Лекция 3

Лексика языка Java

Содержание

- ***Текст программы. Лексемы. Внутреннее устройство языка.***
- **Типы данных. Переменные. Простейшие и ссылочные типы, операции над значениями различных типов.** Приведение типов. Класс Class.
- Система именования элементов языка в Java. Пакеты (packages). Область видимости имени. Конфликт имен и соглашения по именованию.

Текст программы.

- Кодировка – Unicode (2 байта, 65535 символов)

\u0401

A10 : 16 = Q + a, Если Q != 0 A = Q … до тех пор пока a<16/Q = 0

2508:16 =156+12 -- XXXC, 156:16=9+12 – XXCC – 09CC

- Текст программы
  - Пробелы (whitespaces)
  - Комментарии (comments)
  - Основные лексемы (tokens)

Пробелы

```
if (a == b) { if (a == c) nMax = 3; else { if (a > c) nMax = 2; else nMax = 1; } } else { if (a > b) {
    if (a == c) nMax = 2; else nMax = 1; } else { if (b == c) nMax = 2; else nMax = 1; } }
```

```
if (a == b) {
        if (a == c) nMax = 3;
        else {
                if (a > c) nMax = 2;
                else nMax = 1;
        }
}
else {
        if (a > b) {
                if (a == c) nMax = 2;
                else nMax = 1;
        } else {
                if (b == c) nMax = 2;
                else nMax = 1;
        }
 }
```

```
int x = 3;

int
x
=
3
;
```

Пробелы

- Пробел
  - «пробел» \u0020
  - «табуляция» \u0009
  - Form feed \u000c – символ перевода страницы
  - Символ завершения строки
- Завершение строки
  - Carriage return \u000d
  - Line feed \u000a – символ новой строки
  - CR + LF

Комментарии

- Строчные

  // ……. <завершение_строки>

- Блочные

  /* ……. */

“/* ……. */” – часть строки

/* … /* ……. */ …. */ - нельзя делать вложенными

Можно располагать внутри оператора

int /* ……. */ x = 1;

circle. /* ……. */ getR();

circle. get /* ……. */ R();

Комментарии

- Комментарии javadoc
  - Для автоматического создания документации кода
  - перед описанием классов, интерфейсов, методов, полей, если написаны в других местах, не попадут в документацию
  - допустима HTML-разметка и специальные тэги javadoc

```
/**
     ……
*/
```

Лексемы

- Идентификаторы (identifiers)
  - Имена, присвоенные элементам языка для упрощения доступа к ним. Идентификатор не может начинаться с цифры, может быть написан на любом языке при помощи Unicode, длина не ограничена (пакеты, классы, интерфейсы, поля, методы, аргументы, локальные переменные).
- Ключевые слова (keywords)
  - Зарезервированные лексемы, выполняющие различные задачи языка
- Литералы (literals)
  - Значения для цифровых, строковых и других выражений
- Разделители (separators)
  - Служебные символы `( ) [ ] { } ; . ,`
- Операторы (operators)
  - Лексемы, обозначающие специальные действия

Ключевые слова

Зарезервированные лексемы, выполняющие различные задачи языка

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

goto и const зарезервированы, но не используются,
true, false, null – литералы, не являются ключевыми

Литералы

(значения, неименованные константы)

- IntegerLiteral

  43213 (10-чный); 0462421 (8-чный); 0x351fa3 (16-чный)

  0L 456l
- FloatingPointLiteral

  4.4; 4.; .4; 1e12; 3.1E-21; 5.2f (float); 4.12d (double)

  1. .1 1e1 1f
  - Обязательные элементы - . (точка + хотя бы одна цифра в целой или дробной части) или e/E (показатель)
- BooleanLiteral

  true; false
- CharacterLiteral

  ‘A’; ‘ ‘; \u0401, \t – табуляция, \n – конец строки, \r – возврат каретки, \’ – одиночная кавычка, \” – двойная кавычка, \\ - косая черта
- StringLiteral

  “Абракадабра”, “text1” + “text2”, “text1 \r\n text2”, “A”
- NullLiteral

  null

Лексемы

- Идентификаторы (identifiers)
  - Имена, присвоенные элементам языка для упрощения доступа к ним. Идентификатор не может начинаться с цифры, может быть написан на любом языке при помощи Unicode, длина не ограничена (пакеты, классы, интерфейсы, поля, методы, аргументы, локальные переменные).
- Ключевые слова (keywords)
  - Зарезервированные лексемы, выполняющие различные задачи языка
- Литералы (literals)
  - Значения для цифровых, строковых и других выражений
- Разделители (separators)
  - Служебные символы `( ) [ ] { } ; . ,`
- Операторы (operators)
  - Лексемы, обозначающие специальные действия

Операторы

Знак, указывающий компилятору на необходимость выполнения определенного действия.

Арифметические, поразрядные (битовые), логические, отношения + присваивания, некоторые доп.операторы.

```
=  >  <  !  ~  ?  :  ==
<=  >=  !=  &&  ||  ++  --  +
-  *  /  &  |  ^  %  <<
>>  >>>  +=  -=  *=  /=  &=  |=
^=  %=  <<=  >>=  >>>=
```

- Оператор присваивания возвращает значение правого операнда
- Оператор сравнения возвращает булевское значение

Операторы

- арифметические

++ -- + - * / %

1/2 =0, 1./2=0.5

x =5; y=++x x =5; y=x++ x,y =?

10%3=1

- сравнения

> < == <= >= !=

Операторы

- оператор присваивания и укороченные операторы присваивания

= += -= *= /= &= |=

^= %= <<= >>= >>>=

переменная = выражение;

переменная = переменная = выражение;

x=x+10 x+=10;

Операторы

- логические

!

& | ^ - оба операнда логические

&& || - укороченные операторы

(лог.выр.1) & (лог.выр.2) vs (лог.выр.1) && (лог.выр.2)

- битовые (целые операнды представляются в двоичном виде)

& | ^ - оба операнда целые, ~ - not

<< >> >>> a<<b – a на b битов сдвигается влево, справа нули, a>>b – a на b битов сдвигается вправо, сдвигаемые биты отбрасываются, a>>>b сдвиг вправо с заполнением нулями (без знака)

Операторы

оператор с условием ? :

(условие) ? (выр.1, выполняется, если условие true) : (выр.2, выполняется, если условие false)

Содержание

- **Текст программы. Лексемы. Внутреннее устройство языка.**
- ***Типы данных. Переменные. Простейшие и ссылочные типы, операции над значениями различных типов.*** Приведение типов. Класс Class.
- Система именования элементов языка в Java. Пакеты (packages). Область видимости имени. Конфликт имен и соглашения по именованию.

Типы данных

Java строго типизированный язык программирования, в процессе компиляции типы операндов проверяются во всех выражениях.

- Простые (primitive)
  - Целочисленные (byte - 1, short - 2, int – 4, long – 8, char – 2, 0..65535)
  - Дробные (float - 4, double - 8)
  - Булевский (boolean)
- Объектные (reference)

классы (пользовательские, стандартные библиотеки), интерфейсы, массивы

Объявление переменных

<Тип> <Имя> [=Значение]

```
int a;
double b = 3.5, c = 3.8;
float d = b+c; динамическая инициализация
int e = a = 5;
```

final double pi = 3.1415; - именованная константа

Объявление переменных

<Тип> <Имя> [=Значение]

Point a = new Point (1, 2);

Point a;

a = new Point (1, 2);

Point a = null;

String – исключение, объекты создаются при записи литералов

“abc” + “def”

```
int a = 3;
int b = a;
a = 5;
b = ?

Chair ch1 = new Chair(4);
Chair ch2 = ch1;
ch1.legN = 3;
ch2.legN = ?
```

Point a = new Point (1, 2);

Point a = null;

JVM всегда занимается подсчетом хранимых ссылок, как только на объект не остается ни одной ссылки, он предназначается для уничтожения сборщиком мусора.

- Оператор instanceof

if (a instanceof Point) { … }

Типы данных

Java строго типизированный язык программирования, в процессе компиляции типы операндов проверяются во всех выражениях.

- Простые (primitive)
  - Целочисленные (byte - 1, short - 2, int – 4, long – 8, char – 2, 0..65535)
  - Дробные (float - 4, double - 8)
  - Булевский (boolean)
- Объектные (reference)

классы (пользовательские, стандартные библиотеки), интерфейсы, массивы

Операторы

Знак, указывающий компилятору на необходимость выполнения определенного действия.

Арифметические, поразрядные (битовые), логические, отношения + присваивания, некоторые доп.операторы.

```
=  >  <  !  ~  ?  :  ==
<=  >=  !=  &&  ||  ++  --  +
-  *  /  &  |  ^  %  <<
>>  >>>  +=  -=  *=  /=  &=  |=
^=  %=  <<=  >>=  >>>=
```

- Оператор присваивания возвращает значение правого операнда
- Оператор сравнения возвращает булевское значение

Класс Object

«Родитель» для всех объектов Java

- Class getClass();
  - получение класса объекта
- boolean equals(Object);
  - сравнение объектов
- int hashCode();
  - хэширование объекта
- String toString();
  - строковое представление объекта
- void finalize();
  - финальная обработка объекта перед сборкой мусора

Класс Class

Метакласс для всех классов Java

Point a = new Point(3, 5);

- Объект типа Point;
- Объект типа Class, описывающий класс Point
- Объект типа Class, описывающий класс Object
- Объект типа Class, описывающий класс Class

Класс Object

«Родитель» для всех объектов Java

- Class getClass();
  - получение класса объекта
- boolean equals(Object);
  - сравнение объектов
- int hashCode();
  - хэширование объекта
- String toString();
  - строковое представление объекта
- void finalize();
  - финальная обработка объекта перед сборкой мусора

Класс String

- Каждая строка – объект класса String
- Каждый объект String неизменяем
  - Изменение строки порождает новый объект
  - Возможный источник неэффективности!
  - Можно использовать класс StringBuffer и подобные
- Сравнение объектов String между собой оператором сравнения дает непредсказуемый результат
  - Сравнение – только с помощью метода equals()

```
String s1 = “abc”, s2 = “abc”;
print (s1.equals(s2));      // true
print (s1 == s2);           // ???
```

Содержание

- Текст программы. Лексемы. Внутреннее устройство языка.
- Типы данных. Переменные. Простейшие и ссылочные типы, операции над значениями различных типов. Приведение типов. Класс Class.
- **Система именования элементов языка в Java. Пакеты (packages). Область видимости имени. Конфликт имен и соглашения по именованию.**

Имена

- Имя <> Идентификатор
  - Простое (simple) – 1 идентификатор
  - Составное (qualified) – >1 идентификатора
- Имена
  - Пакеты
  - Классы
  - Интерфейсы
  - Поля и методы ссылочных типов
  - Аргументы методов и конструкторов
  - Локальные переменные

Область видимости

- Область видимости (scope) предназначена для минимизации конфликтов имен
- При обращении к элементу из его области видимости можно использовать простое имя, в противном случае – составное

Пакеты

- Пакет – способ группировки типов (классов и интерфейсов)

  java.lang.Object

  - Обеспечение логической группировки
  - Эффективное проектирование и разработка
- Пакеты образуют иерархическую структуру
- Пакеты обладают собственным пространством имен
  - Минимизация конфликтов имен

Пакеты

- Элементы пакета
  - Классы
  - Интерфейсы
  - Пакеты
- Хранение элементов пакета
  - Каждому пакету соответствует папка с таким же именем
  - Папки вложены аналогично вложенности пакетов
  - Каждому классу соответствует файл .java, его имя совпадает с именем класса.
  - Пакеты могут паковаться в .jar-файлы

Модуль компиляции

- Модуль компиляции
  - Объявление пакета

    package ru.ifmo.wavelab;
  - import-выражения

    import ru.government.money.*;

    import com.sun.Java;
  - Объявления одного или нескольких типов

    class WaveLaboratory { }

    interface Student { }
- Разграничение доступа
  - public / private / <default>

Область видимости

- Пакет – вся программа
  - Доступ только по полному имени
- Тип верхнего уровня – собственный пакет
  - Из других пакетов – доступ по составному имени или через import
- Элемент типа – все тело собственного типа
  - Из других типов – по составному имени, с помощью this и super.
- Аргументы метода (конструктора) – внутри метода
- Локальная переменная – внутри того блока, в котором она объявлена

Область видимости для JRE

- Переменная CLASSPATH
  - Перечисление имен пакетов и библиотек, доступных для JRE
  - Текущий каталог по умолчанию не виден!

```
set CLASSPATH = .;c:\java\classes;d:\lib\myclasses.jar
```

- Указание полного имени класса при запуске
  - Расширение .class не указывать! (речь о классе, а не о файле)
  - Запуск производится из каталога, в котором находится корневой пакет (а не сам класс)

```
java ru.ifmo.laboratory.Projector;
```

Соглашения по именованию

- Типы – с большой буквы (возможно несколько слов)
  - String; MySpecialType; ArrayIndexOutOfBoundsException
- Интерфейсы – аналогично, с суффиксом –able
  - Runnable; Serializable; StringSerializable;
- Методы – с маленькой буквы (возможно несколько слов)
  - listen(); readAndWrite();
  - getSize(); setWeight(); toString(); isCorrect()
- Константы – только большими буквами
  - MAX_SIZE; CENTER;
- Пакеты – маленькими буквами, часто – по Web-сайту разработчика или его компании
  - com.sun.image.codec.jpeg; org.omg.CORBA.ORBPackage
  - oracle.jdbc.driver.OracleDriver

Содержание

- Текст программы. Лексемы. Внутреннее устройство языка.
- Типы данных. Переменные. Простейшие и ссылочные типы, операции над значениями различных типов. Приведение типов. Класс Class.
- Система именования элементов языка в Java. Пакеты (packages). Область видимости имени. Конфликт имен и соглашения по именованию.

Структура класса

- Именование пакета

  `package` <имя пакета>.<имя класса>
- Заголовок класса

  `class` <имя класса>

  [**`extends`** <имя класса-родителя>]

  { [тело класса] }
- Объявление поля

  <тип поля> <имя поля> [= значение];

Объявление методов

- Объявление метода

  <тип значения | **void**> **<**имя метода> (

  <тип_параметра> <имя_параметра>[, …]])

  { [тело метода]

  return <значение>; }

- Метод main – точка входа в программу

```
public static void main() { }
public static void main(String[] args) {
   System.exit(<код_возврата>);
}
```

Hello world

```
package ru.ifmo;


public class Hello {
  Hello() {}
  public static void main() {
   Hello object = new Hello();
   System.out.println("Hello world!");
   System.exit();
  }
}
```