1. Введение в Java. Типы данных. Переменные.

Класс

- Java является объектно-ориентированным языком, поэтому такие понятия как "класс" и "объект" играют в нем ключевую роль. Любую программу на Java можно представить как набор взаимодействующих между собой объектов.
- Шаблоном или описанием объекта является класс, а объект представляет экземпляр этого класса. Можно еще провести следующую аналогию. У нас у всех есть некоторое представление о человеке - наличие двух рук, двух ног, головы, туловища и т.д. Есть некоторый шаблон - этот шаблон можно назвать классом. Реально же существующий человек (фактически экземпляр данного класса) является объектом этого класса.
- Класс определяется с помощью ключевого слова class:
- class Person {
- }
- В данном случае класс называется Person. После названия класса идут фигурные скобки, между которыми помещается тело класса - то есть его поля и методы.

Подготовка к работе

- Установить JDK - https://www.oracle.com/technetwork/java/javase/downloads/2133151
- IDEA - https://www.jetbrains.com/idea/download/#section=windows
- ECLIPSE - https://www.eclipse.org/downloads/
- Прописать PATH – JAVA HOME JDK
  - Windows - https://bit.ly/2Wi1sRW

```
public class HelloWorld {
        public static void main(String[] args) {
            System.out.println("Hello World!");
        }
}
```

Типы данных

Java позволяет работать со следующими типами данных:

целые числа;

вещественные числа;

строки;

логические значения;

специальные значения;

символьные значения;

универсальные значения (var);

Примитивные типы данных

- В Java существует 8 примитивных типов данных:
- byte (целые числа, 1 байт)
- short (целые числа, 2 байта)
- int (целые числа, 4 байта)
- long (целые числа, 8 байтов)
- float (вещественные числа, 4 байта)
- double (вещественные числа, 8 байтов)
- char (символ Unicode, 2 байта)
- boolean (значение истина/ложь, 1 байт)

BYTE

- byte — диапазон допустимых значений от -128 до 127
- //объявление переменных типа byte.
- byte getByte, putByte;
- // инициализация переменных
- getByte = 0;
- putByte = 0;
- Переменные типа byte полезны при работе с потоком данных, который поступает из сети или файла.

SHORT

- short — диапазон допустимых значений от -32768 до 32767
- //объявление и инициализация переменной типа short.
- short employeeID = 0;

INTEGER

- int — диапазон допустимых значений от -2147483648 до 2147483647
- //объявление и инициализация переменной типа int.
- int max = 2147483647;
- Тип int используется чаще при работе с целочисленными данными, нежели byte и short, даже если их диапазона хватает. Это происходит потому, что при указании значений типа byte и short в выражениях, их тип все равно автоматически повышается до int при вычислении.

LONG

- long — диапазон допустимых значений от -9223372036854775808 до 9223372036854775807
- //Использование переменных типа long.
- long days = getDays();
- long seconds;
- seconds = days * 24 * 60 * 60;
- Тип удобен для работы с большими целыми числами.

FLOAT

- float — диапазон допустимых значений от ~1,4*10^{-45} до ~3,4*10^{38}
- //Объявление и инициализация переменных типа float.
- float usd = 31.24f;
- float eur = 44.03f;
- Удобен для использования, когда не требуется особой точности в дробной части числа.

DOUBLE

- double — диапазон допустимых значений от ~4,9*10(-324) до ~1,8*10(308)
- //Объявление и инициализация переменных типа double.
- double pi = 3.14159;
- Математические функции такие как sin(), cos() возвращают значение double

CHAR

- char — символьный тип данных представляет собой один 16-битный Unicode символ. Он имеет минимальное значение ‘\ u0000’ (или 0), и максимальное значение ‘\ uffff’ (или 65535 включительно). Символы char можно задавать также при помощи соответствующих чисел. Например символ ‘Ы’ соответствует числу 1067. Рассмотрим на примере:

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

    char symb1=1067;
    char symb2 ='Ы';

    System.out.println("symb1 contains "+ symb1);
    System.out.println("symb2 contains "+ symb2);
}
```

STRING

- Тип String не является примитивным типом данных, однако это один из наиболее используемых типов в Java. String предназначен для хранения строк текста. Несколько примеров использования String
- //Создание строки с помощью конструктора
- String myString = new String("The weather was fine");
- //Можно также создать строку используя кавычки ""
- String myString = "The weather was fine";

VAR

- Появился в java 10

1. Переменная var не может быть глобальной переменной
2. Может быть локальной инициализированной переменной

Специальные значения

- Самым распространенным специальным значением является null, обозначающее пустое значение. То есть отсутствие какого-нибудь значения. В некоторых языках программирования null считается совпадающим с нулем. Но в Java эти два значения считаются абсолютно разными.

Ссылочные типы данных

Ссылочные типы входят все классы, интерфейсы, массивы. Описанный выше тип String также относится к ссылочным типам. Этот класс из стандартной библиотеки Java.

Также существуют классы-оболочки:

- Byte
- Short
- Integer
- Long
- Float
- Double
- Character
- Boolean

В отличие от примитивных типов, они пишутся с заглавной буквы.

ПЕРЕМЕННЫЕ

- При выполнении различных операций, значения могут меняться. Чтобы хранить меняющиеся значения, используются переменные.

Переменная - область памяти в которой хранится значение. Часто говорят, что переменная - именованная область памяти. То есть у каждой переменной есть свое имя, чтобы отличать их друг от друга. Имя переменной используется, чтобы указать, что именно это значение используется и никакое другое.

Локальные переменные

- Локальные переменные объявляются в методах, конструкторах или блоках.
- Локальные переменные создаются, когда метод, конструктор или блок запускается и уничтожаются после того, как завершиться метод, конструктор или блок.
- Модификаторы доступа нельзя использовать для локальных переменных.
- Они являются видимыми только в пределах объявленного метода, конструктора или блока.
- Локальные переменные реализуются на уровне стека внутри.
- В Java не существует для локальных переменных значения по умолчанию, так что они должны быть объявлены и начальное значение должны быть присвоено перед первым использованием.

Пример

```
public class Test {
    public void pupAge(){
        int age = 0;
        age = age + 7;
        System.out.println("Возраст щенка: " + age);
    }
    public static void main(String args[]){
         Test test = new Test();
         test.pupAge();
     }
}
```

Переменная экземпляра

- Переменные экземпляра объявляются в классе, но за пределами метода, конструктора или какого-либо блока.
- Когда для объекта в стеке выделяется пространство, создается слот для каждого значения переменной экземпляра.
- В Java переменные экземпляра создаются тогда, когда объект создан с помощью ключевого слова «new» и разрушаются тогда, когда объект уничтожается.
- Переменные содержат значения, которые должны ссылаться более чем на один метод, конструктор или блок, или на основные части состояния объекта, которые должны присутствовать на протяжении всего класса.
- Переменные экземпляра могут быть объявлен на уровне класса, до или после использования.
- Модификаторы доступа могут быть предоставлены для переменных экземпляра.
- Переменные экземпляра в Java являются видимыми для всех методов, конструкторов и блоков в классе. Как правило рекомендуется сделать их private (уровень доступа). Однако можно сделать их видимыми для подклассов этих переменных с помощью модификаторов доступа.
- Переменные экземпляра имеют значения по умолчанию. Для чисел по умолчанию равно 0, для логических — false, для ссылок на объект — null. Значения могут быть присвоены при объявлении или в конструкторе.
- Переменные экземпляра в Java могут быть доступны непосредственно путем вызова имени переменной внутри класса. Однако в статических методах и различных класса (когда к переменным экземпляра дана доступность) должны быть вызваны используя полное имя — ObjectReference.VariableName.

Пример

```
public class Employee{
    public String name;
    private double salary;
 public Employee (String empName){
    name = empName;
 }
 public void setSalary(double empSal){
    salary = empSal;
}
public void printEmp(){
    System.out.println("имя: " + name );
    System.out.println("зарплата:" + salary);
}
public static void main(String args[]){
    Employee empOne = new Employee("Олег");
    empOne.setSalary(1000);
    empOne.printEmp();
  }
}
```

Переменные класса или статические переменные в Java

- Переменные класса, также известные в Java как статические переменные, которые объявляются со статическим ключевым слово в классе, но за пределами метода, конструктора или блока.
- Там будет только одна копия каждой статической переменной в классе, независимо от того, сколько объектов создано из него.
- Статические переменные или переменные класса в Java используются редко, кроме когда объявляются как константы. Константы - переменные, которые объявлены как public/private, final и static. Константы никогда не меняются от первоначального значения.
- В Java статические переменные создаются при запуске программы и уничтожаются, когда выполнение программы остановится.
- Видимость похожа на переменную экземпляра. Однако большинство статических переменных объявляются как public, поскольку они должны быть доступны для пользователей класса.
- Значения по умолчанию такое же, как и у переменных экземпляра. Для чисел по умолчанию равно 0, для данных типа Boolean — false; и для ссылок на объект — null. Значения могут быть присвоены при объявлении или в конструкторе. Кроме того, они могут быть присвоены в специальных блоках статического инициализатора.
- Статические переменные могут быть доступны посредством вызова с именем класса ClassName.VariableName.
- При объявлении переменных класса как public, static, final, имена находятся в верхнем регистре. Если статические переменные такими не являются, синтаксис такой же, как у переменных экземпляра и локальных.

Пример

```
public class Employee{
private static double salary;

public static final String DEPARTMENT = "Разработка";

public static void main(String args[]){
      salary = 1000;
      System.out.println(DEPARTMENT+"средняя зарплата: "+salary);
   }
}
```