1. Java – высокоуровневый статически-типизированный объектно-ориентированный язык программирования, являющийся платформенно-независимым, используемым для разработки информационных систем.

Java-платформа – вычислительная платформа, основное назначение которой выполнять программы, написанные на языке Java.

1. Имя java-файла должно совпадать с названием класса, объявленном в этом файле. Регистр букв важен.
2. JVM – Java Virtual Machine – виртуальная машина Java, часть программного обеспечения Java, интерпретирующая байт-код, описываемый в файлах классов.

JDK – Java Development Kit – средство разработчика Java, включающее в себя набор утилит, стандартные библиотеки с их исходным кодом и набор демонстрационных примеров.

JRE – Java Runtime Environment – среда выполнения Java, предназначенная для запуска готовых приложений. Содержит реализацию виртуальной машины и набор стандартных библиотек.

1. Компиляция: javac filename.java.

Запуск: java filename.

В переменных среды окружения необходимо прописывать пути к установленному jdk для того, чтобы передавать информацию не только процессу, но и операционной системе. Таким образом, можно управлять некоторыми аспектами поведения операционной системы.

1. private – ограничение видимости данных в пределах одного класса;

default (package visible) – по умолчанию – класс + все классы внутри пакета;

protected – все классы пакета и классы-наследники;

public – видимость везде // для использования конечным пользователем.

1. Пакеты – контейнеры классов, которые используются для разделения пространства имён классов. На диске пакеты представляют собой вложенные папки. Многие именуют по имени компании/сайта. Создание пакета: Проект-Файл-Создать-Пакет.
2. Для подключения класса или пакета используется import. Импортировать надо те классы, интерфейсы и перечисления которых используются. При импорте пакета импортируются только классы, лежащие в этом пакете, подпакеты не импортируются. java.lang импортируется по умолчанию.
3. Объект – экземпляр класса, его конкретная реализация.

Ссылка на объект – указатель на данный объект.

1. int, short, long, float, double, Boolean, char, byte.

int x = 10; void myMethod(int u, double d);

1. Преобразование типов – процесс помещения значения менее разрядного типа в больший (или наоборот, но в таком случае сужение типов данных происходит с отбрасыванием лишних битов). Необходимо, когда, имея величину какого-либо определённого типа, нам необходимо присвоить её переменной другого типа. Boolean не приводится ни к какому типу.
2. Неявное преобразование – автоматическое. Происходит тогда, когда оба типа совместимы и длина конечного типа больше или равна длине исходного типа. Явное преобразование – когда необходимо чётко прописывать, что необходимо привести.

Повышающее (расширяющее) преобразование – неявное. Конечный тип имеет больший диапазон значений.

Сужающее (понижающее) преобразование всегда явное.

1. Литералы – явно заданные значения в коде программы (константы определённого типа, находящиеся в коде в момент запуска). Классификация – числовые (целочисленные и с плавающей точкой), строковые, символьные, логические, null(единственный объект-литерал).
   1. целочисленные: 0B11010, 01232, 456. 0x141;
   2. с плавающей запятой: 3.14159, 4.05E-13F;
   3. строковые: “Hello World”;
   4. символьные: ‘%’, ‘\u00F7’;
   5. логические: true, false.
2. При делении необходимо явно преобразовывать одно из чисел к числу с плавающей запятой, чтобы получить как результат число с плавающей запятой.
3. Класс-оболочка – объект-аналог примитивных типов (он копирует то, что уже существует, но добавляет новые возможности). Примитивы занимают меньше места, но при помощи класса-оболочки можно выполнять спецоперации (к примеру, перевод текста в число). Объект любого из классов-оболочек представляет собой экземпляр в динамической памяти, в которой хранится её неизменяемое значение.
4. Примитивные переменные хранят фактические значения, а ссылочные переменные хранят ссылку (указатель) на объект, на которые они ссылаются. Java передаёт и примитивные, и ссылочные переменные по значению, но разница заключается в том, что при передаче ссылочной переменной она предоставляет копию ссылки, т.е. ссылку на изначальный объект.
5. Автоупаковка – функция преобразования примитивных типов в объектные (распаковка, соответственно, преобразует объектные типы в примитивные). Автоупаковка используется, когда примитив передаётся в метод как параметр, но метод ожидает объект класса-оболочки; или когда значение примитивного типа присваивается объекту соответствующего класса-оболочки. Распаковка используется, когда объект передаётся как параметр методу, который ожидает примитив, или когда объект присваивается переменной соответствующего примитивного типа.
6. Арифметические операторы в порядке убывания приоритета: ++, --; +, -; \*, /, %; +, -.

Логические операторы в порядке убывания приоритета: !; &, ^, |, &&, ||.

Битовые операторы в порядке убывания приоритета: ~; <<, >>; >>>; &; ^; |.

При равности приоритетов операторов они применяются слева направо (за исключением постфиксных инкремента и декремента).

1. Нельзя использовать оператор сравнения == при сравнении чисел с плавающей запятой. Можно использовать пороговое значение (достаточно маленькое число, которое должно быть меньше, чем модуль разности сравниваемых чисел). Можно использовать класс BigDecimal – он исключает ошибки округления. Представление в экспоненциальном виде числа (знак числа s, мантисса m, степень e: ). Деление на ноль приведёт к результату бесконечность (знак бесконечности зависит от знака делимого). При делении нуля на ноль результат будет NaN (Not a Number). Проверка того, является ли число NaN, состоит в том, чтобы сравнить число с самой собой: NaN никогда не равно самому себе, оно не равно ничему. Так же есть функции Double.isNaN(), Float.isNaN().
2. Статический импорт – import static packet\_name.class\_name.\* – используется для импорта статических полей и методов класса, в результате чего нет необходимости использовать имя класса перед вызовом метода.
3. Примеры:
   1. if – ветвление. Выбор одного из случаев, но никогда какие-либо два сразу.

bool canDoSomething; /\*присвоение значения\*/

if (canDoSomething) {

System.out.println(“canDoSomething = true”);

}

* 1. while – цикл с условием.

while (canDoSomething) {

/\*программа делает что-нибудь. важно сделать так, чтобы значение canDoSomething когда-нибудь стало false, иначе цикл будет выполняться вечно\*/

}

* 1. do-while – цикл с условием. Главное отличие от while состоит в том, что выполняется хотя бы один раз.

do {

/\*программа делает что-нибудь. важно сделать так, чтобы значение canDoSomething когда-нибудь стало false, иначе цикл будет выполняться вечно\*/

} while(canDoSomething);

* 1. for – цикл на определенное количество шагов.

for (int i = 0; I < 5; i++) {

System.out.println(i);

}

* 1. for-each – цикл на определённое количество шагов. Главное отличие от for’а состоит в том, что он предназначен для работы с контейнерами.

int[] myArrayy;

for (int I : myArray) {

doSomething();

}

1. instanceof нужен, чтобы проверить, был ли объект создан на основе другого объекта. null instanceof Something == false (согласно Java Language Specification).