Программирование
1 семестр

ИТМО

# Введение

# Преподаватели

- Гаврилов Антон Валерьевич

  avgavrilov@itmo.ru
  - лекции
  - рубежка
  - зачет / экзамен
- Преподаватели практики (у всех свои)
  - лабораторные (проверка, защита)

- Цель курса:
  - научиться писать программы на языке Java
  - изучить принципы объектно-ориентированного программирования
  - научиться использовать классы стандартной библиотеки,

# Что будем изучать

ИТМО

- 1 семестр:
  - базовый синтаксис Java
  - основы ООП
- 2 семестр
  - стандартная библиотека (Java API)
  - коллекции, ввод-вывод, потоки данных, базы данных, сеть, многопоточность, графический интерфейс

# Темы лекций (1 семестр)

1) Введение. Типы данных, переменные и выражения
2) Ветвление, циклы, массивы и подпрограммы
3) Основы ООП, объекты и классы. Поля и методы
4) Инкапсуляция, конструкторы, наследование
5) Полиморфизм, интерфейсы
6) Объектно-ориентированный дизайн, стиль кодирования
7) Ошибки, исключения и отладка программ
8) Пакеты и модули, контрольная работа

# Структура курса и баллы

- 1 семестр
  - 8 лекций (16 часов)
  - 8 практических занятий (32 часа)
- Нужно выполнить и сдать
  - 3 (4) лабораторных (36 - 60 баллов)
  - 1 (2) контрольных (12 - 20 баллов)

  **ОБЯЗАТЕЛЬНО!**
  - 1 зачет (12 - 20 баллов)
- 60 и более баллов - предмет сдан

# Где искать материалы

- Официальный сайт Java
  - https://docs.oracle.com/java/
  - документация, примеры
- Книги
- Сайты

- Видео

# Где искать задания и задавать вопросы

- Сайт https://se.ifmo.ru,
  - раздел Программирование
  - слайды, методички
  - задания, журнал

- Группа в Телеграм

# Что потребуется

- JDK 17 или 21 или 25 (скоро)
- Текстовый редактор
- SSH (для доступа к helios)
- IDE
  - Intellij IDEA
  - NetBeans
  - Eclipse
  - VS Code

- Сервер helios.cs.ifmo.ru
  - JDK 17 и 21
  - Текстовый редактор
  - SSH
- Лабы сдаются тут!

# Почему Java?

ИТМО

- Популярность и востребованность
- Высокая надежность программ
- Практика хорошего кода
- Хорошая производительность
- Переносимость программ
- Корпоративные решения

# Что и как делать

- Посещать занятия, спрашивать, если не понятно
- Писать код самостоятельно
- Разбираться в своем коде, не копировать бездумно
- Осознавать свои действия, понимать, зачем это нужно
- Искать информацию, читать документацию
- Писать программы небольшими этапами
- Не доверять нейросетям

# Программы и программирование

- Программирование - создание или написание программ
- Программа - набор инструкций для выполнения задачи исполнителем по определенному алгоритму
- Исполнитель может исполнять инструкции, входящие в его систему команд
- Программист пишет программы на языке программирования
- Язык программирования и система команд могут не совпадать
- Одну и ту же программу можно написать по-разному

# Языки программирования

ИТМО

- Высокого уровня
  - Ближе к человеческим
  - Выше абстракция
  - Просто писать программы
  - Не зависят от "железа"
  - Сложно исполнять
- Низкого уровня
  - Ближе к машинному коду
  - Ниже абстракция
  - Сложно писать программы
  - Зависят от "железа"
  - Просто исполнять

# Трансляция программы

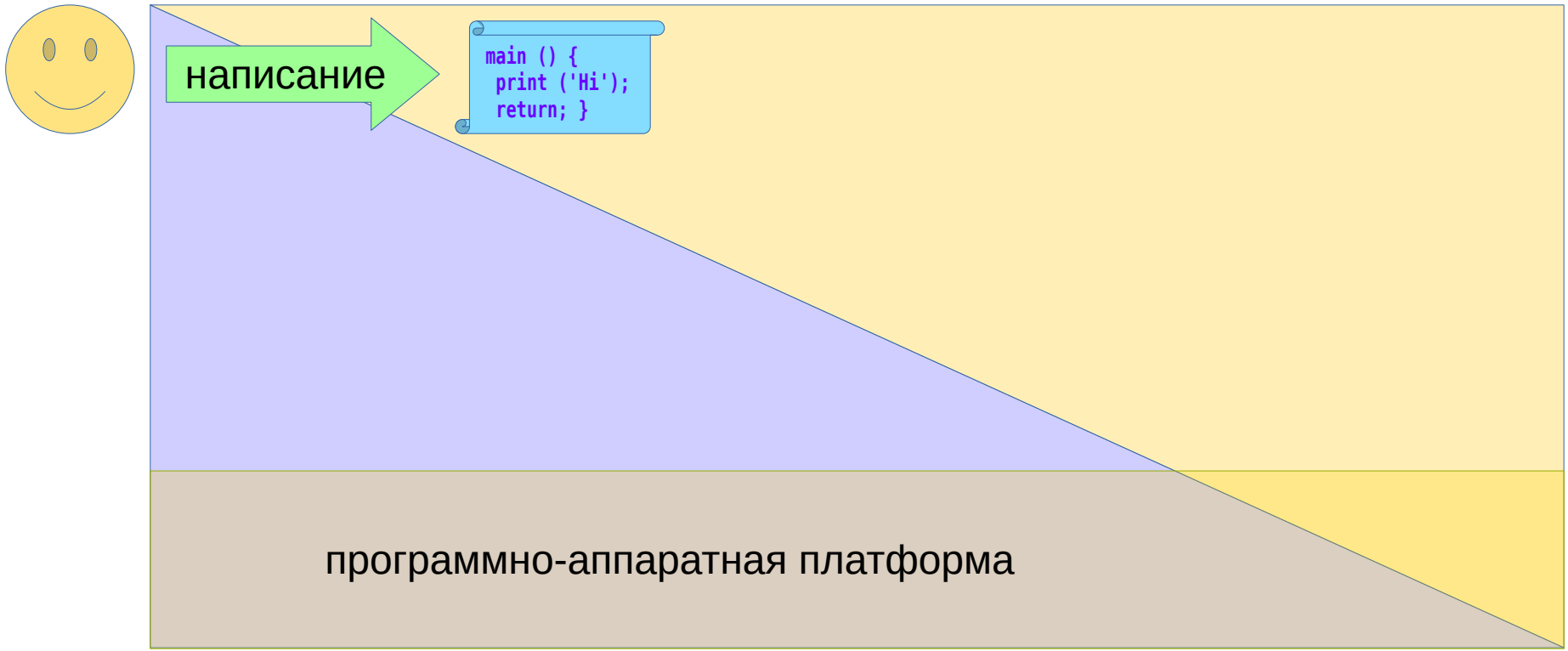
ИТМО

- На входе - исходный код на языке программирования
- Разбор текста, построение синтаксического дерева
- Распределение памяти, создание таблицы символов, ...

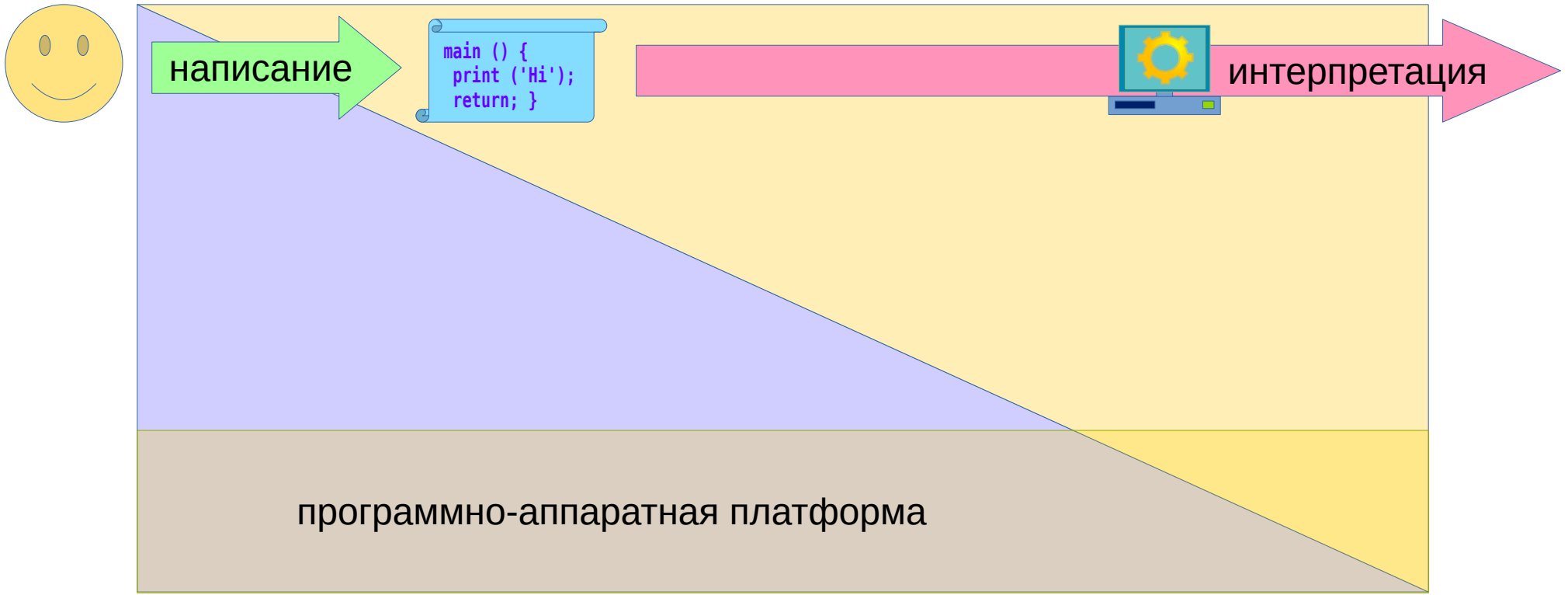
- Компилятор
  - Создает машинный код
  - Под конкретную платформу
  - Не нужен при исполнении

- Интерпретатор
  - Исполняет программу
  - Не зависит от платформы
  - Занимает память

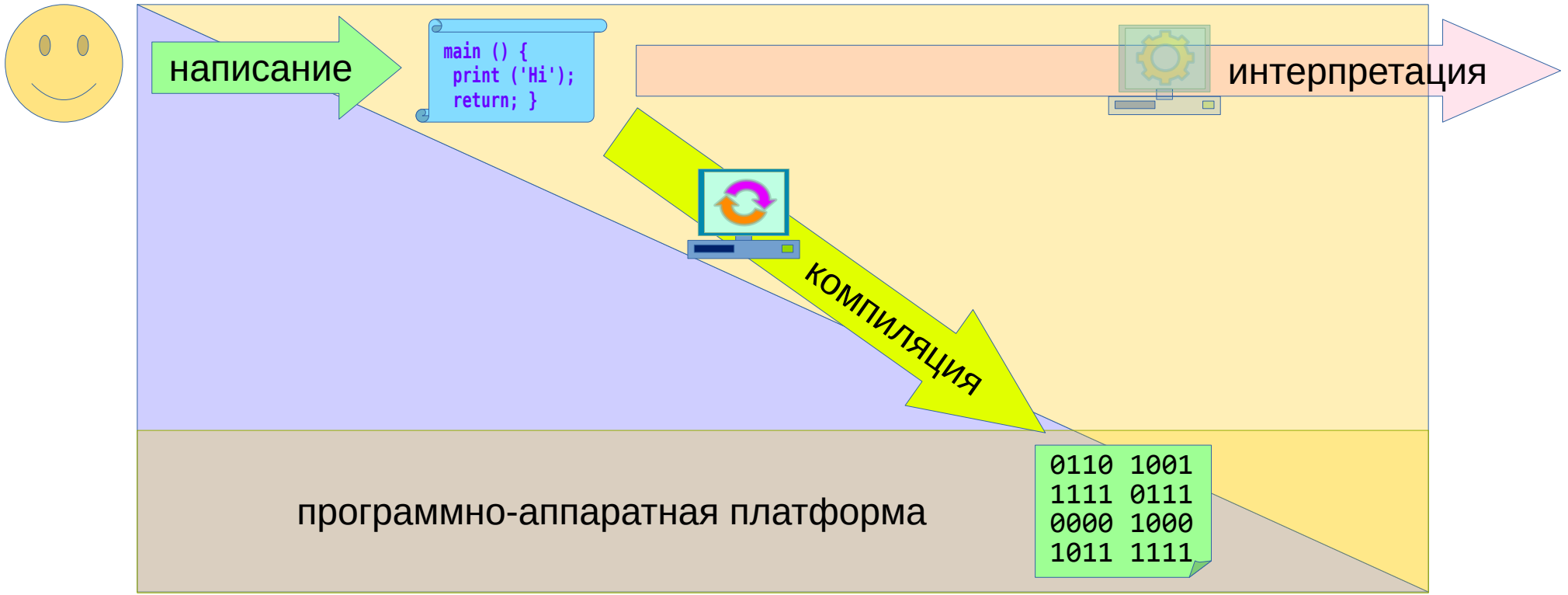
# Написание и исполнение программ



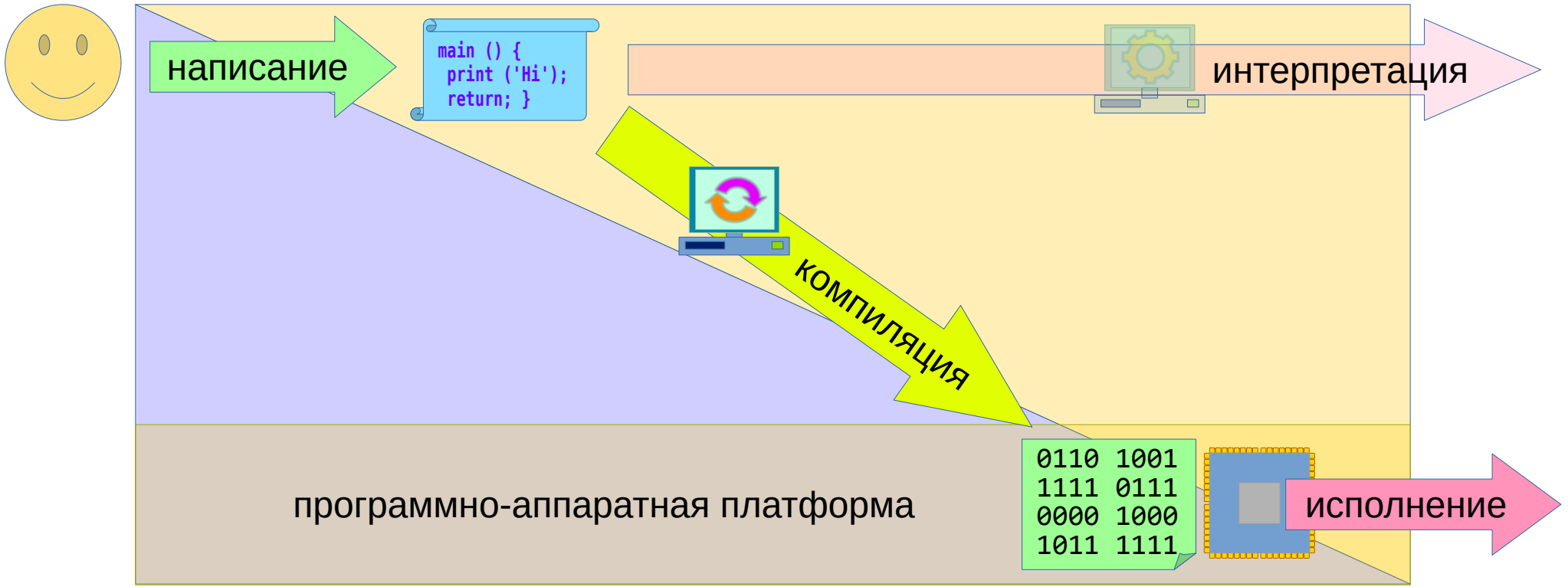
# Написание и исполнение программ



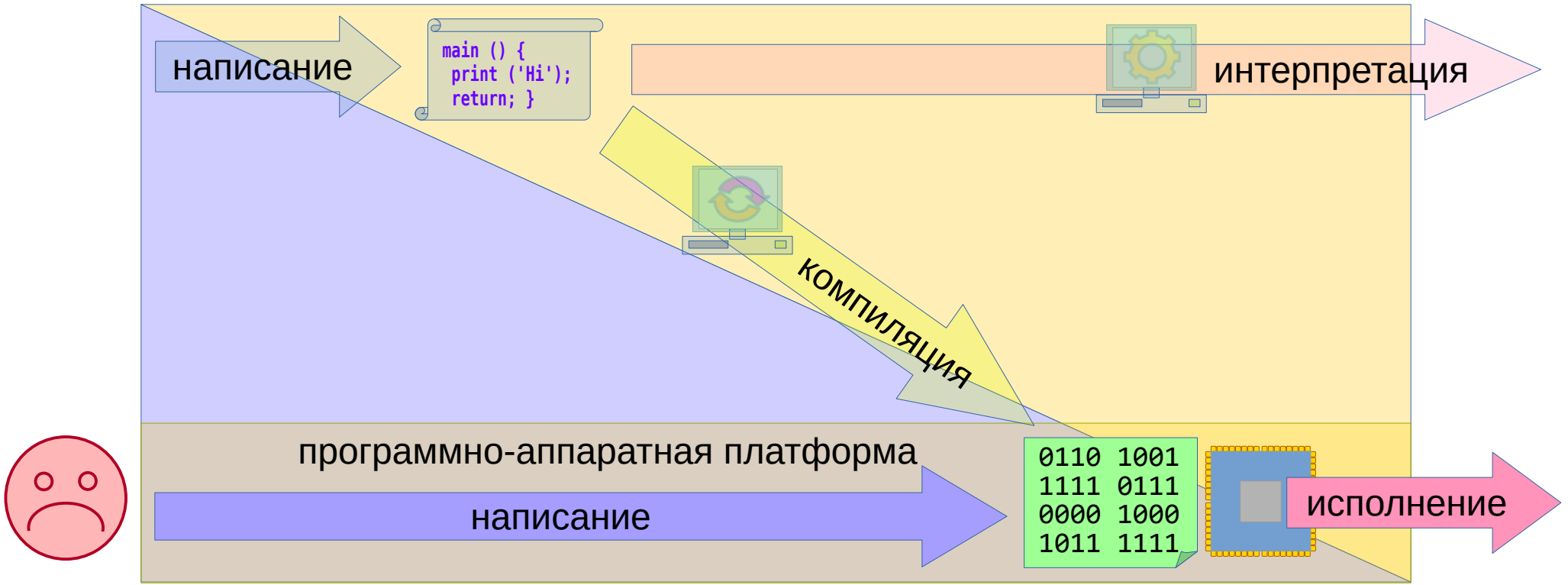
# Написание и исполнение программ



# Написание и исполнение программ

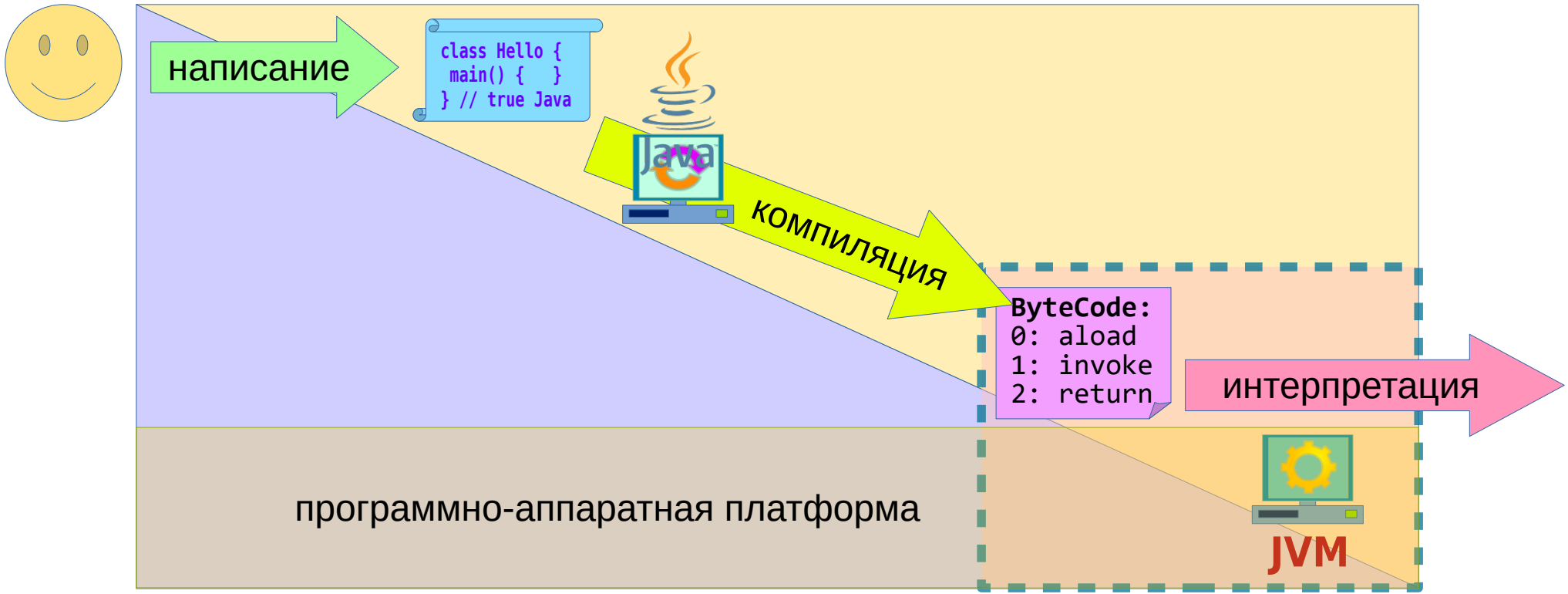


# Написание и исполнение программ



# Написание и исполнение программ

# Написание и исполнение программ

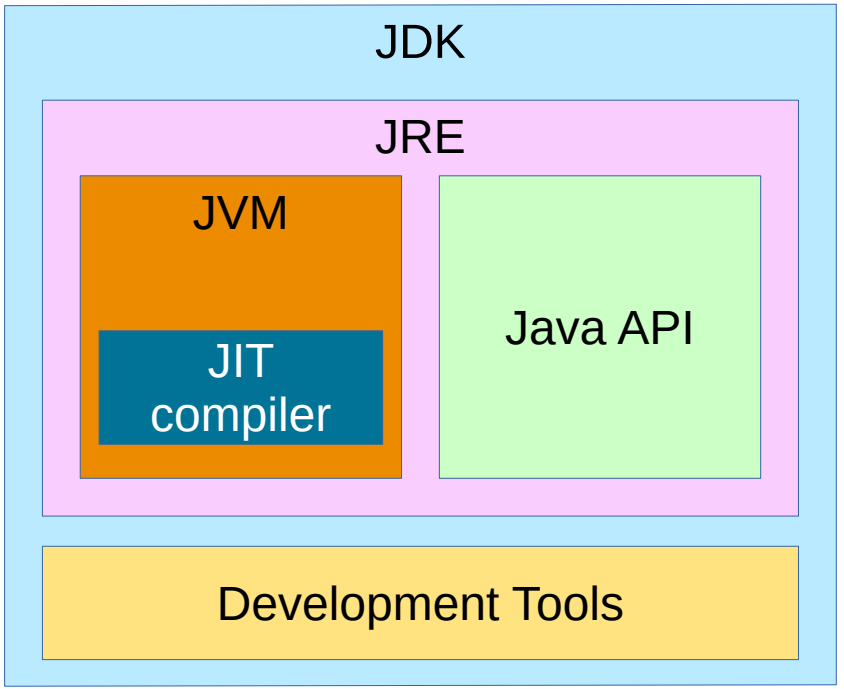


# Основные свойства языка Java

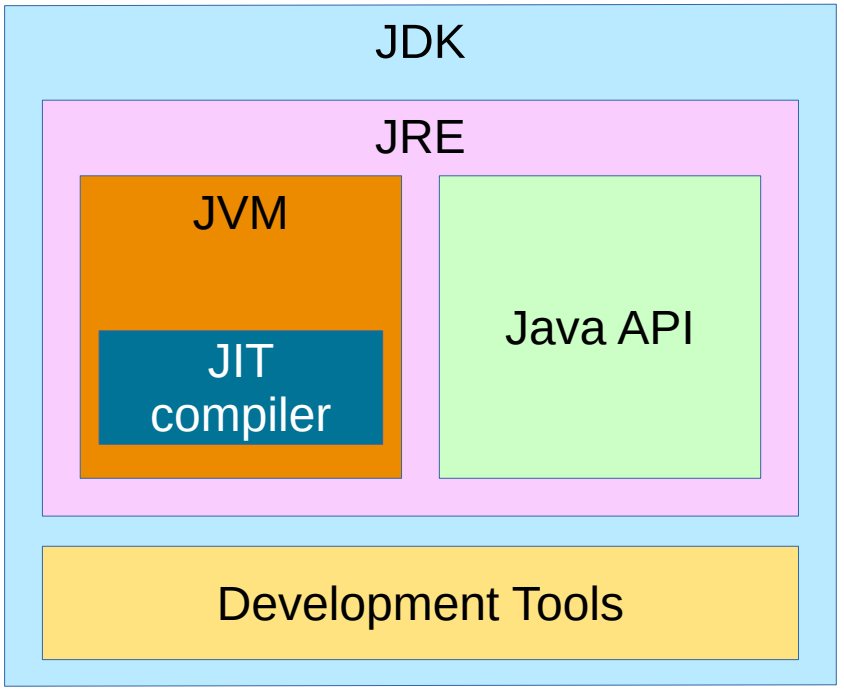
ИТМО

- Кросс-платформенность
  - Программа транслируется в байт-код
  - Формат байт-кода стандартный для всех платформ
  - Java-машина интерпретирует байт-код
  - Библиотеки распространяются в виде байт-кода
- Программа исполняется на разных платформах без перекомпиляции (если на платформе работает JVM)

# Java-платформа


# Java-платформа и другие языки








# Краткая история и версии Java

ИТМО

- 1996 JDK 1.0.2 - Sun Microsystems
- 1997 JDK 1.1
- 1998 Java 2 SE 1.2
- 2000 Java 2 SE 1.3
- 2002 Java 2 SE 1.4
- 2004 Java SE 5.0 (1.5)
- 2006 Java SE 6 (1.6)
- 2011 Java SE7 (1.7) - Oracle
- 2014 Java SE 8 (1.8) - LTS

| | Март | Сентябрь |
|---|---|---|
| 2017 | | Java SE 9 |
| 2018 | Java SE 10 | Java SE 11 |
| 2019 | Java SE 12 | Java SE 13 |
| 2020 | Java SE 14 | Java SE 15 |
| 2021 | Java SE 16 | Java SE 17 |
| 2022 | Java SE 18 | Java SE 19 |
| 2023 | Java SE 20 | Java SE 21 |
| 2024 | Java SE 22 | Java SE 23 |
| 2025 | Java SE 24 | Java SE 25 |

# Hello World

ИТМО

```
/* Первая программа
   файл: Hello.java
*/

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

# Hello World

ИТМО

**Исходный код программы**

```
/* Первая программа
   файл: Hello.java
*/

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

Расширение .java

Имена должны совпадать

Размер имеет значение

# Hello World

ИТМО

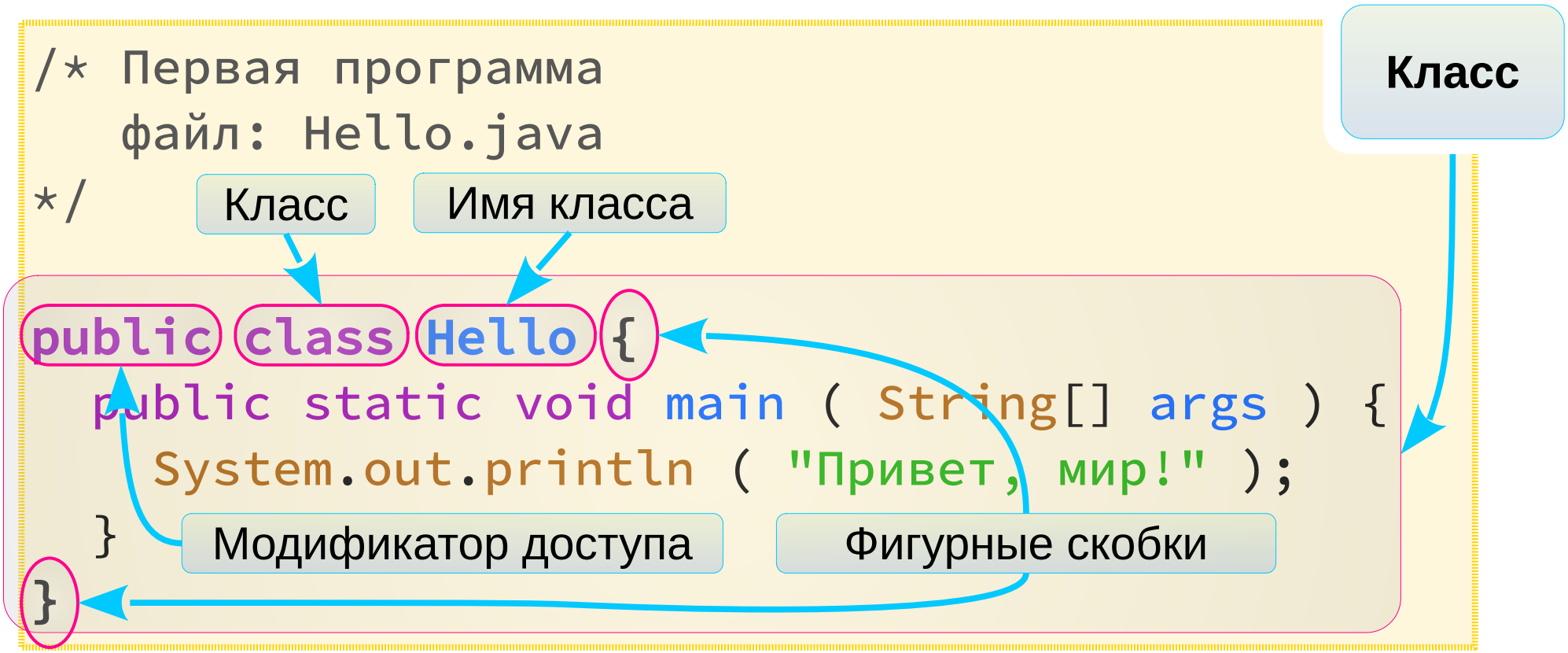
```
/* Первая программа
   файл: Hello.java
*/

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

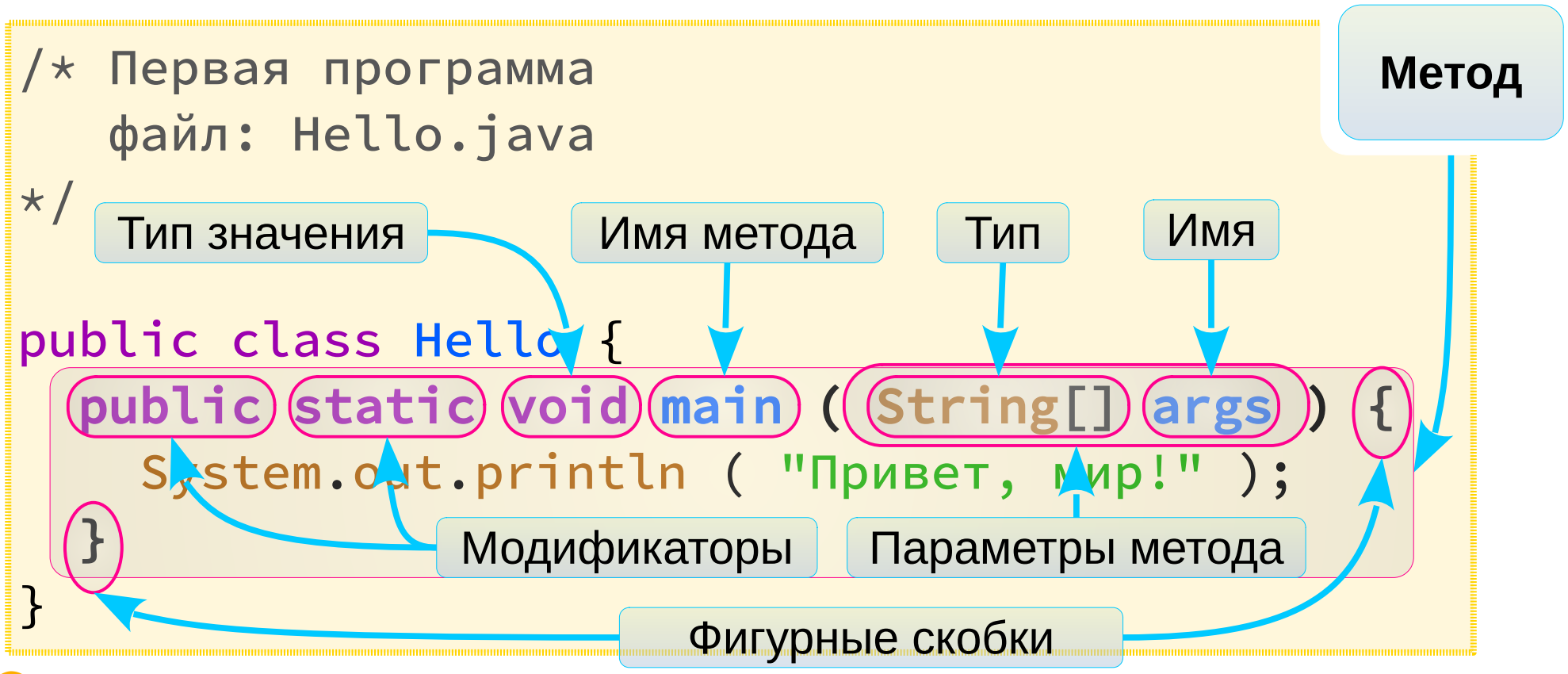
**Комментарии**

| | |
|---|---|
| `//` | однострочный |
| `/* */` | блочный |
| `/** */` | документирующий |

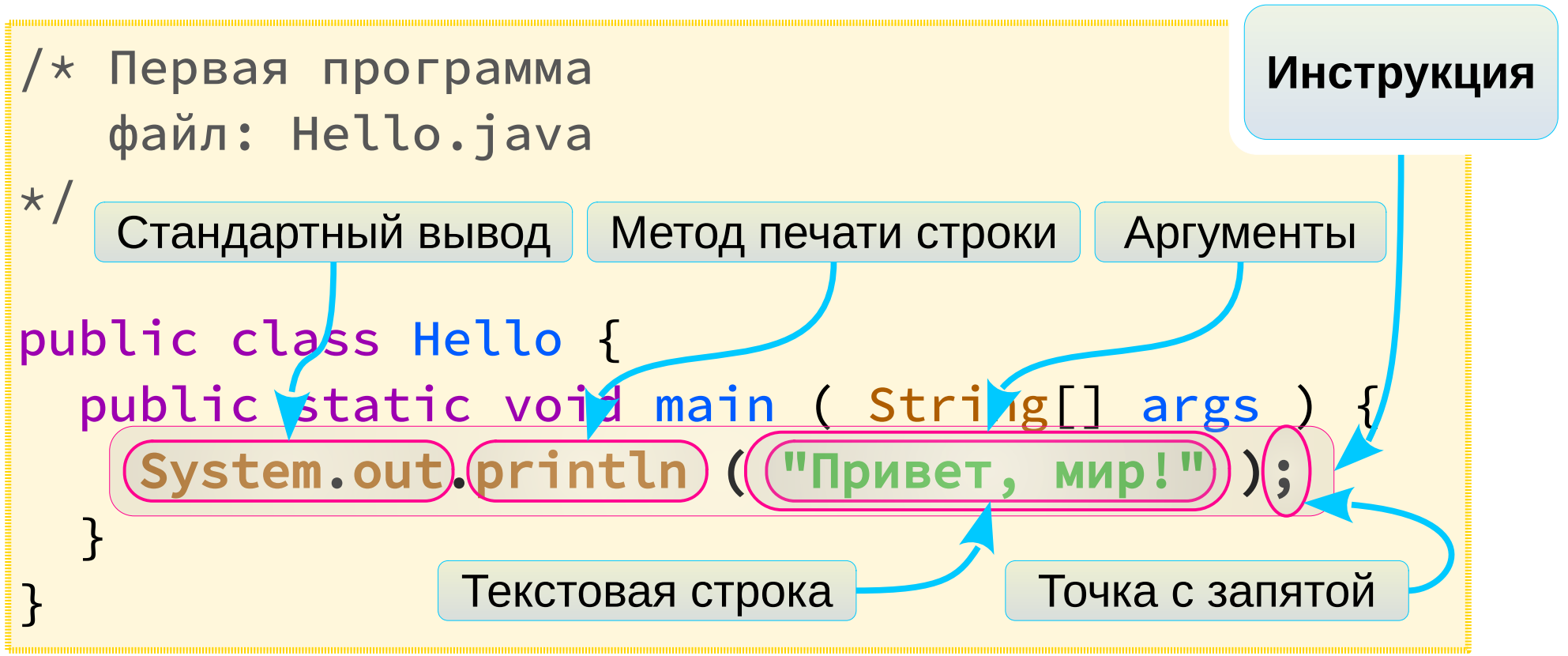
# Hello World


Класс
/* Первая программа
файл: Hello.java
*/
Класс
Имя класса
public class Hello {
public static void main ( String[] args ) {
System.out.println ( "Привет, мир!" );
}
}
Модификатор доступа
Фигурные скобки

# Hello World


/* Первая программа
файл: Hello.java
*/
public class Hello {
public static void main ( String[] args ) {
System.out.println ( "Привет, мир!" );
}
}
Метод
Тип значения
Имя метода
Тип
Имя
Модификаторы
Параметры метода
Фигурные скобки

# Hello World


Инструкция
/* Первая программа
файл: Hello.java
*/
Стандартный вывод
Метод печати строки
Аргументы
public class Hello {
public static void main ( String[] args ) {
System.out.println ("Привет, мир!");
}
}
Текстовая строка
Точка с запятой

# Hello World

ИТМО

```
/* Первая программа
   файл: Hello.java
*/

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

# Compact source (Java 25 - сентябрь 2025 г.)

ИТМО

```
/* Первая программа
   файл: Hello.java
*/

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

# Compact source (Java 25 - сентябрь 2025 г.)

```
/* Первая программа
   файл: Hello.java
*/


                void main (                    ) {
            IO.println ( "Привет, мир!" );
  }
```

# Compact source (Java 25 - сентябрь 2025 г.)

ИТМО

```
/* Первая программа
   файл: Hello.java
*/


  void main ( ) {
    IO.println ( "Привет, мир!" );
  }
```

# Компиляция

ИТМО

- Исходный код (Hello.java)

```
javac Hello.java
```

- Байт-код (Hello.class)
- Если нет сообщений - все хорошо!

# Запуск программы

- Виртуальная машина Java (JVM)
  - Запускает метод main класса Hello
- Байт-код (Hello.class)

```
java Hello .class
Привет, мир!
```

# Стартовый метод программы

ИТМО

- Точка входа в приложение (entry point)

```
public static void main (String[] args) { }
```

  - метод должен называться main
  - должен иметь модификаторы public static
  - должен возвращать значение типа void (ничего)
  - должен иметь один параметр типа String[]
  - имя параметра значения не имеет

# Запуск программы (однофайловой)

ИТМО

- Все классы в одном файле
- В первом классе есть метод main
- Упрощенный способ запуска (без явной компиляции)
- Файл .class не создается

```
java Hello.java
```

# Упаковка в jar-архив

- Для передачи приложений
- Формат - аналогичен zip
- Содержит классы приложения (.class)
- Содержит файл META-INF/MANIFEST.MF
  - Main-Class: Hello

# Упаковка в jar-архив

ИТМО

- Набор классов приложения (Hello.class + *.class)
- Архиватор jar создает архив
- jar-архив (hello.jar)

```
jar -c -f hello.jar -e Hello *.class
```

что положить в архив

создать

имя архива

точка входа для манифеста

# Упаковка в jar-архив

- Набор классов приложения (Hello.class + *.class)
- Архиватор jar создает архив
- jar-архив (hello.jar)

```
jar -c -f hello.jar -e Hello *.class
java -jar hello.jar
Привет, мир!
```

запускаем метод main()
Main-Class-а из манифеста

# Типы данных

ИТМО

- Тип данных
  - Набор допустимых значений
  - Набор возможных действий
- Система типов в ЯП
  - Статическая или динамическая
  - Сильная (строгая) или слабая
  - Явная или неявная

# Типы данных

ИТМО

- Тип данных
  - Набор допустимых значений
  - Набор возможных действий
- Система типов в ЯП
  - Статическая или динамическая
  - Сильная (строгая) или слабая
  - Явная или неявная

Java

# Система типов Java

ИТМО

- Примитивные
  - Целые (byte, short, int, long)
  - Символьный (char)
  - Вещественные (float, double)
  - Логический (boolean)
- Ссылочные (объекты)
  - Массивы
  - Классы
    - Строки
    - Перечисления
    - Записи
  - Интерфейсы
    - Аннотации

# Примитивные типы

ИТМО

| тип | байт | бит | диапазон | default |
|---|---|---|---|---|
| byte | 1 | 8 | -127 ... +128 | 0 |
| short | 2 | 16 | -32768 ... +32767 | |
| int | 4 | 32 | -2 147 483 648 ... +2 147 483 647 | |
| long | 8 | 64 | -9 223 372 036 854 775 808 ... +9 223 372 036 854 775 808 | 0L |
| float | 4 | 8+24 | $\pm 1.4 \cdot 10^{-45}$ ... $\pm 3.4028235 \cdot 10^{38}$ | 0.0F |
| double | 8 | 11+53 | $\pm 4.9 \cdot 10^{-324}$ ... $\pm 1.7976931348623157 \cdot 10^{308}$ | 0.0 |
| char | 2 | 16 | 0 ... 65535 | '\u0000' |
| boolean | 1* | 1 ... 32 | false, true | false |

# Литералы

ИТМО

- числа
  - целые
    - десятичные - `1_999_000`
    - восьмиричные - `017`
    - 16-ричные - `0xf4e0L`
    - двоичные - `0b0101_1011`
  - с плавающей точкой

    `3.1415926F`

    `2.9981e25`
- символы
  - одиночные - `'ъ'`, `'\t'`
  - строки - `"Java\u2122"`
  - блоки - `"""`

    `many`

    `lines"""`
- логические (false != 0)
  - `false`
  - `true`

# Переменные

ИТМО

- Переменная
  - хранит значение
  - имеет тип и имя
- Объявление (declaration)
- Присваивание (assignment)
- Инициализация
- Получение значения

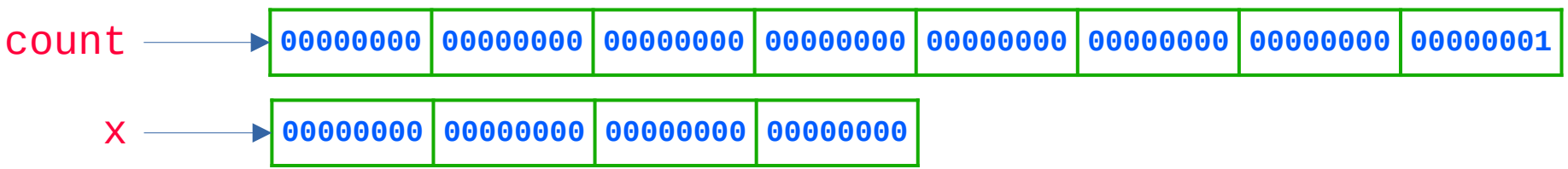
*<type> <name> = <value>*

```
int x;
x = 31;
long count = 1L;
System.out.print(x);
```

# Для добавления заглавия щёлкните мышью

- Примитивные типы
  - Выделяется место (X байт)
  - Имя связывается с областью хранения значения

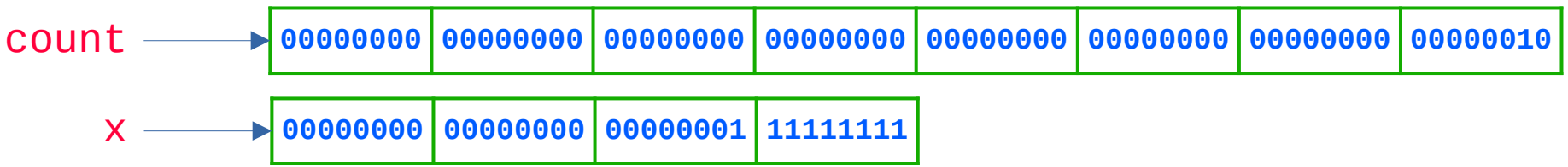
```
long count = 1L;
int x;    // x = 0
```



# Присваивание

- Старое значение заменяется новым

```
count = 2L;
x = 511;
```

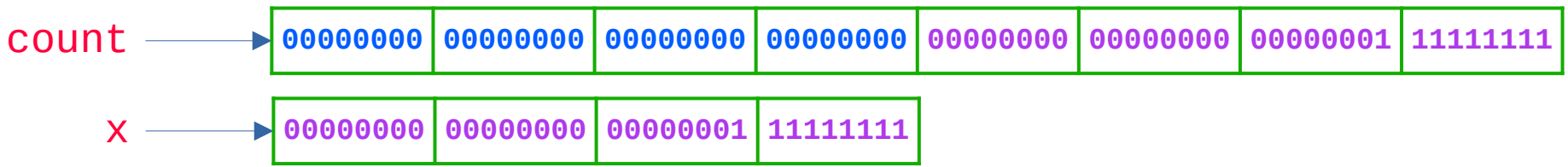


# Преобразование примитивных типов

ИТМО

- Разрешено расширяющее преобразование
- Запрещено неявное сужающее преобразование

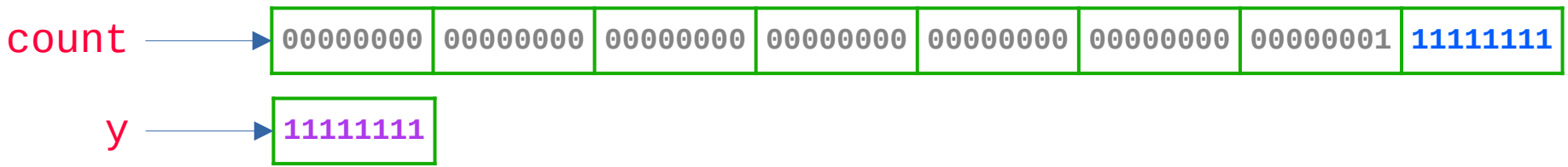
```
x = count;
count = x; // 511
```




# Преобразование примитивных типов

- Разрешено расширяющее преобразование
- Разрешено явное сужающее преобразование

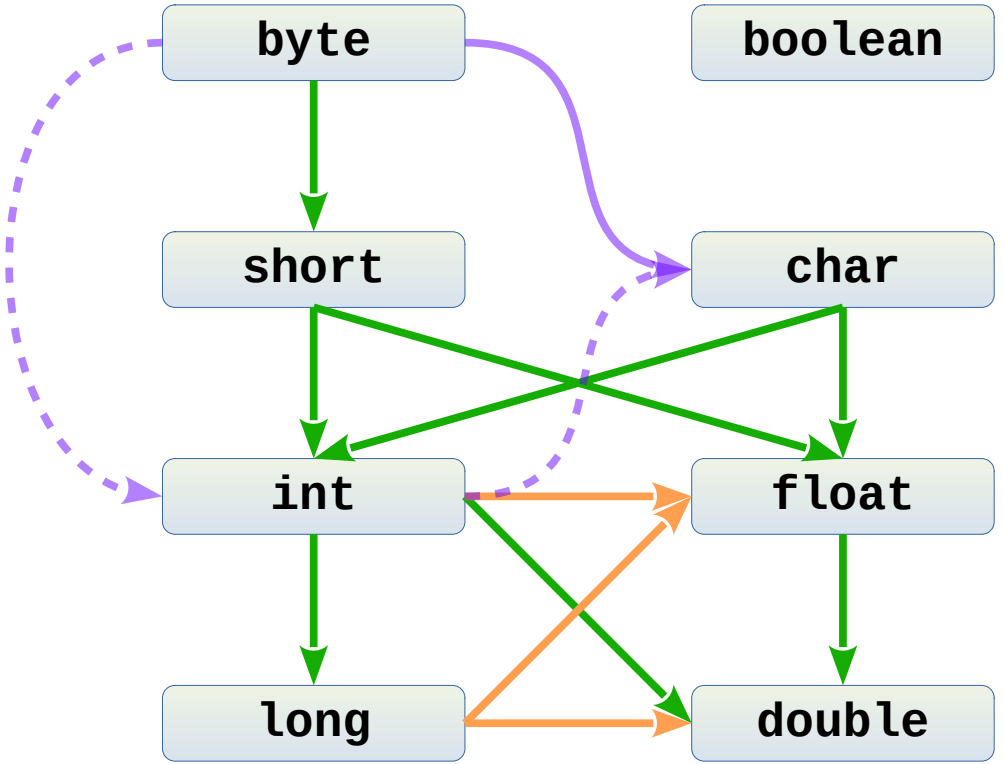
```
byte y = (byte) count; // -1
```




# Преобразование примитивных типов

| boolean | byte | char | short | int | long | float | double |
|---|---|---|---|---|---|---|---|
| byte | 1 | →int→ | | | | | |
| char | | 2 | | | | | |
| short | | | 2 | | | | |
| int | | | | 4 | | | |
| long | | | | | 8 | | |
| float | | | | | | 4 | |
| double | | | | | | | 8 |

# Преобразование примитивных типов




# Константы (финальные переменные)

- final - значение переменной можно присвоить один раз
  - Инициализация
  - Объявление и однократное присваивание

```
final int a = 50; // OK

a = 100; // ошибка!

final int b;

b = 50; // OK
```

# Операторы

ИТМО

- Арифметические

  `+  -  *  /  %`

- Инкремент и декремент

  `++  --`

- Побитовые и сдвиги

  `~  &  |  ^  <<  >>  >>>`

- Сравнения

  `<  >  <=  >=  ==  !=`

- Присваивания

  `=  +=  -=  *=  /=  %=`

- Логические

  `!  &&  ||`

# Выражения

ИТМО

- Оба операнда приводятся к одинаковому типу
- (`byte`, `char`, `short`) → `int` → `long` → `float` → `double`
- Тип результата обычно такой же как тип операнда

```
byte b = 2;
byte c = 3;
byte d = b + c; // ошибка
byte e = (byte) (b + c); // OK
int result = 15 / 4;  // 3
double dr = 15 / 4.0; // 3.75
```

```
float f = 4.0 * 15; // ошибка
float f = 4.0; // ошибка
float f = 4.0f // OK
f *= 11f; // 44.0f
f /= 10;
f += 4.8f; // 9.200001
```

# Приоритеты операторов

| | |
|---|---|
| 1 | `( )` |
| 2 | `x++  x--` |
| 3 | `++x  --x  +x  -x  ~x  !x` |
| 4 | `(int) x` |
| 5 | `x*y  x/y  x%y` |
| 6 | `x+y  x-y` |
| 7 | `x<<y  x>>y  x>>>y` |

| | |
|---|---|
| 8 | `x<y  x<=y  x>=y  x>y` |
| 9 | `x==y  x!=y` |
| 10 | `x&y  x\|y  x^y` |
| 11 | `x&&y` |
| 12 | `x\|\|y` |
| 13 | `c ? x : y` |
| 14 | `x=y  x+=y  x*=y  x>>=y` |

# Инкремент и декремент

ИТМО

- `x++  x--  5++`
- Постинкремент и постдекремент
- Операция после значения
- `y = x++;`
- `y = x; x = x + 1;`

```
x++++++x = x++ + ++x
```

- `++x  --x  ++1`
- Преинкремент и предекремент
- Операция до значения
- `y = --x;`
- `x = x - 1; y = x;`

# Двоичное представление типов

- Целые числа
  - без знака
  - со знаком
- Вещественные числа
- Символы
  - ASCII
  - Unicode

- $0 \ldots 2^N - 1$
- Прямой двоичный код

| | | | | | |
|---|---|---|---|---|---|
| | **0** | **0** | **0** | **0** | 0 |
| | **1** | **1** | **1** | **1** | 15 |
| + | **0** | **0** | **0** | **1** | 1 |
| 1 | **0** | **0** | **0** | **0** | 0 |

# Двоичное представление типов

- Целые числа
  - без знака
  - со знаком
- Вещественные числа
- Символы
  - ASCII
  - Unicode

- $-2^{N-1} \ldots 2^{N-1} - 1$
- Дополнительный код

| | | | | | |
|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 0 | - 8 |
| | 1 | 1 | 1 | 1 | - 1 |
| + | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |

# Двоичное представление типов

- Целые числа
  - без знака
  - со знаком
- Вещественные числа
- Символы
  - ASCII
  - Unicode

- $\pm M \cdot 2^E$, $M = 1.0000 \ldots 1.1111$
- IEEE 754

| 0 | E | E | E | M | M | M | M |
|---|---|---|---|---|---|---|---|

| E | M | значение |
|---|---|---|
| 000 | 0000 | ± 0 |
| 111 | 0000 | ± ∞ |
| 111 | ≠ 0000 | NaN |

# Арифметические операции

- Целые типы
  - Результат точный
  - Возможно переполнение и смена знака

2 147 483 647 + 1 = - 2 147 483 648

  - Делить на 0 нельзя

- Типы с плавающей точкой
  - Дроби не всегда точные
  - Возможна потеря точности

1.0 + 1e-70 = 1

  - Есть бесконечность

1.0 / 0,0 = Infinity

  - Есть не число

0.0 / 0.0 = NaN

# Двоичное представление типов

- Целые числа
  - без знака
  - со знаком
- Вещественные числа
- Символы
  - ASCII
  - Unicode

- Код символа 0 ... 255
  - Знаки препинания 32 ... 64
  - Цифры 48 ... 57
  - Латинские буквы 65 ... 127
  - Расширение 128 ... 254
- Кодировки
  - UTF-8
  - iso8859-5, koi8-r, cp1251

# Двоичное представление типов

ИТМО

- Целые числа
  - без знака
  - со знаком
- Вещественные числа
- Символы
  - ASCII
  - Unicode

- Код символа 0 ... 65535
  - Кодировки
    - UTF-8 (Файлы, Веб)
    - UTF-16 (Java / char)

# Имена переменных

ИТМО

- Java Naming Conventions
- Имена должны быть значащими и понятными
- Имена переменных - с маленькой буквы camelCase
- Имена констант - большими буквами SNAKE_CASE
- Не использовать магические числа

```
int total = 8 * 5 * 4 * 12;
```

# Имена переменных

ИТМО

- Java Naming Conventions
- Имена должны быть значащими и понятными
- Имена переменных - с маленькой буквы camelCase
- Имена констант - большими буквами SNAKE_CASE
- Не использовать магические числа

```
final int HOURS=8, DAYS=5, WEEKS=4, MONTHS=12;
int workingDays = HOURS * DAYS * WEEKS * MONTHS;
```

# Математические функции

ИТМО

- Класс java.lang.Math - стандартная библиотека
- Константы
  - double Math.PI, double Math.E
- Функции (параметр - double, результат - double)
  - Math.sin(), Math.cos(), Math.tan(), Math.asin(), Math.sinh(),
  - Math.log(), Math.log10(), Math.exp(), Math.pow(), Math.sqrt(),
  - Math.abs(), Math.round(), Math.min(), Math.max()

# Конкатенация строк

ИТМО

- Оператор конкатенации (склеивания) строк: +
- При конкатенации операнды приводятся к строке

```
"Hello" + " " + "world"  // Hello world

"Hello" + 5 + 10  // Hello510

"Hello" + (5 + 10) // Hello15

10 + 5 + "Hello" + 5 + 10 // 15Hello510
```

# Методы печати в стандартный вывод

ИТМО

- `System.out`
- `.println() / .print()`
- `.printf() / .format("x = %4d, y = %02.1f%n", x, y)`
  - `%05d` - десятичное целое число с 5 знаками, в начале - нули
  - `%8.4f` - десятичная дробь, всего 8 знаков, 4 после запятой
  - `%9.2e` - научный формат, всего 9 знаков, 2 после запятой
  - `%7.3g` - либо дробь, если не лезет - научный формат

# Вопросы?