

Введение в программирование на Java

Лекция 1: Обзор языков высокого уровня, синтаксис Java и инструменты (IntelliJ IDEA)

Александр Глускер

РУТ МИИТ/ВИШ

5 декабря 2025 г.

- 1 Обзор языков высокого уровня
- 2 Java сегодня: роль языка и JVM
- 3 Инструменты: JDK, IntelliJ IDEA, CLI, JShell
- 4 Синтаксис Java: базовые конструкции
- 5 Ввод/вывод и обработка ошибок
- 6 Массивы и методы (без ООП)
- 7 Итоги и мини-практика

Цели

- Понять место Java и философию JVM
- Установить JDK и настроить IntelliJ IDEA
- Научиться писать и запускать простые программы
- Подготовить базу к занятию по ООП

Навигация по лекции

- 1 Обзор языков высокого уровня
- 2 Java сегодня: роль языка и JVM
- 3 Инструменты: JDK, IntelliJ IDEA, CLI, JShell
- 4 Синтаксис Java: база до ООП
- 5 Ввод/вывод и обработка ошибок
- 6 Массивы и методы (минимум до ООП)
- 7 Итоги

Высокоуровневые языки: карта местности

- **Типизация:** статическая (Java, C#), динамическая (Python, JS)
- **Компиляция:** в машинный код (Go, Rust, Kotlin), в байткод под VM (Java, Kotlin, Scala), JIT/интерпретация (Python/JS)
- **Управление памятью:** сборка мусора (Java, C#), владение/заимствование (Rust), ручная (C)
- **Экосистемы:** JVM (Java/Kotlin/Scala), .NET, нативные (C/C++/Rust), веб (JS/TS)

Почему нам Java

- Сильная статическая типизация, крупная экосистема, стабильные LTS-релизы
- Универсальность: бэкенд (но...), инструменты (но...), Android (исторически), desktop/CLI (но...)

Навигация по лекции

- 1 Обзор языков высокого уровня
- 2 Java сегодня: роль языка и JVM**
- 3 Инструменты: JDK, IntelliJ IDEA, CLI, JShell
- 4 Синтаксис Java: база до ООП
- 5 Ввод/вывод и обработка ошибок
- 6 Массивы и методы (минимум до ООП)
- 7 Итоги

Java в 2025: трезвый взгляд

- **Сервер/Enterprise:** де-факто стандарт (Spring, Jakarta EE)
- **Android:** Kotlin-first для нового кода; Java остаётся в существующих проектах и библиотеках
- **JVM-платформа:** множество языков (Kotlin, Scala и др.) на одном рантайме

Версии языка

- LTS: 17 (2021), 21 (2023), 25 (план — сентябрь 2025)

Философия JVM

- *Write once, run anywhere:* компиляция в байткод, исполнение на JVM
- HotSpot JIT оптимизирует «на лету», есть современные GC
- OpenJDK — *референсная реализация* Java SE

Экосистема Java кратко

Компонент	Что это
JDK	Компилятор <code>javac</code> , рантайм, инструменты (<code>jar</code> , <code>javadoc</code> , <code>jshell</code>)
JVM (HotSpot)	Виртуальная машина: загрузка классов, JIT, GC
OpenJDK	Открытая референсная реализация Java SE
LTS	Версии с длительной поддержкой (17/21/25)
Билд-системы	Maven/Gradle (зависимости, сборка, тесты)
IDE	IntelliJ IDEA (Community/Ultimate)

Навигация по лекции

- 1 Обзор языков высокого уровня
- 2 Java сегодня: роль языка и JVM
- 3 Инструменты: JDK, IntelliJ IDEA, CLI, JShell**
- 4 Синтаксис Java: база до ООП
- 5 Ввод/вывод и обработка ошибок
- 6 Массивы и методы (минимум до ООП)
- 7 Итоги

Установка и настройка

- 1 Установите **IntelliJ IDEA Community** (хватает для чистой Java, Maven/Gradle)
- 2 В IDEA: New Project → Java →, укажите название проекта (English), остальное по умолчанию
- 3 Запуск: зелёный треугольник слева от `main`

JShell: REPL для быстрых экспериментов

```
1 $ jshell
2 jshell> int x = 2 + 2;
3 x ==> 4
4 jshell> "Hello, " + "Java"
5 $2 ==> "Hello, Java"
```

- Полезно для проверки выражений, без создания проекта

Навигация по лекции

- 1 Обзор языков высокого уровня
- 2 Java сегодня: роль языка и JVM
- 3 Инструменты: JDK, IntelliJ IDEA, CLI, JShell
- 4 Синтаксис Java: база до ООП**
- 5 Ввод/вывод и обработка ошибок
- 6 Массивы и методы (минимум до ООП)
- 7 Итоги

Структура минимальной программы

```
1 // Файл Main.java
2 public class Main {
3     public static void main(String[] args) {
4         System.out.println("Hello, Java!");
5     }
6 }
```

- Пока игнорируем ООП: мыслите о `class Main` как о контейнере для `main`

Комментарии, идентификаторы, литералы

```
1 // Однострочный комментарий
2 /* Многострочный */
3 /**
4  * Доккомментарий–
5  */
6 int answer = 42;
7 double pi = 3.1415;
8 char letter = 'A';
9 String s = "Строка";
```

Типы и переменные

```
1 int i = 10; long L = 10L;  
2 double d = 2.5; float f = 2.5f;  
3 boolean ok = true;  
4 char c = 'Ж';  
5 String text = "Привет";  
6 var sum = i + 5; // локальноевыведениетипа (Java 10+)
```

Замечания

- var — только для локальных переменных; тип всё равно статический
- Преобразования: явные касты, (int) d, и методы вроде Integer.parseInt

Операторы

```
1 int a = 7, b = 3;  
2 int sum = a + b;      // + - * / %  
3 int mod = a % b;  
4 boolean gt = a > b;   // > < >= <= == !=  
5 boolean both = (a > 0) && (b > 0); // && || !  
6 a++; --b; // инкремент/декремент/
```

Условия: if/else и тернарный оператор

```
1 int x = -3;
2 if (x >= 0) {
3     System.out.println("Неотрицательное");
4 } else {
5     System.out.println("Отрицательное");
6 }
7 String sign = (x >= 0) ? "non-negative" : "negative";
```


switch: классический и современный (Java 14+)

```
1 // Классический
2 int day = 2;
3 switch (day) {
4     case 1: System.out.println("Пн"); break;
5     case 2: System.out.println("Вт"); break;
6     default: System.out.println("Другое");
7 }
8
9 // Выражение switch (Java 14+)
10 int n = 3;
11 String name = switch (n) {
12     case 1 -> "Один";
13     case 2, 3 -> "Два или три ";
14     default -> "Иное";
15 };
```

Циклы: while, do-while, for, for-each

```
1 int i = 0;
2 while (i < 3) { System.out.println(i); i++; }
3
4 int j = 0;
5 do { System.out.println(j); j++; } while (j < 3);
6
7 for (int k = 0; k < 3; k++) { System.out.println(k); }
8
9 int[] arr = {1, 2, 3};
0 for (int v : arr) { System.out.println(v); } // for-each
```

Навигация по лекции

- 1 Обзор языков высокого уровня
- 2 Java сегодня: роль языка и JVM
- 3 Инструменты: JDK, IntelliJ IDEA, CLI, JShell
- 4 Синтаксис Java: база до ООП
- 5 Ввод/вывод и обработка ошибок**
- 6 Массивы и методы (минимум до ООП)
- 7 Итоги

Стандартный ввод: Scanner

```
1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         try (Scanner sc = new Scanner(System.in)) {
6             System.out.print("Введите целое число : ");
7             String line = sc.nextLine(); // читаем строку полностью
8             int x = Integer.parseInt(line.trim()); // парсим явно
9             System.out.println("Квадрат: " + Math.multiplyExact(x, x));
10        } catch (NumberFormatException e) {
11            System.out.println("Ошибка: введите целое число ");
12        } catch (ArithmeticException e) {
13            System.out.println("Переполнение при вычислениях ");
14        }
15    }
16 }
```

Почему так

Файлы: чтение/запись в одну строку (NIO)

```
1 import java.nio.file.*; import java.io.IOException;
2 import java.util.*;
3
4 public class Main {
5     public static void main(String[] args) throws IOException {
6         Path p = Path.of("data.txt");
7         Files.writeString(p, "Hello\n", java.nio.charset.StandardCharsets.UTF_8);
8         String content = Files.readString(p);
9         System.out.println(content);
10    }
11 }
```

Навигация по лекции

- 1 Обзор языков высокого уровня
- 2 Java сегодня: роль языка и JVM
- 3 Инструменты: JDK, IntelliJ IDEA, CLI, JShell
- 4 Синтаксис Java: база до ООП
- 5 Ввод/вывод и обработка ошибок
- 6 Массивы и методы (минимум до ООП)**
- 7 Итоги

Массивы: создание и базовые операции

```
1 int[] a = {3, 1, 4};  
2 int[] b = new int[5];           // заполнен нулями  
3 System.out.println(a.length);  
4 Arrays.sort(a);                 // [1,3,4]  
5 int pos = Arrays.binarySearch(a, 3); // позиция 3
```

Методы (статические функции)

```
1 public class Main {  
2     static int sum(int x, int y) {  
3         return Math.addExact(x, y);  
4     }  
5     public static void main(String[] args) {  
6         System.out.println(sum(2, 3));  
7     }  
8 }
```

Важно

- Пока обходимся static методами; принципы ООП — на следующем занятии
- Перегрузка методов разрешена (одинаковое имя, разные параметры)

Навигация по лекции

- 1 Обзор языков высокого уровня
- 2 Java сегодня: роль языка и JVM
- 3 Инструменты: JDK, IntelliJ IDEA, CLI, JShell
- 4 Синтаксис Java: база до ООП
- 5 Ввод/вывод и обработка ошибок
- 6 Массивы и методы (минимум до ООП)
- 7 Итоги**

Что нужно запомнить

- Java = язык + платформа JVM; OpenJDK — референсная реализация
- Актуальные LTS: 17, 21, (план) 25; для курса — JDK 21
- IntelliJ IDEA Community достаточно для старта; JShell для экспресс-проверок
- База синтаксиса: типы, операторы, if/switch, циклы, массивы, методы
- Ввод/вывод: Scanner + явный парсинг;
Files.readString/writeString