19.09.25

Лексика языка Java

План

1) Структура исходного кода
2) Типы данных
3) Описание классов
4) Точка входа программы

- Общая структура
- Поля
- Методы
- Конструкторы
- Блоки инициализации

Кодировка

Java ориентирован на Unicode

Java чувствителен к регистру

Символы Unicode задаются с помощью escape-последовательностей: \u262f \uu2042 \uuu203d

обозначают Unicode символы

Исходный код разделяется на:

- Пробелы
- Комментарии → // Однострочный | /*Многострочный*/ | /**Документирования*/

  Допускают наличие HTML-тэгов кроме заголовков

  Специальные тэги @see @param @deprecated
- Лексемы
  - Идентификаторы
  - Служебные слова public, class, ...
  - Литералы
  - Разделители
  - Операторы

Идентификаторы

- Состоят из букв и цифр, знаков _ и $
- Не допускают совпадения со служебными словами
- Длина не регламентируется

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

Типы данных

Значащие, примитивные (простые)

Предназначены для работы со значениями естественных, простых типов

- Булевский (логический) тип
  - `boolean` – допускает хранение значений true или false
- Целочисленные типы
  - `char` – 16-битовый символ Unicode
  - `byte` – 8-битовое целое число со знаком
  - `short` – 16-битовое целое число со знаком
  - `int` – 32-битовое целое число со знаком
  - `long` – 64-битовое целое число со знаком
- Вещественные типы
  - `float` – 32-битовое число с плавающей точкой (IEEE 754-1985)
  - `double` – 64-битовое число с плавающей точкой (IEEE 754-1985)

Ссылочные (классы, массивы и т.д.)

- Предназначены для работы с объектами
- Переменные содержат ссылки на объекты
- Тип переменной определяет контракт доступа к объекту
- Ссылка - это не указатель
- Передаются и сравниваются по значению в ссылке

Литералы

- Булевы
  `true false`
- Символьные
  `'a' '\n' '\\' '\377' '\u0064'`
- Целочисленные
  `29 035 0x1D 0X1d 0xffffL`
  `0b11110000 0B11110000 (Java 1.7)`
  - По умолчанию имеют тип `int`
- Числовые с плавающей запятой
  `1. .1 1e1 1e-4D 1e+5f`
  - По умолчанию имеют тип `double`
- Строковые
  `"Это строковый литерал" ""`

Описание класса

Содержит:

- Поля
- Методы
- Вложенные типы

```
class Body {
  public long idNum;
  public String name;
  public Body orbits;

  public static long nextID = 0;
}
```

Модификаторы объявления класса

- `public`
  Признак общедоступности класса. В одном файле может содержаться определение только одного public-класса
- `abstract`
  Признак абстрактности класса
- `final`
  Завершенность класса (класс не допускает наследования)
- `strictfp`
  Повышенные требования к операциям с плавающей точкой

Поля класса

Объявление поля:

[модификаторы] <тип> {<> [= <инициализирующее выражение>]};

double sum = 2.5 + 3.7;

Если поле явно не инициализируется, ему присваивается значение по умолчанию

- Модификаторы полей:
  - модификаторы доступа
  - **static**
    поле статично (принадлежит контексту класса)
  - **final**
    поле не может изменять свое значение после инициализации
  - **transient**
    поле не сериализуется (влияет только на механизмы сериализации)
  - **volatile**
    усиливает требования к работе с полем в многопоточных программах

static + final = const

Но const в Java не существует

Методы

[модификаторы] <тип> <сигнатура> [throws исключения] {<тело>};

Если метод ничего не возвращает, то тип - void

К сигнатуре метода относятся: Название и Список и типы параметров

Модификаторы методов

- Модификаторы доступа
- **abstract**
  абстрактность метода (тело при этом не описывается)
- **static**
  метод принадлежит контексту класса
- **final**
  завершенность метода (метод не может быть переопределен при наследовании)
- **default (Java 8)**
  реализация метода по умолчанию, используется в интерфейсах

Модификаторы реализации

- **synchronized**
  синхронизированность метода (особенности вызова метода в многопоточных приложениях)
- **native**
  «нативность» метода (тело метода не описывается, при вызове вызывается метод из native-библиотеки)
- **strictfp**
  повышенные требования к операциям с плавающей точкой

Особенности методов

- На время выполнения метода управление передаётся в тело метода
- Возврат управления осуществляется либо после вызова return, либо после конца выполнения (если метод ничего не возвращает)
- Возвращается ОДНО значение

Переменное количество аргументов

Синтаксис: `methodName(type ... arg)`

массив

Ограничения:

- Может быть только один такой аргумент.
- Должен быть последним в списке.
- Может вызывать неоднозначность при перегрузке.

Создание объектов

```
MyClass obj = new MyClass();
```

Оператор new создает объект и возвращает ссылку на него.

Вызов конструктора

Конструкторы

Правила:

- Имя класса = имя конструктора
- Нет возвращаемого типа (даже void)
- Можно использовать только модификаторы доступа.
- Могут быть перегружены и вызывать друг друга через this().

Не определяются разработчиком

Определяются разработчиком

По умолчанию

```
MyClass() {
  num = 0;
  text = "";
}
```

Без параметров

```
MyClass() {
  num = 0;
  text = "";
}
```

С параметрами

```
MyClass(int number, string text) {
  num = number;
  text = text;
}
```

Деструкторы и сборка мусора

В Java нет деструкторов.

Сборка мусора (Garbage Collection)

Автоматически

При нехватке памяти или спустя время

Вручную

System.gc() или Runtime.getRuntime().gc()

На объект нет ссылок?

Да

Очистить

Нет

Ничего не делать

Точка входа в программу (main)

```
public static void main(String[] args)
```

static void — всегда!

args — аргументы командной строки

Блоки инициализации

Нестатический

- Для НЕстатических полей
- Выполняется при создании объекта

```
MyClass() {
   static num = 0;
   text = "";

   {
      text = "default";
   }
}
```

Статический

- Для статических полей
- Выполняется при загрузке класса в ВМ

```
MyClass() {
   static num = 0;
   text = "";

   static {
      num = 29;
   }
}
```

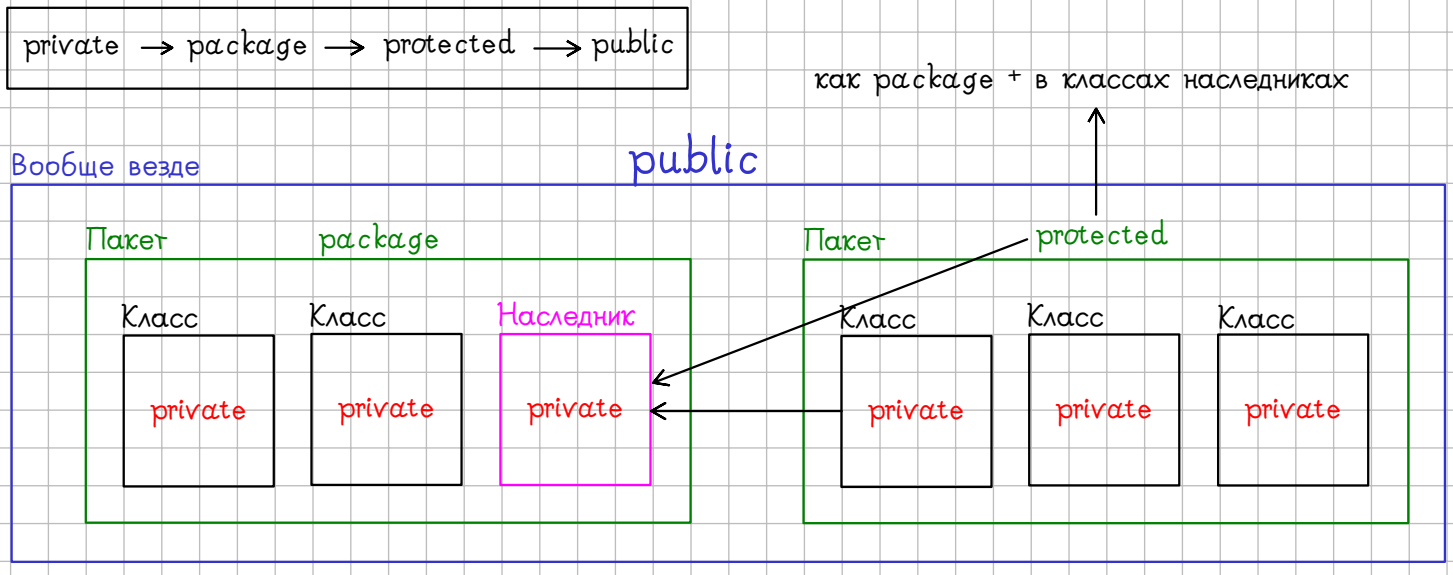
Порядок инициализации объекта:

Значения по умолчанию. → Блоки инициализации → Тело конструктора

Модификаторы доступа

По убыванию строгости

private → package → protected → public



Перечисления (enum)

0 1

enum Apple { Jonathan, GoldenDel }

Особенности:

- Не создаются через new
- Наследуются от java.lang.Enum

Пример:

```
Apple a = Apple.Jonathan;
if (a == Apple.GoldenDel) {}
```

Записи (record)

Назначение: неизменяемый класс для данных

public record Name(String field1, int field2) {}

Особенности:
- нельзя наследовать
- final-класс

Name rec = new Name("Hello", 1);