Назначение лекции

Вы узнаете, где применяется язык Java и выглядит программный код. Познакомитесь со средой разработки и формированием простейшего консольного приложения, а также с понятием класса и объекта. Вы узнаете, что такое данные в программе, переменные, их тип и действия по записи в переменные различных значений. Вспомните основные операторы Java, пришедшие из С++.

Новые слова

**Java** - объектно-ориентированный язык программирования.

**IntelliJ IDEA** – среда разработки для Java.

**Класс** – программный шаблон, который используется для создания объекта. Содержит описание всех свойств и методов объекта. Сам код класса не может выполнятся. Для выполнения действий, описанных в классе, требуется создать объект.

**Объект** – экземпляр класса. Наследует свойства и методы класса.

**Модификатор** – определяет характер доступа к членам класса (общедоступный, локальный и пр.)

**Конструктор клас**са – специальная подпрограмма класса, которая автоматически выполняется при создании объекта. Имя конструктора должно совпадать с именем класса.

**Тип данных** – понятие в программировании, определяющее количество и структуру памяти для размещения данных.

**Переменная** – символьный адрес места в памяти для данных.

**Присваивание** – запись в переменную конкретных данных.

**Тринарный оператор** – краткая запись условия, которое можно использовать в правой части присваивания.

# Введение

Java — объектно-ориентированный язык программирования, который был разработан Джеймсом Гослингом из компании Sun Microsystems (которая позже была приобретена компанией Oracle Corporation).

Первая версия была выпущена в 1995 году. В настоящее время Java является одним из самых популярных языков программирования.

Язык может использоваться для настольных приложений, веб-приложений и приложений для мобильных устройств.

Одна из главных особенностей Java — *платформенная независимость*. Это означает, что программа, написанная на Java, может выполняться в любой операционной системе (Window, Mac или Linux).

Механизм платформенной независимости заключается в следующем: Java на выходе получает промежуточный код (байт-код). Этот код поступает на вход **виртуальной** машины Java. А эта машина преобразуется в машинный код той операционной системы, где установлена Java машина.

Как следствие на Java можно писать программу, не думая о том, где она будет работать. Это в теории, а на практике – это не совсем так и немного думать приходиться.

Второе следствие – это замедление выполнения программы. Это критично для задач системного характера или задач реального времени, но для других приложений это не имеет большого значения.

# Код Java

Синтаксически Java основан на языках С и С++. Пример кода на Java:

int counter = 1;  
int sum = 0;  
while (counter < 5)

{  
 sum = sum + counter;  
 counter++;

}

# Среда разработки

Прежде чем браться за разработку приложений на Java, необходимо загрузить среду разработки (*IDE- Integrated Development Environment*). Их несколько. Одну из них мы будем использовать: IntelliJ IDEA. Это бесплатная версия, но есть и платная.

Кроме нее есть и другие, например, *Eclipse* и *NetBeans*. Оба продукта доступны для бесплатной загрузки.

Разработчику Java следует знать несколько сокращений:

*JDK (Java Development Kit)* — бесплатный набор инструментов от компании *Oracle*, упрощающих разработку приложений Java. К числу таких инструментов относится компилятор для преобразования написанного кода в байт-код (javac.exe), архиватор для упаковки и распространения файлов Java (jar.exe) и генератор документации для построения документации в формате HTML на основе кода Java (javadoc.exe).

Кроме того, *JDK* также включает исполнительную среду Java (*Java Runtime Environment*, или *JRE*). *JRE* содержит виртуальную машину Java (*JVM*) и ресурсы, необходимые *JVM* для запуска программ Java.

*IntelliJ IDE* все эти компоненты содержит и отдельно устанавливать какой-либо не требуется.

# IntelliJ IDE Community 2022.1

Рассмотрим интерфейс системы, в объеме, достаточном для выполнения лабораторных работ. Более подробно эту информацию можно получить в сети Интернет. Там этой информации более, чем достаточно.

Главное окно системы:

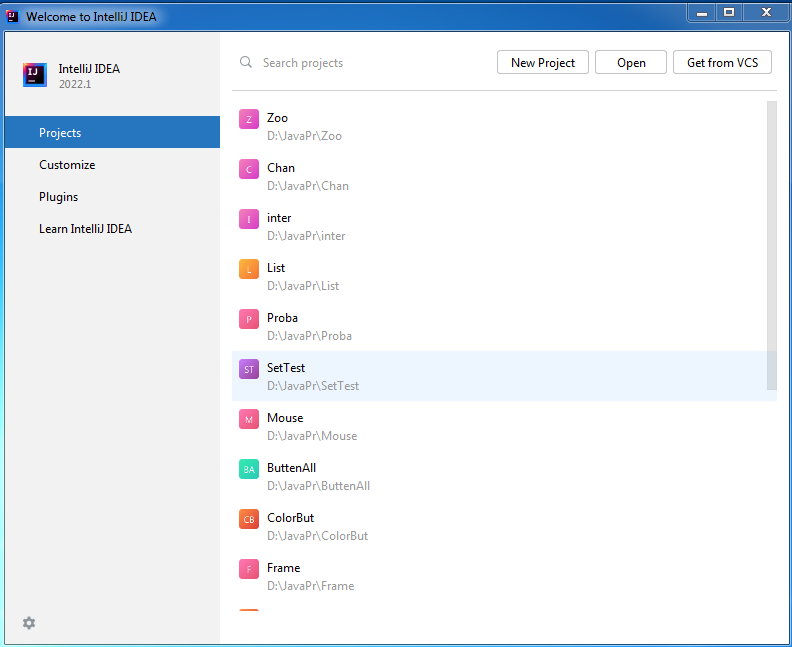


Рис.1. Главное окно IntelliJ IDE

New Project – создание нового проекта

Open – открытие старого проекта.

Для открытия старого проекта достаточно выделить его и нажать «Open». Проект будет открыт.

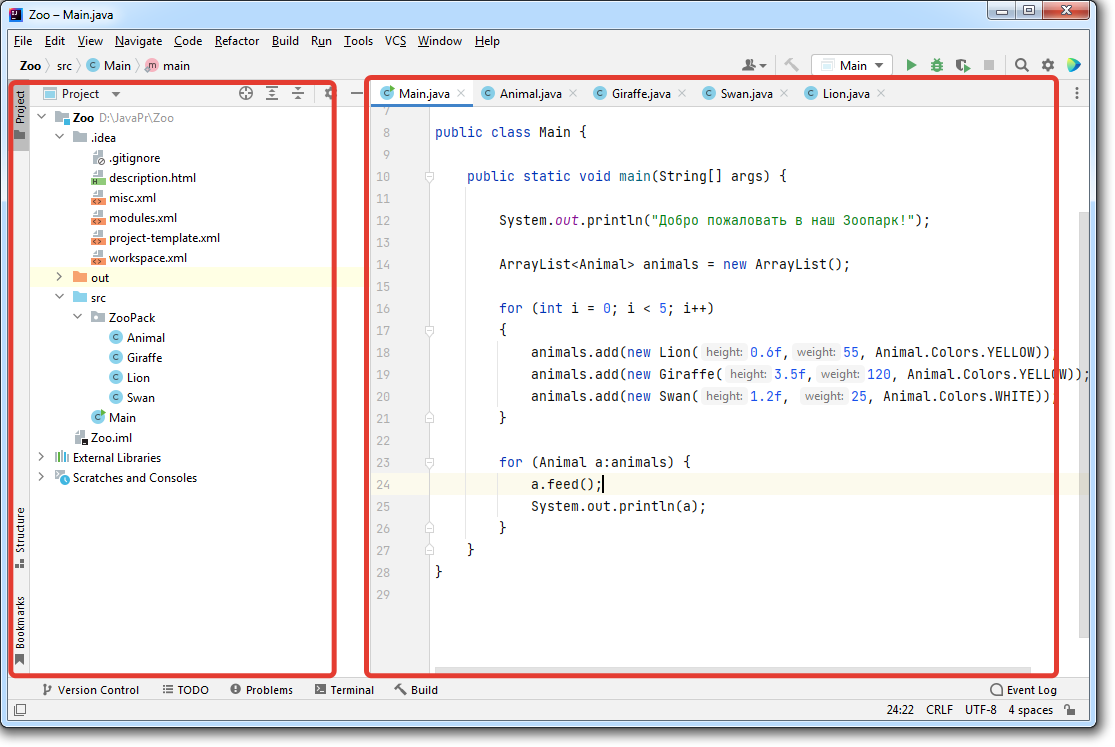


Рис. 2. Проект

Окно состоит из двух главных частей:

Левое – структура проекта

Правая – код главной программы и всех классов

При нажатии кнопки «Выполнение» (рис. 3) проект выполняется.

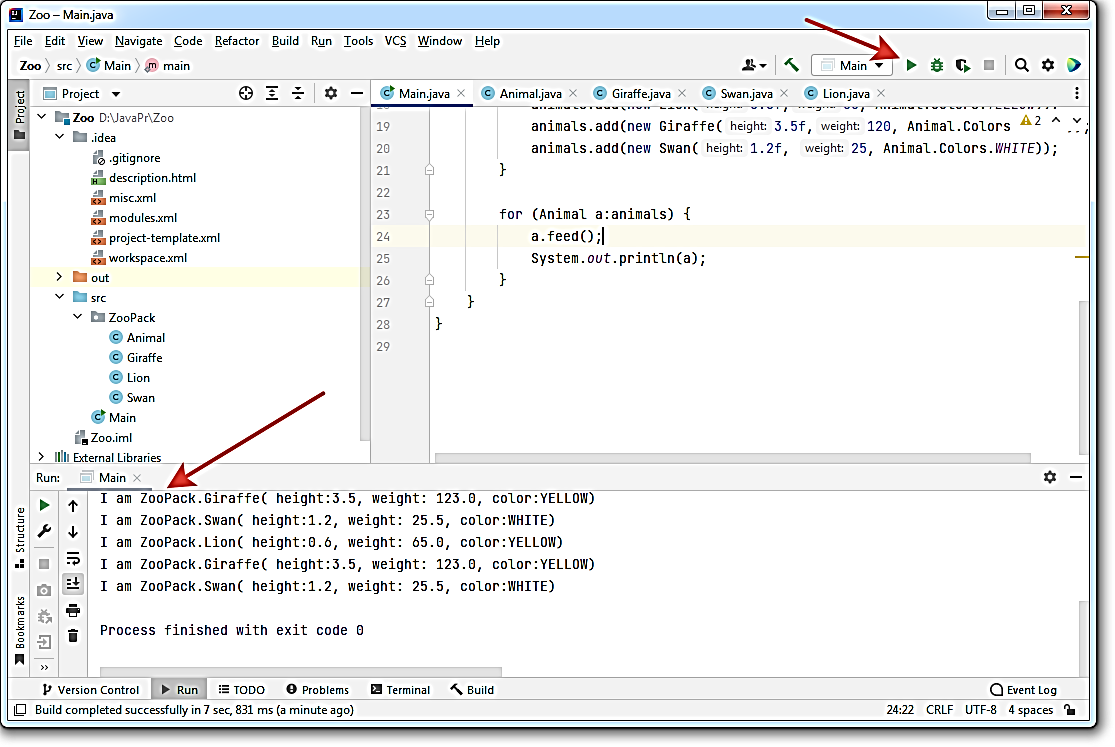


Рис. 3. Выполнение

Если выбрано «New Project», то автоматически формируется код простейшей программы Java:

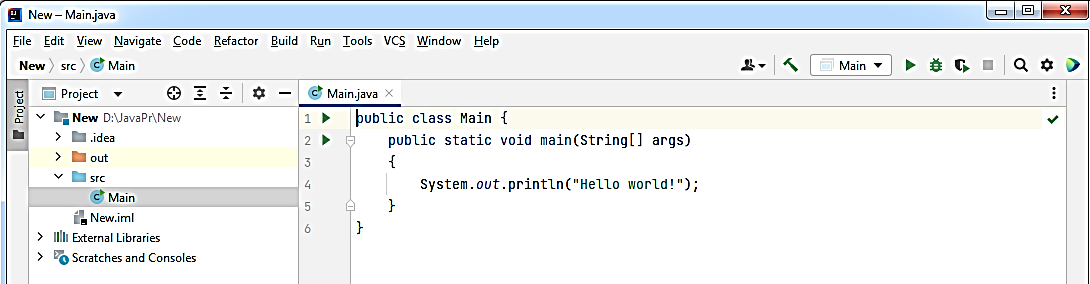


Рис.4

Легко заметить, что программа *Main* объявляется как *class* и внутри находится еще одна *main.* Это так называемый *конструктор* класса.

Внутри конструктора используется оператор выдачи сообщения на экран: *System.out.println(“Hello world”);*

Для выполнения проекта достаточно нажать соответствующую кнопку *run:*

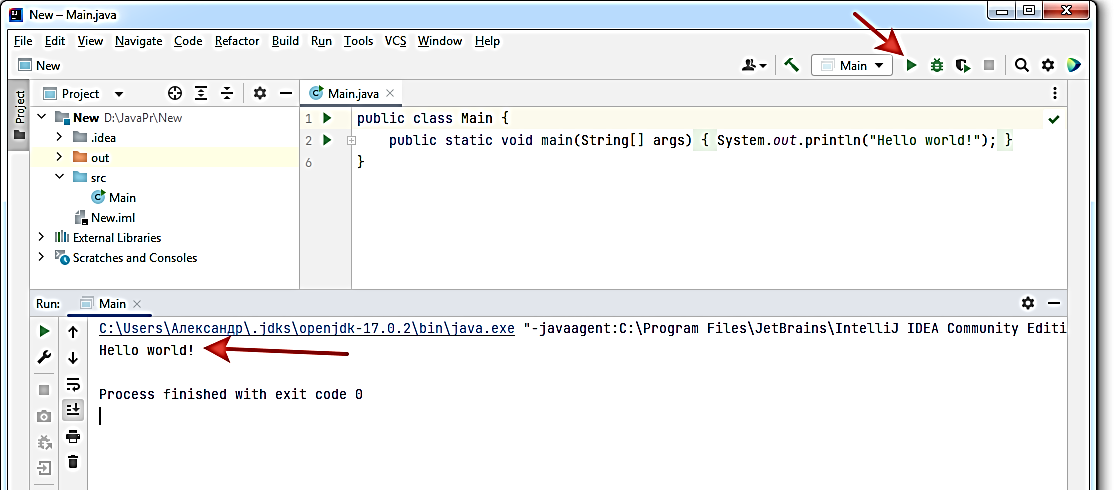


Рис.5. Выполнение программы

Также можно подметить сходство с С++: *main*, фигурные скобки, точка с запятой – в конце операторов. И это не случайность, т.к. Java разрабатывалась как развитие языков С и С++.

Развитие определялось объектно-ориентированными возможностями нового языка. В современном программировании, понятие объект рассматривает как структурную основу при разработки программной системы. Т.е. программная система – это набор объектов, которые взаимодействуют друг с другом.

Каждый объект – это совокупность *свойств* объекта (атрибутов), таких как цвет, размер, вес, количество и пр. и *методов*, которые определяют поведение объекта, меняют состояние объекта или значение свойств. Свойства – это в общем случае переменные или массивы (или др. структуры)

Объект может получать входные данные и формировать выходные данные. Объекты могут взаимодействовать между собой.

Некоторые объекты могут и не содержать свойства или методы, но обычно они присутствуют. Современное программирование состоит в выделении объектов будущего прикладного решения и обеспечение их взаимодействия.

Для того чтобы объект можно было использовать в программе, его надо описать специальным образом. Это описание называется **классом**. Именно в классе описываются все свойства и методы.

Примечание. В Java название класса пишется латинскими буквами и начинается с большой буквой. Например, *Main*

Если класс оформлен, то на основе этого класса можно создать объект.

Таким образом класс - это абстрактный шаблон будущего объекта, а объект — это конкретная реализация класса - экземпляр, который располагается в памяти и может функционировать. Сам класс не может выполнятся. Это возможно только после создания объекта данного класса.

Например, для класса *Employee*, который описывает сотрудника и его зарплату, можно объявить объект на одного сотрудника:

*Employee sotrudnik = new Employee;*

Объект *sotrudnik* получит все свойства и методы класса *Employee*

Пример 2: для класса DomJ, который описывает домашнее животное (название, имя, вес и пр.), можно объявить объект domj для конкретного животного:

***D****omJ* ***d****omj* *= new* ***D****omJ();*

Можно заметить, что название класса начинается с заглавной буквы, а название объекта – с маленькой. Кроме того, у DomJ появились круглые скобки, а в первом примере их не было. Отличие заключается в том, что у класса домашнего животного DomJ есть конструктор и происходит автоматическое обращение к этому конструктору.

Конструктор класса, выполняет некие первоначальные действия, например, устанавливает первоначальные значения свойств объекта. Имя конструктора совпадает с именем класса.

В отличии от традиционных языков программирования типа C++ , программа на Java состоит из набора из классов и объектов, включая и главную программу Main, которая оформлена как класс.

Рассмотрим простейшую программу, на которую ссылаются множество публикаций, в том числе в сети. Это программа «Здравствуй Мир!»:

**public class Main**

**{  
 public static void main(String[] args)  
 {  
 System.*out*.println("Здравствуй Мир!");  
 }  
}**

При запуске программа выдает на экран консоли сообщение «Здравствуй Мир».

Программа состоит из класса Main (с большой буквы), причем общедоступный класс (public), хотя в данном случае можно и без общедоступности:

**class Main**

**{  
 public static void main(String[] args)  
 {  
 System.*out*.println("Здравствуй Мир!");  
 }  
}**

Внутри класса Main находится еще одна програмка main. Это конструктор класса Main. Конструктор автоматически вызывается при создании нового объекта.

Конструктор main тоже общедоступный (public) и неизменяемый (static). Эти слова (public, static) называются **модификаторами** доступа. Их много и будем рассматривать по мере поступления.

Для выдачи на экран консоли используется команда println (взята из языка С). Команда используется для вывода (out) и берется из библиотеки System:

**System.*out*.println("Здравствуй Мир!");**

Особенность метота println заключается в автоматическом переходе на следующую строку. Если требуется продолжать выдачу на текущей строке, то следует использовать метод print:

**System.*out*.print("Здравствуй Мир!");**

Здесь курсор установится в конце фразы **Здравствуй Мир!**

# Данные

Программист, через посредство языка программирования, работает с данными. Данные условно можно разделить на входные, выходные и внутренние.

Входные данные передают программе информацию из внешнего мира, а выходные данные – это результат работы программы. При этом могут использоваться и внутренние данные, например, какие-либо константы (пи, g и пр.)

Данные могут сильно отличаться друг от друга, по размеру, по назначению. Они могут занимать различное количество памяти компьютера и по-разному размещаться в памяти.

# Типы данных

Для того, чтобы это учесть, разработчики языков программирования ввели понятие «**тип**» данных: разным данным присваивается разный тип. В этом случае компьютер понимает сколько памяти выделить для данных и как их там разместить.

В языке Java различают следующие **примитивные** типы данных (программист может определить и свои, не примитивные типы):

**boolean** – хранит один бит информации;

**byte** – хранит один байт информации, и соответственно целочисленные значения от -128 до 127 включительно;

**short** – хранит два байта информации, и соответственно целочисленные значение от -32768 до 32767 включительно;

**int** – хранит четыре байта информации, и соответственно целочисленные значение от -2147483648 до 2147483647 включительно;

**long** – хранит восемь байт информации, и соответственно целочисленные значения от – 9223372036854775808 (-2 в степени 63) до 223372036854775807 (2 в степени 63 минус 1) включительно;

**float** – хранит четыре байта информации, и соответственно значения с плавающей точкой от 1.4х10-45 до 3.4х1038;

**double** – хранит восемь байт информации, и соответственно значения с плавающей точкой от 4.9х10-324 до 1.7х10308;

**char** – хранит 16 бит (2 байта), и соответственно символ в формате UTF.

В основном используют int, float, boolean и char т.к. они покрывают большинство нужд программиста.

В лекции не рассматривается физическое размещение данных в памяти. В сети Интернет достаточно публикаций на эту тему.

Таким образом благодаря понятию тип данных «компьютер» знает **сколько** и **как** разместить данные в памяти. Остается определить **где** они размещены.

# Переменные

Для того чтобы знать **где** располагаются данные в памяти необходимо знать их **адрес.** Вся память компьютера адресуемая. По сути дела, каждый байт памяти имеет свой адрес. Для простоты не будем различать физическую и виртуальную память.

Компьютерный адрес – это целое число от 0 и до максимально возможного на данном компьютере. Все данные размещаются по этим адресам, и не обязательно подряд. Но для программиста такая адресация неудобна. Более привычный символьный адрес. Например, fio, name, sym и пр.

Для удобства программиста, в языках программирования вводится понятие **переменная.** Переменная – это и есть символьный адрес данных, вернее указатель на место размещения данных. Операционная система, когда необходимо, преобразует имя переменной в физический адрес.

Для имен переменных Java используются латинские буквы (кириллица не рекомендуется). Регистр имеет значение: Fio и fio – это разные переменные.

Примеры переменных с типами

int god;

float t;

char sym;

boolean zn;

Имена объектов имеют аналогичный смысл и правила формирования имен соответствуют переменным.

# Присваивание

Итак, мы рассмотрели понятия «тип» и «переменная». Но о самих данных не говорили. Данные – это числа, символы, строки и пр. Данные можно «записать» в переменные по их имени. На самом деле, данные будут записаны в память, адрес которых соответствует имени переменной. Для записи используется **присваивание**. Примеры присваивания:

int god=2022;

float t=15.123;

char sym=’a’;

boolean zn=true;

# Арифметические операции с переменными примитивного типа

С переменными можно выполнять различные арифметические действия: сложение, вычитание, деление и пр.

Например,

int a = 15;

int b = 20;

int c = a + b;

System.out.println («Сумма:" + c); // вывод: Сумма:35

Аналогично,

int a = 15;

int b = 20;

c = a – b;

System.out.println («Разность:" + c); // вывод: Разность:-5

c = a \* b;

System.out.println («Произведение:" + c); // вывод: Произведение:300

c = a / b;

System.out.println («Частное:" + c); // вывод: Частное:0

a/b получилось равным нулю. Это произошло потому, что переменная **с** объявлена целой. Результат 0.75 не является целым и “копьютер” отбрасывает дробную часть, после чего остается ноль.

Как в этом случае следует поступать. Если бы мы работали с языком С++, то достаточно было бы написать:

float d = a / b;

и выдать d на экран (0.75). Но Java более строго относится к типам данных и при таком варианте результат будет d=0.0

Для решения правильно будет написать:

double d = (double) a / b;

System.out.println (”Частное:” + d); // вывод: Частное:0.75

# Сокращение арифметических операций

В языке Java (как и в С++) повсеместно используются сокращения. Эти сокращения не имеют влияние на быстродействие (как было раньше), но сокращают код. И хотя я против таких сокращений (немного путают код), то т.к. в большинстве программ он используется, то целесообразно изложить эти особенности.

Пример 1,

int a = 3;

a = a +1; // результат: 4

int b = 5;

b = b – 1; // результат: 4

можно записать, как

int a = 3;

a++; // Инкремент: переменная a теперь будет иметь значение 4

int b = 5;

b – ; // Декремент: переменная b теперь будет иметь значение 4

Пример 2,

int a = 10;

int b = ++a; // сначала переменная **a** становится 11, затем оно присваивается **b**

Пример 3,

int a = 10;

int b = a++;// переменной **b** присваивается 10 и только потом значение **a**

//увеличивается на 1 и уже равно 11

Пример 4,

int a = 3;

a += 1; // a=a+1, как и инкремент добавляет единицу, результат: 4

Пример 5,

int a = 3;

a += 5; // a=a+5, как и инкремент добавляет 5, результат: 8

Пример 6,

int a = 3;

a - = 1; // a=a-1, как и инкремент убирает единицу, результат: 2

Пример 7,

int a = 3;

a \* = 2; // a=a\*2, как и инкремент умножает на 2, результат: 6

Пример 8,

int a = 3;

a / = 3; // a=a/3, как и инкремент делит на 3, результат: 1

# Операции сравнения

В Java над числами можно не только производить арифметические операции, но также их сравнивать.

Пример 1,

int a = 1;

int b = 2;

boolean comp = a <b;

System.out.println («Результат: " + comp); //результат: **true**

Пример 2,

int a = 1;

int b = 2;

boolean comp = a >b;

System.out.println («Результат: " + comp); //результат: **false**

Пример 3,

boolean a = true;

boolean b = true;

boolean comp = a && b; // логическое И

System.out.println («Результат: " + comp); //результат: **true**

Пример 4,

boolean a = true;

boolean b = false;

boolean comp = a && b;

System.out.println («Результат: " + comp); //результат: **false**

Пример 5,

boolean a = true;

boolean b = false;

boolean comp = a || b; //логическое ИЛИ

System.out.println («Результат: " + comp); //результат: **true**

Пример 6,

boolean a = false;

boolean comp = !a; //логическое НЕ

System.out.println («Результат: " + comp); //результат: **true**

Пример 7,

boolean a = true;

boolean comp = !a; //логическое НЕ

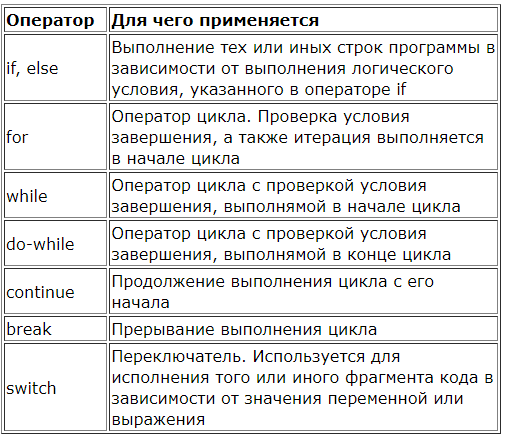
System.out.println («Результат: " + comp); //результат: **false**

# Пример полной программы с данными

**public class Main {  
 public static void main(String[] args) {  
  
 int a = 5;  
 int b = 3;  
 int c = a+b; //c=8  
 int d=c/4; //d=2  
 boolean g=c>d; //g=true  
 boolean h=!g; //h=false  
 System.*out*.println ("Результат: " + h);  
 }  
}**

# Операторы языка Java

В Java вы можете использовать те же самые операторы структурного программирования, что и языке программирования С и С++, а именно:

****

Использование этих операторов не отличается от правил языка С++, например,

**for(i = 0; i < 10; i++)**

**{**

**System.out.print(i + " ");**

**}**

Результат:

**0 1 2 3 4 5 6 7 8 9**

или

**i = 0;**

**while(true)**

**{**

**i++;**

**System.out.print(i + " ");**

**if(i < 9)**

**continue;**

**else**

**break;**

**}**

# Тренарный оператор языка Java

Во многих языках программирования реализован тренарный оператор, который упрощает некоторые действия, связанные с условием. Например, из двух чисел надо выбрать максимальное. Стандартно это можно сделать следующим образом:

**int a = 15;**

**int b = 10;**

**int maxNum;**

**if (a<b) maxNum=b;**

**else maxNum=a;**

**System.out.println («Большее число это: " + maxNum);**

С помощью тренарного оператора это же можно сделать так:

**int a = 15;**

**int b = 10;**

**int maxNum = a <b? b: a;**

**System.out.println («Большее число это: " + maxNum);**

Выражение **a <b? b: a** работает следующим образом: переменной maxNum присваивается значение **b**, если **a<b**, иначе значение **a**

Конец лекции