1. Java платформа – это среда где выполняется код написанный на языке программирования Java.
2. Java-файл содержит в себе понятный для человека код. Имя основного класса, объявленного в файле должен соответствовать имени файла.
3. JVM – виртуальная машина Java, платформа на которой выполняется код.

JDK – инструмент для разработки, содержащий в себе компилятор, стандартный библиотеки, документацию и JRE.

JRE – виртуальная машина с необходимыми библиотеками для выполнения приложения.

1. Компилирование командой **javaс <file.class>.** Пути в переменных средах прописываются, т.к. система по-умолчанию не имеет предустановленной JDK.
2. Public – поля и методы доступны другим классам в текущем пакете и во внешнем.

Private – доступен только в том же классе.

Protected - доступен из любого места в текущем классе или пакете.

1. Пакеты – это по сути директории в файловой системе и служат для организации файлов классов (позволяет избежать повторов в именах файлов классов).
2. import BigInteger, ArrayList, Scanner. Импортированы по-умолчанию ava.lang, java.util, java.io.
3. Сам объект занимает место в памяти, а ссылка на объект содержит в себе только адрес объекта.
4. Bool, short, byte, int, long, double, float, char. Все они различаются хранимыми в себе максимальными числами.

Создание int n = 4; String str = "sd"; double d = 324.324;

Примитивные типы данных передаются в методы как функции:

double x = 4;

double y = 6;

double z = Math.sin(x) + Math.cos(y) / Math.sin(x) - Math.cos(y);

System.out.println(z);

1. short(16 бит), int(32), byte(8), long(64), bouble(64), float(32), Boolean(8), char(16).

Преобразование происходит как правило в большую сторону from int to long. Сужение тоже может быть, но с явным указанием этого.

Не поддаются преобразованию Boolean, char.

1. Явное приведение типов – это когда в коде мы указываем какой тип данных мы хотим использовать или получить. Неявное приведение – это когда в процессе выполнения программы происходит преобразование, например:

byte a = 1;

byte b = 2;

или

~~int value = 4;~~ double value = 4;

System.out.println(8/value);

~~byte result = a + b;~~ byte result = (byte)(a + b);

1. Литерал - явно заданное значение. При помощи литералов можно указывать значения в разных системах счисления:

Десятеричная система: 10

Шестнадцатеричная система: 0x1F4, начинается с 0x

Восьмеричная система: 010, начинается с нуля.

Двоичная система (начиная с Java7): 0b101, начинается с 0b

1. Перед выполнением операции все аргументы автоматически приводятся к более универсальному типу, если это возможно и далее выполняется операция.
2. Класс-обертка содержит в себе значение примитива. Классы-обертки позволяют нивелировать недостатки, которые есть у примитивных типов.

Integer i = new Integer(682);

System.out.println(i.toString());

Поэтому у i есть методы.

Объект класса оболочки – константный объект, потому что значение примитива – константа.

1. Ссылочные переменные хранят адрес ячейки памяти, в которой расположен определенный объект и могут быть переопределены. Примитивы же – константы.

TV telly = new TV();

TV telly2 = new TV();

telly = telly2;

1. Автозапаковка - переменной класса-обертки можно присваивать значение примитивного типа.

Автораспаковка - переменной примитивного типа можно присваивать объект класса-обертки.

int x = 7;

Integer y = 111;

x = y; // автораспаковка

y = x \* 123; // автоупаковка

1. Математические \*, /, +, - (приоритет так же как в математике).

Логические И, ИЛИ, НЕ (приоритет, начиная с высшего: **!** > **&** > **^** > **|** > **&&** > **||**).

Побитовые &, |, ^, <<, <<<, >>>, >> (применимы к целочисленным типам).

1. Разобраться
2. Статический импорт позволяет улучшить читабельность кода.

import static java.lang.Math.cos;

double value = cos(PI \* 4);

System.out.println(value);

Это позволяет получить доступ к статическим переменным без указания имени класса.

1. **If** – оператор принятия решения.

if (a > b) {

System.out.println(a + " > " + b);

}

**Switch** – оператор выбора применим к известному числу возможных ситуаций.

int month = 3;

String monthStr;

switch (month) {

case 1: monthStr = "Январь";

break;

case 2: monthStr = "Февраль";

break;

default: monthStr = "Не извевестно";

break;

}

System.out.println(monthStr);

**While** – Если условие истинно, то тело цикла выполняется:

double b = 3;

while( b < 15 ) {

System.out.print("x < " + b);

b++;

System.out.print("\t");

}

**Do while** – тоже самое, только тело цикла выполнится гарантированно 1 раз.

int i = 1;

do {

i++;

System.out.print(i + " ");

} while(i < 0);

**For** – цикл со счётчиком, пока выполняется условие счётчик ведёт отсчёт с указанным шагом.

int n = 10;

for(int i=0, j = n - 1; i < j; i++, j--){

System.out.println(i \* j);

}

**For-each** – тоже самое, что и for, но менее универсальная. Можно применять там где реализован интерфейс Itereble.

List<String> names = new ArrayList<>();

names.add("Раз");

names.add("Два");

names.add("Три");

for(String name : names){

System.out.println(name);

}

Например, нет возможности удалять элементы при переборе в чистом виде. Нужно создавать ссылочный объект

HashSet<String> names = new HashSet <String>();

names.add("Раз");

names.add("Два");

names.add("Три");

HashSet<String> names2 = new HashSet <String>(names);

for(String name : names2){

System.out.println(name);

}

names2 = null; //чтобы не было утечки памяти.

1. Оператор instanceof нужен для того, чтобы проверить, был ли объект, на которую ссылается переменная X, создан на основе какого-либо класса Y. Оператор instanceof проверяет именно происхождение объекта, а не переменной.

При null будет false, т.к. это просто специальное значение ссылочного типа: