**JAVA**

*Каким образом происходит процесс кодирования и исполнения не на Java:*

* кодирование;
* компиляция (проверяются исходники на правильность синтаксиса и правильность кода);
* в результате получается код, называемый ***объектным кодом*** (он еще не приложение, а промежуточный этап; в нем произошло преобразование текста в какой-то внутренний формат, но для того, чтобы этот код был исполняемым, не хватает деталей);
* линковка (дополняет предыдущий этап) и получаем машинный код (связывание с другими кодами и библиотеками ОП) – ПОЛУЧИЛИ ИСПОЛНЯЕМЫЙ КОД.

Исполняемый код взаимодействует исключительно с ОП напрямую.

*Каким образом происходит процесс кодирования и исполнения в Java:*

* кодирование;
* компиляция;
* в результате получаем код, называемый ***байт-код*** (файлы с расширением class – сам по себе он не исполняем);
* байт-код не связываем ни с чем, и он является единым для всех ОП, а интерпретируем его при помощи интерпретатора под каждой ОП.

Байт-код всегда единый, но интерпретатор разный на разных ОП.

В связи с тем, что существует посредник – интерпретатор, то страдает скорость при работе с java-приложениями.

Java – это Си-подобный язык.

Синтаксис:

- комментарии: 1) многострочный комментарий - /\* \*/; 2) однострочный комментарий - //; 3) \*\* - документация.

\*Третья, особая форма комментариев, предназначена для сервисной программы *javadoc,* которая использует компоненты Java-транслятора для автоматической генерации документации по интерфейсам классов. Соглашение, используемое для комментариев этого вида, таково: для того, чтобы разместить перед объявлением открытого (public) класса, метода или переменной документирующий комментарий, нужно начать его с символов /\*\* (косая черта и две звездочки). Заканчивается такой комментарий точно так же, как и обычный комментарий — символами \*/. Программа javadoc умеет различать в документирующих комментариях некоторые специальные переменные, имена которых начинаются с символа @. Вот пример такого комментария:

***/\*\****

***\* Этот класс умеет делать замечательные вещи. Советуем всякому, кто***

***\* захочет написать еще более совершенный класс, взять его в качестве***

***\* базового.***

***\* @see Java. applet. Applet***

***\* ©author Patrick Naughton***

***\* @version 1. 2***

***\*/***

***class CoolApplet extends Applet { /\*\****

***\* У этого метода два параметра:***

***\* @param key - это имя параметра.***

***\* @param value - это значение параметра с именем key.***

***\*/ void put (String key, Object value) {***

*Типы данных:*

- класс (основной тип данных);

- целочисленный типы данных (byte – 1 байт, short – 2 байт, int – 2 байта, long – 4 байта) – знаковые (-, +) (не могут быть беззнаковыми);

- вещественный тип данных (float – 4 байта, double – 8 байтов);

- символьный (char) – под него выделяются 2 байта (Unicode);

- булеанный тип данных (boolean): true and false (true - это true, а false - это false) – другого не дано; выделяют 1 бит.

В java нет указателей.

**Классы**

*3 кита ООП:*

Инкапсуляция – соединение данных (полей) и их обработку (методов).

Наследование – получаем все, но не ко всему будем иметь доступ.

Полиморфизм – возможность вызова того метода, к которому принадлежит объект.

*Класс состоит из (внутри класса могут описываться):*

- полей;

- методов (функции обработки).

Написать функцию вне класса невозможно, т.к. java – абсолютно ОО язык.

Объект – это переменная типа class.

Объект.поле/метод()

**Создание объекта и обращение к его элементам**

Пример класса

class Person {

String (класс) name;

int age;

void birthday () {

age++;

System.out.println (name + “is now” + age);

}

}

**Массив**

Person mary = new Person();

int myArray[] = new int[5];

or

int myArray[] = {1, 4, 9, 16, 25};

String languages[] = {“Prolong”, “Java”};

Нельзя в кв. скобках писать про количество элементов заранее.

Узнать, сколько в array элементов можно через .length.

***Цикл for***

for (тип объекта: массив) {}

for (int x: myArray) {} – пройдет весь массив целиком без вариаций.

**Передача параметров в метод**

Передачи в программировании параметров бывают:

1. по значению;
2. по ссылке.

В java передача параметров бывает только по значению.

**Статическое поле и метод**

Делят эти поля и методы между собой все объекты класса.

Статические методы могут работать с другими только статическими методами.

ВОПРОСЫ:

- как запускать интерпретатор «ручками»: с помощью командной строки и запускать интерпретацию? С помощью JDR?

- как ознакомиться с документацией более полно?