# Questions

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 является сердцем языка программирования Java. Когда мы запускаем программу, JVM несет ответственность за преобразование байт-кода в машинный код. JVM также зависит от платформы и предоставляет основные функции, такие как управления памятью Java, сборкой мусора, и т.д. Мы также можем выделять определенный объем памяти для JVM. JVM является виртуальной машиной, потому что обеспечивает интерфейс, который не зависит от операционной системы и аппаратных средств.

**Источник.** <https://javadevblog.com/chto-takoe-jdk-jre-i-jvm-v-java.html>

**Что такое JDK?**

**Ответ:** Java Development Kit является основным компонентом среды Java и предоставляет все инструменты, исполняемые и бинарные файлы, которые нужны для компиляции, отладки и выполнения программы на Java.

**Источник:** <https://javadevblog.com/chto-takoe-jdk-jre-i-jvm-v-java.html>

**Что такое JRE?**

**Ответ:** JRE является реализацией JVM, которая предоставляет платформу для выполнения Java-программ. JRE состоит из виртуальной машины Java, бинарных файлов и других классов. JRE не содержит инструменты для разработки (компилятор Java, отладчик и т.д). Если вы хотите запустить любую Java программу, вы должны установить JRE.

**Источник:** <https://javadevblog.com/chto-takoe-jdk-jre-i-jvm-v-java.html>

**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?**

**Ответ.** Класс описывается в своем файле, который имеет расширение .java. Это – ИСХОДНЫЙ КОД. Для исполнения его необходимо скомпилировать. (Это я говорю специально для тех, кто пытается запустить .java-файл). После компиляции (успешной!) появляется файл с расширением .class и тем же именем, что и файл .java.

Описание расширения CPP

Расширение CPP – это исходный код файла, написанного на C++, популярном языке программирования, который добавляет функциональность, такую как объектно-ориентированное программирование на C; может быть отдельной программой или одним из многих файлов в проекте, должен быть скомпилированным с помощью компилятора C++ для необходимой ОС перед работой.

CPP файлы чаще всего редактируются программами, которые обеспечивают подсветку синтаксиса. Но вы можете открыть CPP файлы с помощью любого текстового редактора, но чаще всего используются программы, которые обеспечивают подсветку синтаксиса, автозаполнение и другие полезные инструменты.

C++ является одним из самых популярных языков программирования и, следовательно, файлы .CPP очень часто встречаются.

**Источник.** <http://skipy.ru/technics/likbez.html>

<http://fileext.ru/cpp>

**Перечислите группы примитивных типов данных и состав каждой группы.**

**Ответ.**

* **byte** : Тип byteданных - это 8-битовое знаковое целое число с дополнением до двух. Он имеет минимальное значение -128 и максимальное значение 127 (включительно). Тип byteданных может быть полезен для экономии памяти в больших [массивах](https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html) , где экономия памяти действительно имеет значение. Их также можно использовать вместо тех, intгде их ограничения помогают прояснить ваш код; тот факт, что диапазон переменной ограничен, может служить формой документации.
* **Short**  : shortтип данных - это 16-битовое знаковое целое число с дополнением до двух. Он имеет минимальное значение -32 768 и максимальное значение 32 767 (включительно). Как и в случае byte, применяются те же правила: вы можете использовать a shortдля экономии памяти в больших массивах в ситуациях, когда экономия памяти действительно имеет значение.
* **int** : По умолчанию intтип данных представляет собой 32-битное знаковое целое число с дополнением до двух, которое имеет минимальное значение -2 31 и максимальное значение 2 31 -1. В Java SE 8 и более поздних версиях вы можете использовать intтип данных для представления 32-разрядного целого числа без знака, минимальное значение которого равно 0, а максимальное - 2 32 -1. Используйте класс Integer, чтобы использовать intтип данных как целое число без знака. См. Раздел «Числовые классы» для получения дополнительной информации. В класс были добавлены статические методы, такие как compareUnsignedи divideUnsignedт. Д., [Integer](https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html" \t "_blank)Для поддержки арифметических операций с целыми числами без знака.
* **long** : longтип данных - 64-битное целое число с дополнением до двух. Длинное число со знаком имеет минимальное значение -2 63 и максимальное значение 2 63 -1. В Java SE 8 и более поздних версиях вы можете использовать longтип данных для представления 64-битной длины без знака, которая имеет минимальное значение 0 и максимальное значение 2 64 -1. Используйте этот тип данных, когда вам нужен более широкий диапазон значений, чем предоставленный int. [Long](https://docs.oracle.com/javase/8/docs/api/java/lang/Long.html" \t "_blank)Класс также содержит методы , такие как compareUnsigned, и divideUnsignedт.д. , чтобы поддерживать арифметические операции для беззнаковых долго.
* **float** : floatтип данных - 32-битное число с плавающей запятой одинарной точности IEEE 754. Диапазон его значений выходит за рамки этого обсуждения, но указан в разделе « [Типы, форматы и значения](https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.3) с [плавающей запятой»](https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.3) Спецификации языка Java. Как и в случае с рекомендациями для byteи short, используйте float(вместо double), если вам нужно сохранить память в больших массивах чисел с плавающей запятой. Этот тип данных никогда не следует использовать для точных значений, таких как валюта. Для этого вам нужно будет вместо этого использовать класс [java.math.BigDecimal](https://docs.oracle.com/javase/8/docs/api/java/math/BigDecimal.html" \t "_blank) . Обложки [Numbers and Strings](https://docs.oracle.com/javase/tutorial/java/data/index.html" \t "_top)BigDecimal и другие полезные классы, предоставляемые платформой Java.
* **double** : doubleтип данных - 64-битное число с плавающей запятой двойной точности IEEE 754. Диапазон его значений выходит за рамки этого обсуждения, но указан в разделе « [Типы, форматы и значения](https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.3) с [плавающей запятой»](https://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.3) Спецификации языка Java. Для десятичных значений этот тип данных обычно выбирается по умолчанию. Как упоминалось выше, этот тип данных никогда не следует использовать для точных значений, таких как валюта.
* **boolean** : Тип booleanданных имеет только два возможных значения: trueи false. Используйте этот тип данных для простых флагов, которые отслеживают истинные / ложные условия. Этот тип данных представляет один бит информации, но его «размер» не определен точно.
* **char** : charтип данных - один 16-битный символ Unicode. Он имеет минимальное значение '\u0000'(или 0) и максимальное значение '\uffff'(или 65 535 включительно).

|  |  |  |  |
| --- | --- | --- | --- |
| Тип | Размер | По умолчанию | Значения |
| boolean | 8 | false | true , false |
| byte | 8 | 0 | −128..127 |
| char | 16 | '\0' | 'a', '\60', '\n' |
| short | 16 | 0 | −32768..32767 |
| int | 32 | 0 | −2147483648..2147483647 |
| long | 64 | 0 | 922372036854775808L |
| float | 32 | 0.0f | 3.40282347E+38 |
| double | 64 | 0.0 | 1.797693134486231570E+308 |

**Источник.** https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

**Какой размер данного булевского типа?**

**Ответ.** 8 бит.

**Источник.** https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

**Какой размер данного символьного типа?**

**Ответ.** 16 бит.

**Источник.** https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

**Какими литералами можно инициализировать символьную переменную?**

**Ответ.**

Язык программирования Java также поддерживает несколько специальных последовательностей для charи Stringлитералов: \b(BackSpace), \t(вкладка), \n(строки), \f(форма подачи), \r(возврат каретки), \"(двойные кавычки), \'(одинарная кавычка), и \\( обратная косая черта).

Также есть специальный nullлитерал, который можно использовать как значение для любого ссылочного типа. nullможет быть присвоено любой переменной, кроме переменных примитивных типов. Вы мало что можете сделать с nullценностью, кроме проверки ее наличия. Поэтому nullчасто используется в программах как маркер, указывающий на недоступность какого-либо объекта.

**Источник.** https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

**Какие есть виды переменных?**

Ответ.

В языке программирования Java определены следующие типы переменных:

* **Переменные экземпляра (** нестатические поля **) С** технической точки зрения объекты хранят свои индивидуальные состояния в «нестатических полях», то есть в полях, объявленных без staticключевого слова. Нестатические поля также известны как *переменные экземпляра,* поскольку их значения уникальны для каждого *экземпляра* класса (другими словами, для каждого объекта); currentSpeedодного велосипеда не зависит от currentSpeedдругого.
* **Переменные класса (статические поля)***переменная класса* является любое поле объявляется с помощью staticмодификатора; это сообщает компилятору, что существует ровно одна копия этой переменной, независимо от того, сколько раз был создан экземпляр класса. Поле, определяющее количество передач для конкретного типа велосипеда, можно было бы пометить как, staticпоскольку концептуально одинаковое количество передач будет применяться ко всем экземплярам. Код static int numGears = 6;создаст такое статическое поле. Кроме того, finalможно добавить ключевое слово , чтобы указать, что количество передач никогда не изменится.
* **Локальные переменные** Подобно тому, как объект сохраняет свое состояние в полях, метод часто сохраняет свое временное состояние в *локальных переменных* . Синтаксис объявления локальной переменной аналогичен объявлению поля (например, int count = 0;). Не существует специального ключевого слова, определяющего переменную как локальную; это определение полностью исходит из того места, в котором объявлена ​​переменная - между открывающей и закрывающей фигурными скобками метода. Таким образом, локальные переменные видны только тем методам, в которых они объявлены; они недоступны для остального класса.
* **Параметры** Вы уже видели примеры параметров как в Bicycleклассе, так и в mainметоде "Hello World!" применение. Напомним, что подпись mainметода есть public static void main(String[] args). Здесь argsпеременная является параметром этого метода. Важно помнить, что параметры всегда классифицируются как «переменные», а не «поля». Это также применимо к другим конструкциям, принимающим параметры (например, конструкторам и обработчикам исключений), о которых вы узнаете позже в этом руководстве.

Источник. https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html

**В какой области памяти хранятся локальные переменные?**

**Ответ:** Локальные переменные реализуется на внутреннем уровне стека

**Источник:** <https://elekt.tech/java/java_basics/types_of_variables_in_java.html>

**В какой области памяти хранятся аргументы метода?**

**Ответ:** В стеке выделяется область памяти для хранения аргументов метода.

**Источник:**

**В какой области памяти хранятся объекты?**

**Ответ:** Объекты храниться в так называемой Яме или Heap-памяти.

**Источник:** <https://javadevblog.com/chto-takoe-heap-i-stack-pamyat-v-java.html>

**В какой области памяти хранятся ссылки на объекты?**

**Ответ:** Ссылки на объекты в методе храниться в Стеке.

**Источник:**

**В какой области памяти хранятся массивы?**

**Ответ**. Любой проинициализированный массив объектов в Java хранит ссылки на объекты. Ссылки храниться в Стеке. Следовательно, и массив хранится в стеке.

**Источник**. <https://qa-help.ru/questions/vydelenie-pamyati-pod-massiv-3>

**Дайте определение массива.**

**Ответ.** Массив — это совокупность переменных одного типа, хранящихся в смежных ячейках постоянного участка оперативной памяти.

*Массив* представляет собой объект контейнера , который содержит определенное количество значений одного типа. Длина массива устанавливается при создании массива. После создания его длина фиксируется.

**Источник.** **Конспект jse18.** https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

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

**Ответ.**

*Первый этап — объявление (declaration).*

На этом этапе определяется только переменная типа ссылка (reference) на массив, содержащая тип массива.

*Второй этап — определение (installation).*

На этом этапе указывается количество элементов массива, выделяется место для массива в оперативной памяти, переменная-ссылка получает адрес массива. Память выделяет JVM.

*Третий этап — инициализация (initialization).*

На этом этапе элементы массива получают начальные значения.

**Источник**. **Конспект jse18.**

**Приведите различные способы инициализации массива.**

**Ответ.**

Третий этап — *инициализация* (initialization).

На этом этапе элементы массива получают начальные значения.

Например,

а[0] = 0.01; а[1] = -3.4; а[2] = 2.89; а[3] = 4.5; а[4] = -6.7;

for (int i = 0; i < 100; i++) b[i] = 1.0 /i;

for (int i = 0; i < 50; i++) ar[i] = 2 \* i + 1;

Первые два этапа можно совместить:

double[] a = new double[5];

double[] b = new double[100];

Можно сразу задать и начальные значения, записав их в фигурных скобках через запятую в виде констант или константных выражений. При этом даже необязательно указывать количество элементов массива, оно будет равно количеству начальных значений.

double[] а = {0.01, -3.4, 2.89, 4.5, -6.7};

Можно совместить второй и третий этап:

а = new double[] {0.1, 0.2, -0.3, 0.45, -0.02};

Можно даже создать безымянный массив, сразу же используя результат операции new, например, так:

System.out.println(new char[] {'H', 'e', 'l', 'l', 'o'});

**Источник. Конспект jse18.**

**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.**

1. Первый цикл for -это общая форма цикла.

2. У forоператора также есть другая форма, предназначенная для итерации через [Коллекции](https://docs.oracle.com/javase/tutorial/collections/index.html) и [массивы.](https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html) Эту форму иногда называют *расширенным* оператором *for(foreach)* , и ее можно использовать, чтобы сделать ваши циклы более компактными и удобными для чтения.

**Источник.** https://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html

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

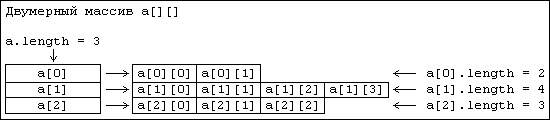
**Ответ**. Ссылке можно присвоить "пустое" значение null, не указывающее ни на какой адрес оперативной памяти:

ar = null;

После этого массив становится недоступным, если на него не было других ссылок.

**Источник.** Конспект Jse-18.

**Что собой представляет двумерный массив? Многомерный массив?**

Ответ. 

Для двумерного массива не существует такого понятия как его размеры. Можно определить размер массива только по первому индексу. Причина кроется в организации массива. Он представляет собой массив ссылок на массивы, каждый из которых содержит реальные данные. И эти массивы могут иметь разную длину!

Рассмотрим вот этот двумерный массив. Его длина по первому индексу равна 3. Ее можно получить через a.length. Элементами этого массива являются ссылки на массивы с данными. Длина каждого из этих массивов может быть получена так же через .length. Доступ к каждому из этих массивов осуществляется через его индекс – первый индекс в массиве a.

**Источник**. http://skipy.ru/technics/arrays.html

Перечислите ссылочные типы.

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

**Какие из элементов структуры являются обязательными?**

**Ответ.**

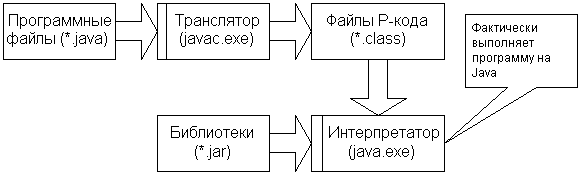


Рис. Жизненный цикл программы на Java

Исходная Java-программа – файл с расширением **java**.

Программа транслируется в байт-код компилятором **javac.exe**.

Транслированная в байт-код программа имеет расширение **class**.

Для запуска программы нужно вызвать интерпретатор **java.exe**, указав в параметрах вызова, какой байт-код следует выполнять.

При установленной переменной JAVA\_HOME библиотеки добавляются интерпретатором.

Иначе необходимо указать JVM, [где искать классы](http://skipy.ru/technics/likbez.html#lookup).

**Источник. Конспект Jse-18.**

**Для чего предназначены пакеты?**

**Ответ.**

 Пакет – область в пространстве имен классов.

Физически пакету соответствует директорий (имена должны совпадать).

Оператор **package** определяет пакет.

Внутри указанной области можно выделить подобласти, используя тот же оператор **package**.

**Источник.** **Конспект Jse18.**

**Как выглядит полное имя класса?**

**Ответ.**

 Полное имя класса имеет вид packagename.ClassName.

**Источник.** **Конспект Jse18.**

**Для чего предназначен оператор** import**?**

**Ответ.**

 Чтобы получить доступ к классу из другого пакета, перед именем

такого класса указывается имя пакета: by.bsu.eun.objects.CommonObject.

Чтобы избежать таких длинных имен при создании объектов классов, используется ключевое слово import.

В практическом программировании следует использовать индивидуальный import

класса, чтобы при анализе кода была возможность быстро определить месторасположение используемого класса.

**Источник. И.Н. Блинов, В.С. Романчик**

**Для чего предназначен статический импорт?**

**Ответ.**

При вызове статических методов и обращении к статическим константам

приходится использовать в качестве префикса имя класса, что утяжеляет код

и снижает скорость его восприятия.

package by.bsu.stat;

public class ImportDemo {

public static void main(String[ ] args) {

System.out.println(2 \* Math.PI \* 3);

System.out.println(Math.floor(Math.cos(Math.PI / 3)));

}

}

Статические константы и статические методы класса можно использовать

без указания принадлежности к классу, если применить статический импорт,

import static java.lang.Math.\*;

как это показано в следующем примере.

package by.bsu.stat;

import static java.lang.Math.\*;

public class ImportDemoLux {

public static void main(String[ ] args) {

System.out.println(2 \* PI \* 3);

System.out.println(floor(cos(PI / 3)));

}

}

**Источник. И.Н. Блинов, В.С. Романчик**

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

**Ответ.**

Создаем класс публичный класс Runner. Прописываем в нем публичный класс BusinessTrip.

Ide выдает: Класс BusinessTrip является общедоступным, его следует объявить в файле с именем BusinessTrip.java.

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

**Ответ.**

Проверяем. Создаем обычный класс в дефолтном пакете. Называем 1 по условию. После нажатия на Enter ide выдает: There is not a valid Java qualified name (Недействительное полное имя Java)

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

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

**Это единственный способ?**

**Ответ**.

Наиболее правильный

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

}

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

Ответ.

Вводим экспериментальный код:

public class Runner {  
 public static void main(String[] args) {  
 }  
}  
public class Runner{  
 public static void main(String[] args){  
 }  
}

Компилятор выводит: Ошибка: (5, 8) java: повторяющийся класс: Runner

**Можно ли определить метод main() с другими модификаторами?**

**Ответ.**

 Вводим экспериментальный код.

public class Runner {  
 protected/private static void main(String[] args) {  
 }  
}

Компилятор выводит: Ошибка: основной метод не найден в классе Runner, определите основной метод как:

public static void main (String [] args)

или класс приложения JavaFX должен расширять javafx.application.Application

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

**Если да, то приведите пример.**

**Ответ.**

Используем экспериментальный код:

public class Runner {  
 public static void main(String[] args) {  
 }  
 public static void main (String [] args) {  
 }  
}

Компилятор выводит: Ошибка: (4, 24) java: метод main (java.lang.String []) уже определен в классе Runner

**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.**

Используем экспериментальный код:

public class Runner {  
 public static void main(String[] args) {  
 }  
 public void main (String [] args) {  
 }  
}

Без модификатора static второй метод main невозможно вызвать.

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

Создайте bat-файл для запуска приложения на выполнение.

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.

Имеется корректный файл helloWorld.jar.

Создайте bat-файл для запуска приложения на выполнение.

**Какие есть виды комментариев?**

**Ответ**.

Документирование кода В языке Java используются блочные и однострочные комментарии /\* \*/ и //, аналогичные комментариям, применяемым в C++. Введен также новый вид комментария /\*\* \*/, который может содержать описание документа с помощью дескрипторов вида:

@author — задает сведения об авторе;

@version — задает номер версии класса;

@exception — задает имя класса исключения;

@param — описывает параметры, передаваемые методу;

@return — описывает тип, возвращаемый методом;

@deprecated — указывает, что метод устаревший и у него есть более совершенный аналог;

@since — определяет версию, с которой метод (член класса, класс) присутствует; @throws — описывает исключение, генерируемое методом;

@see — что следует посмотреть дополнительно.

**Источник** **И.Н. Блинов, В.С. Романчик**