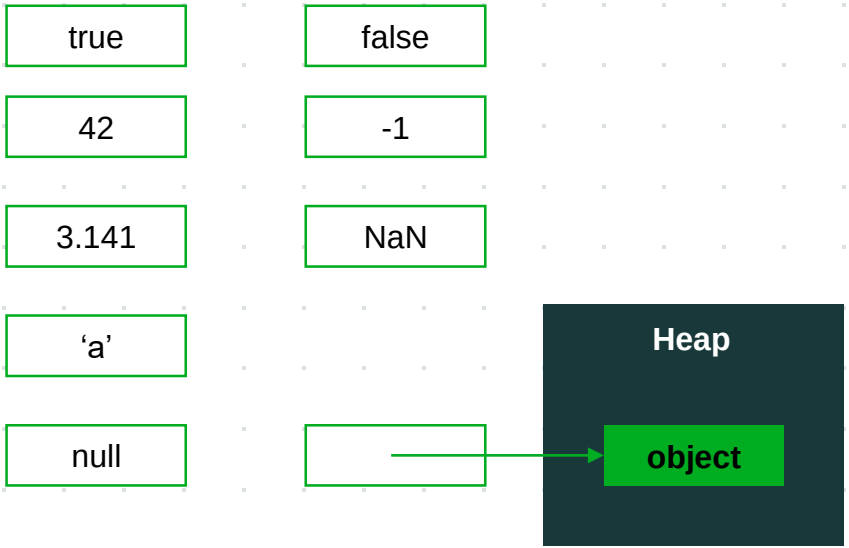
- Логическое : boolean
- Целочисленные: byte, short, int, long
- С плавающей точкой: float, double
- Символьные: char

Ссылочные

- Все остальные

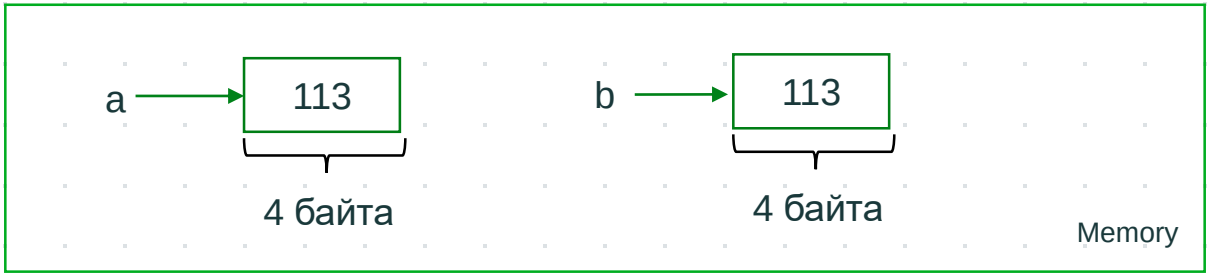
Типы данных

- Логическое : boolean
- Целочисленные: byte, short, int, long
- С плавающей точкой: float, double
- Символьные: char
- Ссылочные



nexign

Устройство примитивных типов




```
int a = 113;
int b = a;   // копирование по значению
```


nexign

Константы (final)

final применительно к переменным, указывает на запрет изменения значений после инициализации

```
final int a;
...
a = 42; // инициализация

...
a = 43; // compilation error:
// variable a might already have been initialized
```

nexign

Типы данных. Примитивы

Типы данных

- Логическое : boolean
- Целочисленные: byte, short, int, long
- С плавающей точкой: float, double
- Символьные: char

Типы данных

- **Логическое : boolean**
- Целочисленные: byte, short, int, long
- С плавающей точкой: float, double
- Символьные: char

Boolean

- Литералы: **false, true**
- Результат любого сравнения — **boolean**:

  < > == <= >= !=
- Нет преобразования между **boolean** и другими примитивными типами

```
boolean b = true;
//не сработает
if (b == 1) {...}
```

Boolean

Логические операторы

Оператор	Описание
&&	Логический оператор «И»
||	Логический оператор «ИЛИ»
!	Логический оператор «НЕ»
^	Логический оператор исключающее или (XOR)

- && и || — вычисление по сокращенной схеме
- & и | — вычисление по полной схеме

```
if (c instanceof Integer && c > 5) {...}
```

Операторы сравнения

Оператор	Описание
==	равны
!=	не равны
>	строго больше
<	строго меньше
>=	больше или равно
<=	меньше или равно

Типы данных

- Логическое : boolean
- **Целочисленные: byte, short, int, long**
- С плавающей точкой: float, double
- Символьные: char

Целочисленные

Type	Storage	Range
byte	1 byte	–128 to 127
short	2 bytes	–32,768 to 32,767
int	4 bytes	–2,147,483,648 to 2,147,483, 647
long	8 bytes	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

- Размер фиксирован, одинаков для всех платформ
- Все типы знаковые, беззнаковых вариантов нет

Целочисленные литералы

- Long 100L
- Underscores 1_000_000
- Hexadecimal 0xCAFEBABE
- Binary 0b0101
- Octal 010

```
long a = 100L;

int b = 1_000_000;

int c = 0x2D;

int d =0b0101;

int x = 010;
```

Арифметические операции

Оператор	Описание
+	Складывает значения по обе стороны от оператора
-	Вычитает правый операнд из левого операнда
*	Умножает значения по обе стороны от оператора
/	Оператор деления делит левый операнд на правый операнд
%	Делит левый операнд на правый операнд и возвращает остаток
++	Инкремент - увеличивает значение операнда на 1
--	Декремент - уменьшает значение операнда на 1

Арифметические операции

- a++, ++a
- a--, --a
- +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=, >>>=

```
y= 10;
z = ++y * y--;
```

```
y = y + 1;
z = 11 * 11;
y = y - 1;
```

Приоритет операторов

Категория	Оператор	Ассоциативность
Постфикс	() [] . (точка)	Слева направо
Унарный	++ - - ! ~	**Справа налево**
Мультипликативный	* / %	Слева направо
Аддитивный	+ -	Слева направо
Сдвиг	>> >>> <<	Слева направо
Реляционный	> >= < <=	Слева направо
Равенство	== !=	Слева направо
Побитовое «И» («AND»)	&	Слева направо
Побитовое исключающее «ИЛИ» («XOR»)	^	Слева направо
Побитовое «ИЛИ» («OR»)	|	Слева направо
Логическое «И» («AND»)	&&	Слева направо
Логическое «ИЛИ» («OR»)	||	Слева направо
Условный	?:	**Справа налево**
Присваивание	= += -= *= /= %= >>= <<= &= ^= |=	**Справа налево**
Запятая	,	Слева направо

nexign

Типы данных

- Логическое : boolean
- Целочисленные: byte, short, int, long
- **С плавающей точкой: float, double**
- Символьные: char

Типы с плавающей точкой

Type	Storage	Range
float	4 bytes	Approximately ±3.40282347E+38F (6–7 significant decimal digits)
double	8 bytes	Approximately ±1.79769313486231570E+308 (15 significant decimal digits)

По умолчанию: **double**

Литералы для типов с плавающей точкой

- 2.998e8
- 1.0 (1.)
- 3.14F
- 0x1.0p-3 (0.125, 2 в минус третьей степени)
- Double.POSITIVE_INFINITY
- Double.NEGATIVE_INFINITY
- Double.NaN

Проблемы float и double

- Побитовые операции не поддерживаются
- System.out.println(2.0 - 1.1) выводит 0.8999999999999999
- System.out.println(2.0 / 0.0) выводит 'Infinity` (но 2 / 0 — divizion by zero)
- System.out.println(0.0 / 0.0) выводит 'NaN`
- Сравнение с бесконечностью и NaN не работает, надо проверять

  с помощью Double.isNaN, Double.isInfinite.
- strictfp – вычисления соответствующие стандартам double и float

BigInteger, BigDecimal

- Лучше когда арифметика и сравнения просто работают
- Корректность гарантирована, корректность почти всегда важнее скорости
- Нет предыдущих проблем с double и float
- Очень большой диапазон

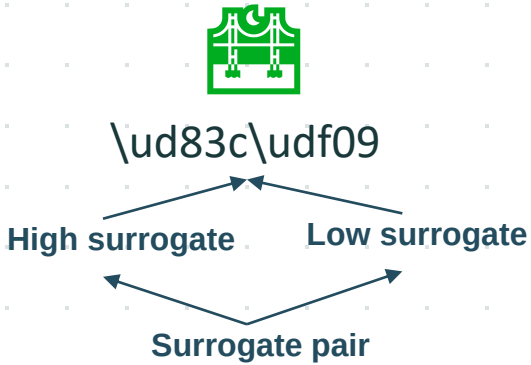
```
BigDecimal num1 = BigDecimal.ONE;
BigDecimal num3 = new BigDecimal(3);
System.out.println(num1.divide(num3, 3, RoundingMode.CEILING));
//0.334
System.out.println(num1.divide(num3, 3,RoundingMode.HALF_UP));
//0.333
```

Типы данных

- Логическое : boolean
- Целочисленные: byte, short, int, long
- С плавающей точкой: float, double
- **Символьные: char**

Char

- char — 16 бит, беззнаковый (0 .. 2 16 − 1)
- Представляет номер символа в кодировке Unicode
- Литералы: символ в одинарных кавычках: ’a’
- Шестнадцатеричный код символа: ’\u78bc’
- Спецпоследовательности: ’\t’, ’\n’, ’\r’, ’\”, ’\\’
- Свободно конвертируется в числовые типы и обратно




```
ch1 = 67; // код переменной
ch2 = 'a'; // сам символ
ch3 = 116; // код переменной
System.out.println(“Кот по-английски: " + ch1 + ch2 + ch3);
```


nexign

nexign

Типы данных. Преобразование типов

Расширяющее

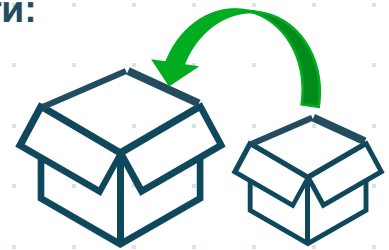
byte → short → int → long → float → double

char

```
short s = b;
int i = s;
long l = i;
float f = l;
double d = f;
```

Потеря точности:

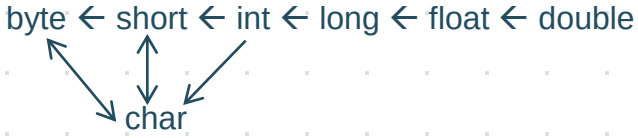
- int → float
- long → float
- long → double



Примитивный тип	Размер в памяти
byte	8 бит
short	16 бит
char	16 бит
int	32 бита
long	64 бита
float	32 бита
double	64 бита
boolean	8 (при использовании в массивах), 32 (при использовании не в массивах)

nexign

Сжимающее




```
a = (int) b;
```

(целевой_тип) значение

Потеря точности:

Всегда

- Оператор приведения типа: (typename)
- При приведении более емкого целого типа к менее емкому старшие биты просто отбрасываются
- При приведении типа с плавающей точкой к целому типу дробная часть отбрасывается (никакого округления)
- Слишком большое дробное число при приведении к целому превращается в MAX_VALUE или MIN_VALUE
- Слишком большой double при приведении к float превращается в Float.POSITIVE_INFINITY или Float.NEGATIVE_INFINITY

nexign

Автоматическое расширение

При вычислении выражения (a @ b) аргументы a и b преобразовываются в числа, имеющие одинаковый тип:

- если одно из чисел double, то в double;
- иначе, если одно из чисел float, то в float;
- иначе, если одно из чисел long, то в long;
- иначе оба числа преобразуются в int

```
byte b = 1;
byte c = b + 1; // compilation error
```

```
double a = Long.MAX_VALUE;
long b = Long.MAX_VALUE;
int c = 1;
System.out.println(a+b+c);
System.out.println(c+b+a);
```

nexign

Неявное приведение

- Сокращенная запись

  var @= expr раскрывается в

  var = (typename) (var @ (expr))

- Неявно срабатывает приведение типа,

  в том числе с потерей данных

```
byte b = 1;
b += 1; // OK
```

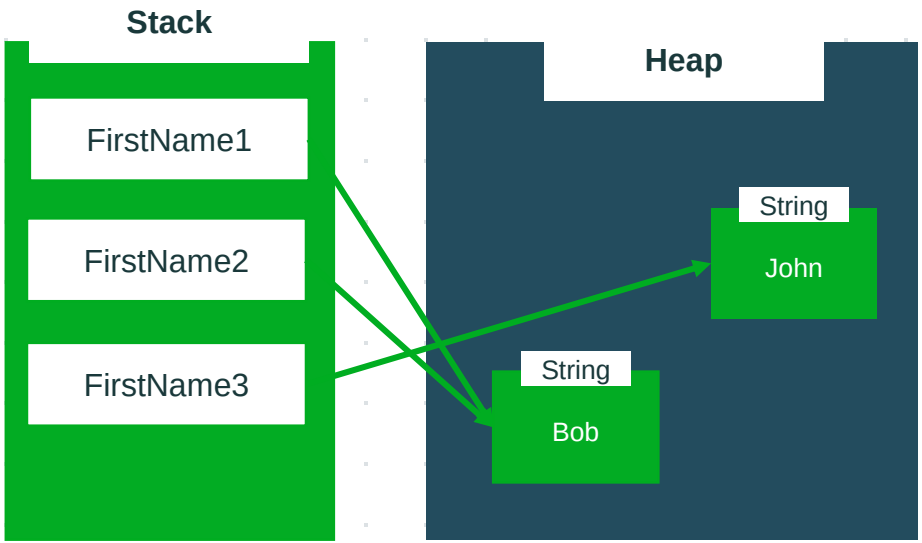
nexign

nexign

Типы данных. Ссылочные типы

Ссылочные типы

- Все остальные, кроме примитивных
- Передаются по ссылке
- Являются объектами
- Имеют поля и методы
- Ссылка может принимать значение null



nexign

Создание объекта

Для создания объекта в памяти используется оператор **new**

```
BigInteger number;
number = new BigInteger("12123");
...
BigInteger number2 = new BigInteger("123123");
```

nexign

Классы обёртки (Wrapper Classes)

Примитивные типы	Wrapper Classes
int	Integer
short	Short
long	Long
byte	Byte
float	Float
double	Double
char	Character
boolean	Boolean

Wrapper Classes

Объекты классов обёрток *кэшируются*, поэтому лучше их не создавать через **new**, а через **valueOf**.

Кэширование

Integer	Byte	Short/Long	Character	Boolean	Float/Double
-128…127	-128…127	-128…127	0…127	TRUE/FALSE	нет

Можно поднять верхнюю границу через параметр

-XX:AutoBoxCacheMax=<size>

Сравнение ссылок

- Оператор == сравнивает ссылки на объекты и выдает true тогда и только тогда, когда они указывают на один и тот же объект в памяти
- Для сравнения объектов по содержанию применяется метод equals(…), доступный всем классам в Java

```
String s1 = "abc";
String s2 = "ab" + "c";
System.out.println(s1 == s2);
System.out.println(s1.equals(s2));
```

Boxing/unboxing

- Autoboxing: примитивное значение → объект-обертка
- Autounboxing: объект-обертка → примитивное значение

```
Integer i = 1;
Integer j = i + 1;
int k = i + j;
```

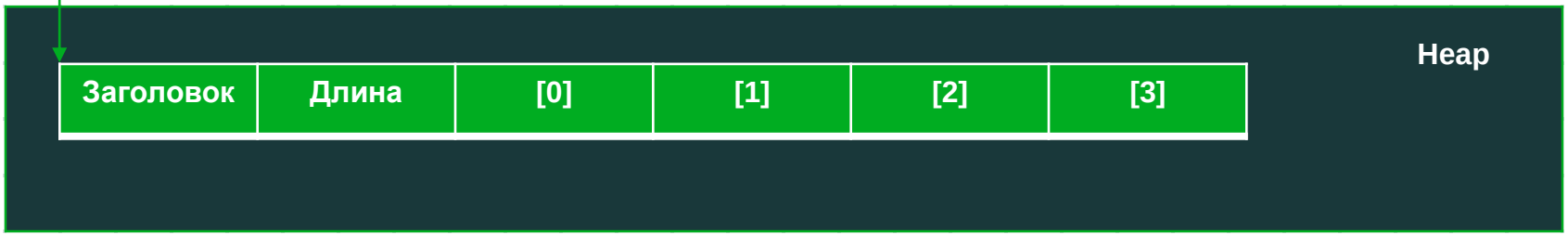
nexign

Типы данных. Ссылочные типы. Массивы

Массивы

- Из любого типа можно построить массив данного типа.
- Длина массива может быть определена в runtime, но после создания не может быть изменена.
- Массив аллоцируется в куче и передаётся по ссылке.
- Массивы проверяют тип данных (ArrayStoreException) и границы (ArrayIndexOutOfBoundsException) в run-time.

```
int[] ints;
```



nexign

Инициализация массивов

Шаблон

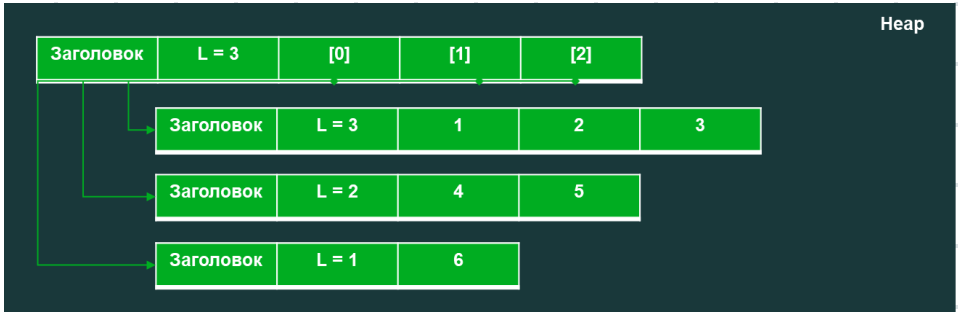
тип имя_переменной [ ] =
new тип [размер] {список_значений};

```
int  week_days[] = new int []
{1, 2, 3, 4, 5, 6, 7};
```

```
int numbers[] = {1, 2, 3, 4, 5};
int arrayLength = numbers.length ;
int firstNumber = numbers[0];
```

Многомерные массивы

```
int[][] initTable = {{1,2,3}, {4,5}, {6}};
```



nexign

Массивы передаются по ссылке

```
int[] smallPrimes = {0, 1, 2, 3, 4, 5, 6}
int[] luckyNumbers = smallPrimes;
luckyNumbers[5] = 12; // теперь smallPrimes[5] тоже 12

luckyNumbers = Arrays.copyOf(luckyNumbers, 2 * luckyNumbers.length);
//теперь luckyNumbers это отдельный массив
//и он стал в два раза длиннее
```

nexign

Операции с массивами

```
int [] ints = {1,2,3,2,1};                                  //5
System.out.println(ints.length);                            //1
System.out.println(ints[0]);                                //ints[1] = 3
ints[1] +=1;                                                //ArrayIndexOutOfBoundsException
ints[10] =10;
int[] copy = ints.clone();                                  //false
System.out.println(ints == copy);                           //false
System.out.println(ints.equals(copy));                      //true
System.out.println(Arrays.equals(ints, copy));              /[I@54bedef2
System.out.println(ints);                                   //[1, 3, 3, 2, 1]
System.out.println(Arrays.toString(ints));                  //[1,1,2,3,3,]
Arrays.sort(ints);
```

nexign

nexign

Типы данных. Ссылочные типы. Строки

Строки

- Для работы с текстовыми данными в **Java** чаще всего используется класс **String**.
- Класс **String** имеет две фундаментальные особенности:
  - **immutable** (неизменный) класс;
  - **final** класс.
- Класс **String** поддерживается на уровне виртуальной машины.
- Перегружен оператор **"+".**
- Неизменность класса **String**, порождает новые экземпляры строк, а старые отбрасываются, создавая большое количество мусора.

Работа со строками

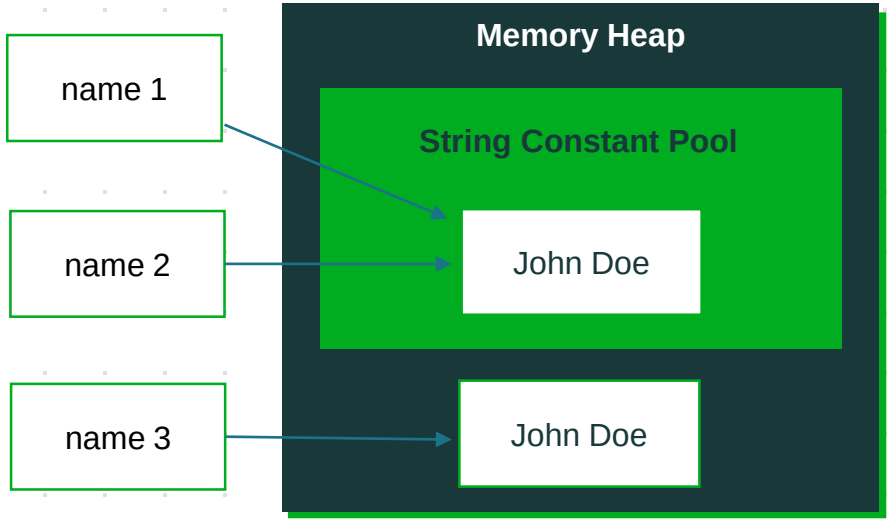
```
String s1 = "abc";              // Упрощенная инициализация строки s2:
String s2 = "abc";              // Инициализация сроки s3
String s3 = "ab" + "c";         // конкатенация
String s4 = new String("abc"); // Классическая инициализация строки
```

Внутреннее представление

- До Java 9 — char[]
- После Java 9 — byte[] и byte coder
- UTF-16 / Latin1 ('Compact Strings')

```
String name1 = "John Doe";
String name2 = "John Doe";

String name3
        = new String("John Doe");
```



Возврат в String Pool

```
name3 = name3.intern();
```

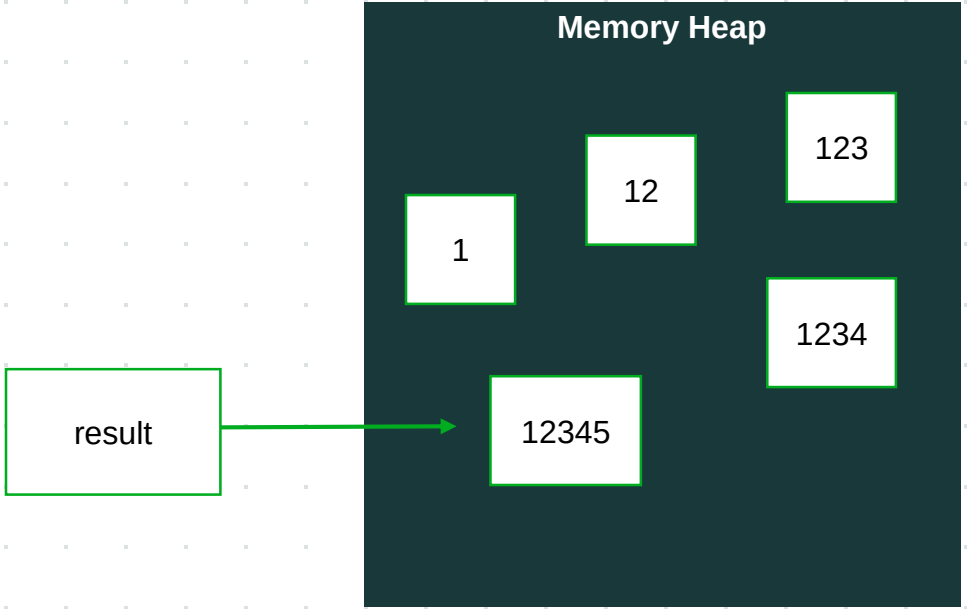

nexign

Методы String

```
String text = "Some Text";
int length() //9
char charAt(int index) //text.charAt(0) - S
int compareTo(String other) //"роза".compareTo("азор":)/ -16
boolean equals(Object other) //text.equals("text") /false
boolean  equalsIgnoreCase(Object other) //text.equals("some text") /true
boolean startWith(String prefix) //text.startWith("S") /true
boolean endsWith(String prefix) //text.endsWith("t") /true
String toLowerCase() / String toUpperCase() //text.toLowerCase()/SOME TEXT
String trim() //"   some text    ".trim()/ "sometext"
String join(CharSequence delimiter, CharSequence... elements)
//String.join("$","Demo","Text"); / Demo$Text
```

Проблемы неизменяемости

```
String result = "";
for (int i = 0; i < 5; i++)
        result += i;
        return result;
```



nexign

StringBuffer & StringBuilder

- Класс **StringBuffer** рекомендуется использовать в тех случаях, когда используются потоки (threads), так как он, в отличие от классов **String** и **StringBuilder**, обеспечивает синхронизацию строк.
- Класс **StringBuilder**, введённый начиная с **JDK 1.5**, полностью ему подобен, но синхронизации не поддерживает. Зато обеспечивает большую скорость работы со строками (что обычно бывает важно только в лексических анализаторах).

String vs StringBuffer vs StringBuilder

Особенности	String	StringBuffer	StringBuilder
Изменяемость	Immutable (нет)	mutable (да)	mutable (да)
Расширяемость	final (нет)	final (нет)	final (нет)
Потокобезопасность	Да, за счет неизменяемости	Да, за счет синхронизации	Нет
Когда использовать	Редкая модификация строк	Частая модификация строк в многопоточной среде	Частая модификация строк в однопоточной среде

Q&A