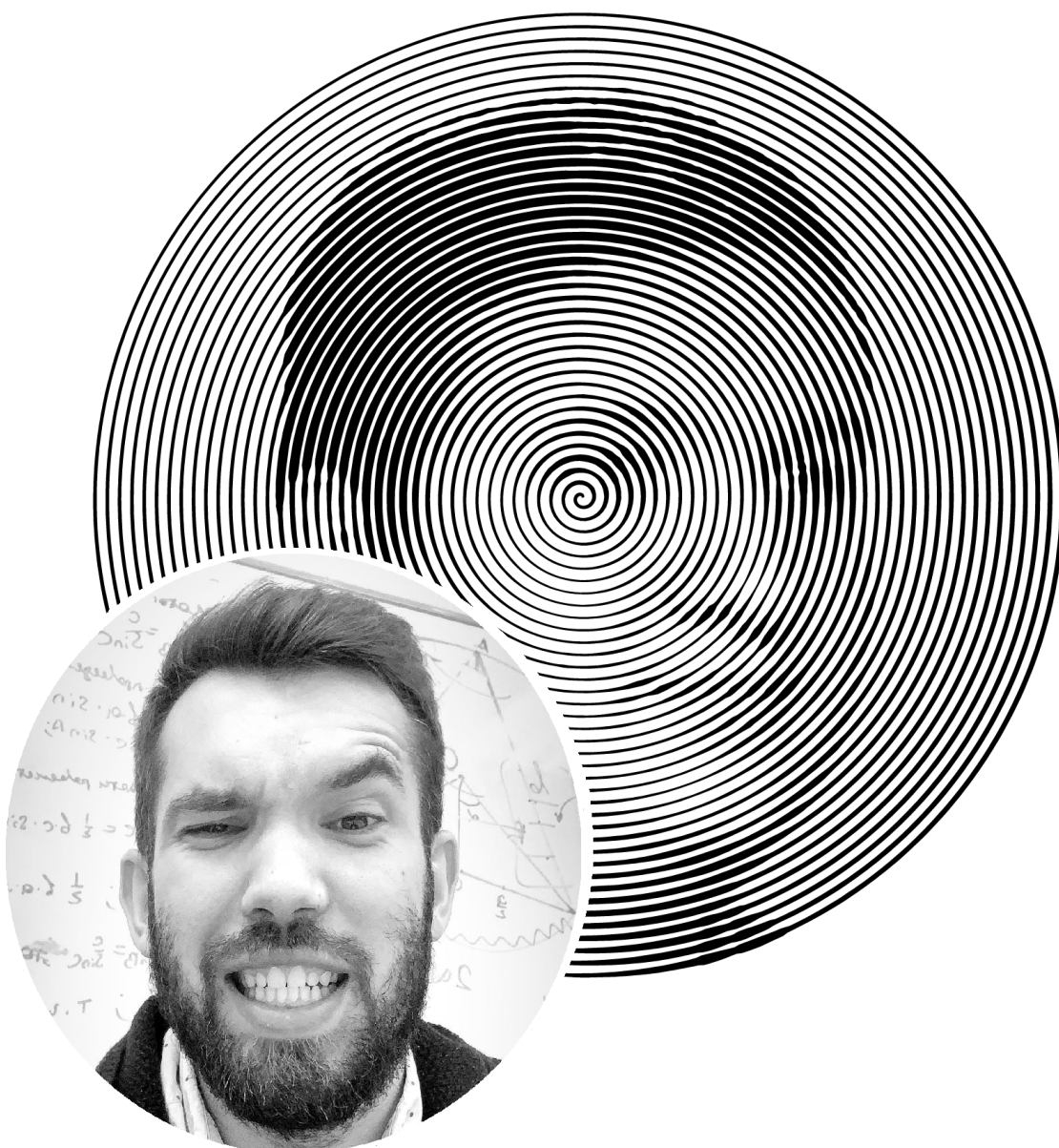


**Н.О. Платонов, Н.Р. Кудлай**

**Основы Java для чайников, электрических  
веников и микроволновых печей**

**Учебное пособие для тех,  
кто по каким-то причинам  
ещё не сдал еще 1 лабу по проге**



**Санкт-Петербург**

**2021**

МИНИСТР ОБРАЗОВАНИЯ И НАУКИ РЗ111 И РЗ118

НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
“ИНСТИТУТ ТЕПЛЫХ МУЖСКИХ ОТНОШЕНИЙ”

**Н.О. Платонов, Н.Р. Кудлай**

**Основы Java для чайников, электрических  
веников и микроволновых печей**

**Учебное пособие для тех,  
кто по каким-то причинам  
ещё не сдал еще 1 лабу по проге**



**УНИВЕРСИТЕТ ИТМО**

**Санкт-Петербург**

**2021**



## Перед началом работы...

Перед тем как разбирать азы синтаксиса нужно немного подготовиться, чтобы потом не тратить на них время.

Первое что можно, но не обязательно нужно — поставить far manager. Он нужен для простой и быстрой передачи файлов на helios. Если вам по какой-то причине это не нужно, то этот пункт можно пропустить.

Ссылка для скачивания: <https://farmanager.ru>

Для передачи файла на helios:

- Открываем Far
- Нажимаем Alt+F1
- Выбираем NetBox
- Нажимаем Shift+F4, заполняем поля и нажимаем ОК. Теперь после выполнения предыдущего пункта можно будет просто выбрать сессию и все. Если будут выскакивать ошибки типа — данный протокол не защищен и т. п. - просто нажимаем ОК.
- Для передачи файла в одной части менеджера зайдите на helios, а во второй откройте папку содержащую файл, который вы хотите передать и перетащите его из одного окна в другое.
- P.S. Если нажать Ctrl+O, то можно будет работать с helios в командной строке

Второе, что нужно сделать в обязательном порядке — поставить java на компьютер. Есть 2 хорошие IDE, одну из которых я советую вам поставить. 1 — Eclipse(более простой и понятный интерфейс, но нет мощного автодополнения и некоторые вещи придется делать руками). 2 — IntelliJ IDEA(более сложный интерфейс, но она намного удобнее). Если у вас еще нет java на компьютере, то IDEA предложит ее скачать. Лучше выбрать 1.8, т.к. на helios стоит именно эта версия. Если java стоит, но IDEA ее почему-то не нашла, то просто укажите ей путь. Теперь создайте проект и подождите пока IDEA закончит его делать(может занят некоторое время, профессиональные IDE они такие). ПКМ по папке scr → New → Package. Назовите его main. ПКМ по main → New → Java Class. Назовите его Main. Зачем это надо — так принято. Пакеты обычно называются с маленькой буквы, а классы с большой. Рекомендую загрузить Ultimate версию с лицензией от JetBrains для студентов. Если планируете использовать не только IDEA, но и другие продукты от JetBrains и не только советую поставить ToolBox.

## ***Первая прога***

Откройте класс Main и напишите следующее внутри класса

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

Должно выглядеть примерно так:

```
package main;
```

```
public class Main{  
    public static void main(String[] args){  
        // код писать тут  
    }  
}
```

Что черт побери тут творится мы разберем во 2 пособии, сейчас просто нужно знать, что весь ваш код нужно писать в void main. Напишите там System.out.println("Hello world!"). Возле void main появится зеленая стрелка, нажав на которую и выбрав run вы запустите программу и покажете IDE точку входа в прогу и теперь сможете запускать свой код через зеленый треугольник.

На консоли должно было вывести - «Hello world!». Если это так — значит все хорошо. System.out – позволяет выводить на консоль данные, которые вы ей передадите

- System.out.print() - выводит данные на консоль
- System.out.println() - тоже что и прошлый пункт, но после вывода переводит строку на новую
- System.out.printf() - форматированный вывод

# Типы данных

В Java есть несколько примитивных типов данных. Главное их отличие — хранение на стеке, а не в куче, как у отсальных.

Числовые типы данных:

- byte – 1 байт на хранение ()
- short – 2 байта на хранение ()
- int – 4 байта на хранение ()
- long – 8 байт на хранение ()

Прочие:

- boolean – хранит true/false
- char – 2 байта на хранение (хранит символы)

А String? String – это класс, примитивным типом он не является.

P.S. Если вдруг вы хотите увеличить/уменьшить значение переменной на 1, то можно использовать пред- и пост- инкременты/декременты. Если написать ++, то значение переменной увеличится на 1, а если –, то уменьшится. Если поставить их до переменной, то значение переменной изменится до операции, а если после, то после.

Например:

```
a = 1;
```

```
System.out.println(a++); // Выведет 1
```

```
System.out.println(++a); // Выведет 3
```

## ***Инициализация и объявление***

С типами данных мы познакомились, теперь нужно научиться их применять. Для объявления примитивных типов:

<тип данных> <имя переменной>; - объявление

<имя переменной> = <значение>; - присвоение

<тип данных> <имя переменной> = <значение>; - инициализация

Примеры:

объявление — `int a;`

присвоение — `a = 2;`

инициализация — `char k = 'j';`

Со сложными типами все немного сложнее, т.к. для них требуется динамическое выделение памяти.

<тип данных> <имя переменной> = new <тип данных>;

Например, `int[] arr = new int[6]` – мы выделили массив типа `int` на 6 ячеек. Ключевое слово `new` говорит компилятору, что мы хотим выделить новую память. Данный сложный тип данных - массив, общее объявление которого - `<тип данных>[] <имя переменной> = new <тип данных>[<размер массива>];`. Если вы хотите создать многомерный массив, то просто увеличивайте количество квадратных скобок, но не забудьте указать размер.

## ***Оператор if***

Оператор `if` – оператор ветвления. Работает он примерно так: Если что-то, то сделай так-то, иначе — то-то. Синтаксически это выглядит так:

```
if (<условие>){
```

```
    Если условие - верно
```

```
} else {
```

```
    Если условие - ложно
```

```
}
```

Разумеется никто не запрещает сделать так

```
if (<условие1>){
```

```
    Если условие1 - верно
```

```
} else if (<условие2>) {
```

Если условие1 — ложно, а условие2 - верно

```
}
```

....

```
else if (<условие n>){
```

Если условие n — верно, а все предыдущие условия ложны

```
} else {
```

Если все условия ложны

```
}
```

Разумеется, блок else можно и вовсе не добавлять.

Для составления сложных условий используются следующие операторы:

- == - эквивалентность(возвращает true только когда оба операнда равны. Для примитивных типов данных сравниваются значения, а для сложных — ссылки!!!!)

- ⑩ != - операция обратная эквивалентности

- >, <, <=, >= - стандартные операции из математики (возвращают true только если левый операнд больше/меньше/не больше/не меньше правого соответственно)

- &&, ||, ! - соответствуют конъюнкции, дизъюнкции и отрицанию из булевой алгебры соответственно

## **Циклы**

в Java есть 3 типа циклов:

### 1. Цикл for

Синтаксическое объявление:

```
for (<блок для объявления переменных>; <условие выхода из цикла>;  
<вещи, которые будут выполнены после каждой итерации(обычно здесь  
операции со счетчиками)>) {}
```

Пример: for (int i = 0; i < 10; i++) {} - 10 итераций цикла

### 2. Цикл while

Синтаксическое объявление:

```
while (<условие выхода из цикла>) {}
```

Пример: while (a < 10) {} будет выполняться пока a < 10

### 3. Цикл do while

Синтаксическое объявление:

```
do {} while(<условие выхода>)
```

Единственное его отличие от while – сначала выполняется действие, а потом проверяется условие.

Do while – цикл с постусловием, а for и while – с предусловием.

P.S. Кроме этого есть 2-я версия цикла for, которая реализуется таким образом

```
for (<тип итерируемой переменной> <имя переменной> : <что итерируем>)  
{  
  
}
```

На каждой итерации цикла в переменную будет записываться значение следующего объекта в коллекции.

По своей сути такая реализация является синтаксическим сахаром и отличается от стандартного for только отсутствием явного счетчика.

Применяться он может только к массивам и объектам, реализующим интерфейс Iterable.

## **Импорт**

Порой возникает ситуация, когда вам нужно что-то использовать, но писать самому что-то лень/сложно. В таком случае вам пригодятся импортируемые библиотеки. Для этого просто пишем

```
import <что импортируем>;
```

например, `import java.lang.Math`; Сейчас мы импортировали в наш класс класс `Math` и теперь можем его использовать. Конечно, можно писать и без импорта, каждый раз делая так `java.lang.Math.abs()`, например, но это не очень то и удобно. Если мы хотим импортировать всю библиотеку, а не ее отдельный класс, то просто пишем вместо имени класса `*`. `Math` является абстрактным классом, поэтому для использования его функций создавать объект не нужно.

На этом конец первого пособия.

Все пособия под редакцией Никиты Кудлая распространяются на бесплатной основе, так как они помогают усваивать материал и созданы не для капитализации, но тираж уже вскоре достигнет полусотни, а себестоимость составляет 5р, и если есть желание поддержать авторов, то вот счет Сбера: 40817810552097130180

Главным автором данного пособия не является Н.Кудлай, так что прошу указать пособие, за которое жертвуете свои кровные