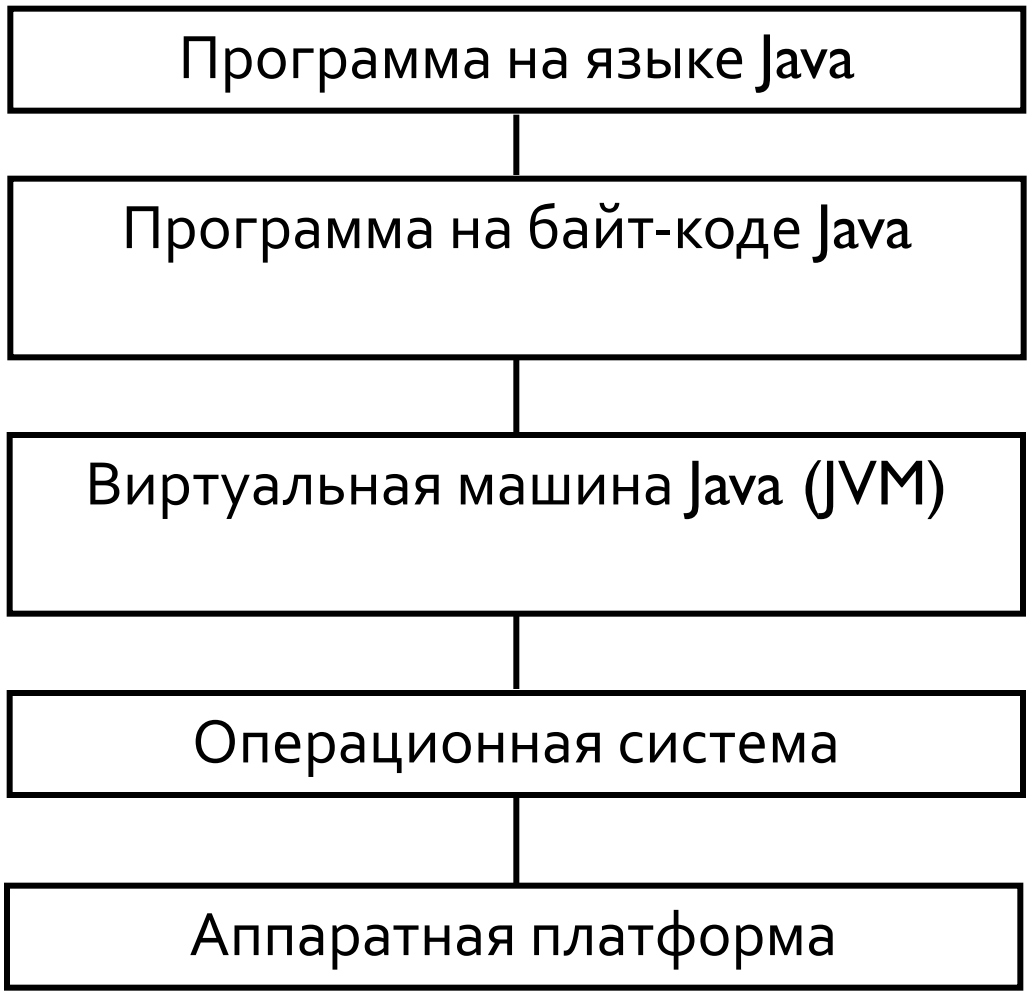
Лекция 1

Язык Java: история появления и развития, архитектура и основные понятия, лексика языка

Архитектура ПО



Основные понятия

JVM – Java Virtual Machine (Виртуальная машина Java)

JRE – Java Runtime Environment (Среда выполнения Java-программ)

JDK – Java Development Kit (Инструментарий разработчика)

J2SE – Java 2 Standard Edition (Стандартная редакция)

J2EE – Java 2 Enterprise Edition (Корпоративная редакция)

J2ME – Java 2 Mobile Edition (Редакция для мобильных устройств)

JavaScript – Модификация языка Java для программирования на стороне Web-клиента

Основные концепции Java

В основу языка Java были положены следующие основные концепции:

- простота;
- безопасность;
- переносимость;
- объектная ориентированность;
- устойчивость;
- многопоточность;
- архитектурная нейтральность;
- интерпретируемость;
- высокая производительность;
- распределенность;
- динамический характер.

История развития

- Java 1.0 – 1996 г.
- Java 1.1 – 1997 г.
- Java 1.2 – Java 2 (J2SE): Swing, Collections Framework – 1998 г.
- Java 1.3 – 2000 г.
- Java 1.4 – assert, цепочки исключений – 2002 г.
- Java 1.5 – обобщения, аннотации, автоупаковка, автораспаковка, перечисления, for-each, varargs. – 2004 г.
- Java 1.6 – Java SE6 – 2006 г.
- Java 1.7 – Java SE7 – некоторые улучшения синтаксиса, расширение стандартных библиотек – 2011 г.
- Java 1.8 – Java SE8 – 2014 г.

Язык Java

Язык Java является объектно-ориентированным языком программирования и поддерживает основные принципы ООП:

- инкапсуляция,
- наследование,
- полиморфизм.

Пример простейшей программы

```
class HelloWorld{
  public static void main(String[] argv){
    System.out.println(“Hello world!”);
  }
}
```

Исходный текст в файле с расширением .java

Компиляция: javac HelloWorld.java

Скомпилированный класс в файле с расширением .class

Запуск программы: java HelloWorld

Пример

```
class Example{
  public static void main(String[] argv){
    int a = 10;
    System.out.println(“Value a=”+a);
    a *= 3;
    System.out.println(“Value a=”+a);
  }
}
```

Лексика языка

Язык Java является языком свободной формы. При написании программ не требуется следовать никаким специальным правилам в отношении отступов.

Символами-разделителями лексем в языке Java являются:

- пробел,
- табуляция,
- перевод строки.

Лексика языка: идентификаторы

Идентификаторы используются для именования переменных, атрибутов, классов, методов и т.д.

Допустимые в идентификаторах символы: ‘0’ .. ‘9’, ‘a’.. ‘z’, ‘A’ .. ‘Z’, ‘_’, ‘$’.

Идентификатор не может начинаться с цифры. Буквы различаются по регистру:

Value и value – разные идентификаторы.

Лексика языка: константы

Целочисленные константы: 100

Вещественные константы: 86.15

Символьные константы: ‘S’

Строковые константы: “My string”

Лексика языка: разделители

( )	Используются для передачи списка параметров при определении и вызове методов, для определения приоритетов операций, для указания типа при приведении типов
{ }	Используется для указания значений автоматически инициализируемых массивов, для определения блоков кода, классов, методов и локальных областей определения
[ ]	Используются при объявлении типов массивов, а также при разыменовании значений массивов
;	Завершает операторы
,	Разделяет последовательные идентификаторы в объявлении переменных, передаче параметров в методы, в цикле for
.	Используется для разделения имен пакетов от подпакетов и классов, а также отделения атрибута или метода от ссылочной переменной

Лексика языка: ключевые слова

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

Лексика языка: комментарии

Три вида комментариев:

- Строчный - //…
- Блочный - /* … */
- Документирования:

```
/**
…
*/
```

Создание документации для класса осуществляется с помощью javadoc

Типы данных

Язык Java – строго типизированный язык:

- каждая переменная обладает типом, каждое выражение имеет тип, каждый тип строго определен;
- все присваивания, как явные, так и посредством передачи параметров в вызовах методов, проверяются на соответствие типов;
- в Java отсутствуют какие-либо средства автоматического приведения или преобразования типа;
- компилятор Java проверяет все выражения и параметры на предмет совместимости типов.

Типы данных

В языке Java заложены 8 элементарных типов данных, разделенных на следующие группы:

- целочисленные,
- вещественные,
- символьные,
- логические.

Целочисленные типы

long	64 бита	-9223372036854775808 .. 9223372036854775807
int	32 бита	-2147483648 .. 2147483647
short	16 бит	-32768 .. 32767
byte	8 бит	-128 .. 127

Вещественные типы

Вещественный тип двойной точности

double

64 бита, от 4.9e-324 до 1.8e+308

Вещественный тип одинарной точности

float

32 бита, от 1.4e-45 до 3.4e+38

Символьный тип

Символьный тип данных в Java:

char

16 бит, 0 .. 65535

Содержит символы Unicode

Может рассматриваться как символы и как целые числа:

char ch = ‘A’, Ch = 65;

Логический тип

Логический тип в Java:

boolean

Может принимать два значения:

true и **false**

Все логические операции и операции отношения возвращают значения этого типа.

Целочисленные константы

Целочисленные константы в языке Java можно указывать в трех системах исчисления:

- десятичной (23),
- восьмеричной (027),
- шестнадцатеричной (0x17).
- двоичной (0b10001) (Java 1.7+)

По умолчанию целочисленные константы имеют тип **int.** Для их приведения к типу **long** необходимо указать суффикс L: 123L.

Целочисленные константы могут быть присвоены переменным типа **byte**, **short** или **char**, если они находятся в их диапазонах:

```
short a = 30000;
byte b = 343; //ошибка!!!
```

В Java 1.7 численные константы можно записывать с символом-разделителем «_»: 123_456_789

Вещественные константы

Вещественные константы могут записываться в стандартной или научной форме.

Для распознания типа вещественной константы используются суффиксы:

D или d – тип **double** (по умолчанию);

F или f – тип **float**.

Примеры: 123.45 354.67f

Символьные константы

Символьные константы в Java заключаются в одинарные кавычки: ‘F’

\ddd	Восьмеричный символ
\uxxxx	Шестнадцатеричный символ Unicode
\’	Одинарная кавычка
\”	Двойная кавычка
\\	Обратная косая черта
\r	Возврат каретки
\n	Новая строка (перевод строки)
\f	Подача страницы
\t	Табуляция
\b	Возврат на одну позицию (забой)

Булевы и строковые константы

В языке Java присутствуют две булевы константы: false – ложь, true – истина.

Строковые константы в языке Java указываются в двойных кавычках:

“My string”

“Hello \”world\”!”

Строки в **Java** не простейший (встроенный) тип, а класс из пакета java.lang

Переменные

Объявление переменных:

тип имя [=значение][, имя [=значение] …]

Примеры:

```
int a=3, b, c=10;
byte d = 12, e = -23;
double x, y = 1.5;
char ch = 'Z';
boolean fl1 = true, fl2 = false;
```

Переменные

В языке Java допускается «динамическое» объявление переменных:

```
int a = 10;
…
int b = 2*a;
…
```

Две области доступа:

- переменная-атрибут класса – объект;
- переменная в методе – блок операторов, в котором она объявлена.

Преобразование и приведение типов

Автоматическое приведение типа возможно:

- оба типа совместимы,
- длина целевого типа больше длины исходного.

Автоматическое приведение:

byte -> short -> int -> long -> float -> double

Невозможно автоматическое приведение к типам char или boolean

Явное приведение типа

В языке Java можно использовать явное приведение типа:

(целевой тип)значение

Примеры:

```
byte a;
int i = 123;
double x = 234.5;
a = (byte)i;
i = (int)x;
```

Автоматическое повышение типа в выражениях

При выполнении целочисленных вычислений тип выражения автоматически повышается до int если параметрами выражения являются byte, short или int, и до long, если хотя бы один из параметров является типа long.

Пример:

```
short x = 12;
short y = (short)(x * 2);
```

Автоматическое повышение типа в выражениях

Если хотя бы один из параметров выражения имеет тип **float**, то и все выражение повышается до **float**.

Если хотя бы один из параметров выражения имеет тип **double**, то и все выражение повышается до **double**.

Пример:

```
int x = 10, r;
double k = 1.5;
r = (int)(x*k);
```