

JAVA SE

Базовый курс. Лекция 1

Меня хорошо видно && слышно?

Напишите “+” в чат, если все ок.

Сообщите в чате, если есть проблемы!



Преподаватель

Елена Ошкина

- ведущий разработчик ПАО ВТБ
- 4 года в профессиональной разработке ПО (Java, Kotlin, Python, C, C++)
- опыт разработки образовательного проекта со студентами WPI (штат Массачусетс, США)
- стажировка в институте GSI (Дармштадт, Германия)

Образование:

- Технопарк, курс Системный архитектор (Mail.ru Group)
- МГТУ им. Н. Э. Баумана (Информатика и вычислительная техника)



Правила вебинара

1. Активно участвуем
2. Задаем вопросы в чат
3. Обсуждение ДЗ в основном будем вести в группе телеграм

Определение целей

Добавляйтесь в группу, представьтесь в чате, расскажите о своих целях, как давно интересуетесь программированием, изучали ли ранее Java?

Какие у вас ожидания от курса?



Группа в телеграм:

https://t.me/joinchat/DYQp_hCAxpTrO5j59jauUA



Чему мы научимся?



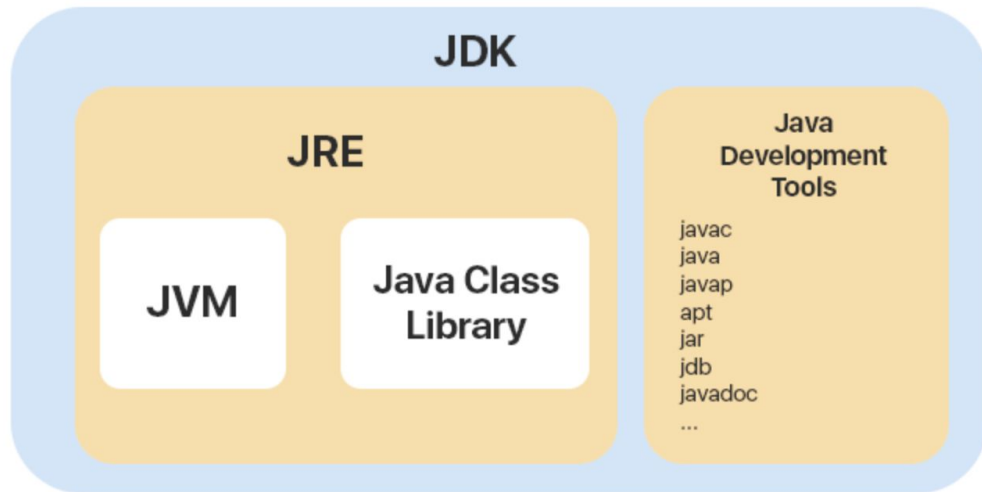
- Базовым принципами построения программы через ООП-подход
- Использовать Java Collections
- Работе с Git/GitHub
- Изучим наиболее популярные шаблоны проектирования
- Работе с инструментом сборки Maven
- Писать модульные тесты и оформлять свой код, так как это принято в профессиональном сообществе

ООП	Коллекции	Функциональное программирование
Шаблоны проектирования	Синтаксис языка	Инструменты

План лекции

1. Простые программы
2. Знакомство с переменными
3. Базовые типы данных
4. Приведение типов
5. Основные операторы
6. Использование основных операторов

Программное обеспечение



Java development tools включают в себя около 40 различных тулов: `javac` (компилятор), `java` (лаунчер для приложений), `javap` (java class file disassembler), `jdb` (java debugger) и др.

Среда выполнения JRE — это пакет всего необходимого для запуска скомпилированной Java-программы. Включает в себя виртуальную машину **JVM** и библиотеку классов Java — Java Class Library.

JVM — это программа, предназначенная для выполнения байт-кода. Первое преимущество JVM — это принцип *“Write once, run anywhere”*.

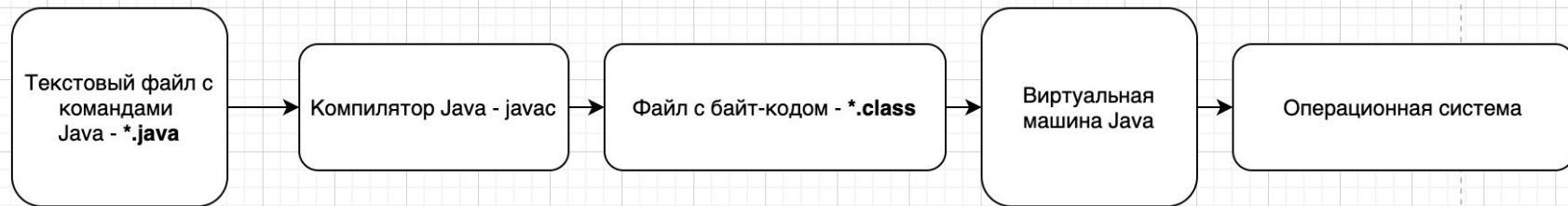
[IntelliJ IDEA](#)



[Пакетный менеджер](#)

[Инструкция по установке Java 8](#)

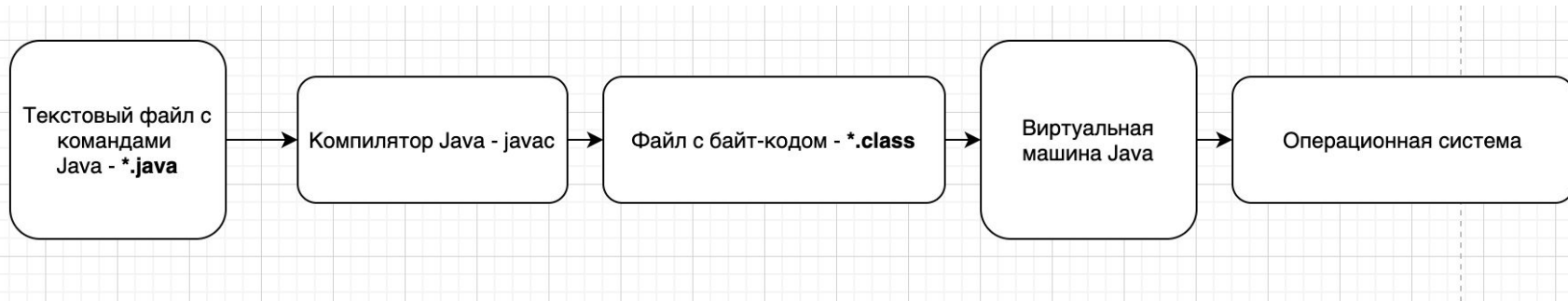
Как работает Java?



1. Чтобы запустить программу на языке Java - нужно текстовый файл с командами преобразовать в инструкции, понятные компьютеру (этот процесс выполняет компилятор)

2. Полученный файл - называется файлом с байт-кодом, чтобы компьютер понял этот код его необходимо запустить через виртуальную машину Java.

Как работает Java?



3. Виртуальная машина позволяет запустить байт-код на любых операционных системах. Программист не тратит время на адаптацию программы для работы в Windows, Linux, Mac.

4. Виртуальная машина – посредник, между вашей программой и операционной системой.



Нажимая кнопку Run в IntelliJ IDEA – среда выполняет все эти действия

Компиляция и запуск java-программы в терминале

```
MacBook-Pro-Elena:src elenaoshkina$ ls
```

```
DemoHelloWorld.java      ru
```

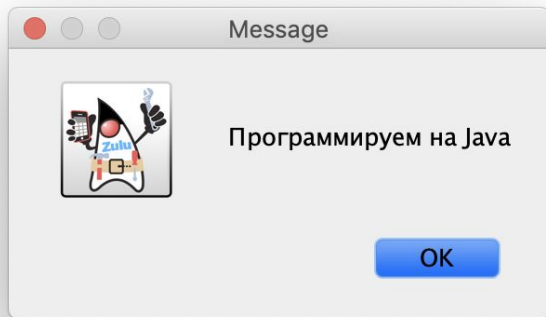
```
MacBook-Pro-Elena:src elenaoshkina$ javac DemoHelloWorld.java
```

```
MacBook-Pro-Elena:src elenaoshkina$ ls
```

```
DemoHelloWorld.class     DemoHelloWorld.java      ru
```

```
MacBook-Pro-Elena:src elenaoshkina$ java DemoHelloWorld
```

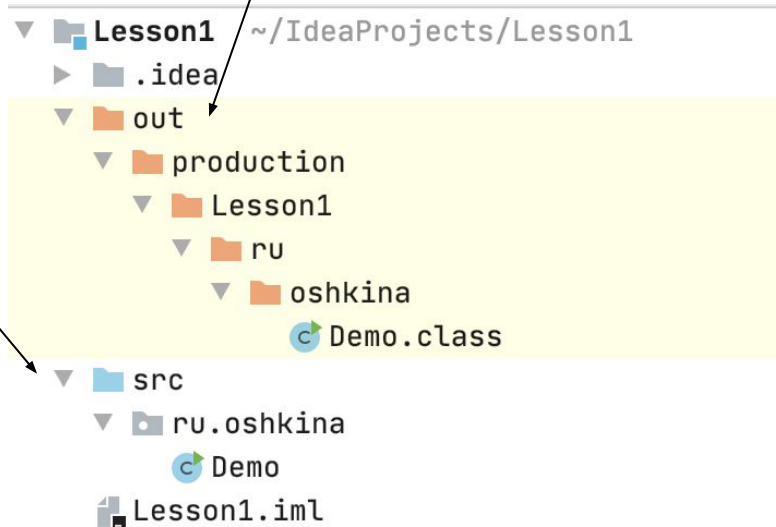
← компилируем программу



Структура проекта

Папка "src" (source - англ. "источник") - для размещения файлов с кодом наших программ.

Папка "Out" - содержит файлы с байт кодом (генерируются после запуска программы)



Принято всегда разделять исходный код программы и скомпилированные файлы (!)

Структура проекта

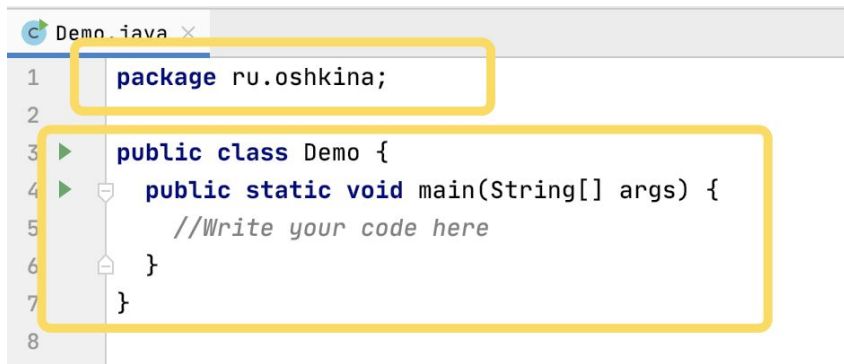
```
MacBook-Pro-Elena:Lesson1 elenaoshkina$ tree
```

```
.
├── Lesson1.iml
├── out
│   └── production
│       └── Lesson1
│           └── ru
│               └── oshkina
│                   └── Demo.class
└── src
    ├── ru
    │   └── oshkina
    │       └── Demo.java
```

Все файлы с исходным кодом на языке java должны иметь расширение *.java.

* Утилита tree - вывод директорий/файлов в виде дерева

Содержимое файла Main.java



```
1 package ru.oshkina;
2
3 public class Demo {
4     public static void main(String[] args) {
5         //Write your code here
6     }
7 }
8
```

Исходный код файла можно условно разбить на две области: заголовок и тело.

(!) В Java есть соглашение, что файл с кодом должен иметь такое же имя, как и имя блока тела кода (Main.java - public class Main).

Содержимое файла Main.java

Внутри самого класса код разбивается на подгруппы операторов - методы.

```
public static void main(String[] args) {  
    //Write your code here  
}
```

На данном этапе в каждом классе в вашей программе будет только один метод - main.

Содержимое файла Main.java

Самым простым элементом программы является оператор. Оператор - это неделимый кусок кода.

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

Он служит для вывода информации на консоль.

* "консоль" обозначает программу для вывода тестовых сообщений. Это программа не имеет графического интерфейса и повсеместно используется при администрировании серверных приложений. В курсе мы периодически будем касаться тем работы с консолью.

Пакеты в Java

Как правило, в Java классы объединяются в пакеты. Пакеты позволяют организовать классы логически в наборы. По умолчанию java уже имеет ряд встроенных пакетов, например, `java.lang`, `java.util`, `java.io` и т.д. Кроме того, пакеты могут иметь вложенные пакеты.



Организация классов в виде пакетов позволяет избежать конфликта имен между классами. Ведь нередко ситуации, когда разработчики называют свои классы одинаковыми именами. Принадлежность к пакету позволяет гарантировать однозначность имен.



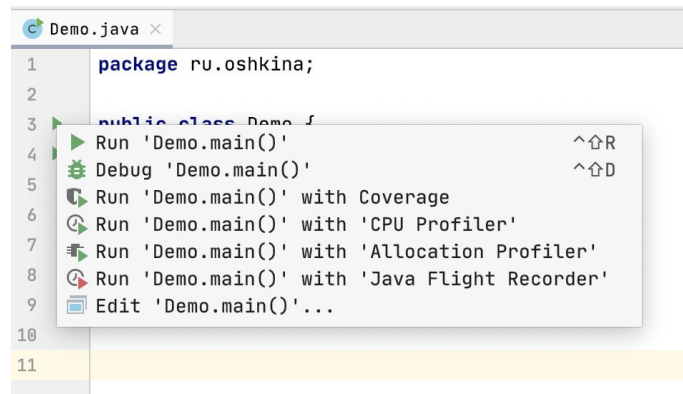
Содержимое файла Main.java

Весь код класса будет выглядеть следующим образом:

```
package ru.oshkina;

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

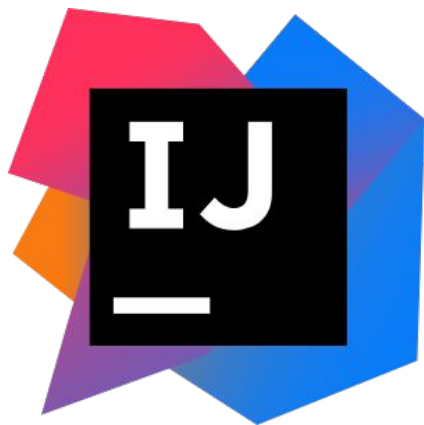
Теперь давайте запустим код:



Результат вывода в консоль IDEA:



Live-демонстрация (Demo1.java, Demo2.java)



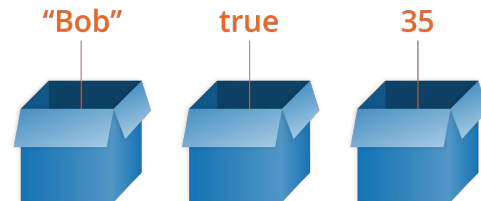
Типы данных

Переменная - элементарный контейнер для хранения данных.

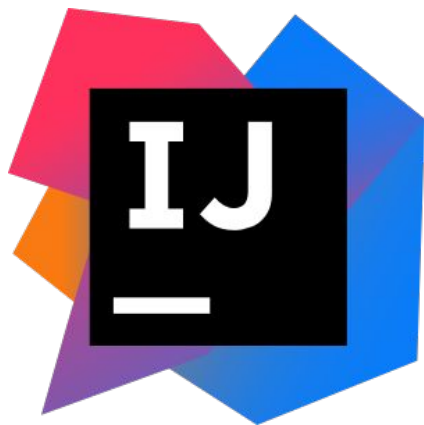
Как и в реальной жизни, контейнеры создают для хранения определенных вещей. Например: жидкости, горючих или сыпучих веществ и т.д. То есть у каждого контейнера есть определенный **тип хранимых данных**.

Шаблон для создания переменной в Java:

ТИП_ДАННЫХ ИМЯ_ПЕРЕМЕННОЙ = ЗНАЧЕНИЕ_ПЕРЕМЕННОЙ.



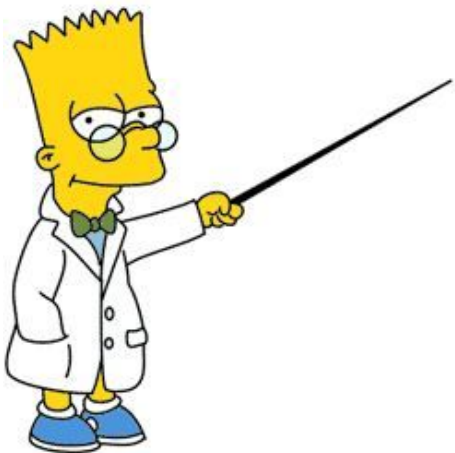
Live-демонстрация (Demo3.java)



Типы данных. Примитивные типы

Обратите внимание, что для каждого типа данных зарезервировано ключевое слово. Его нельзя написать по-другому. Все примитивные типы пишутся со строчной буквы.

Описание	Пример использования	Класс оболочки
byte – используется для хранения байтовой информации в памяти (8 бит).	<code>byte mem = 1;</code>	Byte
short – целочисленный тип (16 бит)	<code>short size = 1;</code>	Short
int – целочисленный тип (32 бита)	<code>int length = 22900;</code>	Integer
long – целочисленный тип (64 бита)	<code>long money = 900500;</code>	Long
float – числа с плавающей точкой (32 бита)	<code>float size = 45.6F;</code>	Float
double – числа с плавающей точкой (64 бита)	<code>double size = 55.88;</code>	Double
boolean – описывают логический тип (1 бит)	<code>boolean exists = true;</code>	Boolean
char – символьный тип (16 бит)	<code>char exit = 'Y';</code>	Character



Типы данных. Примитивные типы

В определении типа данных есть характеристика, указывающая, какие данные могут быть записаны в эту переменную.

byte – числа от -128 до 127.

short – числа от -32,768 до 32,767

int – числа от -2 в степени 31 до 2 в степени 31 минус 1.

long – числа от -2 в степени 63 до 2 в степени 63 минус 1.

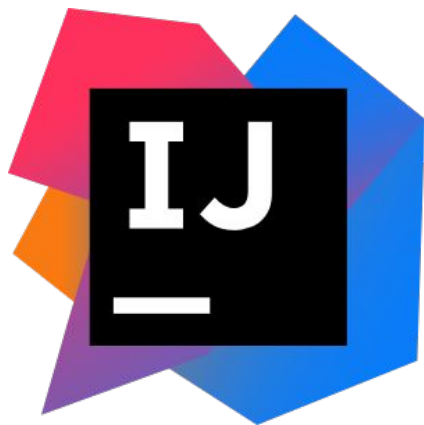
float – числа с плавающей точкой. Диапазон примерно такой же, как и у `int`.

double – числа с плавающей точкой. Диапазон примерно такой же, как и у `long`.

boolean – `true` или `false`.

char – любой символ: `char choice = 'b'; char one = '1';`

Live-демонстрация (Demo4.java, Demo5.java)



Приведение типов

Базовые правила автоматического приведения типов:

1. Типы переменных, входящих в выражение должны быть совместимы.
2. Целевой тип (тип к которому выполняется приведение) должен быть шире исходного типа.
3. Перед выполнением арифметических операций типы `byte`, `short`, `char` расширяются до типа `int`.
4. Если в выражении есть операнды типа `long`, то расширение выполняется до типа `long`.
5. Если в выражении есть операнды типа `float`, то расширение выполняется до типа `float`.
6. Если в выражении есть операнды типа `double`, то расширение выполняется до типа `double`.
7. Литералы, обозначающие целые числа, интерпретируются как значения типа `int`.
8. Литералы, обозначающие вещественные числа, интерпретируются как значения типа `double`.

Live-демонстрация (Demo6.java, Demo7.java)



Арифметические операции

С примитивными типами можно производить следующие арифметические операции:

- | | |
|-------------------------------------|--------------|
| • Сложение | $c = a + b$ |
| • Вычитание | $c = a - b$ |
| • Умножение | $c = a * b$ |
| • Деление | $c = a / b$ |
| • Деление по модулю | $c = a \% b$ |
| • Инкремент | $a++$ |
| • Декремент | $a--$ |
| • Сложение с присваиванием | $a += b$ |
| • Вычитание с присваиванием | $a -= b$ |
| • Умножение с присваиванием | $a *= b$ |
| • Деление с присваиванием | $a /= b$ |
| • Деление по модулю с присваиванием | $a \% = b$ |

Live-демонстрация (Demo8.java)



Побитовые и логические операции

- Логическая операция «И»

$a \& b$

0011	3
0101	5
<hr/>	
0001	1

Пример: $3 \& 5 = 1$

- Логическая операция «ИЛИ»

$a | b$

0011	3
0101	5
<hr/>	
0111	7

Пример: $3 | 5 = 7$

- Логическая операция «Исключающее ИЛИ»

$a \wedge b$

0011	3
0101	5
<hr/>	
0110	6

Пример: $3 \wedge 5 = 6$

- Логическая операция «НЕ»

$\sim a$

101010
↓
010101

Пример: $\sim 15 = -16$

- Сдвиг вправо с учетом знака

$a >> b$

001010
↓
000101

Пример: $10 >> 1 = 5$

- Сдвиг влево с учетом знака

$a << b$

001010
↓
100100

Пример: $10 << 2 = 40$

Можно заметить: при сдвиге вправо на 1 порядок число делится на 2. При сдвиге влево наоборот - умножается на 2.

Операции сравнения

	Оператор	Пример использования	Возвращает true, если
1. Больше	>	$a > b$	a больше b
2. Меньше	<	$a < b$	a меньше b
3. Больше или равно	>=	$a >= b$	a больше или равно b
4. Меньше или равно	<=	$a <= b$	a меньше или равно b
5. Равно	==	$a == b$	a равно b
6. Не равно	!=	$a != b$	a не равно b

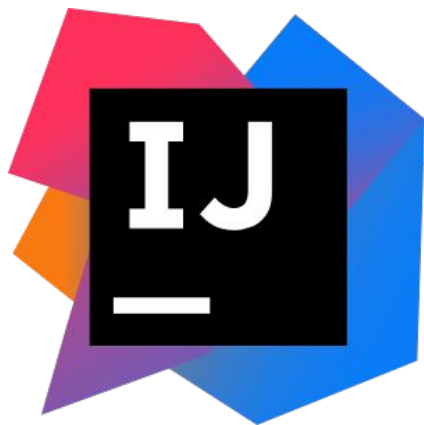
Пример операции:

$(a >= b) \mid (c < 100) \ \& \ !(true) \ ^ \ (q == 5),$

где a, b, c, q - целые числа



Live-демонстрация(Demo9.java)



Переназначение переменной

1. Предположим, что нам нужно вычислить два раза сумму, с разными числами.

```
public class Calculator {  
  
    public static void main(String[] args) {  
        int one = 1;  
        int two = 2;  
        int onePlusTwo = one + two;  
        System.out.println(onePlusTwo);  
    }  
}
```

2. Чтобы это сделать, нам нужно создать новые переменные с уникальными именами:

```
public class Calculator {  
    public static void main(String[] args) {  
        int one = 1;  
        int two = 2;  
        int onePlusTwo = one + two;  
        System.out.println(onePlusTwo);  
        int ten = 10;  
        int eleven = 11;  
        int tenPlusEleven = ten + eleven;  
        System.out.println(tenPlusEleven);  
    }  
}
```


Переназначение переменной

Придумывать каждый раз новые имена со временем станет сложно. Поэтому в Java есть возможность присваивать новые значения переменным. Давайте создадим переменную `age` и присвоим ей значение 18. Потом переназначим ее на 19.

```
int age = 18; /* создаем переменную. */  
System.out.println(age);  
age = 19; /* присваиваем новое значение. */  
System.out.println(age);
```

Давайте перепишем код класса `ru.oshkina.calculator.Calculator` таким образом, чтобы у нас повторно использовались переменные:

```
public class Calculator {  
    public static void main(String[] args) {  
        int one = 1;  
        int two = 2;  
        int result = one + two;  
        System.out.println(result);  
        one = 10;  
        two = 11;  
        result = one + two;  
        System.out.println(result);  
    }  
}
```

Методы

В чем проблема этого кода?

```
public class Calculator {  
    public static void main(String[] args) {  
        int one = 1;  
        int two = 2;  
        int result = one + two;  
        System.out.println(result);  
        one = 10;  
        two = 11;  
        result = one + two;  
        System.out.println(result);  
    }  
}
```

Методы



Проблема

Если нам нужно повторить операцию для других чисел, то придется заново копировать код.

В данном примере вычисляемая операция примитивна, но в больших программах вычисление может занимать сотни строк кода.

А чем плохо копирование? Копирование кода с ошибкой порождает новые ошибки. Скопировали 100 раз часть кода с ошибкой, это значит добавили в проект еще 100 ошибок.

```
public class Calculator {  
    public static void main(String[] args) {  
        int one = 1;  
        int two = 2;  
        int result = one + two;  
        System.out.println(result);  
        one = 10;  
        two = 11;  
        result = one + two;  
        System.out.println(result);  
    }  
}
```

Методы

Метод - это именованный блок команд, которые выполняются при вызове метода. Другими словами есть набор команд и этот набор команд имеет имя.

Чтобы выполнить метод, нужно указать имя класса и через точку имя метода.

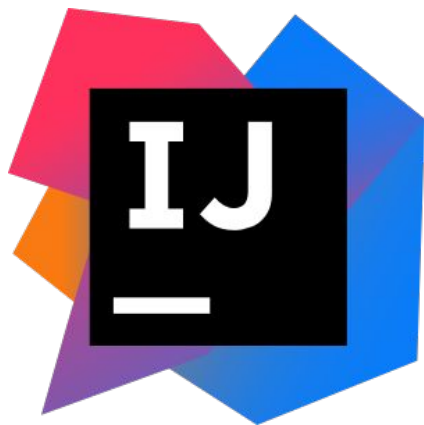
Calculator.plus(1, 2);

```
public class Calculator {  
  
    public static void plus(int first, int second) {  
        int result = first + second;  
        System.out.println(result);  
    }  
  
    public static void main(String[] args) {  
        Calculator.plus(1, 2);  
        Calculator.plus(10, 11);  
    }  
}
```

Обратите внимание, что после имени метода указываются аргументы метода:

public static void plus(int first, int second) {

Live-демонстрация(Demo10.java)



Задание

Ниже приведен код с ошибками. Ваша задача поправить код, чтобы он компилировался.

```
public class ArgMethod {

    public static void hello(String name) {
        System.out.println("Hello, " + name);
    }

    public static void main(String[] args) {
        String name = "Ivan Ivanov";
        int age = 33;

        ArgMethod.hello();

        ArgMethod.hello(name, age);

        ArgMethod.hello(age);

        ArgMethod.hello(name, name, name);
    }
}
```

Резюме

1. Java является полностью объектно-ориентированным языком. Для создания даже самой простой программы необходимо описать по крайней мере один класс. Этот класс содержит главный метод со стандартным названием `main()`. Выполнение программы отождествляется с выполнением главного метода.
2. В методе `main()` можно объявлять переменные. Для объявления переменной указывается тип переменной и ее имя. Переменной одновременно с объявлением можно присвоить значение (проинициализировать). Переменная должна быть проинициализирована до ее первого использования.
3. Существует несколько базовых типов данных. При вычислении выражений выполняется автоматическое преобразование типов. Особенность автоматического преобразования типов в Java в том, что они выполняются без потери значений. Также можно выполнять явное приведение типов.
4. Основные операторы в Java делятся на арифметические, логические, побитовые и операторы сравнения.
5. В java существуют составные операторы присваивания: `a = a + b -> a +=b`