

ОСНОВНЫЕ КОНСТРУКЦИИ

Java

(обзор)

Как работает Java

```
import java.awt.*;
import java.awt.event.*;
class Party {
public void buildInvite() {
    Frame f = new Frame();
    Label l = new Label("Вечеринка у Тима");
    Button b = new Button("Ваша ставка");
    Button c = new Button("Сбросить");
    Panel p = new Panel();
    p.add(l);
    // Еще код...
}
```

Исходник

1

Наберите свой исходный код.

Сохраните его как **Party.java**.

```
File Edit Window Help Plead
% javac Party.java
```

Компилятор

2

Скомпилируйте файл **Party.java**, запустив утилиту `javac` (приложение-компилятор). Если все пройдет без ошибок, вы получите еще один файл с именем **Party.class**.

Файл **Party.class**, сгенерированный компилятором, состоит из байт-кода.

```
Method Party()
0 aload_0
1 invokespecial #1 <Method java.
lang.Object.>
4 return
Method void buildInvite()
0 new #2 <Class java.awt.Frame>
3 dup
4 invokespecial #3 <Method java.
awt.Frame.>
```

Вывод (код)

3

Скомпилированный код: файл **Party.class**.

```
File Edit Window Help Swear
% java Party
Вечеринка у Тима
[Ваша ставка] [Сбросить]
```

Виртуальные машины

4

Выполните программу, запустив виртуальную машину Java (Java Virtual Machine, или JVM) с файлом **Party.class**. JVM транслирует байт-код в такой формат, который поймет целевая платформа, и запустит вашу программу.



Структура кода в Java

**Добавляем класс
в исходный файл.**

Добавляем метод в класс.

**Добавляем выражения
в метод.**

Что творится внутри ИСХОДНОГО файла

Файл с исходным кодом (с расширением *java*) содержит определение одного *класса*. Класс — это *часть* вашей программы; совсем маленькие приложения могут обойтись единственным классом. Содержимое класса должно находиться внутри парных фигурных скобок.

```
public class Dog {
```

КЛАСС

Что происходит внутри класса

Класс может иметь один или несколько *методов*. Метод *bark* из класса Dog будет хранить инструкции о том, каким образом собака должна лаять. Методы должны быть объявлены *внутри* класса (иными словами, между его фигурными скобками).

```
public class Dog {  
    void bark() {
```

метод

Что происходит внутри метода

Инструкции для метода должны быть размещены между его фигурными скобками. Если говорить упрощенно, *код* метода — это набор выражений. Думайте о методе как о функции или процедуре.

```
public class Dog {  
    void bark() {  
        statement1;  
        statement2;  
    }  
}
```

выражения

Структура класса

Когда JVM начинает свою работу, она ищет класс, который ей передали через командную строку. Затем она ищет метод, записанный особым образом, например так:

```
public static void main (String[] args) {  
    // Здесь размещается ваш код  
}
```

Далее JVM выполняет все, что находится между фигурными скобками { } главного метода. Любая программа на языке Java содержит по меньшей мере один **класс** и как минимум один метод **main** (один для всего *приложения*, а не для каждого *класса*).

Публичный, так что все могут иметь к нему доступ.

Это класс (еще бы!).

Имя класса.

Открывающая фигурная скобка класса.

```
public class MyFirstApp {
```

Аргумент для метода. Этому методу нужно передать массив со строками под названием args.

Открывающая скобка метода.

Это мы отложим на потом.

Тип возвращаемого значения. Здесь void говорит о том, что ничего возвращаться не будет.

Имя метода.

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

```
System.out.print ("Я управляю");
```

Здесь выполняется вывод в стандартный поток (по умолчанию это командная строка).

Строка, которую вы хотите напечатать.

Каждое выражение должно заканчиваться точкой с запятой.

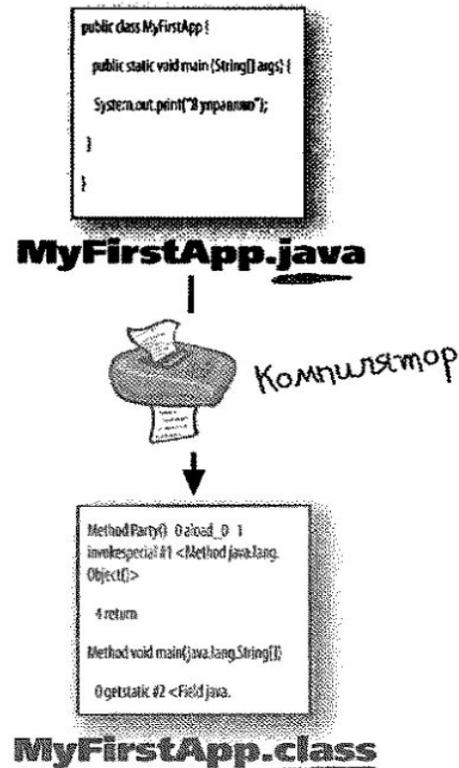
Закрывающая скобка главного метода.

Закрывающая скобка класса MyFirstApp.

```
}  
}
```

Все начинается с метода `main()`. Именно с него программа приступает к выполнению.

Неважно, насколько велика ваша программа (то есть неважно, сколько классов она содержит), всегда должен быть метод `main()`, который приведет в движение остальной код.



```
public class MyFirstApp {

    public static void main (String[] args) {
        System.out.println("Я управляю");
        System.out.println("миром");
    }

}
```

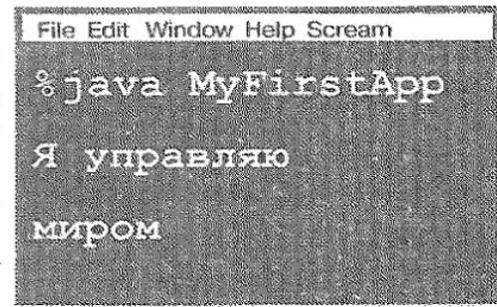
1 Сохраняем

`MyFirstApp.java`

2 Компилируем

`javac MyFirstApp.java`

3 Запускаем



Что можно сделать внутри метода?

1

Сделать что-то

Выражения: объявления, присваивания, вызовы методов и т. д.

```
int x = 3;  
String name = "Кинжал";  
x = x * 17;  
System.out.print("x равен " + x);  
double d = Math.random();  
// Это комментарий
```

Что можно сделать внутри метода?

2 Делать что-то снова и снова

Циклы: *for* и *while*

```
while (x > 12) {  
    x = x - 1;  
}
```

```
for (int x = 0; x < 10; x = x + 1) {  
    System.out.print("Теперь x равен " + x);  
}
```

Что можно сделать внутри метода?



Сделать что-то при условии

Ветвление: условия *if/else*.

```
if (x == 10) {  
    System.out.print("x должен быть равен 10");  
} else {  
    System.out.print("x не равен 10");  
}  
  
if ((x < 3) & (name.equals("Кинжал"))) {  
    System.out.println("Осторожно");  
}  
  
System.out.print("Эта строка выполняется  
в любом случае");
```

Пример цикла while

```
public class Loopy {  
    public static void main (String[] args) {  
        int x = 1;  
        System.out.println("Перед началом цикла");  
        while (x < 4) {  
            System.out.println("Внутри цикла");  
            System.out.println("Значение x равно " + x);  
            x = x + 1;  
        }  
        System.out.println("После окончания цикла");  
    }  
}
```

```
% java Loopy  
Перед началом цикла  
Внутри цикла  
Значение x равно 1  
Внутри цикла  
Значение x равно 2  
Внутри цикла  
Значение x равно 3  
После окончания цикла
```

*← Это результат
работы цикла.*

КЛЮЧЕВЫЕ МОМЕНТЫ

- Выражения заканчиваются точкой с запятой ;.
- Блоки кода задаются парными фигурными скобками { }.
- Целочисленная переменная объявляется с типом и именем: **int x**;
- Оператор **присваивания** состоит из одинарного символа: =.
- В операторе сравнения используется два таких символа: ==.
- Цикл *while* выполняет все, что находится внутри его блока (заданного фигурными скобками), пока *проверка условия* возвращает значение **true**.
- Если проверка условия вернула **false**, то блок кода внутри цикла *while* не выполнится, а JVM проследует вниз по коду и начнет выполнять выражение, находящееся сразу *после* блока с циклом.
- Условие размещается внутри скобок:

```
while (x == 4) { }
```

Синтаксические забавы

★ Каждое выражение должно заканчиваться точкой с запятой.

```
x = x + 1;
```

★ Однострочный комментарий начинается двумя слешами.

```
x = 22;
```

```
// Эта строка меня волнует
```

★ Большинство пробелов, табуляций, символов переноса строки и т. д. игнорируются.

```
x = 3 ;
```

★ Переменные объявляются с помощью имени и типа (в главе 3 вы изучите все типы, доступные в языке Java).

```
int weight;
```

```
// Тип: int, имя: weight
```

★ Классы и методы должны объявляться внутри парных фигурных скобок.

```
public void go() {  
    // Здесь будет  
    // восхитительный код  
}
```

Условное ветвление

```
class IfTest2 {  
    public static void main (String[] args) {  
        int x = 2;  
        if (x == 3) {  
            System.out.println("x должен равняться 3");  
        } else {  
            System.out.println("x не равен 3");  
        }  
        System.out.println("Эта строка выполняется в любом  
случае");  
    }  
}
```

```
% java IfTest2
```

```
x не равен 3
```

```
Эта строка выполняется в любом случае
```

← Новый результат
работы программы



Наточите свой карандаш

Текст, который нужно вывести:

```
% java DooBee  
DooBeeDooBeeDo
```

Впишите недостающий код:

```
public class DooBee {  
    public static void main (String[] args) {  
        int x = 1;  
        while (x < _____) {  
            System.out._____("Doo");  
            System.out._____("Bee");  
            x = x + 1;  
        }  
        if (x == _____) {  
            System.out.print("Do");  
        }  
    }  
}
```

1

// Создайте три набора слов для выбора. Добавляйте собственные слова!

```
String[] wordListOne = {"круглосуточный", "трех-звенный",  
"30000-футовый", "взаимный", "обоюдный выигрыш", "фронтэнд",  
"на основе веб-технологий", "проникающий", "умный", "шесть  
сигм", "метод критического пути", "динамичный"};
```

```
String[] wordListTwo = {"уполномоченный", "трудный",  
"чистый продукт", "ориентированный", "центральный",  
"распределенный", "кластеризованный", "фирменный",  
"нестандартный ум", "позиционированный", "сетевой",  
"сфокусированный", "использованный с выгодой", "выровненный",  
"нацеленный на", "общий", "совместный", "ускоренный"};
```

```
String[] wordListThree = {"процесс", "пункт разгрузки",  
"выход из положения", "тип структуры", "талант", "подход",  
"уровень завоеванного внимания", "портал", "период времени",  
"обзор", "образец", "пункт следования"};
```

2

// Вычисляем, сколько слов в каждом списке

```
int oneLength = wordListOne.length;
```

```
int twoLength = wordListTwo.length;
```

```
int threeLength = wordListThree.length;
```

3

// Генерируем три случайных числа

```
int rand1 = (int) (Math.random() * oneLength);
```

```
int rand2 = (int) (Math.random() * twoLength);
```

```
int rand3 = (int) (Math.random() * threeLength);
```

4

// Теперь строим фразу

```
String phrase = wordListOne[rand1] + " " +
```

```
wordListTwo[rand2] + " " + wordListThree[rand3];
```

5

// Выводим фразу на экран

```
System.out.println("Все, что нам нужно, — это " + phrase);
```

```
}
```

```
}
```

A

```
class Exerciselb {  
    public static void main(String [] args) {  
        int x = 1;  
        while ( x < 10 ) {  
            if ( x > 3) {  
                System.out.println("БОЛЬШОЙ ИКС");  
            }  
        }  
    }  
}
```

B

```
public static void main(String [] args) {  
    int x = 5;  
    while ( x > 1 ) {  
        x = x - 1;  
        if ( x < 3) {  
            System.out.println("маленький икс");  
        }  
    }  
}
```

C

```
class Exerciselb {  
    int x = 5;  
    while ( x > 1 ) {  
        x = x - 1;  
        if ( x < 3) {  
            System.out.println("маленький икс");  
        }  
    }  
}
```