Язык Java. Синтаксис.

Java – это мультипарадигменный (часто объектно-ориентированный), сильно типизированный, кроссплатформенный язык общего назначения. Является статистическим типизированным языком.

Java применяется:

Серверные технологии

Высоконагруженные системы и базы данных

Распределенные системы

Веб-сервисы

Десктопные приложения

Игры

Телефоны (J2ME, Android)

Апплеты для смарт-карт

Интерактивный контент Blu-Ray

Плюсы Java

+ Java не умирает

+ Совместимость

+Статистическая (и сильная типизация)

+Строгая спецификация языка и виртуальная машина

+Библиотека

+Инструментарий

+Модель памяти

+Скорость

+Безопасность

Инструменты в Java.

Инструментами программиста являются:

Java – язык программирования

JDK (java development kit) – инструментарий разработчика

JVM (Java Virtual machine) – виртуальная машина

JRE (Java Runtime Environment) – среда исполнения

Жизненный цикл.

В языке Java используется определенный жизненный цикл. Но стоит учитывать то, что в скриптовых языках(JavaScript, Python, PHP и т.п.) нет промежуточного представления.

1. Исходный код,
2. Промежуточное представление 1,
3. Промежуточное представление N,
4. Интерпретация (исполнение),

В целом промежуточное представление делится на 2 стадии:

В памяти

В процессоре

Управление памяти.

В языке Java используются *кучи* – это область нагромождения большого количества объектов, по форме обычно по конусу. Поэтому язык Java в своем роде динамичен в этом плане. Ты можешь ввести объект, использовать его и забыть, а сборщик мусора удалит.

*Сборщик мусора*(garbage collection) находит мусор и очищает его. Он страж, который кричит: «Ты, не пройдешь!» и сбивает его в пропасть. Именно он позволяет тебе писать более гибкие программы и просто писать его легче. Управлять им не обязательно, хотя Java представляет способы призвать его.

Базовый синтаксис Java

Инструкция, выражения и «;».

Инструкции или оператор – это наименьшая законченная часть кода.

Выражение – конструкция, созданная из переменных, операций и т.п.

«;» - разделяет инструкции.

Значение и их типы.

В языке java присутствуют значении. У каждого значения свой тип. В общем, существует всего лишь 5 значений:

* Логические.
* Целочисленные.
* Дробные.
* Символьные.
* Ссылочные.

Логические

Логические типы отвечают за логические действия.

К логической группе относят тип *boolean*. В общем boolean может применять true (истинна) и false (ложь). Все они обычно обозначают *условие.*

Целочисленные

Для целых чисел определенно 4 типа: *byte*, *int*, *long*, *short*. В java целочисленные поддерживают положительные и отрицательные числа.

Byte – 8 разрядный тип с диапазоном -128: 127. Является наименьшей по длине в байтах.

Int – 32-разрядный тип с диапазоном -2147483648: 2147483647. Является наиболее распространенным в использовании.

Short - 16 разрядный с диапазоном 32768: 32767.

Long – 64 разрядный с диапазоном -9223372036854775808: 9223372036854775807.

Символьные.

Символьный тип – это знак поддерживающий кодировку Unicode, также с ними можно производить операции.

char ch= 88 эквивалентен char ch=”X” Также его можно декрементировать и инкрементировать.

Ссылочный.

Ссылочный – хранит не самого его, а ссылку на него. К таким можно отнести String и Object, в среде разработки ссылочный тип подсвечивается белым шрифтом. Стоит добавить то, что каждый объект является ссылочным типом данных.

Факты о значениях.

1) Значение есть результат вычисления выражения (за исключение типа void)

2)Значения можно сохранять в переменные (поля, константы и т.д.)

3)Можно передать значение в методы аргументом и принимать в виде параметра.

4)При передаче в метод, присваивании значений копируется.

5)Значение нельзя изменить, но можно присвоить переменной новое значение.

Объявление локальных переменных.

Синтаксис: *поля тип имя = инициализирующее\_значение*

Например:

int I;

I = 256;

final f = 24;

Переменные.

1. Локальная переменная видима в блоке кода, в котором объявлена, включая вложенные блоки.

2. Нельзя объявить переменную с ранее используемым именем.

1. Значение final –переменной можно объявить только один раз.

Операции

Логические.

Логические операции обычно применяются с boolean и используются для логических задач. Основные операции:

|  |  |  |
| --- | --- | --- |
| Операция | Сокращенная операция. | Описание |
| & | && | & обозначает и |
| | | || | | обозначает или |
| ^ |  | Исключающие или ^| |
| == |  | Проверяет на равенство |

Арифметические.

Арифметические операторы — используются в математических выражениях таким же образом, как они используются в алгебре.

Оператор Описание

+ Складывает значения по обе стороны от оператора

- Вычитает правый операнд из левого операнда

\* Умножает значения по обе стороны от оператора

% Делит левый операнд на правый операнд и возвращает остаток

++ Инкремент - увеличивает значение операнда на 1

-- Декремент - уменьшает значение операнда на 1

Полезные вещи

* abc () – позволяет получить модуль числа.
* hash Code () - позволяет получить кэш чего-либо.
* Infinity – бесконечное число.
* NaN – число которое не больше, не меньше и не равно никакому числу (в том числе и себе). Возвращает !себя.
* java.math.BigInteger – позволяет задать любое большое целочисленное число, которое влезет в память.
* java.math.BigDemical - позволяет задать любое большое дробное число, которое влезет в память.

Преобразование.

Преобразование – это приведение переменной одного значения другой с другим типом. Минус преобразования - потеря точности.

Можно выделить два способа: расширяющий и сужающий.

Расширяющий способ преобразования.

Различия Расширяющего способа заключается в том, что не теряется точность числа. Но при этом такую операцию можно использовать, например: целочисленное значение с целочисленным значением. Также в таком способе нужно чтоб одно покрывало другое число. Например:

public class Main {

public static void main (String[] args) {

int i = Integer.MAX\_VALUE;

long l= i;

System.out.println (l);

}

}

Сужающий способ преобразования.

Различие Сужающего способа заключается в том, что теряется точность. Но при этом такую операцию можно провести со всеми типами сразу. Например:

public class Main {

public static void main (String [] args) {

double i = Double.MAX\_VALUE;

long l=(long) i;

System.out.println(l);

}

}

Массивы

Массив – объект в куче, содержащий ряд чего-либо. В общем, массив объявляется:

тип [] имя = new тип [] {содержимое}

import java.util.Arrays.\*;

public class Main {

public static void main(String[] args) {

int [] i = new int[] {34,345};

i[1] +=2;

System.out.println(i.length);

System.out.println(i[1]);

int[] copy = i.clone();

System.out.println(copy);

System.out.println(i==copy);

System.out.println(i.equals(copy));

double [][] fd ={{23.5, 37.8, 65.05}, {453.3, 3.2, 5.5}};

System.out.print(Arrays.toString(i));

}

}

Управляющие конструкции.

Предложение блок.

Предложение-блок – это строки внутри блока. Предложение-блок обычно используется в сложных управляющих конструкциях.

{

// предложение

// предложение

}

if

Синтаксис:

if (условие) предложение

[else предложение]

Например:

int i;

i= -23;

if (i>0) {

System.out.println("positive Number");

} else {

System.out.println ("negative or null");

}

Усложненная цепочка:

public class Main {

public static void main(String[] args){

int i;

i= -23;

if(i>0){

System.out.println("positive number");

}else if(i==null){

System.out.println("null number" );

}else{

System.out.println("negative number");

}

}

}

Условный (тернарный) оператор.

Тернарный оператор – это выражение, схожее с условной конструкцией if. Пример:

public class Main {

public static void main (String [] args) {

int i = 12;

System.out.println (i<0 ? "negative" : "positive");

}

}

Switch

Switch применим к известному числу в разных операциях. Switch довольно бесполезная условная конструкция.

public class Main {

public static void main (String [] args) {

int j = 12+ 40;

switch (j) {

case 52:

System.out.println("52");

break;

case 57:

System.out.println("57");

break;

default:

System.out.println("other");

}

}

}

Циклы.

While

Цикл while простой и удобный цикл в применении. Синтаксис:

while (*условие){*

*предложение*

*}*

public class Main {

public static void main (String [] args) {

int i = 14;

while (i>0){

System.out.println (i);

i--;

}

}

}

Также существует бесконечный цикл. В целом он бесполезен, но можно над ним посмеяться.

public class Main {

public static void main (String [] args) {

while (true) {

System.out.println ("Snatch");

}

}

}

Do-while

public class Main {

public static void main(String[] args) {

int i=1;

do{

i++;

System.out.println(i);

}while(i<9);

}

}

For

Цикл for – цикл со счетчиком. Он выполняется до тех пор, пока логическое выражение не станет ложным. Его синтаксис:

for (*инициализация*; *логическое\_значение*; *итерация*) {

//Action

}

Разберем все по отдельности. *Инициализация*

public class Main {

public static void main(String[] args) {

int i;

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

System.out.println(i);

}

}

}

Цикл перебора.

У нас имеются массивы. Ранее мы циклично перебирали от 0 и n-длинны. Но если вам необходимо просто перебрать массив, то можно воспользоваться циклом for-each.

Он имеет такой синтаксис:

for(*тип* имя : *массив*) {

//Действие

}

Вот пример такого цикла:

int [] i = {1, 45, 56};

for (int g : i){

System.out.println (g);

}

Выход из цикла.

Соответственно встает вопрос, а как выйти из цикла? Есть два ключевых слов, которые позволяют выйти из цикла. Первый это break выходящий сразу из цикла. Второй это continue, позволяющий сразу перепрыгнуть на следующую итерацию.

for (int i = 0; i < 10; i++) {

if(i == 5)

continue;

if(i == 8)

break;

System.out.println (i);

}