**Лабораторная работа №1**

**Цель работы:** разработка консольного приложения для реализации простейшей программы на Java.

Реализация первой программы на языке Java выглядит следующим образом:

class MyFirstProgramm{

public static void main(String[] args){

System.out.println("My first Java programm!!");

}

}

Вот и все, всего пять строчек! Но даже на этом простом примере можно заметить целый ряд существенных особенностей языка Java.

* Всякая программа представляет собой один или несколько классов, в этом простейшем примере только один класс (class).
* Начало класса отмечается служебным словом class, за которым следует имя класса, выбираемое произвольно, в данном случае Helloworld. Все, что содержится в классе, записывается в фигурных скобках и составляет тело класса (class body).
* Все действия производятся с помощью методов обработки информации, коротко говорят просто метод (method). Это название употребляется в языке Java вместо названия "функция", применяемого в других языках.
* Методы различаются по именам. Один из методов обязательно должен называться main, с него начинается выполнение программы. В нашей простейшей программе только один метод, а значит, имя ему main.
* Как и положено функции, метод всегда выдает в результате (чаще говорят, возвращает (returns)) только одно значение, тип которого обязательно указывается перед именем метода. Метод может и не возвращать никакого значения, играя роль процедуры, как в нашем случае. Тогда вместо типа возвращаемого значения записывается слово void, как это и сделано в примере.
* После имени метода в скобках, через запятую, перечисляются аргументы (arguments) -или параметры метода. Для каждого аргумента указывается его тип и, через пробел, имя. В примере только один аргумент, его тип — массив, состоящий из строк символов. Строка символов — это встроенный в Java API тип String, а квадратные скобки — признак массива. Имя массива может быть произвольным, в примере выбрано имя args.
* Перед типом возвращаемого методом значения могут быть записаны модификаторы (modifiers). В примере их два: слово public означает, что этот метод доступен отовсюду; слово static обеспечивает возможность вызова метода main () в самом начале выполнения программы. Модификаторы вообще необязательны, но для метода main () они необходимы.

|  |
| --- |
| **Замечание.** После имени метода ставятся скобки, чтобы подчеркнуть, что это имя именно метода, а не простой переменной. Все, что содержит метод, тело метода (method body), записывается в фигурных скобках. |

Единственное действие, которое выполняет метод main () в примере, заключается в вызове другого метода со сложным именем System.out.println и передаче ему на обработку одного аргумента, текстовой константы "My first Java programm!!". Текстовые константы записываются в кавычках, которые являются только ограничителями и не входят в состав текста.

Составное имя System.out.println означает, что в классе System, входящем в Java API, определяется переменная с именем out, содержащая экземпляры одного из классов Java API, класса PrintStream, в котором есть метод println() . Все это станет ясно позднее, а пока просто будем писать это длинное имя.

Действие метода println () заключается в выводе своего аргумента в выходной поток, связанный обычно с выводом на экран текстового терминала, в окно MS-DOS Prompt или Command Prompt или Xterm, в зависимости от вашей системы. После вывода курсор переходит на начало следующей строки экрана, на что указывает окончание ln, слово println — сокращение слов print line. В составе Java API есть и метод print (), оставляющий курсор в конце выведенной строки. Разумеется, это прямое влияние языка Pascal.

|  |
| --- |
| **Замечание.** Язык Java различает строчные и прописные буквы, имена main, Main, MAIN различны с "точки зрения" компилятора Java. В примере важно писать String, System с заглавной буквы, a main с маленькой. Но внутри текстовой константы неважно, писать "MY FIRST JAVA PROGRAMM!!" или "My First Java Programm!!", компилятор вообще не "смотрит" на нее, разница будет видна только на экране. |

Итак, программа написана в каком-либо текстовом редакторе, например, Notepad. Теперь ее надо сохранить в файле, имя которого совпадает с именем класса, содержащего метод main (), и дать имени файла расширение java. Это правило очень желательно выполнять. При этом система исполнения Java будет быстро находить метод main() для начала работы, просто отыскивая класс, совпадающий с именем файла.

|  |
| --- |
| **Замечание.** Называйте файл с программой именем класса, содержащего метод main (), соблюдая регистр букв. |

В нашем примере, сохраним программу в файле с именем **MyFirstProgramm.java** в текущем каталоге. Затем вызовем компилятор, передавая ему имя файла в качестве аргумента:

javac MyFirstProgramm.java

Компилятор создаст файл с байт-кодами, даст ему имя MyFirstProgramm.class и запишет этот файл в текущий каталог.

Осталось вызвать интерпретатор, передав ему в качестве аргумента имя класса (а не файла):

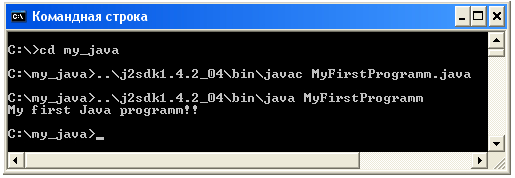
java MyFirstProgramm

На экране появится:

My first Java programm!!

|  |
| --- |
| **Замечание.** Не указывайте расширение class при вызове интерпретатора. |

На рис. 1. показано, как все это выглядит в окне Command Prompt.



*Рис. 1. Окно  Command Prompt*

Выше было показано, что компилируется и запускается программа на Java, не содержащая объявления пакета. На практике, однако, все нетривиальные программы содержат объявления package. При наличии объявления пакета процесс компиляции и запуска программы на Java несколько усложняется. Как было замечено ранее, программа на Java должна записываться в файл с именем, совпадающим с именем класса. Когда класс включается в пакет, добавляются требование, состоящее в том, что имя каталога, в котором храниться этот файл, совпадало с именем пакета.

Давайте введем в **MyFirstProgramm.java** объявление пакета:

package ivanov;

class MyFirstProgramm{

public static void main(String[] args){

System.out.println("My first Java programm!!");

}

}

Создайте теперь каталог, в котором будет храниться данная программа, а именно **/ivanov.** Скопируйте программу **MyFirstProgramm.java** в этот каталог. Получится файл с именем: **/ivanov/ MyFirstProgramm.java**

Перед компиляцией перейдите в каталог **/ivanov** и запустите компилятор как раньше.

javac MyFirstProgramm.java

Результатом будет файл **MyFirstProgramm.class**.

Для вызова интерпретатора необходимо выйти из папки **/ivanov** и далее передать ему в качестве аргумента имя класса следующим образом:

**java ivanov.MyFirstProgramm**

**Пример консольного java-приложения**

Задание: Ввести n строк с консоли. Вывести на консоль строки и их длины, упорядоченные по возрастанию.

Решение:

import java.io.IOException;

import java.util.InputMismatchException;

import java.util.Scanner;

public class Main

{

public static void main(String[] args)

{

int n = 0;

while ( true ) // ввод числа строк

{

System.out.println("Введите число строк");

Scanner sc1 = new Scanner(System. in );

try

{

n = sc1.nextInt();

break;

}

catch(InputMismatchException fg)

{

// если введенное значение не является числом

System.out.print("Вы ввели не число. " );

}

}

// создание массива строк

String[] str = new String[n];

Scanner sc2 = new Scanner(System.in);

for (int i = 0; i < n; i++)

{

System. out.println( " Введите строку №" + (i+1));

str[i] = sc2.nextLine();

}

//сортировка массива строк по длине

for ( int i = 0; i < str. length -1; i++)

{

for (int j = i+1; j < str.length; j++)

{

if (str[i].length()<str[j].length())

{

String temp = str[i];

str[i] = str[j];

str[j] = temp;

}

}

}

int maxlength = str[0].length();

System.out.println("Строки в порядке убывания длины:");

for ( int i = 0; i < str. length ; i++)

{

System.out.print(str[i]);

for (int j = 0; j < maxlength - str[i].length(); j++)

System.out.print(" ");

System. out.println( " её длина = " + str[i].length());

}

}

}

**Задания к лабораторной работе**

1. Ввести n строк с консоли, найти самую короткую строку. Вывести эту строку и ее длину.
2. Ввести n строк с консоли. Упорядочить и вывести строки в порядке возрастания их длин, а также (второй приоритет) значений этих их длин.
3. Ввести n строк с консоли. Вывести на консоль те строки, длина которых меньше средней, также их длины.
4. В каждом слове текста k-ю букву заменить заданным символом. Если k больше длины слова, корректировку не выполнять.
5. В русском тексте каждую букву заменить ее номером в алфавите. В одной строке печатать текст с двумя пробелами между буквами, в следующей строке внизу под каждой буквой печатать ее номер.
6. Из небольшого текста удалить все символы, кроме пробелов, не являющиеся буквами. Между последовательностями подряд идущих букв оставить хотя бы один пробел.
7. Из текста удалить все слова заданной длины, начинающиеся на согласную букву.
8. В тексте найти все пары слов, из которых одно является об­ращением другого.
9. Найти и напечатать, сколько раз повторяется в тексте каждое слово.
10. Найти, каких букв, гласных или согласных, больше в каждом предложении текста.
11. Выбрать три разные точки заданного на плоскости множества точек, составляющие треугольник наибольшего периметра.
12. Найти такую точку заданного на плоскости множества точек, сумма расстояний от которой до остальных минимальна.
13. Выпуклый многоугольник задан на плоскости перечислением координат вершин в порядке обхода его границы. Определить площадь многоугольника.

Источник: <http://www.4stud.info/java-programming/creating-console-application.html>