Введение в Java

3 лекция

Структура программы и первая команда

Давайте начнем создание нашей первой программы. Для этого создадим проект в одной из сред разработки. После небольших манипуляций на экране мы увидим примерно следующий код:

```
class Test{
    public static void main(String[] args){
        тут будет код
    }
}
```

Каждая программа на языке Java должна иметь **класс**. Каждая Java программа начинается с метода **main.** Пока что примите это как условность, мы поймём, зачем они нужны только в начале следующего модуля.

Фигурные скобочки в программировании почти на любом языке означают начало и конец какого-либо блока программы

Свою первую программу мы создали, но теперь хотелось бы вывести, что-нибудь на экран. Для этого существуют две команды:

System.out.println(<вывод>); и System.out.print(<вывод>);

Их различие заключается в том, что первая команда после выполнения выполнения переводит курсор на следующую строку, а вторая - нет. Попробуйте выполнить следующий код:

```
class Test{
    public static void main(String[] args){
        System.out.println("Привет!");
        System.out.println("Эта строчка должна быть на новой строке!");
        System.out.print("И эта, так как предыдущая команда выполнила переход на новую строку");
        System.out.print("--- а это нет, так как просто System.out.print() по");
        System.out.print("сле себя строку не переводит");
    }
}
```

Обратите внимание на несколько очень важных вещей:

1. после каждой команды стоит **";"** - так java понимает, где команда заканчивается;
2. внутри каждого принта текст написан в **двойных кавычках** - это строковые значения, а так принт может выводить почти любое значение;
3. **отступы.** Программисты используют их для лучшего понимания кода. Вы поймёте это, когда напишите свой первый крупный проект. Каждый раз, когда ставится "{", лучше всего делать новый отступ (вам поможет Tab).

1 Задание:

Напишите программу, которая выводит на экран строку "Hello, Java!"

Коротко о системах счисления

Подробно о системах счисления вам рассказали/расскажут на уроках информатики. Но прикреплю вам ссылку на ЯКласс.
Нас больше интересуют 3 позиционные системы счисления: двоичная, восьмеричная и шестнадцатеричная. Давайте быстренько вспомним их и обсудим их значимость в ИКТ:

Название системы	основание	алфавит	где применяется
двоичная	2	0, 1	любые электронные устройства
восьмеричная	8	0, 1, 2, 3, 4, 5, 6, 7	документация более древних ПК
шестнадцатеричная	16	0 - 9, A, B, C, D, E, F	документация современных ПК, запись цвета в RGB

Единицы измерения информации.

Бит - это наименьшая единица информации. это количество информации, необходимое для однозначного определения одного из двух равновероятных событий (Да/Нет). В информатике принято рассматривать последовательности длиной 8 бит. Такая последовательность называется ***байтом***. С помощью одного байта можно записать двоичные коды 256 (2^8) чисел от 0 до 255. Мы с вами привыкли к большим объёмам. Ниже приведена таблица, которая показывает их соотношение.

Единицы информации

Название	обозначение	связь с предыдущим	в байтах
байт	Б	= 8 бит	2^0
килобайт	КБ	= 1024 Б	2^{10}
мегабайт	МБ	= 1024 КБ	2^{20}
гигабайт	ГБ	= 1024 МБ	2^{30}
терабайт	ТБ	= 1024 ГБ	2^{40}
петабайт	ПБ	= 1024 ТБ	2^{50}
эксабайт	ЭБ	= 1024 ПБ	2^{60}
зеттабайт	ЗБ	= 1024 ЭБ	2^{70}
йоттабайт	ЙБ	= 1024 ЗБ	2^{80}

Основные типы данных

Для компьютера очень важно понимать, с каким типом данных мы работаем. Например, "6" он может воспринимать как целое число, действительное число или как строку. И в зависимости от того, как компьютер воспринимает "6", он будет работать с ней по-разному. В любом языке программирование есть понятие примитивных типов данных. В Java их всего 8:

Название	Что означает	Размер
byte	целые числа от -128 (-2^7) до 127 (2^7-1)	1 байт
short	целые числа, от -32768 (-2^{15}) до 32767 ($2^{15}-1$)	2 байта
int	целые числа от -2147483648 (-2^{31}) до 2147483647 ($2^{31}-1$)	4 байта
long	целые числа от -9223372036854775808 (-2^{63}) до 9223372036854775807 ($2^{63}-1$)	8 байт
float	действительные числа примерно от $1,4 * 10^{-45}$ до $3,4 * 10^{38}$	4 байта
double	действительные числа примерно от $4,9 * 10^{-324}$ до $1,8 * 10^{308}$	8 байт
char	символьный тип данных	2 байта
boolean	значение истина/ложь (true/false)	1 байт

Помимо примитивных типов данных, существуют так называемые **ссылочные типы**. Их гораздо больше, и мы поговорим о них позже. Однако один из них настолько популярен, что понадобится нам уже сейчас. Это тип данных **String.** Он предназначен для хранения строк. В отличии от примитивных, тяжело сказать, сколько места будет занимать этот тип, так как он является более сложным и состоит из примитивных. Но если кому-то интересно углубится в изучение этого вопроса, то вот ссылка на статью, где об этом рассказывается.

Переменные

Начнём с понятия. Переменная - именованная область памяти компьютера. когда вы её создаёте, компьютер выделяет память для хранения какого-либо значения, и позволяет обращаться к ней по имени, чтобы вам не пришлось запоминать и каждый раз писать этот самый адрес в памяти. Это похоже на коробочку, в которой может что-то лежать.

Для создания переменной мы сначала указываем тип, а потом имя переменной. По сути имя может быть любым, но есть ряд правил для наименования. Оно:

- не должно начинаться с цифры;
- не должно содержать специальных символов, пробелов и кириллицы;
- должно начинаться с маленькой буквы;
- должно отражать суть того, что в ней хранится;
- если название состоит из нескольких слов, то используют "верблюжий регистр" (например currentDateTime);
- не пишите транслитом, лучше использовать переводчик

Если вам интересно, как правильно называть переменные, то об это можно почитать тут.

Вот пример создания переменных. Обратите внимание, что переменная **a** только создаётся, но не инициализируется (ей не задаётся значение).

```java
class Test_variable{
    public static void main(String[] args){
        int a;
        int b = 5;
        float c = 21.6f;
        String name = "Владислав";
        double myVariable = 0.58386;
        boolean is_end_of_lesson = false;
        System.out.println(b); //да он умеет и значения переменных выводить
        System.out.println("a"); //а тут просто строка, не путайте
    }
}
```

Основные операции над типами данных

Большая часть операций, которые мы рассмотрим, очевидна, но есть и не самые обычные действия:

"+" - сложение чисел **3 + 4 = 7**

"-" - вычитание чисел **9 - 5 = 4**

"*" - умножение чисел **2 * 8 = 16**

Вот с делением уже интереснее, так как в любом программировании существует 3 различных деления:

- просто деление (/)
- целочисленное деление (опять /)
- остаток от деления (%)

Если тип **int** поделить на тип **int**, то автоматически будет выполняться целочисленное деление:

25/4 = 6

Если же в делении присутствует хоть где-нибудь тип **double**, то будет выполняться обычное деление:

25.0/4 = 6.25 25/4.0 = 6.25 25.0/4.0 = 6.25

И, как сказано выше, "%" - остаток от деления

25%4 = 1

Операция "+" умеет не только складывать числа, но и "Склеивать" строки. Этот процесс называется **конкатенация**.

"Hello" + " " + "friend" = "Hello friend"

2 задание

Создайте две переменные: целого типа **a** и действительного **b**, равные 5 и 0.4 соответственно. Выведите построчно на экран их сумму, разность, произведение и частное.

Scanner

Отлично, теперь давайте научимся вводить данные с клавиатуры. Для этой цели служит специальный сканер. Он умеет считывать данные из разных источников. В том числе и с клавиатуры. Для его создания используется строка:

```
Scanner sc = new Scanner(System.in);
```

Можем уже заметить знакомый шаблон создания переменной: сначала указывается тип данных (Scanner), а потом название переменной (sc). В правой части стоит более сложная конструкция. Мы полностью её разберём в начале 2 модуля. А пока обратите своё внимание на **System.in**, именно эта строка говорит о том, откуда будут считываться данные - с клавиатуры.

Для того, чтобы мы смогли создать нашу переменную типа Scanner надо сначала импортировать этот тип (подключить к нашей программе). Для этого до создания класса пропишем строку:

```
import java.util.Scanner;
```

Думаю, что теперь понятно как создать этот сканер, но как с ним работать? Для считывания данных нам достаточно после названия переменной поставить точку, начать вводить **next** и выбрать нужную команду. Их много, но мы в основном будем пользоваться следующими:

```
String s1 = sc.next(); // считает строку до первого пробела
String s2 = sc.nextLine(); //считает следующую строку полностью
int i = sc.nextInt();
double d = sc.nextDouble();
float f = sc.nextFloat();
boolean b = sc.nextBoolean();
и т.д.
```

Как вы поняли, каждая из таких команд считывает соответствующий тип данных.
Давайте посмотрим, как выглядит программа для считывания двух целых чисел и поиска их суммы:

```
import java.util.Scanner;

public class Prog {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        System.out.println(a + b);
    }
}
```

Обратите внимание на самую первую строчку программы. Она нужна для подключения этого самого сканера, так как его нет среди стандартных команд Java. Со временем подобные подключения мы будем встречать гораздо чаще.

Ещё немного про вывод

Если в одной строке нужно вывести несколько значений, то пользуйтесь командой **+.**

```
import java.util.Scanner;

class Prog {
    public static void main(String[] args) {
        int a = 5;
        String b = "end";
        System.out.println("Это ещё не " + b + ". a=" + a);
        System.out.println("a + 4 = " + (a+4)); //используйте скобочки, если значение сначало надо вычислить
    }
}
```

3 задание:

Напишите программу, которая считывает три целых числа и выводит их произведение.