Введение в программирование на Java

Лекция 1. Введение.

Виталий Олегович Афанасьев

05/12 ноября 2025

О содержании курса

18 лекций и 18 семинаров:

1. Введение.
2. Примитивные типы данных. Базовые операции.
3. Условные операции. Циклы.
4. Ссылочные типы данных. Строки. Массивы.
5. Процедурное программирование. Методы.
6. ООП (часть 1). Базовые понятия.
7. ООП (часть 2). Отношения между классами.
8. ООП (часть 3). Интерфейсы. Абстрактные классы.
9. ООП (часть 4). Принципы SOLID.
10. Организация проектов. Пакеты и модули.
11. Верификация ПО. Модульное тестирование.
12. Обработка ошибок. Исключения.
13. Дженерики (часть 1). Базовые коллекции.
14. Дженерики (часть 2). Вариантность.
15. Начала функционального программирования.
16. Stream API.
17. Работа с файлами.
18. Метапрограммирование. Reflection API.

Элементы контроля (1)

- Контесты (не менее 12 штук, срок выполнения $\approx$ 1 неделя)
  - Небольшие задачи на пройденные темы
  - (Почти) после каждого семинара
  - Каждый контест оценивается по 10-бальной шкале как $10 \cdot \frac{\text{кол-во решённых задач}}{\text{кол-во задач в контесте}}$
  - $\text{О}_{\text{КонтЗ}}$ = средняя оценка за **5 лучших** контестов
  - $\text{О}_{\text{КонтИтог}}$ = средняя оценка за **10 лучших** контестов
- Активность на семинарах
  - Именно активность, а не простое присутствие!
  - $\text{О}_{\text{Сем}} = \min(10, \text{кол-во семинаров с активностью})$
- 3 больших домашних задания (срок выполнения $\approx$ 2-3 недели)
  1. Процедурное программирование
  2. Объектно-ориентированное программирование
  3. Объектно-ориентированное и функциональное программирование
- 2 экзаменационных теста (в 3-м модуле и в 4-м модуле)
  - Теоретические вопросы
  - Вопросы на понимание кода

Элементы контроля (2)

Все оценки (кроме оценок за модули и итоговой) округляются до 1 знака после запятой.

Формула оценки за 3-й модуль:

$$0.25 \cdot О_{Конт3} + 0.25 \cdot О_{ДЗ1} + 0.5 \cdot О_{Экз3}$$

Формула итоговой оценки:

$$0.1 \cdot О_{Сем}$$
$$+$$
$$0.15 \cdot О_{КонтИтог}$$
$$+$$
$$0.15 \cdot О_{ДЗ1} + 0.15 \cdot О_{ДЗ2} + 0.15 \cdot О_{ДЗ3}$$
$$+$$
$$0.3 \cdot О_{Экз4}$$

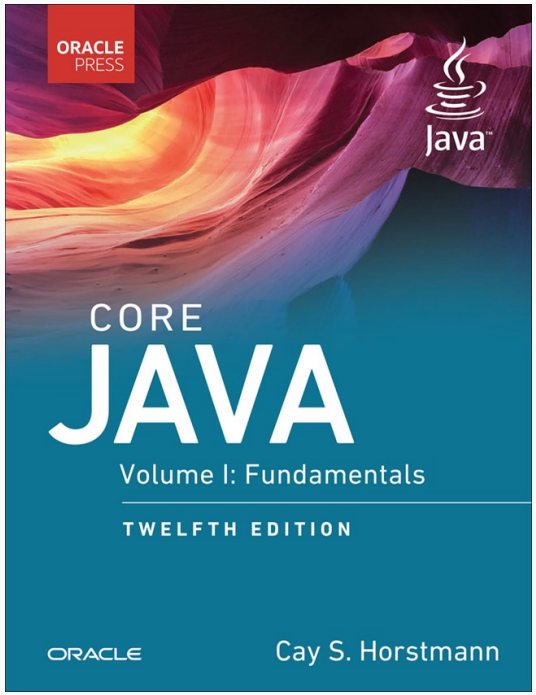
Элементы контроля (3)

- Использование любых ИИ-помощников наказуемо!
  - Курс нужен, чтобы вы натренировали свою "думательную машинку", а не очередную модель от OpenAI.
- При любом подозрении в несамостоятельном выполнении предусматриваются защиты (либо обнуление ДЗ/контеста при грубом нарушении).

Важные моменты

- Не бойтесь задавать вопросы (какими бы глупыми они вам не казались).
- Нужно практиковаться. **Много**.
- **Крайне** рекомендуется читать рекомендованную литературу и разбирать примеры оттуда с компьютером.
- Все информация о курсе будет публиковаться в Telegram-канале.

Основная литература



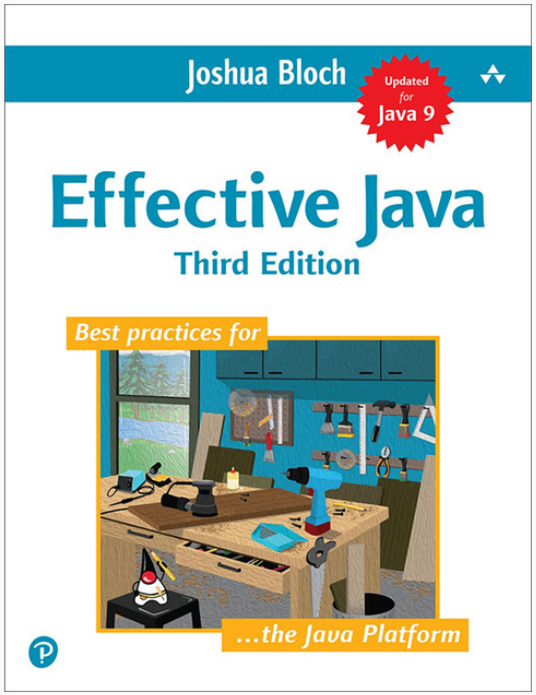
Core Java, Volume I: Fundamentals

12th edition

by Cay Horstmann

Полезная дополнительная литература

Java Notes for Professionals



Effective Java
3rd edition
by Joshua Bloch

Ссылка на чат

Важные понятия в языках программирования

Компиляция и интерпретация (1)

Интерпретация —

Компиляция —

Компиляция и интерпретация (1)

Интерпретация — код выполняется "на лету", без предварительной обработки.

Компиляция — код предварительно обрабатывается компилятором для получения исполняемого файла (либо промежуточного представления, которое можно обработать в дальнейшем).

Компиляция и интерпретация (2)

Но!:

- Программу на "компилируемом" языке можно интерпретировать (например, для языка C существуют такие инструменты ка Ch, TCC, PicoC)
- Программу на "интерпретируемом" языке можно скомпилировать (например, функция `compile` в стандартной библиотеке языка Python)

Тем не менее, языки обычно проектируют так, чтобы они были более предназначены либо для компиляции, либо интерпретации (хотя существуют и "средние" варианты).

Виды типизации

Статическая —

Динамическая —

Виды типизации

Статическая — тип связывается с переменными (функциями, полями, etc) при объявлении и не может быть изменён.

```
int x = 42;
x = "Hello"; // Error!
```

Динамическая — тип определяется на основе значения (т.е. одна переменная может иметь разные типы во время выполнения).

```
x = 42      # typeof(x) -> int
x = "Hello" # typeof(x) -> str
```

О языке Java

Особенности языка Java

- Компилируемый
  (но также и интерпретируемый; об этом дальше)
- Множество парадигм (процедурная,
  объектно-ориентированная, функциональная и т.д.)
- Статическая типизация
- Платформонезависимый
- Управление памятью при помощи сборщика мусора

Мы будем использовать версию **Java 21**. На данный момент это последняя LTS версия.

Платформонезависимость (1)

Вопрос к программирующим на C/C++:

Что нужно сделать, чтобы программа заработала на Windows, Linux, MacOS для процессоров с разными архитектурами (ARM, x86 и т.д.)?

Платформонезависимость (2)

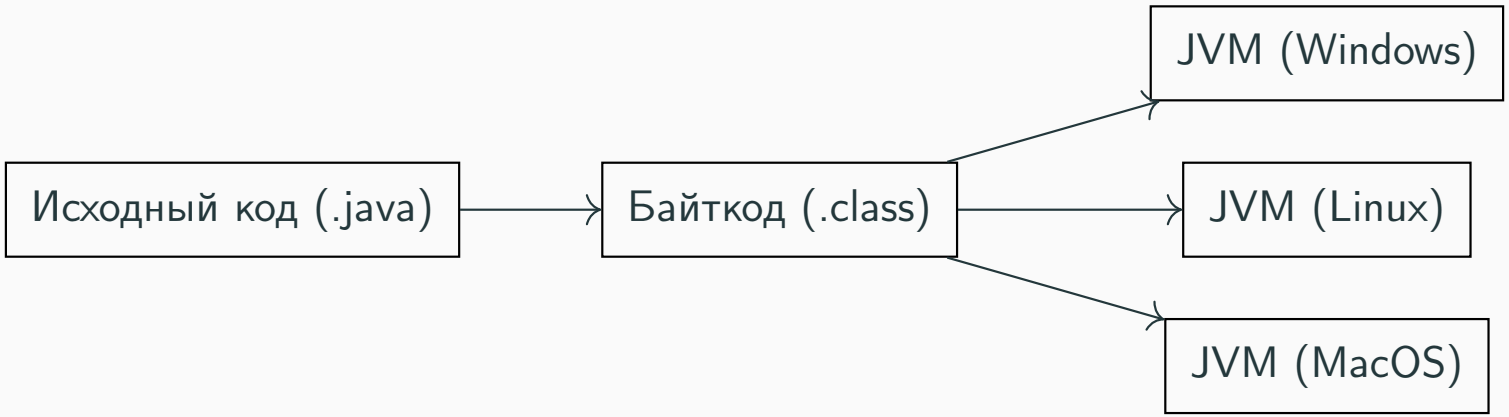
Java использует концепцию **виртуальной машины** (**JVM**, **Java Virtual Machine**):

- Код компилируется в **байткод**, который одинаков для любого hardware и операционных систем
- Реализация JVM под конкретную платформу исполняет (интерпретирует) байткод с учётом особенностей платформы
- "Write once, run everywhere"

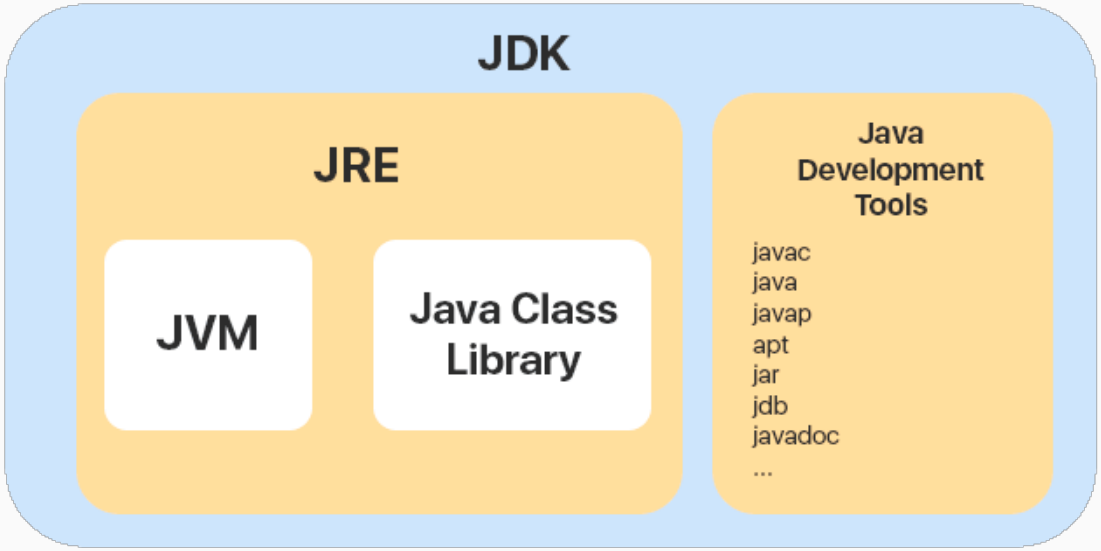
Платформонезависимость (2)

Java использует концепцию **виртуальной машины** (**JVM**, **Java Virtual Machine**):

- Код компилируется в **байткод**, который одинаков для любого hardware и операционных систем
- Реализация JVM под конкретную платформу исполняет (интерпретирует) байткод с учётом особенностей платформы
- "Write once, run everywhere"



JVM, JRE, JDK



JVM (**Java Virtual Machine**) — виртуальная машина, исполняющая скомпилированные программы.

JRE (**Java Runtime Environment**) — компонент, достаточный для выполнения любой программы на Java. Состоит из JVM и стандартной библиотеки.

JDK (**Java Development Kit**) — компонент для разработки программ на Java. Состоит из JRE и "программистских" утилит (компилятора, отладчика и т.п.).

Управление памятью

В языках с ручным управлением памятью необходимо следить за каждым объектом и освобождать выделенную память:

```
int* array = new int[42];
...
delete[] array;
```

Java же использует т.н. **"сборку мусора"** — программист создаёт объекты в своём коде, а виртуальная машина автоматически освобождает неиспользуемую память.

Что пишут на Java?

Что пишут на Java?

Серверные приложения

Что пишут на Java?

Серверные приложения

Десктопные приложения

Что пишут на Java?

Серверные приложения

Десктопные приложения

Мобильные приложения

Что пишут на Java?

Серверные приложения

Десктопные приложения

Мобильные приложения

Видеоигры

Что пишут на Java?

Серверные приложения

Десктопные приложения

Короче говоря — что угодно

Язык — всего лишь инструмент

Мобильные приложения

Видеоигры

Экосистема JVM

Всё не ограничивается одним языком.

Популярность языка Java, его платформонезависимость и обширность стандартной библиотеки послужило началом и для других языков:

- Kotlin
- Scala
- Groovy
- Clojure
- Ceylon
- Jython
- и ещё много-много-много...

Простейшая программа на Java

Файл: Main.java

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Простейшая программа на Java

Файл: Main.java

```
1 public class Main {
2     public static void main(String[] args) {
3         System.out.println("Hello, World!");
4     }
5 }
```

Простейшая программа на Java

Файл: Main.java

```
1 public class Main {
2     public static void main(String[] args) {
3         System.out.println("Hello, World!");
4     }
5 }
```

Простейшая программа на Java

Файл: Main.java

```
1 public class Main {
2     public static void main(String[] args) {
3         System.out.println("Hello, World!");
4     }
5 }
```