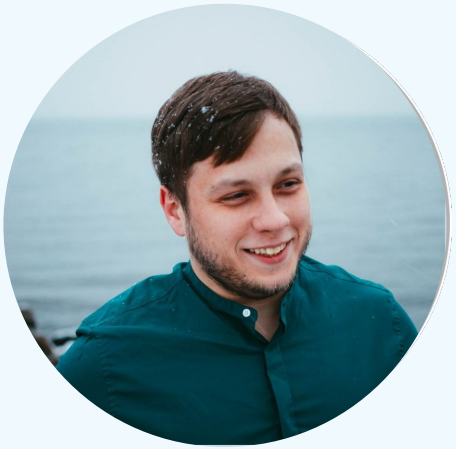
нетология

Структура программы

Филипп
Воронов



Филипп Воронов

Teamlead, VK Group

@Филипп Воронов

План занятия

1. Что такое класс?
2. Как вызвать созданный метод?
3. Как вывести произвольную информацию в консоль?
4. Чтение данных из консоли
5. Пакет и импорт

Что такое класс?

Что такое класс?

В каждом исходном файле Java программист должен создавать так называемый класс `class` — это специальный оператор языка, который определяет содержимое программы.

Пример: Файл `Main.java`

```
class Main {
    // Ваш код
}
```

Программа на Java может состоять из одного или из множества таких классов. Имя каждого файла с исходным кодом обязательно должно совпадать с именем класса внутри него.

Что такое поля и методы класса?

Поля нужны для хранения, изменения и использования значений в программе. В некоторых языках поля называют переменными.

Пример: Создадим класс `Test` с двумя полями, которые смогут хранить целочисленные значения.

```
class Test {

    static int value1;

    static int value2;
}
```

В этом классе мы объявили две целочисленные переменные `value1` и `value2`. Перед объявлением любой переменной должен быть указан тип значения, который будет хранить эта переменная.

Что такое поля и методы класса?

При объявлении переменной можно присвоить ей начальное значение.

Пример: Создадим класс `Test` с двумя полями и присвоим им значения:

```
class Test {

    static int value1 = 10;

    static int value2 = 20;
}
```

Что такое поля и методы класса?

Чтобы создать переменную, которую было бы невозможно изменить в процессе выполнения программы, нужно перед объявлением типа этой переменной добавить оператор `final`.

Пример: Создание константы дня недели:

```
class Test {

    static final String DAY_OF_WEEK = "Monday";
}
```

Такая переменная называется константа, и в случае попытки изменения такой переменной Java выведет ошибку. Любой константе при создании должно быть присвоено значение.

Методы класса

Методы нужны для разделения кода на логические части, они позволяют быстрее читать, дорабатывать и документировать код.

Пример: Метод с именем `printText`, явно он ничего не возвращается так как тип возвращаемого значения void, но печатает на экран строку:

```
class Test {
    static void printText() {
        System.out.println("New program");
    }
}
```

Методы класса

Объявление методов состоит из оператора `static void printText`. На слове `static` мы пока не будет останавливаться, добавляйте его при любом создании метода перед объявлением имени создаваемого метода.

`void` — зарезервированное слово, которое говорит, что метод ничего не возвращает. Его можно заменить на любой другой тип, если мы хотим, чтобы метод вернул значение. После `void` следует имя метода, оно может быть практически любым.

Методы класса

Пример: Метод, который принимает два числа в качестве аргументов и возвращает результат их сложения:

```
class Test {
    static int sum(int value1, int value2) {
        int result = value1 + value2;
        return result;
    }
}
```

Мы создали метод, который возвращает целочисленный тип `int`, а принимает на вход два аргумента — переменные `value1` и `value2`.

Для того чтобы метод вернул значение или завершил работу, нужно вызвать `return` и указать переменную, которую нужно вернуть из метода.

Как вызвать созданный метод?

Метод `main`

Для того чтобы запустить любую программу на Java, нужно найти и запустить метод `main`. Его сигнатура `public static void main(String[] args)`.

Добавим в наш класс метод `main` и вызовем в нем метод `sum`:

```
class Test {

    public static void main(String[] args) {
        sum(4, 5);
    }

    static int sum(int value1, int value2) {
        int result = value1 + value2;
        return result;
    }
}
```

Мы научились вызывать написанный метод, но если запустить программу, никакого результата в консоль выведено не будет.

Как вывести произвольную информацию в консоль?

Вывод данных

Для того чтобы выводить любые данные в консоль, нужно использовать специальный метод, который уже есть в стандартной библиотеке Java — `System.out.println`. В качестве аргумента метод принимает произвольное значение для вывода в консоль.

Добавим в наш код метод `System.out.println`:

```
class Test {

    public static void main(String[] args) {
        int sumOfTwoValues = sum(4, 5);
        System.out.println(sumOfTwoValues);
    }

    static int sum(int value1, int value2) {
        int result = value1 + value2;
        return result;
    }
}
```

Результат

Результат работы программы:

```
9
```

Вывод данных

Такой вывод не всегда удобен и понятен, поэтому принято дополнительно выводить более подробную информацию. Давайте также выведем наименование операции:

```
class Test {

    public static void main(String[] args) {
        int sumOfTwoValues = sum(4, 5);
        System.out.println("Результат сложения: " + sumOfTwoValues);
    }

    static int sum(int value1, int value2) {
        int result = value1 + value2;
        return result;
    }
}
```

А что насчет значений, кроме 4 и 5? Как сложить произвольные значения, не меняя каждый раз код программы?

Чтение данных из консоли

Чтобы запросить данные у пользователя из консоли, в JDK разработан специальный класс — `java.util.Scanner`.

Этот класс умеет считывать данные построчно, а также считывать значения разделенные по умолчанию пробелом (можно заменить на другой разделитель).

Давайте добавим ввод данных из консоли:

```
class Test {

    public static void main(String[] args) {
        java.util.Scanner scanner = new java.util.Scanner(System.in);
        int sumOfTwoValues = sum(4, 5);
        System.out.println("Результат сложения: " + sumOfTwoValues);
    }

    static int sum(int value1, int value2) {
        int result = value1 + value2;
        return result;
    }
}
```

Чтение данных из консоли

В строке `java.util.Scanner scanner = new java.util.Scanner(System.in);`

- `java.util.Scanner` – объявляем тип переменной;
- `scanner` – объявляем имя переменной;
- `new java.util.Scanner(System.in)` – создаем значение переменной.

Теперь переменная scanner содержит методы для чтения данных из консоли:

- `scanner.nextLine()` – прочитает строку из консоли;
- `scanner.nextInt()` – прочитает целочисленное значение из консоли.

Чтение данных из консоли

Запросим пользователя ввести значение и выведем на экран результат сложения:

```
class Test {

    public static void main(String[] args) {
        java.util.Scanner scanner = new java.util.Scanner(System.in);
        System.out.println("Введите первое слагаемое");
        int value1 = scanner.nextInt();
        System.out.println("Введите второе слагаемое");
        int value2 = scanner.nextInt();
        int sumOfTwoValues = sum(value1, value2);
        System.out.println("Результат сложения: " + sumOfTwoValues);
    }

    static int sum(int value1, int value2) {
        int result = value1 + value2;
        return result;
    }
}
```

Почему тип `java.util.Scanner` имеет такое длинное название? Всегда придется писать типы в таком виде?

Пакет и импорт

В полном именовании типа: `java.util.Scanner`, `java.util` — имя пакета; `Scanner` — имя типа.

Пакет (package) в Java позволяет разделять типы/классы между библиотеками.

Добавим к нашему классу пакет `ru.netology.lesson2`. Этот пакет относится только к нашему классу и, если мы захотим использовать этот класс в другом классе, то сможем обратиться к нашему классу по его полному имени `ru.netology.lesson2.Test`.

Имя пакета объявляется в самом начале исходного файла.

Пакет и импорт

```
package ru.netology.lesson2;

class Test {

    public static void main(String[] args) {
        java.util.Scanner scanner = new java.util.Scanner(System.in);
        System.out.println("Введите первое слагаемое");
        int value1 = scanner.nextInt();
        System.out.println("Введите второе слагаемое");
        int value2 = scanner.nextInt();
        int sumOfTwoValues = sum(value1, value2);
        System.out.println("Результат сложения: " + sumOfTwoValues);
    }

    static int sum(int value1, int value2) {
        int result = value1 + value2;
        return result;
    }
}
```

Пакет и импорт

`java.util.Scanner` — такое длинное объявление типа допустимо, но на практике, следуя конвенции общепринятого стиля разработки, а также более удобного написания кода и его чтения был разработан оператор `import`.

Оператор import позволяет сокращать имена типов, добавленных в код программы из внешней библиотеки классов. Для всех типов, не входящих в библиотеку `java.lang`, нужно писать полное имя или использовать `import`.

---

import добавляется сразу после объявления пакета.

Пакет и импорт

```
package ru.netology.lesson2;

import java.util.Scanner;

class Test {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Введите первое слагаемое:");
        int value1 = scanner.nextInt();
        System.out.println("Введите второе слагаемое:");
        int value2 = scanner.nextInt();
        int sumOfTwoValues = sum(value1, value2);
        System.out.println("Результат сложения: " + sumOfTwoValues);
    }

    static int sum(int value1, int value2) {
        int result = value1 + value2;
        return result;
    }
}
```

Пакет и импорт

Добавив `import java.util.Scanner;` внутри программы, мы можем использовать сокращенное имя типа `Scanner: Scanner scanner = new Scanner(System.in);`.

Теперь указывать полное имя типа вместе с именем пакета нет необходимости.

Если получится так, что в нашем классе будет использовано несколько классом с одним именем, то придется у одного из них писать полное имя, import нам уже не поможет сократить код.

Чему мы научились

- Узнали, что такое класс, поля, методы;
- Рассмотрели, как читать данные из консоли и выводить их;
- Узнали, что такое `package` и для чего используется `import`.

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Филипп Воронов

@Филипп Воронов