**Java-Синтаксис языка.**

**Объект** — объекты имеют состояние и поведение. Например: собака может иметь состояние — цвет, имя, а также и поведение — кивать, бежать, лаять, кушать. Объект является экземпляром класса.

**Класс** — может быть определен как шаблон, который описывает поведение объекта.

**Метод** — является в основном поведением. Класс может содержать несколько методов. Именно в методах логически записанные данные манипулируют и выполняют все действия.

**Переменные экземпляра** — каждый объект имеет свой уникальный набор переменных экземпляра. Состояние объекта создается значениями, присвоенными этим переменным экземпляра.

**Итерация** (лат. iteratio «повторение») — повторение какого-либо действия.

**Основы синтаксиса языка Java**

Очень важно знать и помнить следующие моменты в синтаксисе:

* Чувствительность к регистру — Java чувствителен к регистру, то есть идентификатор Hello и hello имеют разный смысл.
* Название классов — для всех первая буква должна быть в верхнем регистре.
* Если несколько слов используются, чтобы сформировать название класса, первая буква каждого внутреннего слова должна быть в верхнем регистре, например, «MyJavaClass».
* Название методов — в синтаксисе Java все имена методов должны начинаться с буквы нижнего регистра.
* Если несколько слов используются, чтобы сформировать имя метода, то первая буква каждого внутреннего слова должна быть в верхнем регистре, например, «public void myMethodName()».
* Название файла программы — наименование файла программы должно точно совпадать с именем класса.
* При сохранении файла, Вы должны сохранить его, используя имя класса (помните о чувствительности к регистру) и добавить «.java» в конце имени (если имена не совпадают, Ваша программа не будет компилироваться), например, «MyJavaProgram» — это название класса, тогда файл должен быть сохранен как «MyJavaProgram.java».
* public static void main(String args[]) — обработка программы начинается с метода main(), который является обязательной частью каждой программы.

**Идентификаторы в Java**

**Идентификаторы** — имена, используемые для классов, переменных и методов. Все компоненты Java требуют имена.

Существует несколько правил в синтаксисе языка Java, которые необходимо помнить об идентификаторе. Они следующие:

* Каждый идентификатор должен начинаться с «A» до «Z» или «a» до «z», «$» или «\_».
* После первого символа может иметь любую комбинацию символов.
* Ключевое слово не может быть использовано в качестве идентификатора.
* Самое главное — идентификатор в Java чувствителен к регистру.
* Пример правильного написания: age, $salary, \_value, \_\_1\_value.
* Пример неправильного написания: 123abc, -salary.

**Перечисления**

Перечисления были введены в Java 5.0. Они ограничивают переменную, чтобы выбрать только одно из нескольких предопределенных значений. Значения в этом перечисляемом списке называются **перечисления**.

С использованием перечисления в Java можно уменьшить количество ошибок в коде.

Например, если рассматривать заявки на свежий сок в магазине, можно было бы ограничить размер упаковки сока как для малых, средних и больших. Это позволяет с помощью использования в Java перечисления сделать так, чтобы никто не заказал другой любой размер упаковки, кроме как малый, средний или большой.

**Пример кода перечисления в Java**

class FreshJuice {

enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }  
 FreshJuiceSize size;  
}

public class FreshJuiceTest {

public static void main(String args[]){  
 FreshJuice juice = new FreshJuice();  
 juice.size = FreshJuice.FreshJuiceSize.MEDIUM;  
 System.out.println("Размер: " + juice.size);  
 }  
}

Полученный результат выше приведенного примера:

Размер: MEDIUM

**Примечание:** в Java перечисления могут быть объявлены как самостоятельно, так и внутри класса. Методы, переменные, конструкторы могут быть определены также внутри перечисления.

**Типы переменных**

Далее в Java мы увидим следующие существующие типы:

* Локальные переменные.
* Переменные класса (статические).
* Переменные экземпляра (динамические).

**Модификаторы**

Как и в других языках, в Java можно модифицировать классы, методы и так далее, с помощью модификаторов. Модификаторы в Java делится на две категории:

* С доступом: default, public, protected, private.
* Без доступа: final, abstract, strictfp.

Более подробно рассмотрим модификаторы класса, модификаторы методов и другие в следующем разделе.

**Массив**

В Java массив является объектом, который хранит несколько переменных одного и того же типа. Тем не менее, сам массив является объектом. Мы рассмотрим, как создать и заполнить массив в последующих главах.

**Комментарии в Java**

Язык Java поддерживает **однострочные и многострочные комментарии**, они очень похожи на используемые в С и С++. Все символы недоступны внутри любых комментариев и игнорируются компилятором.

**Пример комментария**

public class MyFirstJavaProgram{

/\* Это первая моя программа.  
 \* В результате выполнения на экран будет выведено 'Привет мир'  
 \* Это многострочный комментарий, пример написания.  
 \*/

public static void main(String []args){  
 // Однострочный комментарий.  
 /\* Тоже однострочный комментарий. \*/  
 System.out.println("Привет мир");   
 }  
}

**Ключевые слова в Java**

Всего зарезервировано **50 ключевых слов в Java**. Ниже в таблице приведен список зарезервированных ключевых слов.

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

**Важно!** Ключевое слово не может быть употреблено в качестве константы или переменной, или любых других имен идентификаторов.

**Пустая строка**

**Пустая строка** — строки в Java, содержащие только пробелы, возможно с комментарием. Java полностью игнорирует строку имеющую пробелы и комментарии.

**Наследование**

**Наследование** — концепция, которая позволяет повторно воспользоваться полями и методами существующего класса без необходимости переписывать заново код. В этом случае существующий класс называется суперкласс, а производный называется подкласс.

**Интерфейс**

Интерфейс в Java может быть определен как договор между объектами о том, как общаться друг с другом. Он играет жизненно важную роль, когда речь заходит о понятие наследования.

Интерфейс определяет методы, полученного класса (подкласса), и как их следует использовать. Однако осуществление методов полностью зависит от подкласса.