**JAVA**

1. **Начальный уровень**

|  |
| --- |
| System.out.println()/System.out.print() – метод используемый для вывода информации в консоли |
| class – ключевое слово определяющая класс (сущность собирающая в себе определённый набор данных и функций)  class \*имя\_класса\* {  \*имя\_класса\* {} - //Конструктор класса  \*методы и поля класса  } |
| public static void main(String[] args){} – по умолчанию, метод являющийся точкой входа в программу |
| Переменная – сущность, позволяющая хранить значения  \*тип\_данных\_переменной\* \*имя\_переменной\*  int a; - объявление переменной  a = 0; - присвоение значения переменной (если присвоение происходит впервые для переменной, процесс называется инициализацией переменной) (= оператор присваивания)  int a = new int() – объявление переменной вместе с инициализацией (в данном случае инициализация происходит за счёт обращения к конструктору класса базового типа данных)  Типы данных: int, String, double, float, char, boolean |
| Математические операторы:  + сложение (сложение строк называется конкатенацией)  - вычитание  \* умножение  / деление  % остаток от деления  x++ и ++x (x = x + 1) (инкременты с предусловием и постусловием соответственно)  x-- и --x (x = x - 1) (декременты с предусловием и постусловием соответственно)  += прибавить к значению слева значение справа  -= вычесть значение справа из значения слева  \*= умножить значение слева на значение справа  /= разделить значение слева на значение справа |
| //любой\_текст - однострочный комментарий  /\* любой\_текст \*/ - многострочный комментарий |
| import – используется для подключение сторонних классов |
| Ввод данных в консоли для Java требует:  Подключить библиотеку: import java.util.Scanner;  Создать объект класса Scanner: Scanner \*имя\_объекта\_класса\* = new Scanner(System.in);  Непосредственно ввод данных: String name = sc.nextLine();  Класс Scanner имеет еще ряд методов, которые позволяют получить введенные пользователем значения:  next(): считывает введенную строку до первого пробела  nextLine(): считывает всю введенную строку  nextInt(): считывает введенное число int  nextDouble(): считывает введенное число double  nextBoolean(): считывает значение boolean  nextByte(): считывает введенное число byte  nextFloat(): считывает введенное число float  nextShort(): считывает введенное число short |
| if – условный оператор  if (\*условие\*) {  \*выполняемые команды в случае удовлетворения условия\*  }  Операторы сравнения:  < меньше  > больше  != не равно  == равно  <= меньше или равно  >= больше или равно  else – используется с if. Обозначает блок кода, выполняемого при неудовлетворении условия, указанного в if  if (\*условие\*) {  \*выполняемые команды в случае удовлетворения условия\*  }  else {  \*выполняемые команды в случае неудовлетворения условия\*  }  Можно использовать сочетание else if для проверки иного условия  if (\*условие\*) {  \*выполняемые команды в случае удовлетворения условия\*  }  else if (\*условие\*) {  \*выполняемые команды в случае неудовлетворения предыдущего условия\*  }  else {  \*выполняемые команды в случае неудовлетворения условий\*  } |
| switch – оператор множества условий  switch(\*Применяемая\_переменная/значение\*)  {  case \*значение\*:  \*выполняемые команды в случае равенства входящего значения и значения case\*  break;  case \*значение2\*:  \*выполняемые команды в случае равенства входящего значения и значения case\*  break;  default:  \*выполняемые команды в случае несоответствия входящего значения со значениями во всех case\*  break;  } |
| Логические операторы:  && - логическое И  | | - логическое ИЛИ  ! – логическое НЕ |
| while - цикл с условием  while (\*условие\_нахождения\_в\_цикле\*) {  \*выполняемый код в случае удовлетворения условия\*  }  do while – цикл с условием в конце (один раз выполняется в любом случае)  do {  \*выполняемый код в случае удовлетворения условия и без условия но 1 раз\*  } while (\*условие\_нахождения\_в\_цикле\*) |
| for – цикл с условием в промежутке  for (\*начальное значение\*, \*условие\_нахождения\_в\_цикле\*, \*изменение значения\*){  \*выполняемый код в случае удовлетворения условия\*  } |
| break – ключевое слово, используемое для выхода из тела цикла или switch  continue - ключевое слово, используемое для пропуска итерации цикла |

1. **Средний уровень**
2. **Профессиональный уровень**