Что такое технология Java?

**Ответ.** Технология Java - это одновременно язык программирования и платформа.

Язык программирования Java является объектно-ориентированным языком высокого уровня.

Платформа - это аппаратно-программная среда, в которой выполняется java-программа. Платформа Java состоит из двух компонентов:

1. виртуальная машина Java,

2. Java API.

**Источник.** <https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html>

Что такое JVM?

**Ответ.** **JVM** (Java Virtual Machine) - виртуальная машина Java - основная часть исполняющей системы Java, так называемой Java Runtime Environment (JRE). Виртуальная машина Java исполняет байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java (javac). JVM обеспечивает платформо-независимый способ выполнения кода.

**Источник.** <https://www.examclouds.com/ru/java/java-core-russian/jdk-jre-jvm>

Что такое JDK?

**Ответ. JDK** (Java Development Kit) - комплект разработчика приложений на языке Java, включающий в себя компилятор, стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему JRE.

**Источник.** <https://www.examclouds.com/ru/java/java-core-russian/jdk-jre-jvm>

Что такое JRE?

**Ответ.** **JRE** (Java Runtime Environment) - минимальная реализация виртуальной машины, необходимая для исполнения Java приложений, без компилятора и других средств разработки. Состоит из виртуальной машины и библиотек Java классов.

**Источник.** <https://www.examclouds.com/ru/java/java-core-russian/jdk-jre-jvm>

Даны корректные исходные файлы: Hello.java и hello.cpp. Каждый из них преобразуется в исполняемый код, который запускается на выполнение. Назовем данный процесс жизненным циклом приложения. В чем отличие жизненных циклов этих приложений?

Перечислите группы примитивных типов данных и состав каждой группы.

1) целые числа (byte, short, int, long);

2) числа с плавающей точкой (float, double);

3) логический (boolean);

4) символьный (char).

**Источник.** <https://skillbox.ru/media/base/tipy-dannykh-v-java/>

Какой размер данного булевского типа?

**Ответ**. В стандартной реализации Sun JVM и Oracle HotSpot JVM тип **boolean** занимает 4 байта (32 бита), как и тип int. Однако, в определенных версиях JVM имеются реализации, где в массиве boolean каждое значение занимает по 1-му биту.

**Источник.** <https://ru.wikibooks.org/wiki/Java/%D0%A2%D0%B8%D0%BF%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85>

Какой размер данного символьного типа?

**Ответ.** Для char в Java предусмотрена Unicode кодировка. Диапазон значений составляет от 0 до 65536. Размер составляет 2 байта. При этом, в отличие от предыдущих типов переменных, отрицательных значений у char нет. Важно понимать, что в этой переменной не хранится сам символ. Здесь содержится его кодировка из Unicode. Соответственно, мы можем проводить целочисленные операции с символами.

**Источник.** <https://vc.ru/dev/142318-peremennye-i-tipy-dannyh-v-java>

Какими литералами можно инициализировать символьную переменную?

**Ответ.** 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:

char symb1=1067;

char symb2 ='Ы';

**Источник.** <http://study-java.ru/uroki-java/urok-6-sintaksis-java-peremennye-i-tipy-dannyh/>

Какие есть виды переменных?

**Ответ.**

локальные переменные;

переменные экземпляра;

статические переменные или переменные класса.

**Источник.** <http://proglang.su/java/variable-types>

В какой области памяти хранятся локальные переменные?

**Ответ.** Локальные переменные примитивного типа полностью хранятся в стеке. При создании объекта, ссылка хранится в стеке, сам же объект хранится в куче. При создании массива примитивных типов происходит то же самое что и с объектами: ссылка на массив хранится в стеке, а сам массив в куче.

**Источник.** <http://www.quizful.net/interview/java/java-local-variables>

В какой области памяти хранятся аргументы метода?

**Ответ.** Аргументы метода хранятся в стеке, каждый раз при вызове очередного метода, все необходимые аргументы функции заносятся в стек.

**Источник.** <https://ru.stackoverflow.com/questions/1004489/%D0%92%D0%BE%D0%BF%D1%80%D0%BE%D1%81-%D0%BF%D0%BE-%D1%81%D1%82%D0%B5%D0%BA%D1%83-%D0%B8-%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%B0%D0%BC-%D0%B2-java>

В какой области памяти хранятся объекты?

**Ответ.** Переменные-члены объекта хранятся в куче вместе с самим объектом. Это верно как в случае, когда переменная-член имеет примитивный тип, так и в том случае, если она является ссылкой на объект.

**Источник.** <https://habr.com/ru/post/510454/>

В какой области памяти хранятся ссылки на объекты?

**Ответ.** Ссылка хранится в стеке потоков, но сам объект хранится в куче.

**Источник.** <https://habr.com/ru/post/510454/>

В какой области памяти хранятся массивы?

**Ответ.** При создании массива примитивных типов происходит то же самое что и с объектами: ссылка на массив хранится в стеке, а сам массив в куче.

**Источник.** <http://www.quizful.net/interview/java/java-local-variables>

Дайте определение массива.

**Ответ.** Массив — это структура данных, в которой хранятся элементы одного типа. Его можно представить, как набор пронумерованных ячеек, в каждую из которых можно поместить какие-то данные (один элемент данных в одну ячейку). Доступ к конкретной ячейке осуществляется через её номер. Номер элемента в массиве также называют индексом.

**Источник.** <https://javarush.ru/groups/posts/massivy-java>

Назовите 3 этапа создания массива. Какие действия выполняет JVM на каждом этапе?

**Ответ.**

**Объявление (declaration)**

В области памяти стек создается переменная - ссылка на массив, но не указывающая ни на какой массив.

**Создание (instantation)**

Выделяется место для массива в оперативной памяти, переменной-ссылке присваивается оператором **=** адрес массива. Все эти действия производятся оператором **new** за которым следует *тип* элементов массива.

При создании массива с таким синтаксисом все элементы массива автоматически инициализируются значениями по умолчанию. Это false для значений boolean, '\u0000'  для значений char, 0 для целых значений, 0.0 для значений с плавающей точкой и null для объектов или массивов.

**Инициализация  (initialization)**

На этом этапе элементы массива получают начальные значения.

**Источник.** <http://pr0java.blogspot.com/2015/05/1.html>

Приведите различные способы инициализации массива.

**Ответ.** Инициализировать элементы массива значениями можно несколькими способами:

Присвоить каждому элементу массива конкретное значение (это можно сделать например в цикле, но до этого массив уже должен быть объявлен и создан);

**int**[] **ar** = **new** **int**[2];  
ar[0]=1;  
ar[1]=2;

Инициализировать массив при помощи перечисления значений его элементов в фигурных скобках (это можно сделать как на этапе объявления, так и на этапе создания, но синтаксис при этом разный)

Второй способ инициализации можно реализовать по разному.

Инициализацию массива можно совместить с этапом создания, но до этой операции массив уже должен быть объявлен. Например:

**int**[] **ar**; // объявление массива  
ar = **new** **int**[]{1,2}; // создание и инициализация

До создания и инициализации массива **ar** он уже был объявлен.

Так же инициализировать массив можно на этапе его объявления следующим синтаксисом:

**int**[] **ar** = {1,2}; // объявление, создание и инициализация массива

**Внимание!** Этот синтаксис инициализации массива работает только при объявлении массива и совмещает сразу все три операции объявление, создание и инициализацию. Если массив уже объявлен, то такой синтаксис использовать нельзя.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**int**[] **ar** = **new** **int**[]{1,2}; // объявление, создание и инициализация

Хотя этот синтаксис более длинный. Если вы заметили, то данный синтаксис это тоже совмещение всех трех операций: объявления, создания и инициализации.

**Источник.** <http://pr0java.blogspot.com/2015/05/1.html>

Дан массив

int[] values = {...};

//в фигурных скобках корректный список инициализации.

Есть ли различие в проходе по массиву циклами:

1. for(int i=0; i < values.length; i++) {...}

2. for(int value : values) {...}

Если да, то в чем различие заключается?

Как называется каждый цикл?

**Ответ.** между циклом for и for-each есть некоторые тонкие различия. Цикл for требует, чтобы количество итераций было указано заранее. Однако это не относится к циклу for-each, поскольку цикл повторяется от первого элемента к последнему элементу Collection / array и не требует указания количества итераций.

1. «Цикл со счетчиком»;
2. «Для каждого».

**Источник.** <https://javarush.ru/groups/posts/cikly-java>

<https://ru.education-wiki.com/4984678-for-each-loop-in-java>

Можно ли создать массив нулевой длины? Если да, то для каких целей он необходим?

**Ответ.**

Можно создавать и использовать массивы нулевой длины (пустой массив). Например:

**boolean**[] **bits** = **new** **boolean**[0];

Инициализировать такой массив нельзя, так как у него просто нет элементов которые можно инициализировать. Сразу же возникает вопрос, а на кой ляд они тогда вообще нужны эти пустые массивы? Но они нужны и даже очень полезны!

Пустой массив принято использовать в тех местах программы, где заранее неизвестно, будут элементы или нет. Если элементы будут, то возвращается непустой массив, если элементов нет - пустой массив. Примером может служить массив строк который передается в метод main() и содержит аргументы командной строки, а если их нет, то возвращается пустой массив.

Пустой массив лучше, чем **null**, потому что не требует отдельного if'а для обработки. То же верно для списков и других коллекций. Именно поэтому существуют методы Collections.emptyList, emptySet, emptyMap.

**Источник.** <http://pr0java.blogspot.com/2015/05/1.html>

Что собой представляет двумерный массив? Многомерный массив?

**Ответ.** Двумерный массив в Java — это массив массивов, то есть в каждой его ячейке находится ссылка на некий массив. Но гораздо проще его представить в виде таблицы, у которой задано количество строк (первое измерение) и количество столбцов (второе измерение). Двумерный массив, у которого все строки имеют равное количество элементов, называется прямоугольным.

Трёхмерным массивом можно назвать "массив массивов массивов" или "массив, каждым элементом которого является двумерный массив

**Источник.** <https://javarush.ru/groups/posts/mnogomernye-massivy>

Перечислите ссылочные типы.

**Ответ.** Существует [четыре типа ссылочных данных в Java](https://docs.oracle.com/javase/specs/jls/se8/html/jls-4.html#jls-4.3):

* **Классы** ([class types](https://docs.oracle.com/javase/specs/jls/se8/html/jls-8.html#jls-8.1))
* **Интерфейсы** ([interface types](https://docs.oracle.com/javase/specs/jls/se8/html/jls-9.html#jls-9.1))
* **Переменные типов** ([type variables](https://docs.oracle.com/javase/specs/jls/se8/html/jls-4.html#jls-4.4))
* **Массивы** ([array types](https://docs.oracle.com/javase/specs/jls/se8/html/jls-10.html#jls-10.1))

Ссылочные типы хранят ссылку на объект, или же тип данных **null**, то есть нулевую (пустую) ссылку.

**Источник.** <http://pr0java.blogspot.com/2015/04/java-1.html>

Какая структура у java-файла (внутренности класса не указывать)?

Какие из элементов структуры являются обязательными?

**Ответ.**

1. Имя пакета
2. Импортируемые классы и пакеты
3. Имя класса
4. Переменные класса
5. Методы класса

В любой строке программы могут размещаться комментарии

Обязательными являются: имя класса, метод main().

**Источник.-**

Для чего предназначены пакеты?

**Ответ.**  Пакеты позволяют организовать классы логически в наборы. Организация классов в виде пакетов позволяет избежать конфликта имен между классами. Ведь нередки ситуации, когда разработчики называют свои классы одинаковыми именами. Принадлежность к пакету позволяет гарантировать однозначность имен.

Чтобы указать, что класс принадлежит определенному пакету, надо использовать директиву package, после которой указывается имя пакета:

|  |  |
| --- | --- |
| 1 | package название\_пакета; |

**Источник.** <https://metanit.com/java/tutorial/3.2.php>

Как выглядит полное имя класса?

**Ответ.** полным именем класса считается имя, состоящее из всех пакетов, перечисленных через точку и имени класса

java.lang.**String**

Полным именем класса или интерфейса в Java является:

**<имя package>.<имя класса>**

в случае, если класс или интерфейс не является вложенным в какой-либо другой класс или интерфейс.

**<полное имя класса, в который вложен наш класс>.<имя класса>**

в случае, если класс является вложенным

**Источник.** <https://foxford.ru/wiki/informatika/polnoe-imya-klassa-v-java>

Для чего предназначен оператор import?

**Ответ**. оператор import **значительно сокращает** объем вводимого исходного кода. Позволяет не вводить полное используемого класса в программе.

**Источник.** <https://metanit.com/java/tutorial/3.2.php>

Для чего предназначен статический импорт?

**Ответ.** В случае, когда классы имеют статические методы, то благодаря статическому импорту мы можем использовать имена методов без имени класса. Например, писать не Math.sqrt(20), а sqrt(20).

**Источник.** <https://metanit.com/java/tutorial/3.2.php>

Сколько внешних публичных классов можно объявить в одном java-файле?

**Ответ.** В одном физическом java-файле может быть только один публичный класс, имя которого должно совпадать с именем файла и любое количество вложенных классов.

**Источник.** <https://ru.stackoverflow.com/questions/532188/%D0%A1%D0%BA%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE-public-%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2-%D0%BC%D0%BE%D0%B6%D0%B5%D1%82-%D0%B1%D1%8B%D1%82%D1%8C-%D0%B2-%D0%BE%D0%B4%D0%BD%D0%BE%D0%BC-%D1%84%D0%B0%D0%B9%D0%BB%D0%B5>

Может ли синтаксически правильный java-файл иметь имя 1.java? Если да, то какие ограничения?

**Ответ.** Идентификаторы могут содержать цифры, но не могут начинаться с цифры. Более того, идентификаторы не могут содержать знаков пунктуации, за исключением подчеркивания и валютных знаков

**Источник.** <http://pr0java.blogspot.com/2015/03/java-2-java.html>

Допустим, в одном java-файле объявлены два синтаксически правильных внешних класса. Какие могут быть результаты компиляции данного файла?

**Ответ.** Любой Java-класс хранится не более чем в одном файле. В любой файл с исходным кодом можно поместить не более 1-го класса с модификатором public. Если в файле с исходным кодом есть класс с модификатором public, имя файла и имя класса должны быть строго одинаковыми.

В результате компиляции возникнет ошибка на этапе компиляции

**Источник.** <https://javarush.ru/groups/posts/1392-oshibki-nachinajujshikh-java-programmistov-chastjh-1>

Как записывается заголовок метода main() раннер-класса (т.е. класса, который может стартовать из-под JVM)?

Это единственный способ?

**Ответ.**

Метод main() должен объявляться в классе следующим образом:

1. **public** **static** **void** main(String[] args)

Вместо имени массива args можно использовать любое другое имя. Например, следующее описание main() вполне корректно.

1. **public** **static** **void** main(String[] strings)

тип и количество параметров метода main() изменению не подлежат.

**Источник.** <http://zonakoda.ru/vsyo-o-metode-main.html>

Может ли один java-файл содержать два и более раннер-класса?

**Ответ.** Да, у вас может быть больше классов, содержащих метод main(), но по крайней мере один класс, который содержит main(), должен быть общедоступным, чтобы JMV запустил этот класс в качестве основного потока.

**Источник.** <https://overcoder.net/q/1007747/%D0%BC%D0%BE%D0%B3%D1%83-%D0%BB%D0%B8-%D1%8F-%D0%B8%D0%BC%D0%B5%D1%82%D1%8C-%D0%B1%D0%BE%D0%BB%D0%B5%D0%B5-%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE-%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0-%D1%81%D0%BE%D0%B4%D0%B5%D1%80%D0%B6%D0%B0%D1%89%D0%B5%D0%B3%D0%BE-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4-main-%D0%B2-%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B5-java>

Можно ли определить метод main() с другими модификаторами?

**Ответ.** Нельзя. Метод main() загружается первым, до создания какого либо объекта, поэтому для его вызова, он должен быть static, также метод должен быть public для того, чтобы этот метод был виден из любого пакета.

**Источник.-**

Может ли в классе быть два и более метода main()?

Если да, то приведите пример.

**Ответ.** Может, если второй метод будет иметь модификатор доступа не public.

public class A{

public static void main (String[] args) {

}

private static void main(int[] args) {

}

}

**Источник.-**

Может ли в классе быть два метода main(), один из которых раннер, а второй метод main() отличается от первого только отсутствием модификатора static?

Если да, то приведите пример вызова второго метода.

**Ответ.** Не может. При попытке вызова второго метода main из раннер-метода произойдет ошибка компиляции.

**Источник.-**

В корне проекта имеется файл HelloWorld.java, пути к необходимым ресурсам открыты.

Создайте bat-файл для запуска приложения на выполнение.

**Ответ.** Создать текстовый файл с командами приведенными ниже, изменить его расширение на .bat. Запустить созданный файл.

cd /D e:\javalesson

javac HelloWorld.java

java HelloWorld

pause

**Источник.** <https://viarum.ru/kak-sozdat-bat-fayl-v-windows/>

<https://habr.com/ru/post/125210/>

В корне проекта имеется

1. папка src для исходных java-файлов, а в ней – файл HelloArgs.java, который выводит приветствие для каждого аргумента командной строки;

2. папка bin для class-файлов.

Пути к компилятору и JVM открыты, переменная CLASSPATH не установлена.

Создайте bat-файл для запуска приложения с двумя аргументами командной строки clever и 25007 на выполнение.

**Ответ.**

**Источник.**

Имеется корректный файл helloWorld.jar.

Создайте bat-файл для запуска приложения на выполнение.

**Ответ.** Создать текстовый файл с командами приведенными ниже, изменить его расширение на .bat. Запустить созданный файл.

cd /D e:\javalesson

java -jar helloWorld.jar.

**Источник.** <https://coderoad.ru/394616/%D0%97%D0%B0%D0%BF%D1%83%D1%81%D0%BA-%D1%84%D0%B0%D0%B9%D0%BB%D0%B0-JAR-%D0%BD%D0%B0-Windows>

Какие есть виды комментариев?

**Ответ.**

Java поддерживает три типа комментариев –

* Однострочные комментарии – С их помощью вы можете прокомментировать отдельные строки. Компилятор игнорирует все от // до конца строки.
* Многострочные комментарии – Можете прокомментировать несколько строк. Компилятор игнорирует все от / \* до \* /.
* Документационные комментарии – Инструмент Javadoc JDK использует этот вид при подготовке автоматически сгенерированной документации.

**Источник.** <https://hr-vector.com/java/kommentarii-odnostrochnyj-mnogostrochnyj>