Информационные технологии и программирование

Лекция 1

Литература по курсу

- Java. Библиотека профессионала, том 1. Основы (Кей С. Хорстман)
- Java: эффективное программирование (Джошуа Блох)
- Java. Полное руководство (Герберт Шилдт)

Преподаватели

- Лекции - Мосева М.С. (@mmoseva)
- Лабораторные занятия - Харрасов К.Р./Мосева М.С./…

Балльная система

- С 1 сентября по 30 декабря - 18 учебных недель
- Лекции 1 раз в две недели, лаб. - каждую неделю
- 8 лабораторных и 60 тасков
- Всего за семестр можно заработать 60 баллов (за работу в течение семестра)

Балльная система

- 8 лекций - в конце лекции тестирование, за каждое тестирование максимум 1 балл -> 8 баллов
- 8 лабораторных Java - за каждую максимум 3 балла -> 24 балла
- 6 уровней тасков
  - за выполнение каждого 1 балл -> 6 баллов
  - защита тасков на занятиях, максимум 2 балла -> 12 баллов
- коллоквиум -> 10 баллов

Балльная система

- Максимальное количество за работу в семестре: 60 баллов
- Зачет автоматом: 50 баллов
- Зачет не автоматом: доп. вопрос на защиту незащищенных (но выполненных!) лабораторных работ - 1 балл за каждую работу + за сам зачет 15 баллов. Баллы за таски, тесты и коллоквиум не восстанавливаются.

Балльная система

«Полностью сданная лабораторная» представляет из себя:

– защиту лабораторной на занятии;

– отчет в ЭИОС;

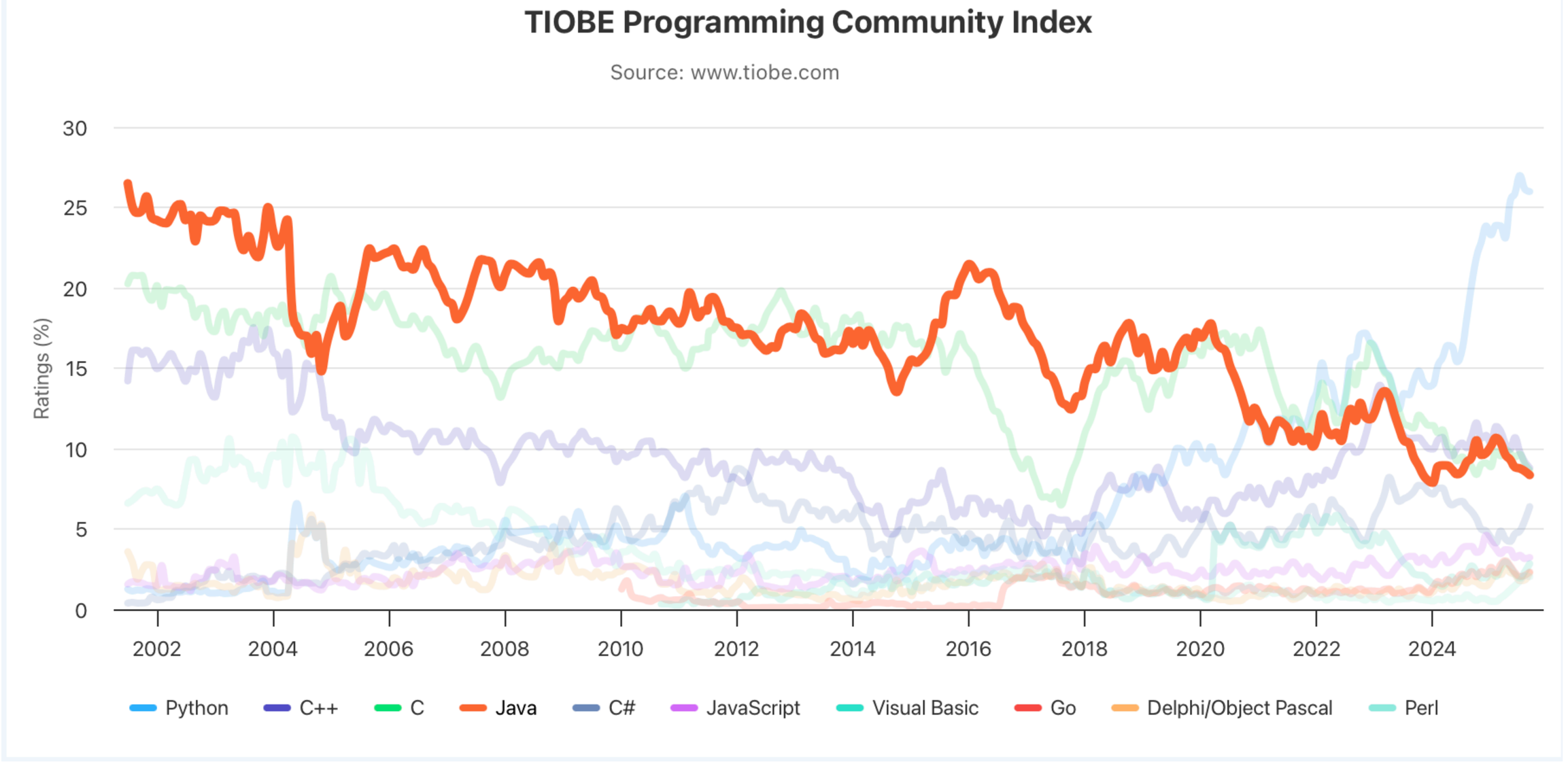
– код программы в Git-репозитории (ссылку также необходимо прикрепить в ЭИОС).

Выполненные таски:

- файл с кодом в ЭИОС

- код в Git-репозитории

Почему Java?


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
Python
C++
C
Java
C#
JavaScript
Visual Basic
Go
Delphi/Object Pascal
Perl

Very long term history

Programming Language	2025	2020	2015	2010	2005	2000	1995	1990	1985
Python	1	3	7	7	7	23	21	-	-
C++	2	4	3	3	3	2	1	2	10
C	3	2	1	2	1	1	2	1	1
Java	4	1	2	1	2	3	-	-	-
C#	5	5	5	6	8	10	-	-	-
JavaScript	6	7	8	9	11	7	-	-	-
Go	7	13	58	183	-	-	-	-	-
Visual Basic	8	12	11	-	-	-	-	-	-
Delphi/Object Pascal	9	186	12	10	9	-	-	-	-
SQL	10	9	-	-	-	-	-	-	-
Fortran	11	33	28	26	16	18	5	3	6
Ada	18	35	30	25	18	20	6	10	3
Lisp	26	29	27	16	15	9	7	5	2

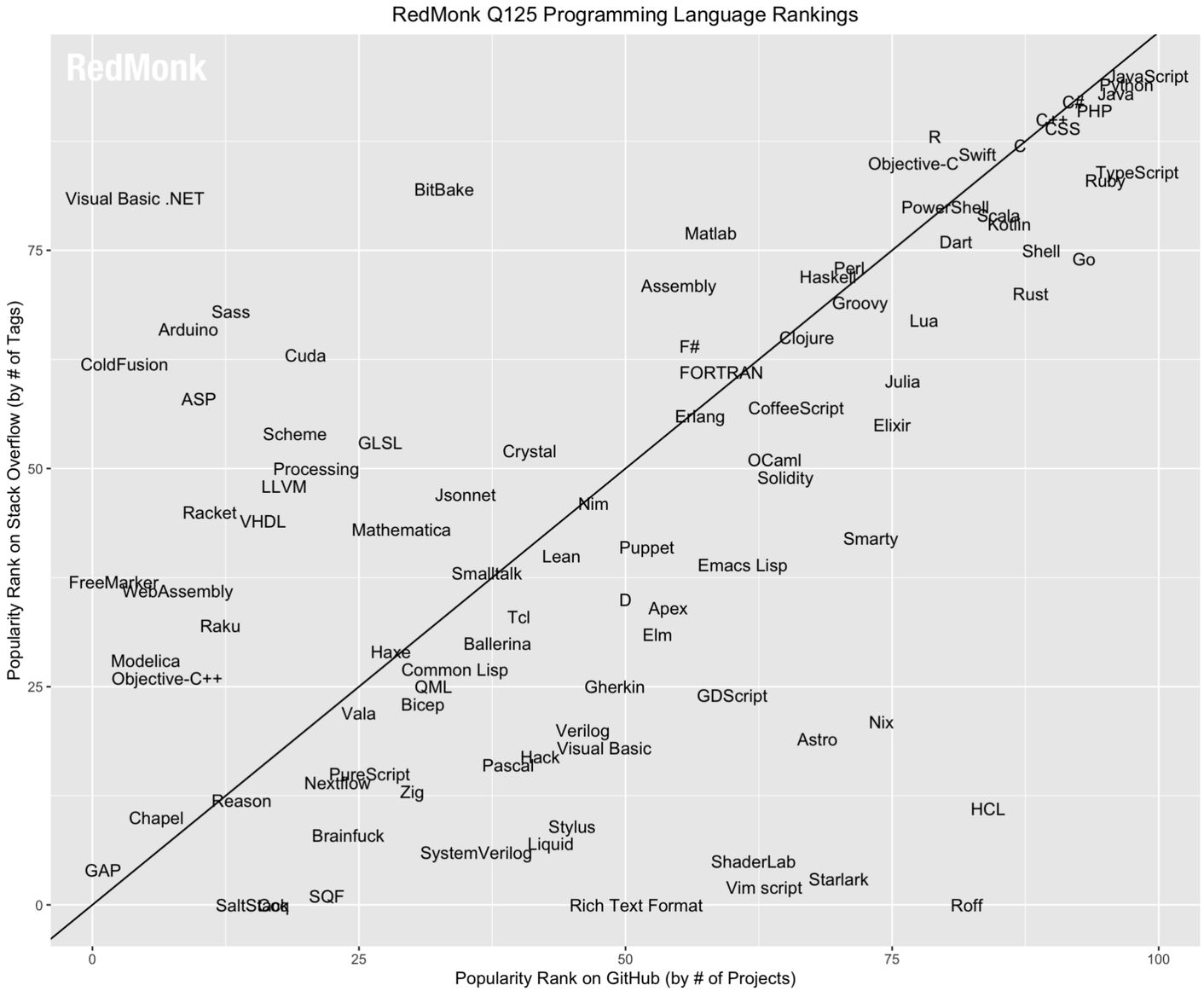
Почему Java?

PYPL PopularitY of Programming Language

Worldwide, Sept 2025 :

Rank	Change	Language	Share	1-year trend
1		Python	29.69 %	+0.2 %
2		Java	14.72 %	-0.7 %
3	↑	C/C++	9.27 %	+2.5 %
4	↓	JavaScript	6.79 %	-1.4 %
5	↑	R	5.26 %	+0.6 %
6	↓	C#	4.81 %	-1.8 %
7	↑↑↑↑	Objective-C	4.17 %	+1.7 %
8	↓	PHP	3.34 %	-0.8 %
9	↑	Rust	2.73 %	+0.2 %
10	↓	Swift	2.72 %	+0.0 %
11	↓↓↓	TypeScript	2.45 %	-0.5 %
12	↑↑↑	Ada	2.11 %	+1.1 %
13	↓	Go	1.56 %	-0.6 %
14		Matlab	1.49 %	+0.0 %

Почему Java?



1 JavaScript
2 Python
3 Java
4 PHP
5 C#
6 TypeScript
7 CSS
7 C++
9 Ruby
10 C
11 Swift
12 Go
12 R
14 Shell
14 Kotlin
14 Scala
17 Objective-C
18 PowerShell
19 Rust
20 Dart

История версий

- JDK 1.0 (January 23, 1996)
- JDK 1.1 (February 19, 1997)
- J2SE 1.2 (December 8, 1998)
- J2SE 1.3 (May 8, 2000)
- J2SE 1.4 (February 6, 2002)
- J2SE 5.0 (September 30, 2004)
- Java SE 6 (December 11, 2006)

— Sun acquisition by Oracle —

- Java SE 7 (July 28, 2011)
- Java SE 8 (March 18, 2014) (LTS)
- Java SE 9 (September 21, 2017)

— 6-monthly release cadence —

- Java SE 10 (March 20, 2018)
- Java SE 11 (September 25, 2018) (LTS)
- Java SE 12 (March 19, 2019)
- Java SE 13 (September 17, 2019)
- Java SE 14 (March 17, 2020)
- Java SE 15 (September 15, 2020)
- Java SE 16 (March 16, 2021)
- Java SE 17 (September 14, 2021) (LTS)
- Java SE 18 (March, 22nd 2022)
- Java SE 19 (September, 20th 2022)
- Java SE 20 (March, 21st 2023)
- Java SE 21 (September, 19th 2023) (LTS)
- Java SE 22 (March, 19th, 2024)
- Java SE 23 (September, 2024)
- Java SE 25 (March, 2025)

Характерные особенности

- простой;
- объектно-ориентированный;
- распределенный;
- надежный;
- безопасный;
- не зависящий от архитектуры компьютера;
- переносимый;
- высокопроизводительный;
- многопоточный;

Характерные особенности

- Простой

«Мы хотели создать систему, которая позволяла бы легко писать программы, не требовала дополнительного обучения и учитывала сложившуюся практику и стандарты программирования. Мы считали C++ не совсем подходящим для этих целей, но, чтобы сделать систему более доступной, язык Java был разработан как можно более похожим на C++. А исключили мы лишь редко используемые, малопонятные и нервразумительные средства C++, которые, по нашему мнению, приносят больше вреда, чем пользы».

- Объектно-ориентированный

«По существу, объектно-ориентированное проектирование - это методика программирования, в центре внимания которой находятся данные (т.е. объекты) и интерфейсы этих объектов. Проводя аналогию со столярным делом, можно сказать, что "объектно-ориентированный" мастер сосредоточен в первую очередь на стуле, который он изготавливает, и лишь во вторую очередь его интересуют необходимые для этого инструменты; в то же время "не объектно-ориентированный" столяр думает в первую очередь о своих инструментах. Объектно-ориентированные средства Java и C++ по существу совпадают».

Характерные особенности

- Распределенный

«Язык Java предоставляет разработчику обширную библиотеку программ для передачи данных по протоколу TCP/IP, HTTP и FTP. Приложения на Java способны открывать объекты и получать к ним доступ по сети с такой же легкостью, как и в локальной файловой системе, используя URL для адресации».

- Надежный

«Язык Java предназначен для написания программ, которые должны надежно работать в любых условиях. Основное внимание в этом языке уделяется раннему обнаружению возможных ошибок, контролю в процессе выполнения программы, а так же устранению ситуации, которые могут вызвать ошибки. Единственное существенное отличие языка Java от C++ кроется в модели указателей, принятой в Java, которая исключает возможность записи в произвольно выбранную область памяти и повреждения данных».

Характерные особенности

- Безопасный

«Язык Java предназначен для использования в сетевой или распределенной среде. По этому причине большое внимание было уделено безопасности. Java позволяет создавать системы, защищенные от вирусов и несанкционированного доступа».

- Независимый от архитектуры

«Компилятор генерирует объектный файл, формат которого не зависит от архитектуры компьютера. Скомпилированная программа может выполняться на любых процессорах, а для ее работы требуется лишь исполняющая система Java. Код, генерируемый компилятором Java, называется байт-кодом. Он разработан таким образом, чтобы его можно было легко интерпретировать на любой машине или оперативно преобразовать в собственный машинный код».

Характерные особенности

- Переносимый

«В отличие от C и C++, ни один из аспектов спецификации Java не зависит от реализации. Разрядность примитивных типов данных и арифметические операции над ними строго определены».

- Производительный

«Обычно интерпретируемый байт-код имеет достаточную производительность, но бывают ситуации, когда требуется еще более высокая производительность. Байт-код можно транслировать во время выполнения программы в машинный код для того процессора, на котором выполняется данное приложение».

- Многопоточный

«К преимуществам многопоточности относится более высокая интерактивная реакция и поведение программ в реальном масштабе времени».

Создатель языка

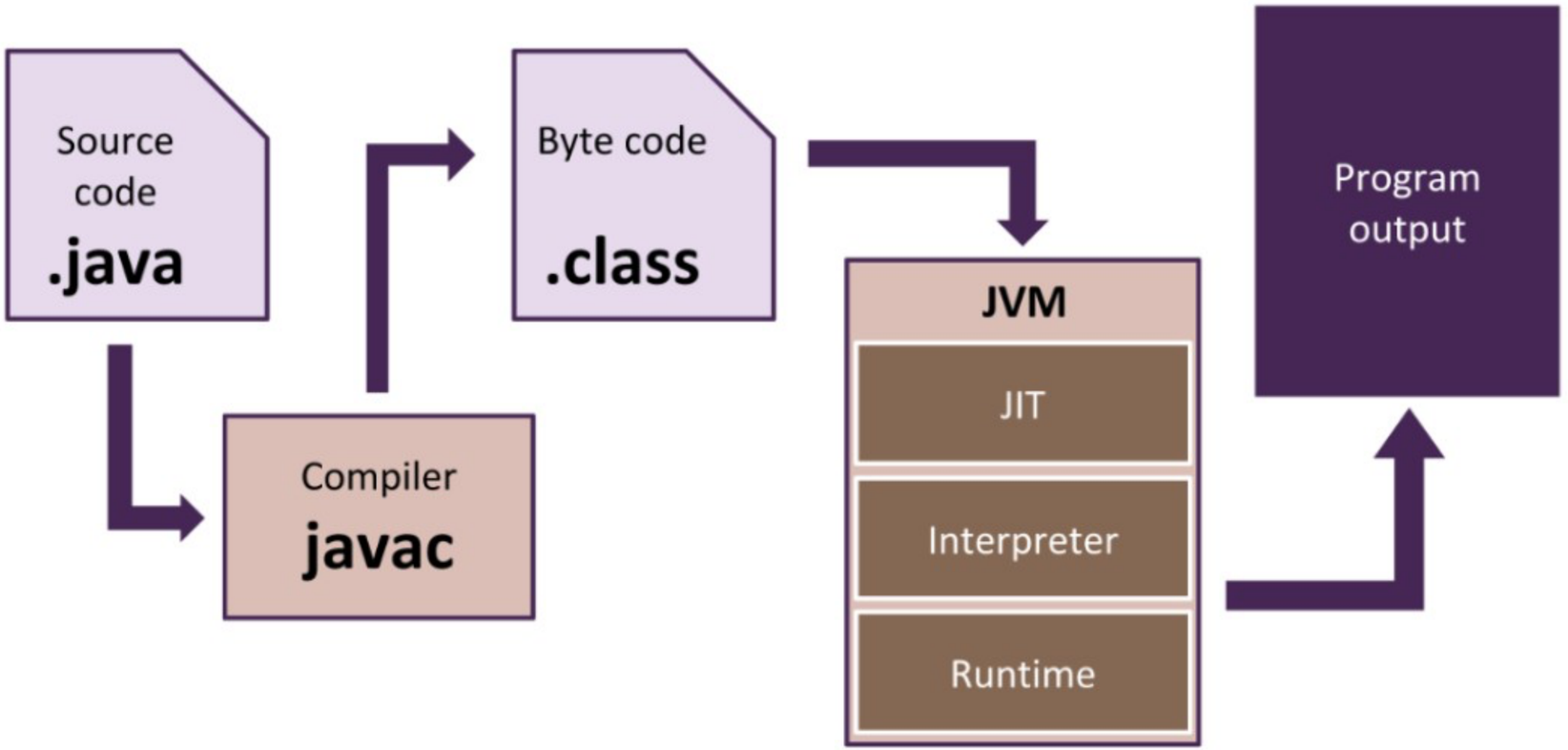
Джеймс Гослинг

- Вдохновлен C++
- Язык для приставок кабельного телевидения
- Принцип **WORA**

Где сегодня используется?

- Desktop
- Embedded Systems
- Server-side (backend)
- Mobile Devices (Android)

Интерпретация и JIT-компиляция



Java Code Style: как правильно оформлять код Java

1. Название файла должно совпадать с именем главного класса в нём

❗ **Неправильно:** файл main.java

✅ **Правильно:** файл называется как класс — Main.java

2. Названия классов и интерфейсов

❗ **Неправильно:** phoneBook, full_name, main, result-list

✅ **Правильно:** PhoneBook, FullName, Main, ResultList

3. Названия методов

❗ **Неправильно:** Print (), get-Size (), Main (), is_hidden ()

✅ **Правильно:** print (), getSize (), main (), isHidden ()

4. Названия переменных

❗ **Неправильно:** PostService, e_mail, post-id, _email

✅ **Правильно:** postService, email, postId, isAlive

Java Code Style: как правильно оформлять код Java

5. Названия констант

✅ Константы:

```
static final int NUMBER = 5; //примитив
enum SomeEnum { ENUM_CONSTANT } // enum
static final ImmutableList<String> NAMES = ImmutableList.of("Ed", "Ann"); // неизменяемый список, с неизменяемыми элементами String
static final ImmutableMap<String, Integer> AGES = ImmutableMap.of("Ed", 35, "Ann", 32); // неизменяемый Map, с неизменяемыми элементами String
```

❗ Не именуются как константы:

```
static String nonFinal = "non-final"; // не final
final String nonStatic = "non-static"; // не статична
static final Set<String> mutableCollection = new HashSet<String>(); // мы можем изменять содержимое Set
static final ImmutableSet<SomeMutableType> mutableElements = ImmutableSet.of(mutable); // изменить количество элементов в коллекции нет возможности, но менять состояние самих элементов коллекции есть возможность
static final String[] nonEmptyArray = {"these", "can", "change"}; // поменять элементы массива возможно
```

Java Code Style: как правильно оформлять код Java

6. Не используйте транслитерацию

❗ **Неправильно:** schet, kolichestvo, soderjitlmya, fio

✅ **Правильно:** account, amount, containsName, fullname

7. Фигурные скобки

Фигурная скобка открывается на той же строке, что и код перед ней.

```
public static void main(String[] args) {
    System.out.println("Hello, code style!");
}
```

Простая программа на Java

```
public class FirstSample {
  public static void main(String[] args) {
    System.out.println("We will not use 'Hello, World!'");
  }
}
```

Комментарии

```
// Однострочный комментарий
public class ClassName {
    public static void main(String[] args) {
    /*Многострочный комментарий.
    Вызов метода без параметров. Переход на следующую строку.
    */
    System.out.println();
    }
}
```

Комментарии

```
/**
* так формировать
* комментарии для
* документации
*/
public class Main {
    public static void main(String[] args) {
        System.out.println();
    }
}
```

Ключевые слова

В языке Java 50 ключевых слов, которых нужно избегать при выборе имен для идентификаторов и объектов:

Типы
byte
short
int
long
char
float
double
boolean

Циклы и ветвления
if
else
switch
case
default
while
do
break
continue
for

Исключения
try
catch
finally
throw
throws

Области видимости
private
protected
public

Объявление\Импорт
import
package
class
interface
extends
implements
static
final
void
abstract
native

Создание\Возврат\Вызов
new
return
this
super

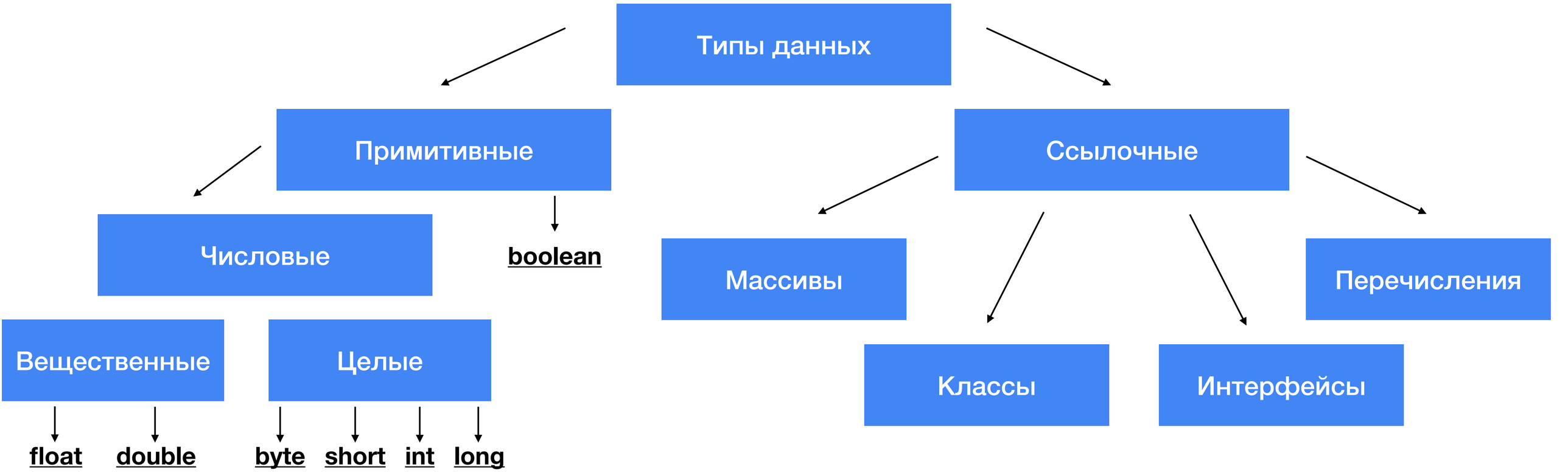
Многопоточность
synchronized
volatile

Зарезервированы, но не используются
const
goto

Остальные
instanceof
enum
assert
transient
strictfp

Зарезервированные литералы: **true, false, null**
const, goto зарезервированы, но не используются

Типы данных



Java - строго типизированный язык

Целочисленные типы данных

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

Разновидности целочисленных литералов

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

Типы данных с плавающей запятой

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)

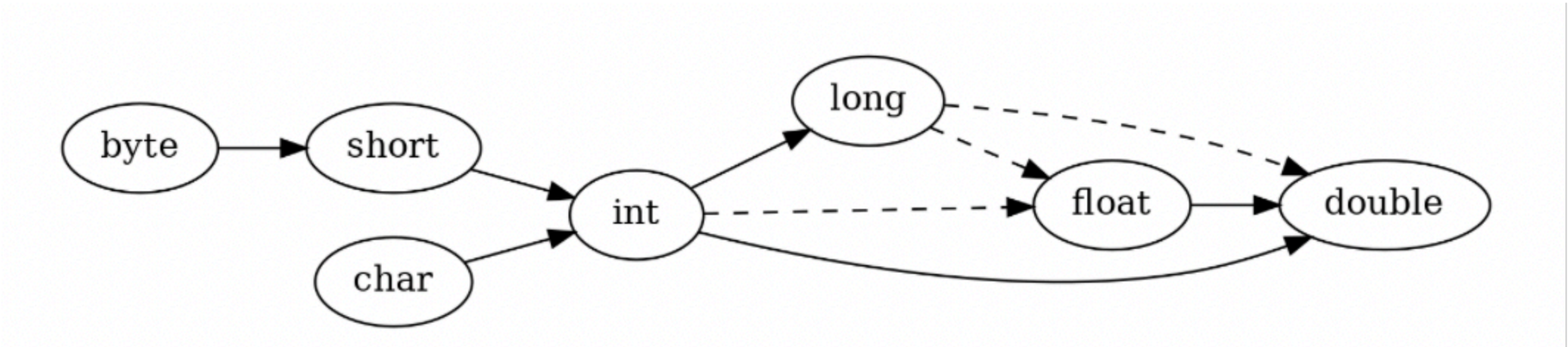
Литералы для типов с плавающей запятой

- 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

Числа с плавающей запятой

- System.out.println(2.0 - 1.1) выводит 0.8999999999999999
- НИКОГДА НЕ СЧИТАЙТЕ ДЕНЬГИ DOUBLE-ОМ!
- `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

Приводимость числовых типов



Правила приведения типов

```
double x = 9.997F; //float в double неявно
int nx = (int) x; //9
nx = (int) Math.round(x); //10
```

✓ Если хотя бы один из операндов относится к типу `double`, то и второй операнд преобразуется в тип `double`.

✓ Иначе, если хотя бы один из операндов относится к типу `float`, то и второй операнд преобразуется в тип `float`.

✓ Иначе, если хотя бы один из операндов относится к типу `long`, то и второй операнд преобразуется в тип `long`.

✓ Иначе оба операнда преобразуются в тип `int`.

Тип char

- 16 bit
- code unit in the UTF-16 encoding
- Не всегда целый символ (хотя почти всегда)!
- Осторожней с Unicode, используйте `String`

Символьные литералы

- ‘ы'
- `‘\u03C0'` (греческая π)
- `‘\''` (escaped single quote)
- Ловушка Unicode Escape: `// Look inside c \users`

Строковые литералы

```
String s = "Hello World!";

// before Java 13

String address = "25 Main Street\n" +
         "Anytown, USA, 12345\n";

//или

String address = "25 Main Street\nAnytown, USA, 12345\n";

String address = """
         25 Main Street
         Anytown, USA, 12345
         """;
```

Тип boolean

- true и false
- В отличие от C и многих других языков, целые числа не сводятся автоматически к `boolean`
- Избегаем ошибок вида `if (x = 0) {...`

Область видимости и время жизни переменной

Время жизни переменной в Java определяется правилом:

- Переменная создается в точке ее описания и существует до момента окончания того блока, в котором находится данное описание.

Областью видимости переменной (scope) является фрагмент программы от точки ее описания до конца текущего блока.

- Область видимости — это статическое понятие, имеющее отношение к какому-то фрагменту текста программы. Время жизни, в отличие от области видимости, — это понятие динамики выполнения программы. Время жизни переменных в Java совпадает с их областью видимости с учетом отличия самих этих понятий.

Область видимости и время жизни объектов

Время жизни объекта определяется следующим правилом.

Объект существует, пока существует хотя бы одна ссылка на этот объект.

Это правило, однако, не утверждает, что объект будет уничтожен, как только пропадет последняя ссылка на него. Просто такой объект становится недоступным и может быть уничтожен.

В Java нет явного уничтожения объектов. Объекты уничтожаются (говорят — утилизируются) *сборщиком мусора* (*garbage collector*), который работает в фоновом режиме параллельно с самой программой на Java.

```
{
    SomeType localReference = new SomeType();
    globalReference = localReference;
}
```

Определение переменных

```
// декларация
double salary;
int vacationDays;
long earthPopulation;
boolean done;
// не очень приветствуется
int i, j;
// технически можно, но . . .
int суммаНДФЛ;
```

Инициализация переменных

```
int vacationDays;
System.out.println(vacationDays); //COMPILATION ERROR
  // variable not initialized
```

Возможные способы инициализации:

```
int vacationDays;
vacationDays = 12;
int vacationDays = 12;
```

Инициализация переменных

```
// before Java 10
int vacationDays = 12;
String greeting = "Hello, world!";
// after
var vacationDays = 12;  // int
var greeting = "Hello, world!"; // String
```

Ключевое слово final

`final` используется в двух смыслах:

- запрещает изменять значение
- запрещает переопределять методы/классы

```
final int a;
...
a = 42; // инициализация
...
a = 43; // compilation error:
// variable a might already have been initialized
```

Ключевое слово final

```
pubic class Constants {

    public static final double CM_PER INCH = 2.54;

    pubic static void main(String[] args) {
        double paperWidth = 8.5;double paperHeight = 11;
        System.out.println("Paper size in centimeters: " +
                           paperWidth * CM_PER_INCH +" by " +
                           paperHeight * CM_PER_INCH);
    }
}
```