Урок 1

Введение в платформу Java

Введение в платформу Java, инструменты разработчика, написание первой программы. Переменные, типы данных, арифметические операции. Простые методы. Условные операторы.

Особенности платформы Java

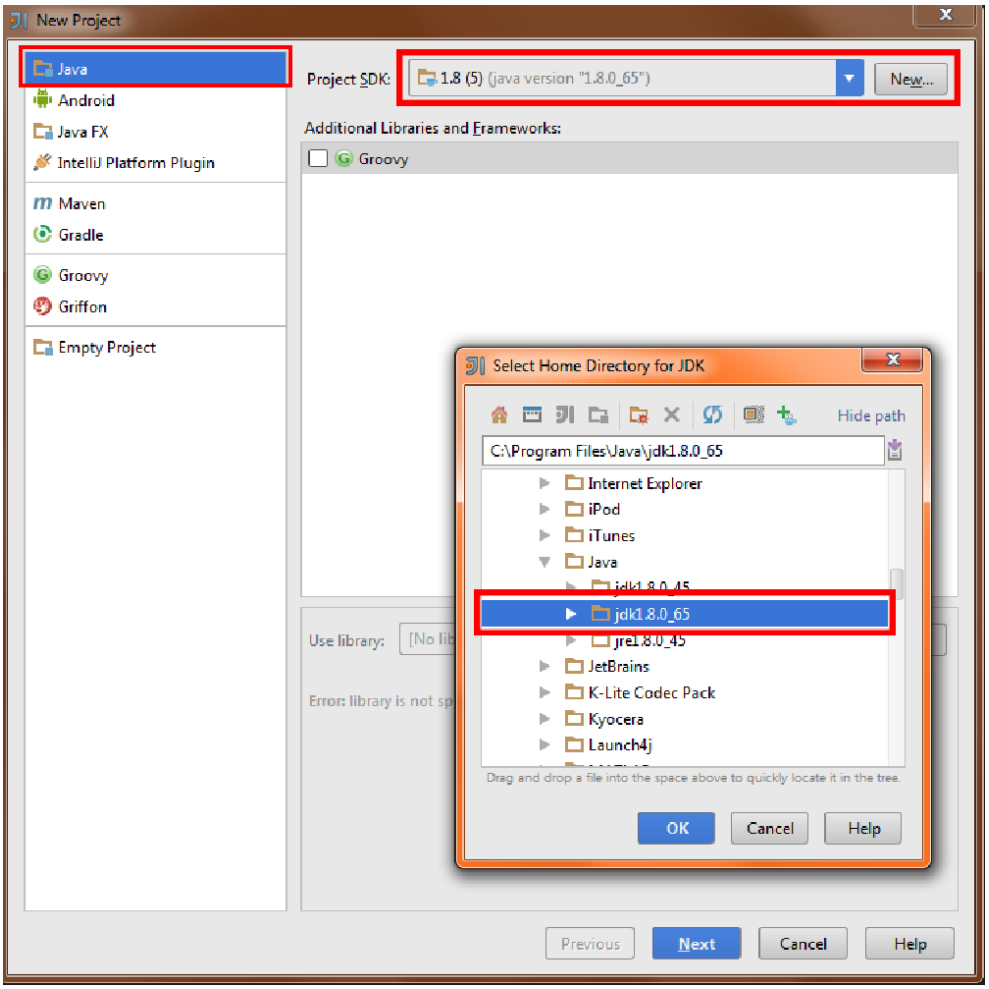
Простота	Язык Java обладает лаконичными, тесно связанными друг с другом и легко усваиваемыми языковыми средствами, был задуман как простой в изучении и эффективный в употреблении язык программирования.
Безопасность	Java предоставляет безопасные средства для создания интернет-приложений.
Переносимость	Программы на Java могут выполняться в любой среде, где есть исполняющая система Java (например, Windows, Linux, Android, MacOS и т.д.).
Объектно-ориентированный характер программирования	В Java воплощена современная философия объектно-ориентированного программирования.
Надежность	Java уменьшает вероятность появления ошибок в программах благодаря строгой типизации данных и выполнению соответствующих проверок во время выполнения. Java исключает ошибки по работе с памятью за счет автоматического управления резервированием и освобождением памяти.
Многопоточность	Язык Java обеспечивает встроенную поддержку многопоточного программирования и предоставляет множество удобных средств для решения задач синхронизации процессов. Это позволяет строить устойчиво работающие интерактивные системы.
Архитектурная независимость	Язык Java не привязан к конкретному типу вычислительной машины или архитектуре операционной системы и следует принципу «написано однажды – работает всегда».
Интерпретируемость и высокая производительность	Java предоставляет байт-код, обеспечивающий независимость от платформы. Компилируя программы в промежуточное представление, называемое байт-кодом, Java позволяет создавать межплатформенные программы, которые будут выполняться в любой системе, где реализована виртуальная машина JVM. Байт-код Java максимально оптимизируется для повышения производительности.

Инструменты разработчика

Для написания программ необходимо установить инструменты разработчика Java Development Kit (JDK), которые свободно предоставляется компанией Oracle. JDK версии 8 можно загрузить здесь: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html. Восьмая версия JDK не является самой последней в настоящий момент, однако ее более чем достаточно для того, чтобы начать изучение языка Java.

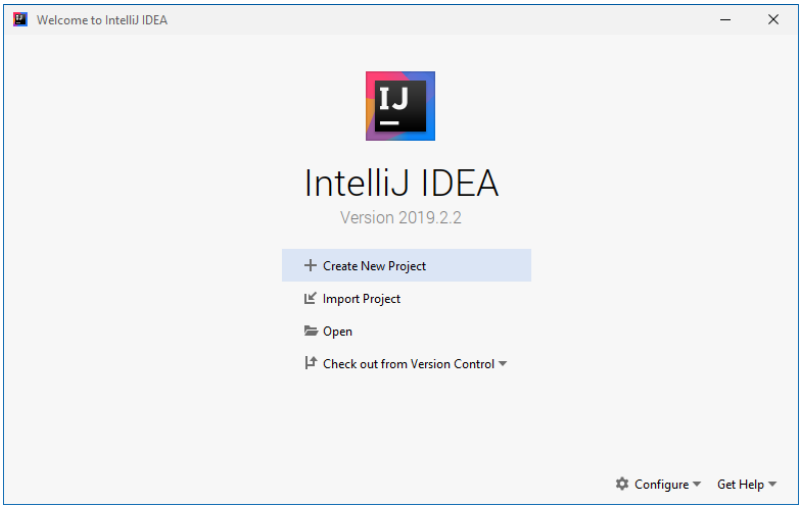
Бесплатную версию среды разработки IntelliJ IDEA Community Edition можно скачать с сайта разработчиков: https://www.jetbrains.com/idea/#chooseYourEdition.

При создании первого проекта в IntelliJ IDEA необходимо указать путь к установленному JDK, как показано на рисунке ниже. Project SDK -> New -> JDK -> путь к папке jdk1.x.x_xxx.

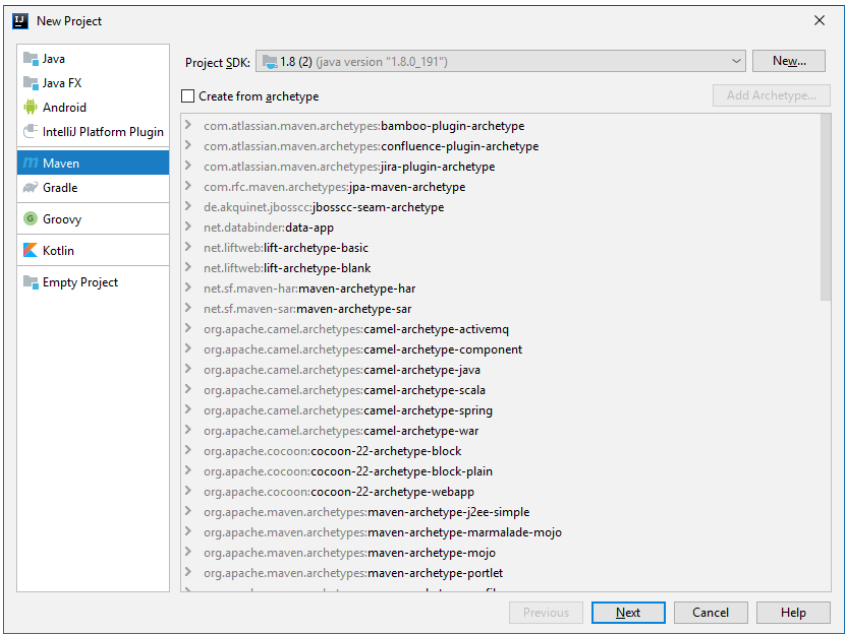


Создаем новый проект

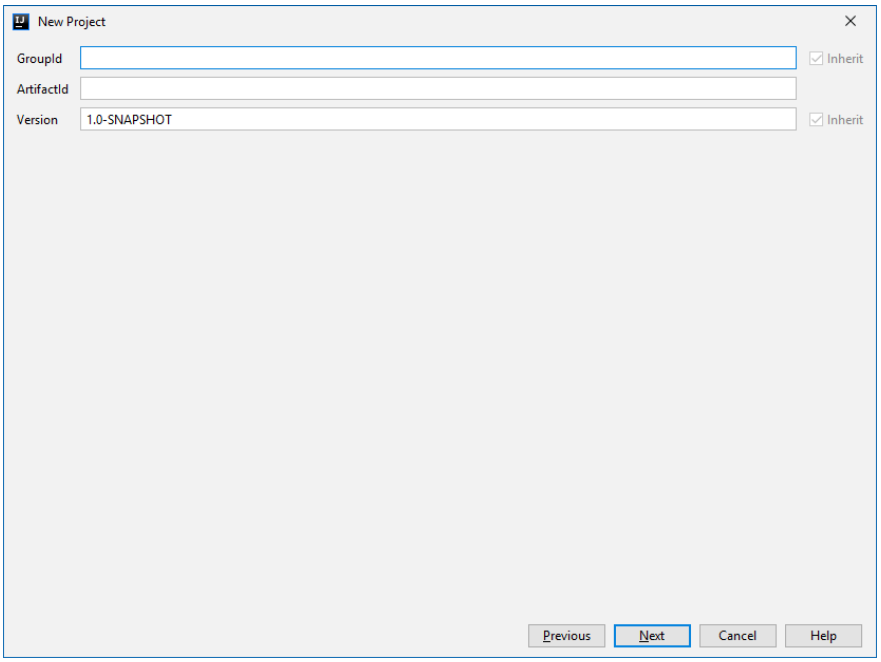
При запуске IntelliJ IDEA появляется окно вида:



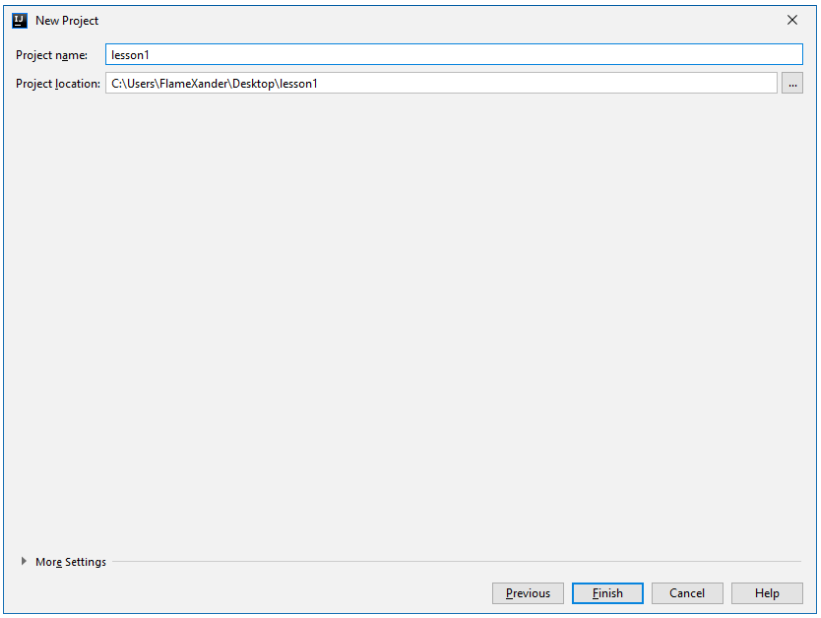
Для создания нового проекта выбираем пункт Create New Project.



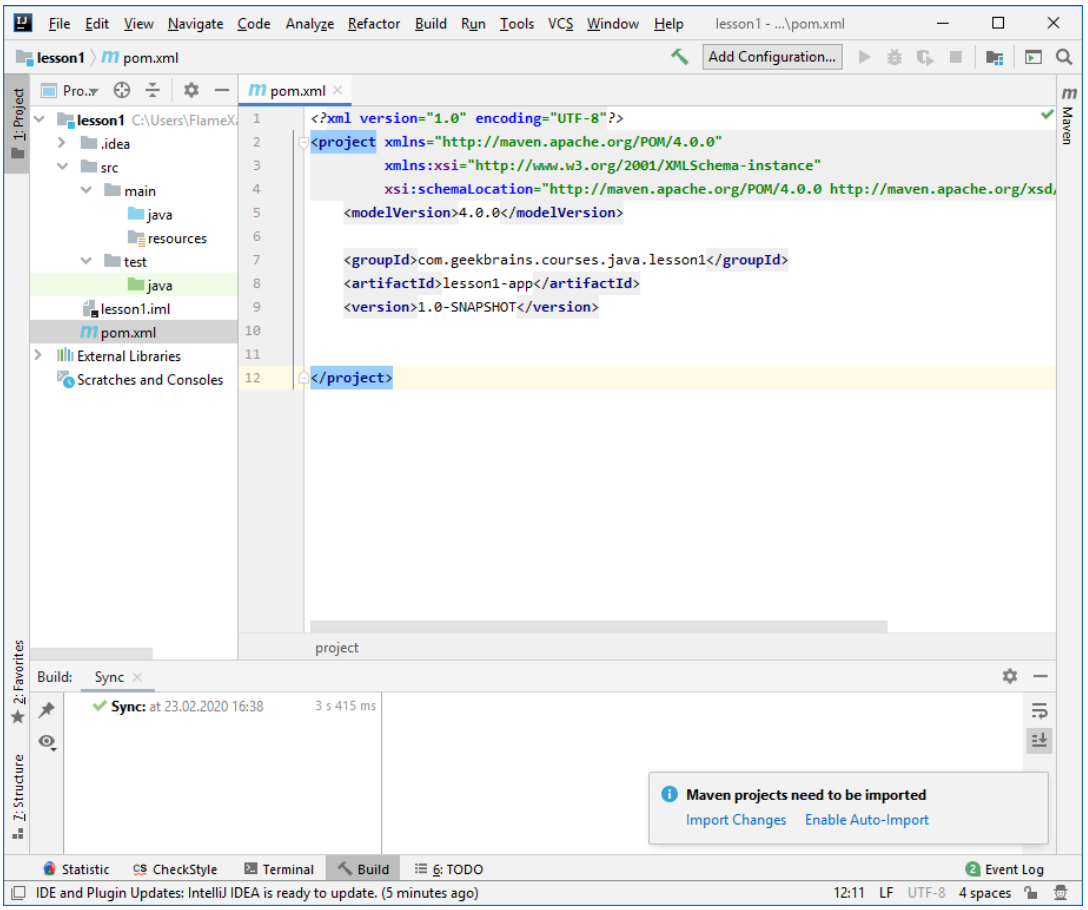
На появившейся странице слева выбираем **Maven**. Это означает что наш проект будет использовать инструмент управления и сборки проекта Apache Maven. В пункте Project SDK должны быть прописана ссылка на JDK.



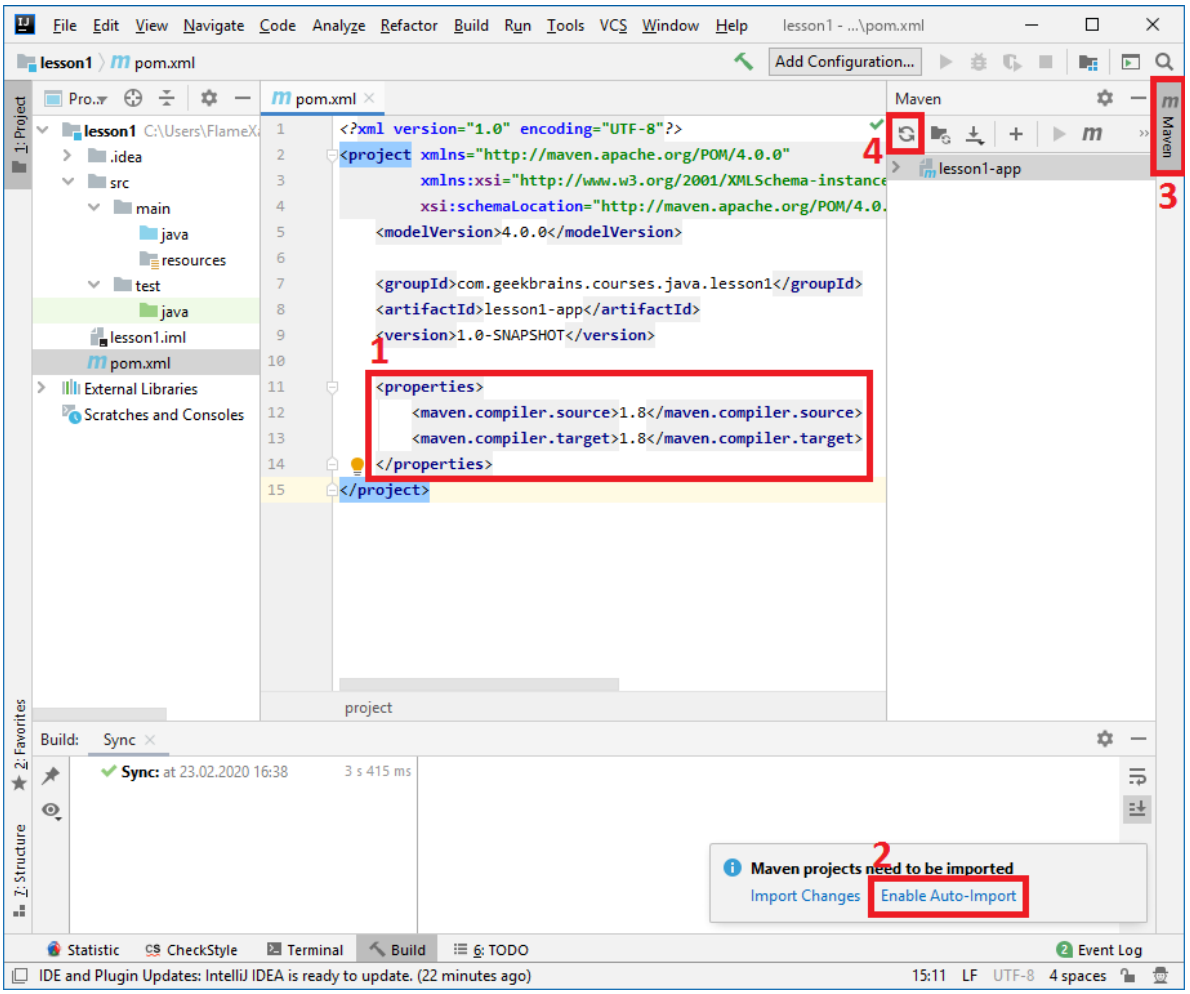
Теперь указываем идентификаторы проекта GroupId, ArtifactId и Version. **GroupId** - идентификатор вашего проекта среди всех остальных проектов, как правило, GroupId начинается с доменного имени, потом может идти название организации/отдела/имени разработчика (элементы этого идентификатора разделяются точками, аналогично пакетам, с которыми вы позднее познакомитесь), и потом название проекта (например, ru.geekbrains.courses.java.alex1234.lesson1, org.apache.maven, com.flamexander.calculator). **ArtifactId** - имя самого проекта (например, lesson1-app, calculator). Version - версия проекта.



Теперь осталось выбрать имя проекта и нажать кнопку Finish. Получим проекта вида:



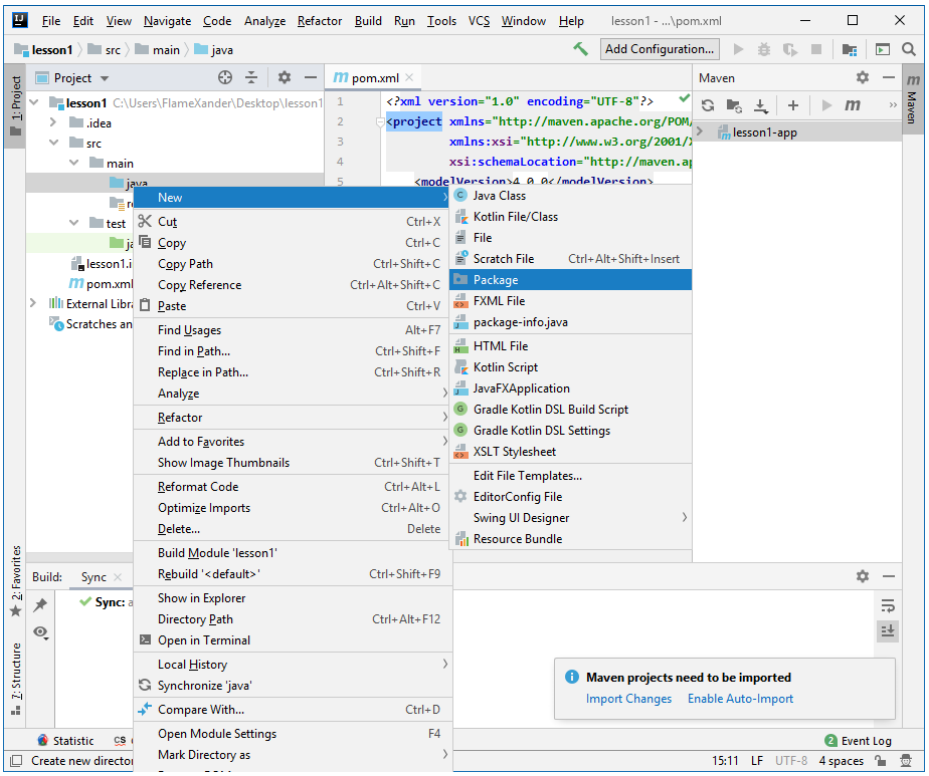
Файл pom.xml является основным конфигурационным файлом проекта. Сразу после создания проекта желательно сделать несколько вещей: указать версию языка, и включить авто-импорт настроек. На рисунке ниже показано как это сделать.



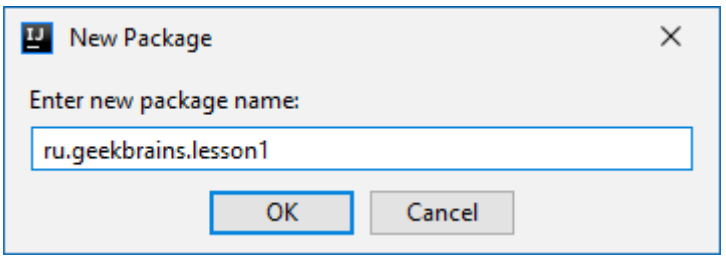
1. Внутри элемента <project></project> прописывается элемент <properties></properties> внутри которого указывается версия языка. В данном случае выбрана версия 1.8, если вы используете JDK 13, то вместо 1.8 необходимо написать 13. **2.** Каждый раз, когда вы вносите изменения в pom.xml необходимо их применять, чтобы обновление происходило автоматически можно включить авто-импорт настроек, кликнув Enable Auto-Import. **3.4.** Авто-импорт не реагирует на изменение версии языка поэтому нужно войти в панель управления Maven (3) и нажать кнопку (4) Reimport All Maven Settings. На этом подготовительная работа закончена.

Теперь немного о структуре проекта. В папке **src > main > java** будут храниться исходники вашего приложения. **src > main > resources** отвечает за ресурсы, которые должны быть упакованы в итоговую сборку проекта. **src > test > java** предназначена для написания тестов. На первом этапе изучения языка вам нужна только папка **src > main > java.**

Для того чтобы начать писать код, нажимайте правой кнопкой на **src > main > java** и выбирайте пункт **New > Package**.

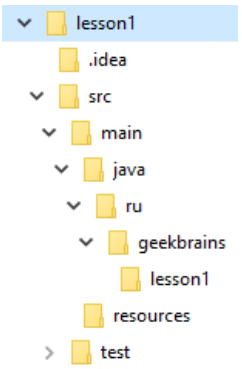


В появившемся окне указываете рабочий пакет. Имя пакета состоит из набора элементов, разделенных точками. Первым элементом является доменное имя (ru, com, org и т.д.), далее может идти наименование организации/отдела/ваше имя, и потом название проекта. Можете указать ru.geekbrains.lesson1

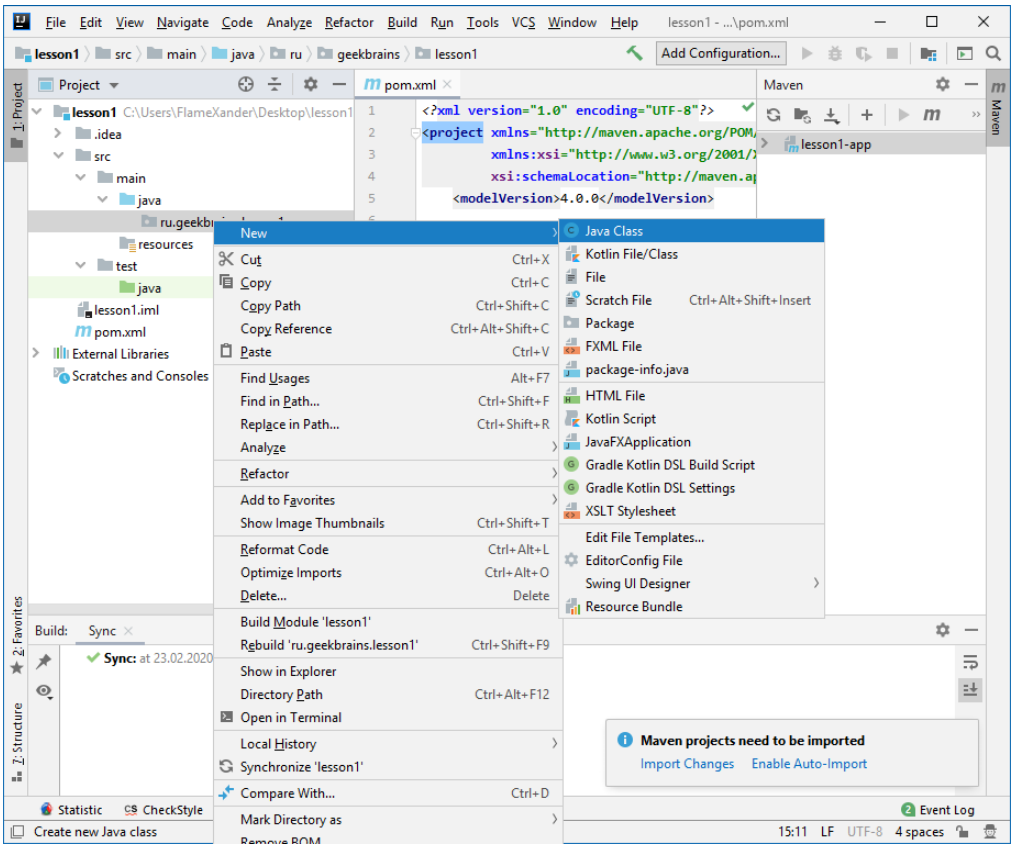


Важно! В названии элементов пакета (в примере выше это ru, geekbrains, lesson1) не следует использовать спец символы (кроме нижнего подчеркивания), заглавные буквы, начинать имя элемента с цифры. Имя элемента также не может совпадать с ключевыми словами зарезервированными языком Java.
Более детально тема пакетов будет рассматривать на занятиях по объектно-ориентированному программированию.

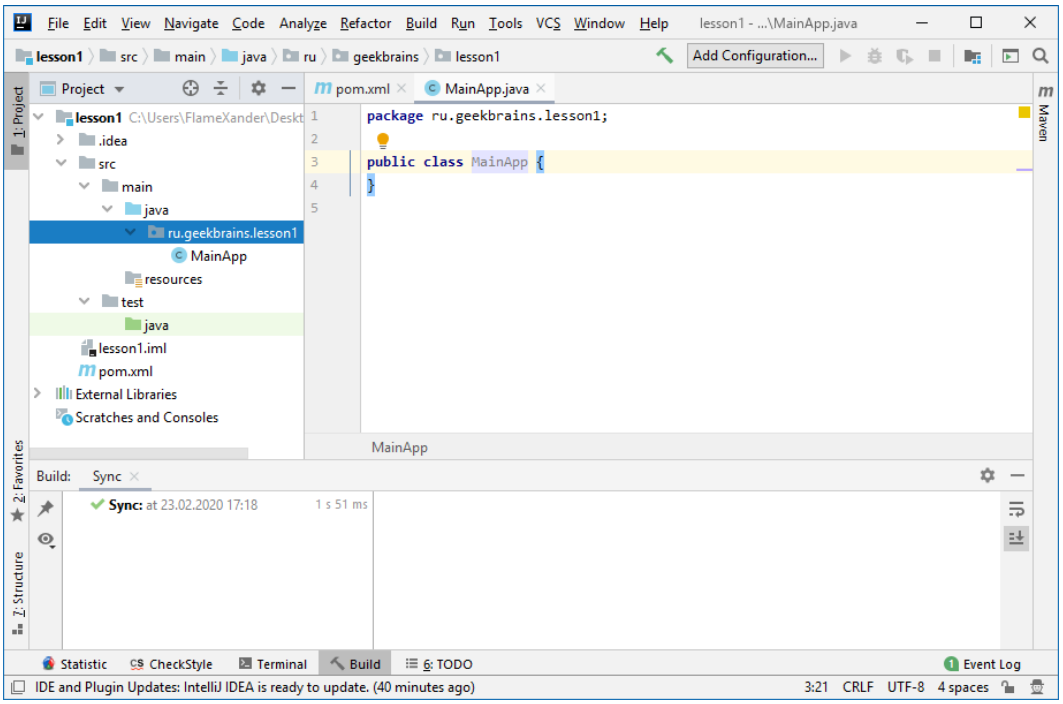
Пакеты нужны для организации структуры проекта. И по сути, ru.geekbrains.lesson1 означает, что в папке java будет вложена папка ru, в папке ru вложена geekbrains, а в geekbrains - lesson1. Наименование пакетов никоим образом не связано ни с какими интернет адресами. Это просто правило их именования.



После того как пакеты созданы, добавляете в него первый Java класс (можете дать ему название MainApp, или любое другое, которое нравится, главное чтобы первая буква в названии была заглавной, и каждое следующее слово в названии также начиналось с заглавной буквы).



Если вы все правильно сделали то проект будет выглядеть вот так.



Как видите первой строкой в классе идет имя пакета с ключевым словом package. В дальнейших примерах эта строка может быть опущена. Но, если ваш класс находится внутри пакета, то эта строка (в данном случае ru.geekbrains.lesson1) всегда будет первой в коде класса.

Первая программа

Давайте посмотрим на то, как выглядит самая простая программа, написанная на языке Java. Создадим новый проект и добавим в него класс FirstApp.java.

```
/**
 * Created by GeekBrains on 15.11.2018.
 */
public class FirstApp {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

В первых строчках кода мы видим так называемые комментарии, они могут быть как однострочные (начинаются с символов //), так и многострочные (блок текста, заключенный в /* … */). Комментарии упрощают работу с кодом, разработчик может делать пометки, которые не будут влиять на размер и скорость выполнения программы.

```
// Однострочный комментарий
/*
...
Блочный комментарий
...
*/
```

После комментариев идёт объявление класса FirstApp.

Важно! Имя класса должно совпадать с именем файла, в котором он объявлен, т.е. класс FirstApp должен находиться в файле FirstApp.java

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

Далее идёт объявление метода main(), с которого начинается исполнение приложения.

Важно! Выполнение любой Java-программы начинается с вызова метода main(). В одном классе может быть объявлен только один метод main(). Если методы main() объявлены в нескольких классах, то вы можете выбирать с какого класса необходимо запускать ваше приложение.

Ключевое слово void сообщает компилятору, что метод main() не возвращает значений. Методы могут также возвращать конкретные значения. Что такое модификаторы доступа (public, private), ключевое слово static и String[] args, будет объяснено позже, когда зайдет разговор об основах ООП и массивах.

Важно! При написании кода на языке Java обязательно необходимо учитывать регистр символов. (Main не равнозначно main, Void и void не одно и то же, System не равнозначно system и т.д.)

Важно! Если вы работаете в IntelliJ IDEA, для упрощения/ускорения написания метода main, вы можете пользоваться сокращением psvm.

В следующей строке кода System.out.println("**Hello, World!**"), которая находится в теле метода main(), в консоль выводится текстовая строка **Hello, World!** с последующим переводом каретки на новую строку. На самом деле вывод текста на экран выполняется встроенным методом println(), который кроме текста может также выводить числа, символы, значения переменных и т.д.

Важно! В языке Java все операторы обычно должны оканчиваться символом ; . После закрытия кодовых блоков (там, где блок закрывается скобкой }) точка с запятой не ставится, потому что эти фигурные скобки не являются операторами.

Практика! Попробуйте создать новый Java проект в IntelliJ IDEA, добавить в него класс FirstApp, прописать в этом классе метод main() и код для вывода сообщения в консоль. Как только все это будет готово - запустите проект, и проверьте что в консоли выведено, то что указано в System.out.println(); Если есть расхождения - сравните с кодом, представленным выше.

Дополнительно: Попробуйте написать в круглых скобках System.out.println() какой-нибудь другой текст, целое или дробное число, и проверьте что получится.

Теперь вы знаете как написать самую простую программу на языке Java и запустить ее в IntelliJ IDEA.

Переменные и типы данных

Переменные представляют собой зарезервированную область памяти для хранения данных. В зависимости от типа переменной операционная система выделяет память и решает, что именно должно в ней храниться.

В Java существует две группы типов данных.

- Примитивные.
- Ссылочные (объектные).

Существует 8 примитивных типов данных.

Тип	Описание	Возможные значения	Пример
byte	8-битное знаковое целое число	от -128 до 127	byte val = -120;
short	16-битное знаковое целое число	от -32768 до 32767	short val = 12442;
int	32-битное знаковое целое число	от -2147483648 до 2147483647	int val = 1000;
long	64-битное знаковое целое число	от -9223372036854775808 до 9223372036854775807	long val = 200000L;
float	32-битное знаковое число с плавающей запятой одинарной точности		float val = 12.23f;
double	64-битное знаковое число с плавающей запятой двойной точности		double val = -123.123;
char	16-битный тип данных, предназначенный для хранения символов в кодировке Unicode	от '\u0000' или 0 до '\uffff' или 65,535	char val1 = ‘*’; char val2 = ‘\u2242’;
boolean	логический тип данных	false, true	boolean val = false;

Важно! Если вы хотите указать float величину, то после числа необходимо поставить букву f.
float floatVal = 12.24f;
Если буква указана не будет, то компилятор будет считать такое дробное число типом double.
При использовании переменной типа long, после числа необходимо ставить букву L.
long longValue = 20000000000L;

Важно! Если вы только начинаете программировать, то нет необходимости запоминать количество бит для хранения переменной определенного типа, или точные границы значений. Достаточно понять какие значения, в каком типе можно хранить.

Заметка. Если вы только начали осваивать программирование, то можете на первых этапах пользоваться для работы с целыми числами переменными типа **int**, для дробных чисел **float**. А при дальнейшем обучении разберетесь с остальными типами.

Ссылочных типов данных существует большое количество, кроме того, можно создавать новые ссылочные типы, но обо всем этом будет рассказано на следующих занятиях.

Общую структуру объявления переменной можно описать так:

```
[тип_данных] [идентификатор(имя_переменной)]; // объявление переменной
[тип_данных] [идентификатор(имя_переменной)] = [начальное_значение]; // объявление переменной с инициализацией
```

```
public class FirstApp {
    public static void main(String[] args) {
        int a = 20;
        float b;
        b = 2.25f;
    }
}
```

Идентификаторы – это имена переменных, которые начинаются с буквы, $ или _, после чего может идти любая последовательность символов. Идентификаторы чувствительны к регистру.

Важно! Имена переменных должны быть написаны в camelCase - первая буква строчная, каждое следующее слово в имени с заглавной буквы и без нижних подчеркиваний или пробелов.
Правильные имена:
floatValue, name, enginePower, firstName, lastName;
Неправильные имена:
Name, Title, EnginePower, First_name, last_name
Имя переменной не должно начинаться со спец символа (кроме нижнего подчеркивания) или цифры.

В качестве идентификаторов нельзя использовать ключевые слова Java.

Заметка. К ключевым словам языка Java относятся: abstract assert boolean break byte case catch char class const continue default do double else enum extends final finally float for goto if implements import instanceof int interface long native new package private protected public return short static strictfp super switch synchronized this throw throws transient try void volatile while

Чтобы переменная не могла менять свое значение в процессе выполнения программы, можно определить её как константу с помощью ключевого слова final, если написать его перед указанием типа данных переменной.

```
public class FirstApp {
```

```
    public static void main(String[] args) {
        final int a = 20;
    }
}
```

Динамическая инициализация переменных. Начальные значения переменных могут рассчитываться на основе значений других переменных. В примере ниже, для заполнения переменной volume используются значения переменных radius и height.

```
public class FirstApp {
    public static void main(String[] args) {
        float radius = 2.0f, height = 10.0f;
        // volume инициализируется динамически во время выполнения программы
        float volume = 3.1416f * radius * radius * height;
        System.out.println("Объем цилиндра равен " + volume);
    }
}
```

Инициализация нескольких переменных в одну строку. Если требуется несколько переменных одного типа, их можно объявить в одном операторе через запятую.

```
public class FirstApp {
    public static void main(String[] args) {
        int x, y, z;
        x = y = z = 10; // присвоить значение 10 переменным x, y и z
        float d = 2.2f, e = 7.2f;
    }
}
```

Арифметические операции

Вы можете выполнять обычные арифметические операции над числовыми переменными.

Операция	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Деление по модулю
++	Инкремент (приращение на 1)
+=	Сложение с присваиванием
-=	Вычитание с присваиванием
*=	Умножение с присваиванием
/=	Деление с присваиванием
%=	Деление по модулю с присваиванием
--	Декремент (отрицательное приращение на 1)

Давайте посмотрим как это будет выглядеть в коде.

```
public class FirstApp {
    public static void main(String args[]) {
        int a = 10;
        int b = 20;
        int c = (a + b - 5) * 2;
        System.out.println("c = " + c);
    }
}
```

В примере выше мы объявили и проинициализировали две целочисленные переменные a и b, а затем объявили и проинициализировали переменную c, которая вычисляется с использованием выражения, состоящего из набора простых арифметических операций.

Давайте еще посмотрим на арифметические операции над переменными.

```
public class FirstApp {
    public static void main(String args[]) {
        int a = 10; // тут a = 10
        a++; // увеличить значение переменной a на 1, получим a = 11
        a--; // уменьшить значение переменной a на 1, получим a = 10
        a += 10; // увеличить значение переменной a на 10, получим a = 20
        a *= 2; // умножить значение переменной a на 2, получим a = 40
        a /= 4; // поделить значение переменной a на 2, получим a = 10
        int b = a + 4; // объявили переменную b и задали ей значение, равное
значению переменной a увеличенной на 4, получим b = 14
    }
}
```

Ещё одна простая программа

Вот так может выглядеть ещё одна программа, написанная на языке Java.

```
public class FirstApp {
    public static void main(String args[]) {
        int a;
        int b;
        a = 128;
        System.out.println("a = " + a);
        b = a / 2;
        System.out.println("b = a / 2 = " + b);
    }
}
```

Важно!

- System.out.println(); отвечает за печать сообщений в консоль. В качестве аргумента вы можете подавать любые значения: строки, числа, объекты, логические типы данных и т.д.
- В процессе обучения вам постоянно придется пользоваться этим методом, поэтому с первого же занятия надо запомнить как он пишется.
- Если вы работаете в IntelliJ IDEA, то для быстрого набора этого метода можно использовать сокращение sout.

Выполнение начинается с первой строки метода main() и идет последовательно сверху вниз. Первые две строки в методе main() означают объявление двух целочисленных переменных с идентификаторами a и b. Затем в переменную a записывается число 128 и выводится сообщение в консоль – «a = 128». Затем значение переменной b вычисляется через значение переменной a, как b = a / 2 (т.е. b = 128 / 2 = 64), и в консоль выводится сообщение «b = a / 2 = 64».

Условный оператор if

Условный оператор if позволяет выборочно выполнять отдельные части программы. Ниже приведена простейшая форма оператора if.

```
if (условие) {
   последовательность_операторов;
}
```

Здесь условие обозначает логическое выражение. Если (условие) истинно (true), последовательность операторов выполняется, если ложно (false) – не выполняется, например.

```
public class FirstApp {
    public static void main(String args[]) {
        if (5 < 10) {
            System.out.println("5 меньше 10");
        }
    }
}
```

В данном примере числовое значение 5 меньше 10, и поэтому условное выражение принимает логическое значение true, а следовательно, выполняется метод println().

Рассмотрим ещё один пример с противоположным условием.

```
public class FirstApp {
    public static void main(String args[]) {
        if (10 < 5) {
            System.out.println("Это сообщение никогда не будет выведено");
        }
    }
```

```
}
```

Теперь числовое значение 10 не меньше 5, а следовательно, метод println() не вызывается, и в консоль ничего не выводится.

Могут быть использованы следующие операторы сравнения.

Оператор	Значение
<	Меньше
<=	Меньше или равно
>	Больше
>=	Больше или равно
==	Равно
!=	Не равно

Обратите внимание, что для проверки на равенство указывается два знака равно.

Для проверки значения типа boolean используется запись.

```
public class FirstApp {
    public static void main(String args[]) {
        boolean bool = true;
        if (bool) { // если bool == true
            // ...
        }
        if (!bool) { // если bool == false
            // ...
        }
    }
}
```

Ниже приведён пример программы, демонстрирующий применение оператора if.

```
public class FirstApp {
    public static void main(String args[]) {
        // объявляем и инициализируем три переменные
        int a = 2, b = 3, c = 0;

        if (a < b) {      // если a меньше b
            System.out.println("a меньше b");
        }
        if (a == b) {    // если a равно b
            System.out.println("a равно b. Это сообщение не будет выведено");
        }
        c = a - b; // переменная c = 2 - 3 = -1
        System.out.println("c = " + c);
        if (c >= 0) {
            System.out.println("c не отрицательно");
        }
        if (c < 0) {
```

```
            System.out.println("c отрицательно");
        }
        c = b - a; // переменная c = 3 - 2 = 1
        System.out.println("c = " + c);
        if (c >= 0) {
            System.out.println("c неотрицательно");
        }
        if (c < 0) {
            System.out.println("c отрицательно");
        }
    }
}

// Результат:
// a меньше b
// c = -1
// c отрицательно
// c = 1
// c неотрицательно
```

Ещё один вариант условного оператора if-else представлен ниже. Если условие верно, выполняется последовательность_операторов_1, если нет – последовательность_операторов_2 (из блока else).

```
if (условие) {
  последовательность операторов 1
} else {
  последовательность операторов 2
}
```

При использовании условий можно составлять более сложные конструкции с помощью логических операторов И(&&) и ИЛИ(||).

```
if (условие1 && условие2) {
   ...
}
if (условие1 || условие2) {
   ...
}
if ((условие1 && условие2) || условие3) {
   ...
}
```

В первом случае (логическое И) для выполнения кода из блока if, необходимо чтобы и условие1, и условие2 одновременно были верны. Во втором случае (логическое ИЛИ) достаточно, чтобы хотя бы одно из условий было верно.

Создаем простые методы

Писать весь код в методе main() не очень хорошая идея, даже если он будет состоять из нескольких сотен строк кода, разбираться в нем будет очень сложно. Представьте что вам захотелось рисовать котов в консоли, и вы написали код вроде этого:

```
public class MainApp {
   public static void main(String[] args) {
       System.out.println("           _..---...,\"\"-._     ,/}/)\n" +
               "         .''         ,       ``..'(/-<\n" +
               "        /    _       {       )          \\\n" +
               "       ;    _ `.      `.    <          a(\n" +
               "     ,'    ( \\  )       `.  \\ __..._ .: y\n" +
               "    (  <\\_-) )'-.____...\\  `._    //-'\n" +
               "     `. `-' /-._)))      `-._)))\n" +
               "       `...'          ");
   }
}
```

Получим:

И по какой-то неведомой причине вам захотелось четыре раза подряд нарисовать этого кота. Во что превратится ваш код? А вот во что:

```
public class FirstApp {
   public static void main(String[] args) {
       System.out.println("           _..---...,\"\"-._     ,/}/)\n" +
               "         .''         ,       ``..'(/-<\n" +
               "        /    _       {       )          \\\n" +
               "       ;    _ `.      `.    <          a(\n" +
               "     ,'    ( \\  )       `.  \\ __..._ .: y\n" +
               "    (  <\\_-) )'-.____...\\  `._    //-'\n" +
               "     `. `-' /-._)))      `-._)))\n" +
               "       `...'          ");
       System.out.println("           _..---...,\"\"-._     ,/}/)\n" +
               "         .''         ,       ``..'(/-<\n" +
```

```
                "       /   _      {      )         \\\n" +
                "      ;   _ `.     `.   <         a(\n" +
                "    ,'   ( \\  )      `.  \\ __.._ .: y\n" +
                "   (  <\\_-) )'-.____...\\  `._   //-'\n" +
                "    `. `-' /-._)))      `-._)))\n" +
                "      `...'         ");
        System.out.println("           _..---...,\"\"-._     ,/}/)\n" +
                "        .''        ,      ``..'(/-<\n" +
                "       /   _      {      )         \\\n" +
                "      ;   _ `.     `.   <         a(\n" +
                "    ,'   ( \\  )      `.  \\ __.._ .: y\n" +
                "   (  <\\_-) )'-.____...\\  `._   //-'\n" +
                "    `. `-' /-._)))      `-._)))\n" +
                "      `...'         ");
        System.out.println("           _..---...,\"\"-._     ,/}/)\n" +
                "        .''        ,      ``..'(/-<\n" +
                "       /   _      {      )         \\\n" +
                "      ;   _ `.     `.   <         a(\n" +
                "    ,'   ( \\  )      `.  \\ __.._ .: y\n" +
                "   (  <\\_-) )'-.____...\\  `._   //-'\n" +
                "    `. `-' /-._)))      `-._)))\n" +
                "      `...'         ");
        System.out.println("           _..---...,\"\"-._     ,/}/)\n" +
                "        .''        ,      ``..'(/-<\n" +
                "       /   _      {      )         \\\n" +
                "      ;   _ `.     `.   <         a(\n" +
                "    ,'   ( \\  )      `.  \\ __.._ .: y\n" +
                "   (  <\\_-) )'-.____...\\  `._   //-'\n" +
                "    `. `-' /-._)))      `-._)))\n" +
                "      `...'         ");
    }
}
```

Что можно сделать, чтобы код выглядел чуть аккуратнее? Мы можем вынести рисование кота в отдельный блок кода - **метод**.

```
public class FirstApp {
    public static void main(String[] args) {
        // ...
    }

    public static void drawCat() {
        System.out.println("           _..---...,\"\"-._     ,/}/)\n" +
                "        .''        ,      ``..'(/-<\n" +
                "       /   _      {      )         \\\n" +
                "      ;   _ `.     `.   <         a(\n" +
                "    ,'   ( \\  )      `.  \\ __.._ .: y\n" +
                "   (  <\\_-) )'-.____...\\  `._   //-'\n" +
                "    `. `-' /-._)))      `-._)))\n" +
                "      `...'         ");
    }
```

```
}
```

Когда вы выносите код в отдельный метод, этому методу дается имя, при этом имя должно отражать назначение метода. Наш метод мы назвали как “нарисовать кота”. Ну давайте посмотрим что нам это дало.

```
public class FirstApp {
   public static void main(String[] args) {
      drawCat();
      drawCat();
      drawCat();
      drawCat();
   }

   public static void drawCat() {
       System.out.println("            _..---...,\"\"-._     ,/})\n" +
               "         .''         ,       ``..'(/-<\n" +
               "        /    _       {       )          \\\n" +
               "       ;    _ `.      `.    <          a(\n" +
               "     ,'    ( \\  )       `.  \\ __..._ .: y\n" +
               "    (  <\\_-) )'-.____...\\  `._    //-'\n" +
               "     `. `-' /-._)))       `-._)))\n" +
               "       `...'          ");
   }
}
```

Неправда ли выглядит значительно аккуратнее? Все что касается рисования кота было вынесено в метод drawCat(), а в методе main() мы 4 раза вызвали метод drawCat(). Вызов можно себе представить как будто бы на месте имени метода, стоит код, который в нем написан.

На первом занятии вы должны запомнить следующую форму создания методов.

```
public class FirstApp {
   public static void имя_метода() {
       // тело_метода
   }
}
```

Имя_метода отражает назначение данного блока кода, **тело_метода** - то, что он должен сделать, если его вызвали. Про то, что такое void и пустые круглые скобки вы узнаете на следующем занятии, пока вам надо научиться работать с простым вариантом методов.

Важно!

- Методы не могут быть объявлены за пределами тела класса;
- Методы должны иметь понятные названия;
- Имя метода записывается в camelCase;.

В конце урока вы узнали ответы на вопросы:

1) С чего начинается выполнение программы на языке Java?
2) Как выглядит объявление метода main?
3) Какие примитивные типы данных представлены в Java, и что в них можно хранить?
4) Что такое переменные и зачем они нужны?
5) Каких правил следует придерживаться при именовании переменных и методов?
6) Что такое методы и зачем они нужны? Что подразумевается под фразой “метод возвращает целочисленное значение”?
7) Для чего нужны условия, что они позволяют сделать?

В конце урока научитесь:

1) Создавать новый проект в IntelliJ IDEA;
2) Создавать метод main();
3) Объявлять и инициализировать переменные;
4) Выводить сообщения в консоль с помощью System.out.println();
5) Писать хотя бы простые условия (без &&, ||, else);
6) Создавать простые методы;

Домашнее задание

1. Создать пустой проект в IntelliJ IDEA, создать класс HomeWorkApp, и прописать в нем метод main().
2. Создайте метод **printThreeWords()**, который при вызове должен отпечатать в столбец три слова: Orange, Banana, Apple.

```
Orange
Banana
Apple
```

3. Создайте метод **checkSumSign()**, в теле которого объявите две int переменные a и b, и инициализируйте их любыми значениями, которыми захотите. Далее метод должен просуммировать эти переменные, и если их сумма больше или равна 0, то вывести в консоль сообщение “Сумма положительная”, в противном случае - “Сумма отрицательная”;
4. Создайте метод **printColor()** в теле которого задайте int переменную value и инициализируйте ее любым значением. Если value меньше 0 (0 включительно), то в консоль метод должен вывести сообщение “Красный”, если лежит в пределах от 0 (0 исключительно) до 100 (100 включительно), то “Желтый”, если больше 100 (100 исключительно) - “Зеленый”;
5. Создайте метод **compareNumbers()**, в теле которого объявите две int переменные a и b, и инициализируйте их любыми значениями, которыми захотите. Если a больше или равно b, то необходимо вывести в консоль сообщение “a >= b”, в противном случае “a < b”;
6. Методы из пунктов 2-5 вызовите из метода main() и посмотрите корректно ли они работают.

Если выполнение задач вызывает трудности, можете обратиться к последней главе методического пособия “Подсказки по домашнему заданию”.

Дополнительные материалы

1. К. Сьерра, Б. Бейтс Изучаем Java // Пер. с англ. – М.: Эксмо, 2012. – 720 с.
2. Кей С. Хорстманн, Гари Корнелл Java. Библиотека профессионала. Том 1. Основы // Пер. с англ. – М.: Вильямс, 2014. - 864 с.

3. [Если набрались опыта в написании кода на Java и хотите понять основы Git] https://youtu.be/SQUcNEO8fxU

Используемая литература

1. Брюс Эккель Философия Java // 4-е изд.: Пер. с англ. – СПб.: Питер, 2016. – 1168 с.
2. Г. Шилдт Java 8. Полное руководство // 9-е изд.: Пер. с англ. – М.: Вильямс, 2015. – 1376 с.
3. Г. Шилдт Java 8: Руководство для начинающих. // 6-е изд.: Пер. с англ. – М.: Вильямс, 2015. – 720 с.

Подсказки по домашнему заданию

Как быть если вы посмотрели вебинар, прочитали методичку, посмотрели на домашнее задание и не понимаете как к нему подойти? Это не страшно, такое вполне возможно, каждый из нас когда-то начинал учиться программировать с нуля. Давайте попробуем вместе выполнить похожее домашнее задание.

1. **Создать пустой проект в IntelliJ IDEA, создать класс HomeWorkApp, и прописать в нем метод main().**

Для выполнения этого задание повторите за преподавателем процесс создания проекта и создайте новый класс с указанным именем. Должно получиться что-то вроде:

```
public class HomeWorkApp {
}
```

Замечательно, переходим к пункту 2.

2. **Создайте метод drawSquare(), который при вызове должен отпечатать рисунок квадрата.** (*Вы же не думали что мы будем решать именно ваше дз?*)

Квадрат наверное может выглядеть вот так:

```
-----
|   |
-----
```

Как это превратить в код?

```
public class HomeWorkApp {
   public static void main(String[] args) {
      drawSquare();
   }

   public static void drawSquare() {
      System.out.println("-----");
      System.out.println("|   |");
      System.out.println("-----");
   }
}
```

Ну вот, ничего сложного. Метод создали, хорошее имя ему дали, сказали что при вызове он должен отпечатать в консоль три строки. Чтобы проверить корректность, да и вообще чтобы метод был выполнен, мы вызвали его из метода main().

3. **Создайте метод checkHomework() в теле которого объявлена int переменная tasksCount (количество выполненных задач). В зависимости от значения переменной, метод должен напечатать в консоль оценку студента.**

```
public class HomeWorkApp {
   public static void main(String[] args) {
      checkHomework();
   }

   public static void checkHomework() {
      int tasksCount = 3;

      if (takskCount == 6) {
         System.out.println("Великолепно, Вы выполнили все задачи!");
      }
      if (tasksCount > 3 && tasksCount < 6) {
         System.out.println("Отлично, Вы выполнили почти все задачи!");
      }
      if (tasksCount <= 3) {
         System.out.println("Вы решили меньше половины, но Вы все равно
молодец, надо же с чего-то начинать!!");
      }

      System.out.println("Проверка завершена");
   }
}
```

В данном случае мы добавили к формулировке дз немного творчества (так делать не надо), это сделано для того, чтобы показать вам как текстовая формулировка задачи может преобразовываться в код. Кроме того, корректнее было бы использовать конструкцию if-else-if, но мы же смотрим пока самые простые варианты.

4. **Создайте метод howIsTheWeatherThere() в теле которого объявлена int переменная temperature. В зависимости от значения переменной, метод должен оценить погоду на улице.**

```
public class HomeWorkApp {
   public static void main(String[] args) {
      howIsTheWeatherThere();
   }

   public static void howIsTheWeatherThere() {
      int temperature = 15;

      if (temperature > 10) {
         System.out.println("На улице очень даже тепло для марта месяца");
      } else if (temperature > -5) {
         System.out.println("Для марта самое то");
      } else {
         System.out.println("Там точно весна?");
      }
   }
}
```

Заметка. Как собрать решение всех задач вместе? А вот так:

```
public class HomeWorkApp {
   public static void main(String[] args) {
      drawSquare();
      checkHomework();
      howIsTheWeatherThere();
   }

   public static void drawSquare() {
      System.out.println("-----");
      System.out.println("|   |");
      System.out.println("-----");
   }

   public static void checkHomework() {
      int tasksCount = 3;

      if (takskCount == 6) {
         System.out.println("Великолепно, Вы выполнили все задачи!");
      }
      if (tasksCount > 3 && tasksCount < 6) {
         System.out.println("Отлично, Вы выполнили почти все задачи!");
      }
      if (tasksCount <= 3) {
         System.out.println("Вы решили меньше половины, но Вы все равно
молодец, надо же с чего-то начинать!!");
      }

      System.out.println("Проверка завершена");
   }

   public static void howIsTheWeatherThere() {
      int temperature = 15;

      if (temperature > 10) {
         System.out.println("На улице очень даже тепло для марта месяца");
      } else if (temperature > -5) {
         System.out.println("Для марта самое то");
      } else {
         System.out.println("А там точно весна?");
      }
   }
}
```

При запуске программы начинает выполняться метод main(), который в свою очередь последовательно выполняет каждый из методов, так как мы их вызываем в теле метода main().