# Java Fundamentals

1. ПояОбъясните, что имеется в виду, когда говорится: Java-язык программирования и Java-платформа.

Язык программирования - формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — ЭВМ) под её управлением.

Язык программирования Java - это универсальный, параллельный, строго типизированный, основанный на классах объектно-ориентированный язык. Он обычно скомпилирован в набор команд байт-кода и двоичный формат.

Компьютерная платфо́рма — в общем смысле, это любая существующая среда выполнения, в которой должен выполняться вновь разрабатываемый фрагмент программного обеспечения или объектный модуль с учётом накладываемых этой средой ограничений и предоставляемых возможностей. Термин платформа может применяться к разным уровням абстракции, включая определенную аппаратную архитектуру, операционную систему или библиотеку времени выполнения.

Программная платформа Java — ряд программных продуктов и спецификаций Sun Microsystems, ранее независимой компании, а ныне дочерней компании корпорации Oracle, которые совместно предоставляют систему для разработки прикладного программного обеспечения и встраивания её в любое кросс-платформенное программное обеспечение. Фактически это имя для [пакета программ](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC) компании Sun, которые позволяют разрабатывать и запускать программы, написанные на языке программирования Java.

1. Поясните как связаны имя java-файла и классы, которые в этом файле объявляются?

* В исходном файле может быть только один публичный класс (public class).
* Исходный файл может иметь несколько "непубличных" классов.
* Название публичного класса должно совпадать с именем исходного файла, который должен иметь расширение .java в конце. Например: имя класса public class Employee{}, то исходный файл должен быть Employee.java.

1. Расшифруйте JVM, JDK, JRE; расшифруйте, где “они находятся” и что собой представляют.

[Java Virtual Machine (сокращенно Java VM, JVM)](https://docs.oracle.com/javase/7/docs/technotes/guides/index.html#javavm) —виртуальная машина Java представляет собой абстрактную вычислительную машину, определенную спецификацией, которая позволяет запускать Java-программы. JVM исполняет байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java (javac). JVM может также использоваться для выполнения программ, написанных на других языках программирования. Программы, предназначенные для запуска на JVM, должны быть скомпилированы в стандартизированном переносимом двоичном формате, который обычно представляется в виде файлов «.class».

Платформа Java Standard Edition предоставляет две реализации виртуальной машины Java (VM): Java HotSpot Client VM (настроена на сокращение времени запуска и объема памяти) и Java HotSpot Server VM (предназначена для максимальной скорости выполнения программы, в компромисе со временем запуска и памятью.)

c:\Program Files\Java\jdk<version>\bin\java.exe

[Java SE Runtime Environment (JRE)](https://docs.oracle.com/javase/7/docs/technotes/guides/index.html#jre-jdk) - предоставляет библиотеки, виртуальную машину Java и другие компоненты, необходимые для запуска апплетов и приложений, написанных на Java.

c:\Program Files\Java\jre<version>\

[Java SE Development Kit (JDK)](https://docs.oracle.com/javase/7/docs/technotes/guides/index.html#jre-jdk) - включает JRE плюс инструменты разработки командной строки такие как компиляторы и отладчики, которые необходимы или полезны для разработки апплетов и приложений.

c:\Program Files\Java\jdk<version>\

1. Объясните как скомпилировать и запустить приложение из командной строки? Зачем в переменных среды окружения прописывать пути к установленному jdk?

При компиляции Java-кода необходимо следить за ходом нескольких вещей:

* Целевой файл, который Вы компилируете.
* Директория, где компилятор ищет .java файлы, импортируемые целевым файлом.
* Директория, где компилятор ищет .class файлы, импортируемые целевым файлом.
* Директория, куда компилятор помещает скомпилированный результат.

По умолчанию компилятор javac считает, что все они являются текущей рабочей директорией, что почти никогда не соответствует нуждам. Следовательно, при компиляции необходимо точно указать каждый из этих элементов.

C:\project> javac -d bin -sourcepath src src\com\elharo\gui\MainFrame.java

Параметр -d указывает конечную директорию (обычно называемую bin, build или classes). Если его не указать, то javac разместит скомпилированный .class файл в той же директории, что и соответствующий .java файл.

-sourcepath указывает директории, в которых компилятор должен искать иерархию исходных файлов.

При запуске программы необходимо указать:

* Сlasspath.
* Полное пакетно-классифицированное имя класса, содержащего метод main()

C:\project> java -classpath bin;C:\classes;E:\lib\junit.jar com.elharo.gui.MainFrame

Переменная PATH содержит список директорий, в которых операционная система пытается искать исполняемые файлы, если пользователь при запуске не указал явно путь к нужному исполняемому файлу. (%JAVA\_HOME%\bin).

JAVA\_HOME должна указывать на директорию, в которую установлена Java. Многие программы используют эту переменную, чтобы определить, где находится Java.

CLASSPATH содержит пути к папкам, в которых VM будет искать файлы \*.class, \*.jar и другие, которые ей нужны. Альтернатива - запускать приложение с ключом –cp(–classpath), после которого идет нужный этому приложению CLASSPATH. Переменные окружения в среде Windows состоят из путей, разделенных ";", в UNIX - ":". Текущая директория обозначается ".".

**>>> <http://www.interface.ru/home.asp?artId=7375> - Управление Java classpath (Windows**) **<<<**

1. Перечислить атрибуты доступа, объяснить их действие.

В Java существуют следующие модификаторы доступа:

* private: члены класса доступны только внутри класса;
* default (package-private) (модификатор, по-умолчанию): члены класса видны внутри пакета (если класс будет так объявлен он будет доступен только внутри пакета);
* protected: члены класса доступны внутри пакета и в наследниках;
* public: члены класс доступны всем;

Последовательность модификаторов по убыванию уровня закрытости: private, default ,protected, public.

Во время наследования возможно изменения модификаторов доступа в сторону большей видимости.

Модификатор доступа у конструкторов, методов и полей может быть любой. Класс может быть только либо public, либо default, причем в одном файле может находиться только один public класс. У блока может быть только один модификатор – default.

Модификатор доступа protected в Java не может быть применен к классу и интерфейсам. Поля в интерфейсе неявно являются public, static, final, а методы в интерфейсе по умолчанию являются public.

1. Что такое пакеты в java-программе, что представляют собой пакеты на диске? Каково соглашение по именованию пакетов? Как создать пакет?

**Java package** (пакет Java) — механизм, позволяющий организовать Java классы в пространстве имен. Обычно в пакеты объединяют классы одной и той же категории, либо предоставляющие сходную функциональность.

* Каждый пакет предоставляет уникальное пространство имен для своего содержимого.
* Допустимы вложенные пакеты.

На файловой системе такая иерархия выглядит в виде вложенных друг в друга директорий с исходниками. При определении классов в пакеты на жестком диске эти классы должны размещаться в подкаталогах, путь к которым соответветствует названию пакета. В проекте на Java всегда существует некоторая корневая директория относительно которой и строится дерево пакетов.

Существует общепринятая схема, где первая часть имени пакета должна состоять из перевёрнутого доменного имени разработчика класса. Так как доменные имена в интернете уникальны, соблюдение этого правила обеспечивает уникальность имён пакетов и предотвратит конфликты.

1. Объясните, какие классы, интерфейсы, перечисления необходимо импортировать в вашу программу, как это сделать. Влияет ли импорт пакета на импорт классов и др., лежащего в подпакетах? Какой пакет в Java импортируется по умолчанию?

Если нам надо использовать классы из других пакетов, то нам надо подключить эти пакеты и классы. Исключение составляют классы из пакета java.lang , которые подключаются в программу автоматически.

Можно импортировать пакеты и классы в проект с помощью директивы import, которая указывается после директивы package

1. Объяснить различия между терминами “объект” и “ссылка на объект”.

Объект – это экземпляр класса или массив.

Значения ссылок (чаще просто ссылки) – это указатели на эти объекты, и специальная ссылка null, которая означает отсутствиет объекта. Другими словами, ссылка — это переменная, содержащая адрес ячейки памяти, в которой хранится объект.

1. Какие примитивные типы в Java вы знаете, как создать переменную примитивных типов. Объяснить процедуру, по которой переменные примитивных типов передаются в методы как параметры.

В Java есть 8 примитивных типов, которые делят на 3 группы:

Целые числа - byte, short, char, int, long

Числа с плавающей точкой (иначе дробные) - float, double

Логический – boolean.

объявление переменной:

int x = 1;

При объявлении переменной, в следующей последовательности указываются:

<тип данных> <имя переменной> = <начальное значение переменной(инициализация)> ;

объявление переменной без инициализации:

int x;

объявление нескольких переменных одного типа:

int x,y;

Java всегда передает параметры в методы по значению. Это означает — «скопировать значение и передать копию.»

1. Каков размер примитивных типов, как размер примитивных типов зависит от разрядности платформы, что такое преобразование (приведение) типов и зачем оно необходимо? Какие примитивные типы не приводятся ни к какому другому?

Примитивные типы Java не являются объектами. К ним относятся:

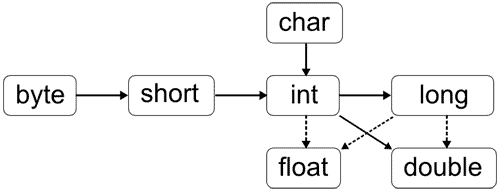
* boolean - булев тип, может иметь значения true или false;
* byte - 8-разрядное целое число;
* short - 16-разрядное целое число;
* int - 32-разрядное целое число;
* long - 64-разрядное целое число;
* char - 16-разрядное беззнаковое целое, представляющее собой символ UTF-16 (буквы и цифры);
* float - 32-разрядное число в формате IEEE 754 с плавающей точкой;
* double - 64-разрядное число в формате IEEE 754 с плавающей точкой;

О ширине (или количестве бит, отводимых для хранения значения) целочисленного типа нельзя думать как о количестве памяти, которую он занимает, а скорее, как о поведении, которое она определяет для переменных и выражений этого типа. Исполнительная среда Java вольна использовать любой размер, какой она хочет, тогда как типы ведут себя согласно их объявлению. Т.е примитивные типы платформонезависимы.

Поскольку код Java является статически типизированным во время компиляции, после объявления переменной ее нельзя объявить повторно или использовать для хранения значений другого типа, если этот тип не преобразуется в тип переменной. Тем не менее иногда может потребоваться скопировать значение в переменную или параметр метода другого типа. Например, может потребоваться передать целочисленную переменную в метод, параметр которого имеет тип double. Или может понадобиться присвоить переменную класса переменной типа интерфейса. Такого рода операции называются *преобразованиями типа*.

тип boolean нельзя привести ни к какому другому типу, отличному от boolean (как обычно- за исключением приведения к строке).

1. Объясните, что такое явное и неявное приведение типов, приведите примеры, когда такое преобразование имеет место.

Неявные преобразования: никакой специальный синтаксис не требуется, поскольку преобразование является строго типизированным и данные не будут потеряны. Примеры включают преобразования из меньших в большие целочисленные типы и преобразования из производных классов в базовые классы.

Явные преобразования (приведения): явные преобразования требуют оператора приведения. Приведение требуется, если в ходе преобразования данные могут быть утрачены или преобразование может завершиться сбоем по другим причинам. Типичными примерами являются числовое преобразование в тип с меньшей точностью или меньшим диапазоном и преобразование экземпляра базового класса в производный класс.

!!! Преобразование значение int в значение типа float. Если у значения int в двоичном представлении больше чем 23 значащих бита, то возможна потеря точности, так как у типа float под целую часть отведено 23 бита. Все младшие биты значения int, которые не поместятся в 23 бита мантиссы float, будут отброшены, поэтому хотя порядок числа сохраниться, но точность будет утеряна. То же самое справедливо для преобразования типа long в тип double.

1. Что такое литералы в Java-программе, классификация литералов, как записываются литералы различных видов и типов в Java-программе?

Литералы — это константы, которые записаны по правилам языка Java.

Целочисленные литералы:

111 — десятичное (десятеричное) число,

056 — число 46 в восьмеричной форме,

0 х АВ — 171 в шестнадцатеричной форме.

0b1010 – двоичная система

Кроме того, в JDK 7 можно вставлять символы подчёркивания для облегчения чтения больших числовых литералов: int x = 123\_456\_789;

Литералы для чисел с плавающей точкой:

18.01 ­ – стандартная форма записи

31.4е-1 , 0.314е1 – научная форма записи

0x14.3p2 ­– запись шестнадцатиричных литерал с плавающей точкой

Символьные литералы. Символы представляют собой 16-битовые значения в наборе символов Unicode. Символьные литералы должны заключаться в одинарные кавычки:

'а' 'b' 'с'

'\143' – вывести нужный символ, используя его восьмеричное значение

'\u0061' – вывести нужный символ, используя его шестнадцатиричное значение

Среди символьных литералов есть так называемые еsсаре-последовательности, которые позволяют произвести какую-либо операцию.

|  |  |
| --- | --- |
| Специальные символы | Действие |
| \а | Предупреждение (звонок) |
| \b | Возврат курсора на шаг |
| \f | Перевод страницы |
| \n | Следующая строка (перевод на новую строку) |
| \r | Возврат каретки |
| \t | Табуляция |
| \\ | Отображение обратной косой черты |
| \’ | Отображение одинарной кавычки |
| \” | Отображение двойной кавычки |
| \ааа | Символ восьмеричного значения (не более 377 — т.е. 255 в десятичной системе)­ |

Строковые литералы - они принадлежат объектам типа String и располагаются между двумя кавычками, например:

"Строка", "Старая строка \n Новая строка", "До табуляций \t После табуляции"

К булевым литералам относятся такие значения, как true (истина) и false (ложь).

Ссылочный литерал — null. Его можно использовать для присвоения значений объекту, т.е. сделать так, чтобы объект не был инициализирован.

1. Как осуществляется работа с типами при вычислении арифметических выражений в Java?

Помимо операций присваивания , преобразование типов происходит и в самих выражениях. В выражении можно свободно смешивать два или более типа данных , при условии их совместимости друг с другом . Когда в выражении смешиваются разные типы данных , они преобразуются в один и тот же тип по порядку следования операций . Преобразования типов выполняются по принятым правилам продвижения типов . Ниже приведен алгоритм , определяемый этими правилами для операций с двумя операндами .

ЕСЛИ один операнд имеет тип double , ТО и второй операнд продвигается к типу double .

ЕСЛИ один операнд имеет тип float , ТО и второй операнд продвигается к типу float .

ЕСЛИ один операнд имеет тип long , ТО и второй операнд продвигается к типу long .

ЕСЛИ один операнд имеет тип int , ТО и второй операнд продвигается к типу int .

ИНАЧЕ оба операнда продвигаются к типу int .

Относительно правил продвижения типов необходимо сделать ряд важных замечаний . Во - первых , не все типы могут смешиваться в выражении . Во - вторых , особого внимания требует последнее из приведенных выше правил . Оно гласит : если ни одно из предыдущих правил не применяется , то все операнды продвигаются к типу int . Следовательно , все значения типа byte и short продвигаются к типу int в целях вычисления выражения . Такое продвижение типов называется целочисленным . Это также означает , что результат выполнения всех арифметических операций будет иметь тип не ниже int . Следует иметь в виду , что правила продвижения типов применяются только к значениям , которыми оперируют при вычислении выражения . Так , если значение переменной типа byte продвигается к типу int внутри выражения , то вне выражения эта переменная по - прежнему относится к типу byte . Продвижение типов затрагивает только вычисление выражения . Но продвижение типов может иногда привести к неожиданным результатам . Если , например , в арифметической операции используются два значения типа byte , то про исходит следующее . Сначала операнды типа byte продвигаются к типу int . А затем выполняется операция , дающая результат типа int . Следовательно , результат выполнения операции , в которой участвуют два значения типа byte , будет иметь тип int .

Аналогичная ситуация возникает при выполнении операций с символьными операндами .

char ch1 = 'a', ch2 = 'b' ;

ch1 = (char) (ch1 + ch2);

Без приведения типов результат сложения операндов ch1 и ch2 будет иметь тип int , и поэтому его нельзя присвоить переменной типа char .

Продвижение типов происходит и при выполнении унарных операций , например с унарным минусом . Операнды унарных операций более мелкого типа , чем int ( byte , sbyte , short и ushort ), т . е . с более узким диапазоном представления чисел , продвигаются к типу int . То же самое происходит и с операндом типа char . Кроме того , если выполняется унарная операция отрицания значения типа int , то результат про двигается к типу long .

Приведение типов можно применять и к отдельным частям крупного выражения . Это позволяет точнее управлять преобразованиями типов при вычислении выражения .

1. Что такое классы-оболочки, для чего предназначены? Что значит “объект класса-оболочки - константный объект”?

Кроме базовых типов данных, в языке Java широко используются соответст­вующие классы-оболочки (wrapper-классы) из пакета java.lang: Boolean, Character, Integer, Byte, Short, Long, Float, Double. Объекты этих классов могут хранить те же значения, что и соответствующие им базовые типы.

Объекты этих классов представляют ссылки на участки динамической памяти, в которой хранятся их значения, и являются классами-оболочками для значений базовых типов. Классы, соответствующие числовым базовым типам, находятся в библиотеке java.lang, являются наследниками абстрактного класса Number и реализуют интерфейс Comparable, представляющий собой интерфейс для определения возможности сравнения объектов одного типа между собой. Объекты классов-оболочек по умолчанию получают значение null.

Когда речь идет о константном объекте имеется ввиду, что невозможно изменить его состояние.

1. Объяснить разницу между примитивными и ссылочными типами данных. Поясните существующие различия, при передаче параметров примитивных и ссылочных типов в методы. Как константные объекты ведут себя при передаче в метод.

Простые типы в Java предопределены заранее и проименованы зарезервированными словами (keywords). Примитивные типы хранят значение.

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

1. Что такое автоупаковка и автораспаковка?

Автоупаковка и распаковка это функция преобразования примитивных типов в объектные и наоборот.

1. Перечислить арифметические, логические, битовые операторы, определить случаи их употребления. Что такое приоритет оператора, как определить, в какой последовательности будут выполняться операции в выражении, если некоторые из них имеют одинаковых приоритет?
2. Укажите правила выполнения операций с плавающей точкой в Java (согласно стандарту IEEE754). Как определить, что результатом вычисления стала бесконечность или не число?
3. Что такое статический импорт, какие элементы можно импортировать при статическом импорте?
4. Объяснить работу операторов if, switch, while, do-while, for, for-each. Написать примеры.
5. Объяснить работу оператора instanceof. Что будет результатом работы оператора, если слева от него будет стоять ссылка, равная null?
6. Определение массиву. Как осуществляется индексация элементов массива, как необходимо обращаться к 1-му элементу массива?
7. Приведите способы объявления и инициализации одномерных и двумерных массивов примитивных и ссылочных типов. Указать разницу между массивами примитивных и ссылочных типов.
8. Что представляет собой двумерный массив в Java, что такое “рваный массив”? Как узнать количество строк и количество элементов в каждой строке для “рваного” массива?
9. Объясните ситуации, когда в Java-коде могут возникнуть следующие исключительные ситуации java.lang.ArrayIndexOutOfBoundsException и java.lang.ArrayStoreException
10. Определения понятиям “класс” и “объект”. Привести примеры объявления класса.
11. Как вы определяете, какие поля и методы необходимо определить в классе, привести примеры.
12. Развернутое объяснение трем концепциям ООП
13. Описать процедуру инициализации полей класса и полей экземпляра класса. Когда инициализируются поля класса, а когда - поля экземпляра класса? Какие значения присваиваются полям по умолчанию? Где еще в классе полям могут быть присвоены начальные значения?
14. Правила, которым должен следовать компонент Java-bean.
15. Определение перегрузке методов. Чем удобна перегрузка методов? Какие методы могут перегружаться и какими методами они могут быть перегруженными? Можно ли перегрузить методы в базовом и производном классах? Можно ли private метод базового класса перегрузить public методом производного? Можно ли перегрузить конструкторы, и можно ли при перегрузке конструкторов менять атрибуты доступа у конструкторов?
16. Что такое раннее и позднее связывание? Перегрузка - ранее или позднее? Объяснить правила, которым следует компилятор при разрешении перегрузки; в том числе, если методы перегружаются примитивными типами, между которыми возможно неявное приведение или ссылочными типами, состоящими в иерархической связи.
17. Что такое неявная ссылка this? В каких методах эта ссылка присутствует, а в каких - нет, и почему?
18. Что такое финальные поля и какие поля можно объявить со спецификатором final? Где можно инициализировать финальные поля?
19. Что такое статические поля, статические финальные поля и статические методы? К чему имеют доступ статические методы? Можно ли перегрузить и переопределить статические методы? Наследуются ли статические методы?
20. Что такое логические и статические блоки инициализации? Сколько их может быть в классе, в каком порядке они могут быть размещены и в каком порядке вызываются?
21. Что представляют собой методы с переменным числом параметров, как передаются параметры в такие методы и что представляет собой такой параметр в методе? Как осуществляется выбор подходящего метода, при использовании перегрузки для методов с переменным числом параметров?
22. Чем является класс Object? Перечислить методы класса, указать их назначение.
23. Что такое хэш-значение? Объяснить, почему 2 разных объекта могут сгенерировать одинаковые хэш-коды.
24. Для чего используется наследование в Java-программе? Привести пример наследования. Поля и методы, помеченные модификатором private, наследуются?
25. Как вызываются конструкторы при создании объекта производного класса? Что в конструкторе класса делает оператор super()? Возможно ли в одном конструкторе использовать операторы super() и this()?
26. Объяснить утверждения: “ссылка базового класса может ссылаться на объекты своих производных типов” и “объект производного класса может быть использован везде, где ожидается объект его базового типа”. Верно ли обратное и почему?
27. Переопределение методов. Зачем они нужны? Можно ли менять возвращаемый тип при переопределении методов? Можно ли менять атрибуты доступа при переопределении методов? Можно ли переопределить методы в рамках одного класса?
28. Правила вызова переопределенных методов. Можно ли статические методы переопределить нестатическими и наоборот?
29. Какие свойства имеют финальные методы и финальные классы? Зачем их использовать?
30. 19.Правила приведения типов при наследовании. Примеры явного и неявного преобразования ссылочных типов. Объясните, какие ошибки могут возникать при явном преобразовании ссылочных типов.
31. Что такое объект класса Class? Чем использование метода getClass() и последующего сравнения возвращенного значения с Type.class отличается от использования оператора instanceof?
32. Укажите правила переопределения методов equals(), hashCode(), toString().
33. Что такое абстрактные классы и методы? Зачем они нужны? Бывают ли случаи, когда абстрактные методы содержат тело? Можно ли в абстрактных классах определять конструкторы? Могут ли абстрактные классы содержать неабстрактные методы? Можно ли от абстрактных классов создавать объекты и почему?
34. Что такое интерфейсы? Как определить и реализовать интерфейс? Указать спецификаторы, которые приобретают методы и поля, определенные в интерфейсе. Можно ли описывать в интерфейсе конструкторы и создавать объекты? Можно ли создавать интерфейсные ссылки и если да, то на какие объекты они могут ссылаться?
35. Для чего служит интерфейс Clonable? Как правильно переопределить метод clone() класса Object, для того, чтобы объект мог создавать свои адекватные копии?
36. Для чего служат интерфейсы Comparable и Comparator? В каких случаях предпочтительнее использовать первый, а когда - второй? Как их реализовать и использовать?
37. Что такое перечисления в Java. Как объявить перечисление? Чем являются элементы перечислений? Кто и когда создает экземпляры перечислений? Могут ли перечисления реализовывать интерфейсы или содержать абстрактные методы? Могут ли перечисления содержать статические методы?
38. Можно ли самостоятельно создать экземпляр перечисления? А ссылку типа перечисления? Как сравнить, что в двух переменных содержится один и тот же элемент перечисления и почему именно так?
39. Что такое внутренние, вложенные и анонимные классы? Как определить классы такого вида? Как создать объекты классов такого вида? (!!!Создание объекта вложенного (nested) класса)
40. Перечислите возможности доступа к членам внешнего класса, которым наделены вложенные, внутренние и анонимные классы. Могут ли классы внутри класса быть базовыми, производными или реализующими интерфейсы? Можно ли анонимный класс создать от интерфейса?
41. Что такое параметризованные классы? Для чего они необходимы? Приведите пример параметризованного класса и пример создания объекта параметризованного класса. Объясните, ссылки какого типа могут ссылаться на объекты параметризованных классов? Можно ли создать объект, параметризовав его примитивным типом данных?
42. Какие ограничения на вызов методов существуют у параметризованных полей? Как эти ограничения снимает использование при параметризации ключевого слова extends?
43. Как параметризуются статические методы, как определяется конкретный тип параметризованного метода? Можно ли методы экземпляра класса параметризовать отдельно от параметра класса, и если “да”, то как тогда определять тип параметра?
44. Что такое wildcard? Приведите примеры его использования.
45. Для чего используется параметризация<? extends Type>, <? Super Type>?
46. Как создать объект класса String? Какие конструкторы класса String вы знаете? Что такое строковый литерал? Объясните, что значит “упрощенное создание объекта String”?
47. Можно ли изменить состояние объекта типа String? Что происходит при попытке изменения состояния объекта типа String? Можно ли наследоваться от класса String? Как вы думаете, почему строковое объекты immutable?
48. Объясните что такое кодировка? Какие кодировки вы знаете? Как создать строки в различной кодировке?
49. Что такое пул литералов? Как строки заносятся в пул литералов? Как занести строку в пул литералов и как получить ссылку на строку, хранящуюся в пуле литералов? Где хранится (в каком типе памяти) пул литералов в Java 1.6 и Java 1.7?
50. В чём отличие объектов классов StringBuilder и StringBuffer от объектов класса String? Какой из этих классов потокобезопасный? Как необходимо сравнивать на равенство объекты классов StringBuilder и StringBuffer и почему?
51. Что такое Unicode? Что такое code point?
52. Как кодируется символ Согласно кодировке utf-8, utf-16 и utf-32?
53. Какие методы класса String используются для работы с кодовыми точками? Как вы думаете, когда следует их использовать?
54. Что для программы является исключительной ситуацией? Какие способы обработки ошибок в программах вы знаете?
55. Что такое исключение для Java-программы? Что значит “программа выбросила исключение”? Опишите ситуации, когда исключения выбрасывается в виртуальной машиной (автоматически), и когда необходимо их выбрасывать вручную?
56. Приведите иерархию классов-исключений, делящую исключения на проверяемые и непроверяемые. В чём особенности проверяемых и непроверяемых исключений?
57. Объясните работу оператора try-catch-finally. Когда данный оператор следует использовать? Сколько блоков catch может соответствовать одному блоку try? Можно ли вкладывать блоки try друг в друга, можно ли вложить блок try в catch или finally? Как происходит обработка исключений, выброшенных внутренним блоком try, если среди его блоков catch нет подходящего? Что называется стеком операторов try? Как работает блок с ресурсами.
58. Укажите правила расположения блоков catch в зависимости от типов перехватываемых исключений. Может ли перехваченное исключение быть в сгенерировано снова, и, если да, то как и кто в этом случае будет обрабатывать повторно сгенерированное исключение? Может ли блок catch выбрасывать иные исключения, и если да, то опишите ситуацию, когда это может быть необходимо.
59. Когда происходит вызов блока finally? Существуют ли ситуации, когда блок finally не будет вызван? Может ли блок finally выбрасывать исключения? Может ли блок finally выполняться дважды?
60. Как выбросить исключение в ручную? Какие объекты (объекты каких классов) могут быть выброшены в качестве исключений? Можно ли выбросить два исключения одновременно? Объясните, как работают операторы throw и throws, в чём их отличия? Объясните правила реализации секции throws при переопределении метода и при описании конструкторов производного класса. Опишите поведения блока throws при работе с проверяемыми и непроверяемыми исключениями. Каков будет результат создания объекта, если конструктор при работе выбросит исключительную ситуацию?
61. Можно ли(и нужно) ли в Java выбрасывать исключения, входящие в пакет JDK? Как в Java создать собственные классы исключений?
62. 9.Перечислите известные Вам методы класс***а Throwable и их назначение?***
63. Что такое XML? Дайте определение тегу, элементу, атрибуту. Перечислите правила XML-документа.
64. Какие документы называются правильными, неправильными и правильно-форматированными? Что такое парсер? Какие документы XML может анализировать парсер согласно спецификации XML?
65. Что такое объявление xml? Приведите пример объявления xml и объясните значения его атрибутов.
66. Что такое пространство имен в xml? Какие правила именования для пространства имен применяются?
67. Что такое префикс пространства имен? Какова область действия пространства имен? Наследуют ли дочерние теги пространство имен родительского? Наследуют ли атрибуты пространство имен своего элемента?
68. В чём отличие явной декларации пространства имен от декларации по умолчанию?
69. Что такое XSD? В каком пространстве имен определены элементы xsd-схемы? Что такое targetNamespace?
70. Какой элемент является корневым элементом xsd-схемы? Приведите пример объявления xsd-схемы? Какую роль играют атрибуты elementFromDefault и attributeFromDefault?
71. Какие базовые типы xsd вы знаете? Как определить тип элемента xsd, содержащего другие элементы: приведите два возможных варианта объявления такого типа. Какие способы определения порядка следования элементов присутствует в xsd?
72. Как определяются новые элементы xsd-схемой? Сколько элементов <element> можно вложить непосредственно в элемент schema?
73. Как xsd позволяет наследоваться (ограничивать) встроенные типы? Что такое Facets, приведите примеры.
74. Как в xsd объявляются атрибуты для элементов? Приведите пример.
75. Как xml-документ связать с xsd-схемой? Какие классы (и из каких пакетов) позволяют провалидировать xml-файл на соответствие xsd-схеме?
76. Что такое DTD? Как DTD можно подключать к xml-файлу?
77. Что такое JDBC? Перечислите основные классы и интерфейсы, входящие в состав JDBC. Укажите их назначение. Какие ещё технологии Java, работающие с БД, вы знаете.
78. Опишите алгоритм получения соединения с БД, выполнения запроса и обработки результатов. Как загрузить драйвер БД и что он собой представляет? Какие типы драйверов баз данных в JDBC вы знаете? Нужно ли регистрировать драйвер БД и, если да, то как это сделать?
79. В чём отличие объектов JDBC типа Statement PreparedStatement? Защищены ли эти объекты от sql-injection? Можно ли работать с несколькими объектами statement или preparedstatement, полученными от от общего объекта-Connection одновременно и может ли такое использование быть небезопасным? Для чего используются объекты типа CallableStatement? Как выполняется вызов хранимых процедур из Java-программы? Что называется batch-командой? Как выполнить batch-команду?
80. Для чего JDBC использует объекты типа ResultSet? Что означает прокручиваемый и непрокручиваемый, обновляемый и необновляемый ResultSet? Как можно получить получить такие различные типы объектов ResultSet? Можно ли через ResultSet-объект изменить значение в БД и, если да, то как это сделать?
81. Как узнать какие типы Java будут конвертированы при выборке sql-типы данных?
82. Дайте определение транзакции, commit и rollback. Как JDBC работает с транзакциями по умолчанию? Как отменить autocommit и как в этом случае следует работать с БД? Что такое точка сохранения и как ее создать? Как откатить транзакцию до точки сохранения или до предыдущего commit-а?
83. Что такое пул соединений с БД, для чего он необходим? Опишите основные принципы создания пула соединения к БД.
84. Что означает термин метаданные? Какую информацию представляют объекты классов DatabaseMetaData, ResultSetMetaData и для чего она может быть использована?
85. Что означает термин “уровень изоляции транзакции”? Какие уровни изолированности транзакций поддерживает JDBC? Как в JDBC задать уровень изолированности транзакций .