**1 Basics of software code development**

Этот список вопросов предназначен для самостоятельного контроля глубины изучаемого материала. Попробуйте выучить материал так, чтобы вы могли ответить (хоть пару слов😊) по каждому вопросу.

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

**Ответ :**

1.Java как язык программирования является высокоуровневым, статически-типизированным и объектно-ориентированным.

2.Java как платформа — это программное обеспечение, представляющее собой рабочую среду для работы программ, написанных на Java (и не только). Она состоит из Java API и Java виртуальной машины (JVM).

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

**Ответ:**

1.Имя java-файла должно быть таким же как имя класса верхнего уровнясодержащегося в этом файле.

**3. Расшифруйте аббревиатуры JVM, JDK и JRE; покажите, где “они находятся” и что собой представляют.**

**Ответ:**

1.JVM (Java Virtual Machine) - это виртуальная машина Java

2.JRE (Java Runtime Environment) - это минимальная реализация виртуальной машины, необходимая для исполнения Java приложений, без компилятора и других средств разработки.

3.JDK (Java Development Kit) - это комплект разработчика приложений на языке Java, включающий в себя компилятор, стандартные библиотеки классов Java.

4.Находятся в корневом каталоге.Проедстовляет из себя набор библиотек.

**4. Объясните, как скомпилировать и запустить приложение из командной строки, а также зачем в переменных среды окружения прописывать пути к установленному jdk.**

**Ответ:**

1. Скомпилировать можно командной : javac appName.java в папке содержащей данный файл.

2. Запустить : командой java appName.

3. Прописывать путь нужно для того чтобы иметь возможность запускать программы через командную строку.

**5. Перечислите атрибуты доступа, объясните их действие.**

**Ответ:**

1.Public - публичный, общедоступный класс или член класса. Поля и методы, объявленные с модификатором public, видны другим классам из текущего пакета и из внешних пакетов.

2.Protected - такой класс или член класса доступен из любого места в текущем классе или пакете или в производных классах, даже если они находятся в других пакетах

3.Private - закрытый класс или член класса, противоположность модификатору public. Закрытый класс или член класса доступен только из кода в том же классе.

4.Default (package visible) - Отсутствие модификатора у поля или метода класса предполагает применение к нему модификатора по умолчанию. Такие поля или методы видны всем классам в текущем пакете.

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

**Ответ:**

1.Пакеты в java-программе разграничивают область видимости для классов.

2.Пакеты на диске представляют из себя папки.

3.Пакеты именуются в lowerСase чтобы не создавать конфликт с названиями классов.

4.Пакеты должны именоваться в трёх уровнях:

1.Домен

2.Имя компании

3.Название проекта

Пример : by.comName.myProject.myPackage

**7. 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?**

**Ответ:**

1.Классы, интерфейсы, перечисления необходимо импортировать в вашу программу те которые используются в используются в данной программе.

2. Импортировать можно с помощью команды import.

3. Импортированный пакет подтягивает все вложенные в него подпакеты и методы.

4.По умолчанию импортируется java.lang

**8. Объясните различия между терминами “объект” и “ссылка на объект”.**

**Ответ:**

1.Объект это - физическая сущность имеющая свою ячейку памяти в heap(это отдел динамической памяти Java).

Создание : Object object = new Object (); - это объект.

2. Ссылка на объект это - это переменная хранящаяся в стеке памяти(отдел памяти JVM), содержащая адрес ячейки памяти, в которой хранится объект.

Создание: Object object; - это ссылка на объект.

**9. Какие примитивные типы Java вы знаете, как создать переменные примитивных типов? Объясните процедуру, по которой переменные примитивных типов передаются в методы как параметры.**

**Ответ:**

1. Примитивные типы Java это - byte,short,int,long,float,double,boolean,char.

2. Создание переменной – ссылка на переменную : int example; - это объявление переменной,

int example = 1; - это инициализация переменной.

3.Примитивные типы данных передаются в метод по значению.Метод создаёт копию передаваемого ему значения и не как не изменяет исходное значение.

**10. 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 типу.**

**Ответ:**

1.Размер примитивных типов:

boolean = 32 бит

byte = 8 бит

short = 16 бит

int = 32 бит

long = 64 бит

char = 6 бит

float = 32 бит

double = 64 бит

2.Зависимость от разрядности платформы : Размер примитивных типов не зависит от разрядности платформы.

3.Что такое преобразование (приведение) типов и зачем оно необходимо : это преобразование одного типа к другому int к byte например.

4.Неприводимые примитивные типы: boolean.

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

**Ответ:**

1.Явное приведение типов : с указанием приводимого типа.

long a = 1;

int b = ( int ) a;

2.Неявное приведение типов : происходит автоматически.

int a = 12345;

long b = a;

3.Преобразование имеет место : например при присвоении целого числа к числу с плавающей точкой и наоборот.

**12. Что такое литералы в Java-программе, какую классификацию литералов вы знаете, как записываются литералы различных видов и типов в Java-программе?**

**Ответ:**

1.Литералы : это явно заданные значения в коде программы.

2.Классифицируются по типам:

1.Числовые : 1,2,3….

2.Стоковые : «Hi»

3.Символьные «\n»

4.Логические true or false

**13. Как осуществляется работа с типами при вычислении арифметически выражений в Java?**

**Ответ:**

1.Как осуществляется работа с типами при вычислении арифметически выражений в Java : будет автоматически выполнятся расширяющее преобразование типов

byte → short → int → long →float → double

глядя на операнды результат будет к преобразован к большему автоматически .

Для сужения диапазона значения необходимо явное преобразование типа (type)value

int number = 1;

byte value = (byte)number;

**14. Что такое классы-оболочки, для чего они предназначены? Объясните, что значит: объект класса оболочки – константный объект.**

**Ответ:**

1.Wrapper предназначен для представления примитивного типа в объект.

2. Объект класса оболочки – константный объект: означает что объект является иммутабельным (неизменяемый, immutable) .

**15. Объясните разницу между примитивными и ссылочными типами данных. Поясните существующие различия, при передаче параметров примитивных и ссылочных типов в методы. Объясните, как константные объекты ведут себя при передаче в метод.**

**Ответ:**

1. Разница между примитивными и ссылочными типами данных: примитивные типы хранят в себе значение а ссылочные ссылку на объект.

2. Передача в метод осуществляется по значению для примитивного типа само его значение

для ссылочных это ссылка на объект в памяти который может быть изменён в случае наличия прав доступа.

3. Константным объектам не может быть присвоено новое значение но может быть изменено его состояние в случае наличия прав доступа.

**16. Поясните, что такое автоупаковка и автораспаковка.**

**Ответ:**

1. Автоупаковка : преобразование примитивных типов в объект.

2. Автораспаковка : преобразование объекта в примитивный тип данных.

**17. Перечислите известные вам арифметические, логические и битовые операторы, определите случаи их употребления. Что такое приоритет оператора, как определить, в какой последовательности будут выполняться операции в выражении, если несколько из них имеют одинаковый приоритет.**

**Ответ:**

1. Арифметические :

+ - сложение.

- - вычитание.

\* - умножение.

/ - деление.

% - остаток от деления.

++ - инкремент ( увеличение на 1).

-- - декремент ( уменьшение на 1).

2.Сравнение :

== - сравнение (для примитивных типов сравнивает значение для объектов сравнивает ссылки на объекты. НЕ САМИ ОБЪЕКТЫ!Для сравнения объектов используется equals

Пример :

Примитив - a == b;

Объект — a.equals( b );

!= - не равно ( true если не равно)

> - больше.

< - меньше.

>= - больше или равно.

<= - меньше или равно.

3. Логические:

! - логическое НЕ.

& - логическое И . True если два операнда true. Проверяет все оперенты.

&& - логическое И сокращённое . То же самое что & но вернёт false если первое (левый) операнд false и второй операнд проверять не будет.

| - логическое ИЛИ. True если есть хоть один true.

|| - логическое ИЛИ сокращённое. То же самое что | но вернёт true если первое (левый) операнд true и второй операнд проверять не будет.

^ - Логическое исключающее ИЛИ (XOR).Возвращает true, если операнды разные и false если оба одинаковые.

4. Побитовые :

~ - Побитовый унарный оператор NOT.

& - Побитовый И. Даёт 1 если оба бита равны 1, в остальных случаях даёт 0.

| - Побитовый ИЛИ. Даёт 1, если любой из бит или оба равны 1. 0 - если оба равны 0.

^ - Побитовый исключающее ИЛИ. Даёт 1 если один из бит равен 1. даёт 0, если оба бита равны 1 или 0.

>> - Сдвиг вправо. Значение левого операнда сдвигается вправо на количество бит правого операнда, старшие биты заполняются битом знака (0 - плюс, 1 — минус).

>>> - Сдвиг вправо с заполнением нулями. Аналогично сдвигу вправо, но старший бит всегда заполняется нулями.

<< - Сдвиг влево. Значение левого операнда сдвигается влево на количество бит правого операнда (пример 0001 << 2 = 0100 ).

**18. Укажите правила выполнения операций с плавающий точкой в Java (согласно стандарту IEEE754). Как определить, что результатом вычисления стала бесконечность или нечисло?**

**Ответ:**

NaN (англ. Not-a-Number) — одно из особых состояний числа с плавающей запятой. Используется во многих

математических библиотеках и математических сопроцессорах. Данное состояние может возникнуть в различных случаях,

например, когда предыдущая математическая операция завершилась с неопределённым результатом, или если в ячейку

памяти попало не удовлетворяющее условиям число.

В соответствии с IEEE 754, такое состояние задаётся через установку показателя степени в зарезервированное

значение 11…11, а мантиссы — во что угодно, кроме 0 (зарезервированное значение для машинной бесконечности).

Знак и мантисса могут нести какую-то дополнительную информацию: многие библиотеки «отрицательный»

NaN выводят как -NaN.

К операциям, приводящим к появлению NaN в качестве ответа, относятся:

все математические операции, содержащие NaN в качестве одного из операндов;

деление нуля на нуль;

деление бесконечности на бесконечность;

умножение нуля на бесконечность;

сложение бесконечности с бесконечностью противоположного знака;

вычисление квадратного корня отрицательного числа[1];

логарифмирование отрицательного числа.

В некоторых языках программирования есть «тихий» и «сигнальный» NaN: первый, попав в любую операцию, возвращает

NaN, второй — вызывает аварийную ситуацию. Обычно «тихий» или «сигнальный» определяется старшим битом мантиссы.

NaN не равен ни одному другому значению (даже самому себе[2]); соответственно, самый простой метод проверки

результата на NaN — это сравнение полученной величины с самой собой.

В Java тип double имеет специальные значения для понятий «плюс бесконечность» и «минус бесконечность».

Положительное число, разделенное на 0.0, дает «плюс бесконечность», а отрицательное – «минус бесконечность».

Этим понятиям соответствуют специальные константы типа Double:

public static final double POSITIVE\_INFINITY = 1.0 / 0.0; плюс бесконечность

public static final double NEGATIVE\_INFINITY = -1.0 / 0.0; минус бесконечность

Строку конвертируем в число, а в ней есть буквы. Ответ — NaN

Бесконечность минус бесконечность. Ответ — NaN

Многие другие ситуации, где в ответе ждут число, а получается неизвестно что.

Любая операция, где есть NaN, дает в результате NaN.

Действия с бесконечностью:

n ÷ ±Infinity 0

±Infinity × ±Infinity ±Infinity

±(не ноль) ÷ ±Infinity

Infinity + Infinity Infinity

±0 ÷ ±0 NaN

Infinity - Infinity NaN

±Infinity ÷ ±Infinity NaN

±Infinity × 0 NaN

double inf = Double.POSITIVE\_INFINITY;

System.out.println(inf + 5);

System.out.println(inf - inf); // same as Double.NaN

System.out.println(inf \* -1);

**19. Что такое статический импорт, какие элементы можно импортировать при статическом импорте.**

**Ответ:**

1. Статический импорт позволяет вызывать методы класса на прямую.

import static java.lang.System.out;  
  
 public class Main {  
 public static void main(String[] args) {  
 out.println("Hi");  
 }  
}

**20. Объясните работу операторов if, switch, while, do-while, for, for-each. Напишите корректные примеры работы этих операторов.**

**Ответ:**

1.Оператор if : if (ЕСЛИ условия выполняются то переходим к выполнению блока ){

out.println (“Hi“);

};

2.Оператор switch : switch(Выражение для сравнения) {

case Совпадение1:

команда;

break;

default:

оператор;

break;

}

3.Оператор while : while ( ПОКА условия положительное блок срабатывает ){

out.println (“Hi“);

};

4.Оператор do-while : do-while ( Тоже что и while с отличием что оно сработает один раз даже если условие ложное){

out.println (“Hi“);

}

5.Оператор for : for (int i = 0 (инициализация ); i < 10 (условие) ; i++ (шаг)){

out.println (“Hi“ + i);

};

6.Оператор for-each : ДЛЯ КАЖДОГО - разновидность цикла for для массивов и коллекций

for-each(type var : array){

out.println (var);

};

**21. Объясните работу оператора instanceof. Что будет результатом работы оператора, если слева от него будет стоять ссылка, равная null?**

**Ответ:**

(object) instanceof (type);

1.*instanceof —  это бинарный оператор, используемый для проверки того, принадлежит ли объект заданному типу. Результат операции либо истина , либо ложь* . Он также известен как оператор сравнения типов, поскольку сравнивает экземпляр с типом.

2.Что будет результатом работы оператора, если слева от него будет стоять ссылка, равная null? - результат будет false!

**2\_Algorithmization**

Этот список вопросов предназначен для самостоятельного контроля глубины изучаемого материала. Попробуйте выучить материал так, чтобы вы могли ответить (хоть пару слов😊) по каждому вопросу.

**1. Дайте определение массиву. Как осуществляется индексация элементов массива. Как необходимо обращаться к i-му элементу массива?**

**Ответ:**

1.Массив - это структура данных, в которой хранятся элементы одного типа.

2.Индексация осуществляется от 0 до n.

3.Обращаться к i-му элементу массива следует по индексу элемента .

Пример:

int [] array = new int [3] — создание массива.

array[0] = 3; - добавление элемента в массив.(инициализация массива)

array[1] = 2;

array[2] = 1;

System.out.println(array[1]); - вывод элемента под индексом 1.

Вывод: 2

Или через цикл for :

for(int i = 0; i < array.length; i++){

System.out.print(i + «,»);

}

Вывод:

1,2,3,

**2. Приведите способы объявления и инициализации одномерных и двумерных массивов примитивных и ссылочных типов. Укажите разницу, между массивами примитивных и ссылочных типов.**

**Ответ:**

int [] array ;

Cat [] cat; - объявления одномерных массивов примитивных и ссылочных типов.

Int [] array = new int[]{1,2,3};

Cat [] cat = new Cat [2] ; - инициализация одномерных массивов примитивных и ссылочных типов.

int [][] array;

Cat[][] cat; - объявления двумерных массивов примитивных и ссылочных типов.

array = new int [][] {{1,2,3,4,5},{6,7,8,9,0}};

cat = new Cat[2][2]; - инициализация двумерных массивов примитивных и ссылочных типов.

Разница между массивами ссылочного и примитивного типа в том что массив примитивного типа содержит значение а ссылочный содержит ссылки на объекты.

**3. Объясните, что значит клонирование массива, как в Java можно клонировать массив, в чем состоит разница в клонировании массивов примитивных и ссылочных типов.**

**Ответ:**

1. Объясните, что значит клонирование массива : создание точной копии.

2. Как в Java можно клонировать массив : так как все массивы являются объектами то можно при помощи метода clone().

3. В чем состоит разница в клонировании массивов примитивных и ссылочных типов : в примитивных типах при клонировании создаётся точно такой массив и заполняется теми же значениями и при изменении клонированного массива это никак

не отразится на исходном массиве.

int [] arraySrc = new int[]{1,2,3};  
int [] arrayClone = arraySrc.clone();  
System.*out*.println(Arrays.*toString*(arraySrc));  
  
arrayClone[0] = 9; // Изменяем значение index[0];  
  
System.*out*.println(Arrays.*toString*(arraySrc));  
  
System.*out*.println(Arrays.*toString*(arrayClone));

В ссылочных массивах тоже создаётся точная копия исходного массива, но ссылки которые его заполняют указывают на те же объекты что и в исходном массиве. Из этого следует что изменение состояния объекта отразится на всех массивах.

Dog [] dogs = new Dog[2];  
for (int i = 0; i < 2; i++) {  
 Dog dog = new Dog("Sam");  
 dogs[i] = dog;  
}  
  
System.*out*.println("Source array before cloneable: " + Arrays.*toString*(dogs));  
  
Dog [] dogsClone = dogs.clone();  
  
dogsClone[0].setName("Tom"); // Изменяем имя объекта index[0];  
  
System.*out*.println("Source array after cloneable: " + Arrays.*toString*(dogs));  
  
System.*out*.println("Cloned array: " + Arrays.*toString*(dogsClone));

**4. Объясните, что представляет собой двумерных массив в Java, что такое “рваный массив”. Как узнать количество строк и количество элементов в каждой строке для “рваного” массива?**

**Ответ:**

1. Двумерных массив в Java - это массив массивов. То есть массив содержит в себе ссылки на вложенные в него массивы.

2. “Рваный массив” — это массив содержащий в себе массивы разных размеров

1 2 3 4

2 2

1 2 3 4 4

3. Как узнать количество строк и количество элементов в каждой строке для “рваного” массива : Длину каждой строки в рваном массиве можно узнать с помощью переменной length. Для этого вызываем строку массива, длину которой мы хотим узнать:

array [2].length; //для строки массива под индексом 2.

**5. Объясните ситуации, когда в java-коде могут возникнуть следующие исключительные ситуации java.lang.ArrayIndexOutOfBoundsException и java.lang.ArrayStoreException.**

**Ответ:**

1. java.lang.ArrayIndexOutOfBoundsException — выход за границы массива , возникает

при попытке обращения к несуществующему элементу массива.

2. java.lang.ArrayStoreException — ошибка хранилища массива , возникает при попытки добавить в массив элемента не соответствующего типа.

**6. Объясните, зачем при кодировании разделять решаемую задачу на методы. Поясните, как вы понимаете выражение: “Один метод не должен выполнять две задачи”.**

**Ответ:**

1.Для возможности переиспользования методов в других программах.

2. Один метод не должен выполнять две задачи : потому что это приведёт к невозможности переиспользования методов в других программах.

**7. Объясните, как в Java передаются параметры в методы, в чем особенность передачи в метод значения примитивного типа, а в чем ссылочного.**

**Ответ:**

1. Передача в метод осуществляется по значению для примитивного типа само его значение

для ссылочных это ссылка на объект в памяти который может быть изменён в случае наличия прав доступа.

**8. Объясните, как в метод передать массив. И как массив вернуть из метода. Можно ли в методе изменить размер переданного массива.**

**Ответ:**

1. Как в метод передать массив : в качестве параметров метода.

2. Как массив вернуть из метода : указать массив в типе возвращаемого значения.

3. В методе его можно изменить путём создания нового массива и копирования в него необходимую информацию.Сам массив неизменяемый.

**9. Поясните, что означает выражение ‘вернуть значение из метода’. Как можно вернуть значение из метода. Есть ли разница при возврате значений примитивного и ссылочного типов.**

**Ответ:**

1.Вернуть значение из метода : это способность метода передавать значение того типа который объявлен в методе.

2. Как вернуть значение из метода : указать тип возвращаемого значения в методе и воспользоваться оператором return.

3. Есть ли разница при возврате значений примитивного и ссылочного типов :

1. При возврате значений примитивного типа : нет.

2. При возврате значений ссылочного типа : нет.

**10. Перечислите известные вам алгоритмы сортировки значений, приведите код, реализующий это алгоритмы.**

**Ответ:**

1.Сортировка пузырьком , выбором, вставками, шелла, быстрая , слиянием : как написать помню пока только BubbleSort )

public int [] bubbleSort(int[] array) {  
 boolean isSorted = false;  
 while (!isSorted) {  
 isSorted = true;  
  
 for (int i = 1; i < array.length; i++) {  
 if (array[i] < array[i - 1]) {  
 int temp = array[i];  
 array[i] = array[i - 1];  
 array[i - 1] = temp;  
 isSorted = false;  
  
  
 }  
 }  
 }  
 return array;  
}

**3 Strings and basics of text processing**

Этот список вопросов предназначен для самостоятельного контроля глубины изучаемого материала. Попробуйте выучить материал так, чтобы вы могли ответить (хоть пару слов😊) по каждому вопросу.

**Strings**

**1. Как создать объект класса String, какие конструкторы класса String вы знаете? Что такое строковый литерал? Объясните, что значит “упрощенное создание объекта String”?**

**Ответ:**

1. Как создать объект класса String : String s = new String();

2. Какие конструкторы класса String вы знаете :

1.String (); - создается объект с пустой строкой .

2.String (byte [] bytes) - создается объект из массива байтов.

3.String (char [] value) - создается объект из массива char.

4.String (StringBuffer str) — приведенная копия объекта класса StringBuffer.

5.String (StringBuilder str) — приведенная копия объекта класса StringBuilder.

3. Что такое строковый литерал : это набор символов заключенный в кавычки.

4. Объясните, что значит “упрощенное создание объекта String” : это создание без оператора new . String s = «abc»;

**2. 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 строковые объекты immutable?**

**Ответ:**

1. Можно ли изменить состояние объекта типа String : нет, так как он immutable (то-есть неизменяемый )

2. Что происходит при попытке изменения состояния объекта типа String : создаётся новый объект String.

3. Что такое строковый литерал : это набор символов, заключённых в двойные кавычки.

4. Можно ли наследоваться от класса String : нет, класс String объявлен как final класс.

5. Как вы думаете, почему строковые объекты immutable : для безопасности,например в многопоточной среде могут использовать несколько потоков один экземпляр строки

без опасения её изменить.

**3. Объясните, что такое кодировка? Какие кодировки вы знаете? Как создать строки в различной кодировке?**

**Ответ:**

1. Объясните, что такое кодировка : это стандарт кодирования символов.

2. Какие кодировки вы знаете : ASCII ,Unicode ,КОИ8

3. Как создать строки в различной кодировке : Никак, java имеет зафиксированную кодировку - unicode. все остальное только при чтении/перезаписи байтов String str=new String(s.getBytes(),"UTF-8");

**4. 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?**

**Ответ:**

1. Что такое пул литералов : это множество строк в Java Heap Memory.

2. Как строки заносятся в пул литералов : при объявлении.

3. Как занести строку в пул литералов и как получить ссылку на строку, хранящуюся в пуле литералов : Занести строку в пул можно с помощью метода String.intern(); который перед созданием String проверит есть ли такой объект в пуле и если есть вернёт его если нет создаст новый.

4. Где хранится(в каком типе памяти) пул литералов в Java 1.6 и Java 1.7 : В 1.6 - находился в PermGen, в 1.7 в Heap.

**5. В чем 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?**

**Ответ:**

1. В чем отличие объектов классов StringBuilder и StringBuffer от объектов класса String : в том что это изменяемые объектами в отличии от String.

2. Какой из этих классов потокобезопасный : StringBuffer.

3. Как необходимо сравнивать на равенство объекты классов StringBuilder и StringBuffer и почему : через equals () предварительно приведя их к строке методом toString(); так как в классе String метод equals () переопределен.

**6. Что такое Unicode?**

**Ответ:**

1. Что такое Unicode : это стандарт кодирования символов.

7. Какие методы класса String используются для работы с кодовыми точками? Как вы думаете, когда следует их использовать?

Ответ:

1. Какие методы класса String используются для работы с кодовыми точками? Как вы думаете : codePointAt; codePointBefore; codePointCount; indexOf.

2. Когда следует их использовать : для изменения кодировок.

**Regular Expressions**

**1. Расскажите, что представляет собой регулярное выражение? Что такое метасимволы регулярного выражения? Какие вы знаете классы символов регулярных выражений? Что такое квантификаторы? Какие логические операторы регулярных выражений вы знаете? Что значит “якорь” для регулярного выражения?**

**Ответ:**

1. Расскажите, что представляет собой регулярное выражение : это формальный язык поиска и осуществления манипуляций с [подстроками](https://ru.wikipedia.org/wiki/Подстрока) в тексте, основанный на использовании метасимволов.

2. Что такое метасимволы регулярного выражения : это символы определяющие производимую операцию.

3. Какие вы знаете классы символов регулярных выражений :

\\w буквенно-цифровой символ (включая \_)

\\W не буквенно-цифровой

\\d числовой символ (тоже, что [0-9])

\\D нечисловой

\\s любой пробел (такой же как [\t\n\r\f])

\\S не пробел

\\h горизонтальный разделитель. Табуляция, пробел и все символы

в Unicode категории «разделители» (space separator Unicode category)

\\H не горизонтальный разделитель

\\v вертикальные разделители. новая строка и все символы

«разделители строк» в Unicode

\\V не вертикальный разделитель

4. Что такое квантификаторы : оператор определяющий сколько раз может встречаться предшествующее выражение.

5. Какие логические операторы регулярных выражений вы знаете : Ленивые \*? ?? +? {n,]? Жадные \* ? + {n,} Ревнивые \*+ ?+ ++ {n,}+

6. Что значит “якорь” для регулярного выражения : они позволяют нам привязать шаблон регулярного выражения к конкретной точке в строк.

**2.** **Какие java-классы работают с регулярными выражениями? В каком пакете они расположены? Приведите пример анализа текста с помощью регулярного выражения и поясните код примера.**

**Ответ:**

1. Какие java-классы работают с регулярными выражениями : Pattern ,Mather

2. В каком пакете они расположены : java.util.regex

3. Приведите пример анализа текста с помощью регулярного выражения и поясните код примера :

*/\*\**  
 *\* Пример текста для анализа.*  
 *\*/*  
String example =  
 "Какие java-классы работают с регулярными выражениями?" +  
 " В каком пакете они расположены? " +  
 "Приведите пример анализа текста с помощью" +  
 " регулярного выражения и поясните код примера.";  
 */\*\**  
 *\* Объявление массива строк для хранения предложений.*  
 *\*/*  
String [] sentence;  
 */\*\**  
 *\* Разделение строки example на предложения при помощи*  
 *\* метода split() класса String и регулярного*  
 *\* выражения [\.\!\?] которое в методе split() разделяет*  
 *\* текст когда встречаются эти знаки ".!?".После заполняется*  
 *\* массив sentence строками (предложениями).*  
 *\*/*  
sentence = example.split("[\\.\\!\\?]");  
 */\*\**  
 *\* Создается Pattern (шаблон).В его метод compile() передается*  
 *\* строка для дальнейшей обработки в классе Matcher.*  
 *\*/*  
Pattern pattern = Pattern.*compile*("пример");  
 */\*\**  
 *\* Цикл for проходит по массиву sentence и передает*  
 *\* каждый объект (предложение) в matcher, далее в*  
 *\* условии оператора if объявляется метод matcher.find().*  
 *\* Если метод find() найдет соответствие в тексте*  
 *\* то вернёт true и сработает блок if, если нет*  
 *\* то сработает блок else.*  
 *\*/*  
for (int i = 0; i < sentence.length; i++) {  
 Matcher matcher = pattern.matcher(sentence[i]);  
 if (matcher.find()) {  
 System.*out*.println("В предложении № " + (i + 1) +  
 " присутствует стово (" + pattern + "): " + example);  
 } else {  
 System.*out*.println("В предложении № " + (i + 1) +  
 " слово (" + pattern + ") отсутствует!");  
 }  
 }

**3. Что такое группы в регулярных выражениях? Как нумеруются группы? Что представляет собой группа номер 0(ноль)? Приведите пример с использованием групп регулярного выражения.**

**Ответ:**

1. Что такое группы в регулярных выражениях : это механизм для поиска и замены текста.

2. Как нумеруются группы : от одного.

3. Что представляет собой группа номер 0(ноль) : это общее вырождение,включающее

в себя все возможные вложенные группы.

4. Приведите пример с использованием групп регулярного выражения :

public static void main(String[] args) {

Main main = new Main();  
 System.*out*.println(main.emailValidationForDomainRuBy("example@mail.ru")); // true  
 System.*out*.println(main.emailValidationForDomainRuBy("example@mail.by")); // true  
 System.*out*.println(main.emailValidationForDomainRuBy("example@gmail.com"));// false  
 System.*out*.println(main.emailValidationForDomainRuBy("@mail.ru")); // false  
  
}  
private boolean emailValidationForDomainRuBy (String mail){  
  
 String regex = "(\\w+@\\w+[.](ru|by))";  
  
 Pattern pattern = Pattern.*compile*(regex);  
  
 Matcher matcher = pattern.matcher(mail);  
  
 if(matcher.matches()){

*/\*\**  
 *\* Пример использования групп в regex.В примере matcher.group(1)*  
 *\* будет выведена группа №1 (ru|by),группа matcher.group(0)*   
 *\* соответствует всему выражению.*  
 *\*/*  
 System.*out*.print(matcher.group(1) + "\s"); // Выводит только домен.

System.*out*.print(matcher.group(0) + "\s"); // Выводит все выражение.

return true;  
 }  
  
 return false;  
}

**4. Programming with classes**

Этот список вопросов предназначен для самостоятельного контроля глубины изучаемого материала. Попробуйте выучить материал так, чтобы вы могли ответить (хоть пару слов😊) по каждому вопросу.

**1. Опишите процедуру инициализации полей класса и полей экземпляра класса. Когда инициализируются поля класса, а когда – поля экземпляров класса. Какие значения присваиваются полям по умолчанию? Где еще в классе полям могут быть присвоены начальные значения?**

**Ответ:**

1. Опишите процедуру инициализации полей класса и полей экземпляра класса : полей static проходит во время загрузки класса , экземпляра во время его объявления.

2. Когда инициализируются поля класса, а когда – поля экземпляров класса :

Поля класса : если оно static то во время загрузки класса,если нет то при создании экземпляра класса.

Экземпляр класса : в месте объявления или в конструкторе.

3. Какие значения присваиваются полям по умолчанию : примут значения для ссылочных null , для примитивов 0 для char « » boolean false .

4. Где ещё в классе полям могут быть присвоены начальные значения : в конструкторе или в блоке.

**2. Дайте определение перегрузке методов. Как вы думаете, чем удобна перегрузка методов? Укажите, какие методы могут перегружаться, и какими методами они могут быть перегружены? Можно ли перегрузить методы в базовом и производном классах? Можно ли private метод базового класса перегрузить public методов производного? Можно ли перегрузить конструкторы, и можно ли при перегрузке конструкторов менять атрибуты доступа у конструкторов?**

**Ответ:**

1. Дайте определение перегрузке методов : это создание методов с одинаковыми названиями но с разными параметрами.

2. Как вы думаете, чем удобна перегрузка методов : можно создать много методов с одним именем но с разными входными параметрами.

3. Укажите, какие методы могут перегружаться, и какими методами они могут быть перегружены : статические методы могут перегружаться нестатическими,и наоборот.

4. Можно ли перегрузить методы в базовом и производном классах : да.

5. Можно ли private метод базового класса перегрузить public методов производного : нет.

6. Можно ли перегрузить конструкторы, и можно ли при перегрузке конструкторов менять атрибуты доступа у конструкторов : да,да.

**3. Объясните, что такое раннее и позднее связывание? Перегрузка – это раннее или позднее связывание? Объясните 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.**

**Ответ:**

1. Объясните, что такое раннее и позднее связывание : это связывание ссылки и кода где она будет применятся на этапе компиляции,а позднее во время исполнения кода.

2. Перегрузка – это раннее или позднее связывание : это раннее связывание.

3. Объясните 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 : правило для перегрузки два метода из одного класса не могут иметь одинаковые сигнатуры(при этом сигнатура не включает в себя возвращаемый параметр). Выбирается наиболее подходящий метод. Расширяющее преобразование примитивов имеет приоритет над автоупоковкой/распоковкой. Сужающее преобразование примитивов автоматически не выполняется. Когда вызывается метод , количество фактических аргументов (и любых явных аргументов типа) и типы аргументов времени компиляции используются во время компиляции для определения сигнатуры метода, который будет вызван.

**4. Объясните, как вы понимаете, что такое неявная ссылка this? В каких методах эта ссылка присутствует, а в каких – нет, и почему?**

**Ответ:**

1. Объясните, как вы понимаете, что такое неявная ссылка this : это указатель на объект вызвавшему его методу.Кроме static методов.

2. В каких методах эта ссылка присутствует, а в каких – нет, и почему : в конструкторах одного типа, не присутствует в static методах.

**5. Что такое финальные поля, какие поля можно объявить со спецификатором final? Где можно инициализировать финальные поля?**

**Ответ:**

1. Что такое финальные поля, какие поля можно объявить со спецификатором final : это поля которое больше не может быть переприсвоены после инициализации. Класс,метод,переменная.

2. Где можно инициализировать финальные поля : в классе.

**6. 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?**

**Ответ:**

1. Что такое статические поля, статические финальные поля и статические методы : статические поля это общие переменные для членов класса , методы не привязаны к объекту класса и могут оперировать статическими полями,финальные поля это глобальные переменные (константы).

2. К чему имеют доступ статические методы : к статическим методам и полям.

3. Можно ли перегрузить и переопределить статические методы : нет , так как обращение к статическому методу происходит через имя класса,которому он принадлежит.

4. Наследуются ли статические методы : нет.

**7. 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?**

**Ответ:**

1. Что такое логические и статические блоки инициализации : логический это код заключённый в фигурные скобки и не принадлежащий ни одному методу в текущем классе,используются для инициализации ролей вызовов методов и обращении к полям текущего класса.

Статический это тоже самое только с приставкой static и вызывается он один раз в жизненном цикле приложения при создании объекта или при обращении к статическому методу(полю) данного класса.

2. Сколько их может быть в классе, в каком порядке они могут быть размещены и в каком порядке вызываются : блоков может быть столько сколько необходимо , порядок размещения неважен , выполняются они последовательно в порядке их размещения после выполнения последнего блока будет вызван конструктор.

**8. 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?**

**Ответ:**

1. Что представляют собой методы с переменным числом параметров, как передаются параметры в такие методы и что представляет собой такой параметр в методе : это параметр в который можно передавать не фиксированное число параметров (Type … args)(Type1 t1, Type … args ) нужен тогда когда точно не известно количество передаваемых аргументов в него можно передавать массив или массивы (Type [] … args) и методы с таким параметром могут быть перегружены.

2. Как осуществляется выбор подходящего метода, при использовании перегрузки для методов с переменным числом параметров : предположу что по типу параметров.

**9. Чем является класс Object? Перечислите известные вам методы класса Object, укажите их назначение.**

**Ответ:**

1. Чем является класс Object : суперкласс всех классов,его свойствами обладают все классы в Java.

2. Перечислите известные вам методы класса Object, укажите их назначение :

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

1. equals() : используется и переопределяется для сравнения двух объектов одинакового типа.

2. hashCode() : вычисляет и возвращает хэш-код объекта (число,

в общем случае вычисляемое на основе значений полей объекта)

3. toString() : возвращает представление объекта в виде строки.

4. clone() : создаёт и возвращает копию объекта.

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

5. notify() : пробуждает один поток, ожидающий на мониторе этого объекта.

6. notifyAll() : пробуждает все потоки.

7. wait() : заставляет поток ждать пробуждения ( notify() или notifyAll()).

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

8.finalize() : пызывается сборщиком мусора для объекта, когда сборщик мусора определяет, что больше нет ссылок на объект

**10. Что такое хэш-значение? Объясните, почему два разных объекта могут сгенерировать одинаковые хэш-коды?**

**Ответ:**

1. Что такое хэш-значение : число, являющееся уникальным идентификатором объекта.

2. Объясните, почему два разных объекта могут сгенерировать одинаковые хэш-коды : потому что хэш-код имеет диапазон int значения -2 147 483 648 до 2 147 483 647 и по этому при множестве объектов могут быть пересечение значений.

**11. Что такое объект класса Class? Чем использование метода getClass() и последующего сравнения возвращенного значения с Type.class отличается от использования оператора instanceof?**

**Ответ:**

1. Что такое объект класса Class : В Java почти все сущности являются объектами, за исключением примитивных типов. У каждого объекта есть класс. Сами классы тоже является объектами, и они принадлежат классу Class.

У класса Class нет публичных конструкторов. Class - это generic тип. Методы Class предназначены для получения информации о классе (объекте типа Class). Например, можно узнать полное имя класса, какие у него аннотации, какие конструкторы и т.п. Эти методы нужны для reflection. С помощью reflection вы можете создавать объекты, которые принадлежат этому классу, и при этом заранее класс объекта вы можете не знать.

2. Чем использование метода getClass() и последующего сравнения возвращенного значения с Type.class отличается от использования оператора instanceof :

**instanceof - проверяет, является ли ссылка на объект с левой стороны экземпляром типа с правой стороны или каким-либо подтипом .**

**getClass() == ... проверяет, идентичны ли типы.**

**12. Укажите правила переопределения методов equals(), hashCode() и toString().**

**Ответ:**

1. equals() :

1.Рефлексивность — объект равен самому себе. if(this == obj) return true;

2.Cимметричность — если x.equals(y) возвращает значение true, то и y.equals(x)

3.Транзитивность — если метод equals() возвращает значение true при срав-

нении объектов x и y, а также y и z, то и при сравнении x и z будет возвра-

щено значение true;

4.Непротиворечивость — при многократном вызове метода для двух не под-

вергшихся изменению за это время объектов возвращаемое значение всегда

должно быть одинаковым;

5.Ненулевая ссылка при сравнении с литералом null всегда возвращает значе-

ние false.

2. hashCode() : его нужно переопределять везде где переопределен метод equals.

1.Все одинаковые по содержанию объекты одного типа должны иметь одина-

ковые хэш-коды.

2.Различные по содержанию объекты одного типа могут иметь различные

хэш-коды

3.Во время работы приложения значение хэш-кода объекта не изменяется,

если объект не был изменен.

3. toString() : нужно переопределять таким образом чтобы он возвращал всю полезную информацию в нужном виде.

**5. Basics of OOP**

Этот список вопросов предназначен для самостоятельного контроля глубины изучаемого материала. Попробуйте выучить материал так, чтобы вы могли ответить (хоть пару слов😊) по каждому вопросу.

**OOP, Classes and Objects, Inheritance, Class Object, Interfaces**

**1. Дайте развернутое объяснение трем концепциям ООП.**

**Ответ:**

1.Инкапсуляция : сокрытие реализации методов и их обработке в классе.

2.Наследование : это передача всех свойств и поведения от одного класса другому.

3.Полиморфизм : это проявление одного поведения разными способами.

**2. Опишите процедуру инициализации полей класса и полей экземпляра класса. Когда инициализируются поля класса, а когда – поля экземпляров класса. Какие значения присваиваются полям по умолчанию? Где еще в классе полям могут быть присвоены начальные значения?**

**Ответ:**

1. Опишите процедуру инициализации полей класса и полей экземпляра класса : полей static проходит во время загрузки класса , экземпляра во время его объявления.

2. Когда инициализируются поля класса, а когда – поля экземпляров класса :

Поля класса : если оно static то во время загрузки класса,если нет то при создании экземпляра класса.

Экземпляр класса : в месте объявления или в конструкторе.

3. Какие значения присваиваются полям по умолчанию : примут значения для ссылочных null , для примитивов 0 для char « » boolean false .

4. Где ещё в классе полям могут быть присвоены начальные значения : в конструкторе или в блоке.

**3. Приведите правила, которым должен следовать компонент java-bean.**

**Ответ:**

1.Класс должен содержать private поля.

2.Конструктор без параметров.

3.Должны быть Get и Set для доступа к полям.

4.Класс должен имплементировать интерфейс Serializable.

5.В класс должны быть переопределены методы equals(), hashCode() и toString().

**4. Дайте определение перегрузке методов. Как вы думаете, чем удобна перегрузка методов? Укажите, какие методы могут перегружаться, и какими методами они могут быть перегружены? Можно ли перегрузить методы в базовом и производном классах? Можно ли private метод базового класса перегрузить public методов производного? Можно ли перегрузить конструкторы, и можно ли при перегрузке конструкторов менять атрибуты доступа у конструкторов?**

**Ответ:**

1. Дайте определение перегрузке методов : это создание методов с одинаковыми названиями но с разными параметрами.

2. Как вы думаете, чем удобна перегрузка методов : можно создать много методов с одним именем но с разными входными параметрами.

3. Укажите, какие методы могут перегружаться, и какими методами они могут быть перегружены : статические методы могут перегружаться нестатическими,и наоборот.

4. Можно ли перегрузить методы в базовом и производном классах : да.

5. Можно ли private метод базового класса перегрузить public методов производного : нет.

6. Можно ли перегрузить конструкторы, и можно ли при перегрузке конструкторов менять атрибуты доступа у конструкторов : да,да.

**5. Объясните, что такое раннее и позднее связывание? Перегрузка – это раннее или позднее связывание? Объясните 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.**

**Ответ:**

1. Объясните, что такое раннее и позднее связывание : это связывание ссылки и кода где она будет применятся на этапе компиляции,а позднее во время исполнения кода.

2. Перегрузка – это раннее или позднее связывание : это раннее связывание.

3. Объясните 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 : правило для перегрузки два метода из одного класса не могут иметь одинаковые сигнатуры(при этом сигнатура не включает в себя возвращаемый параметр). Выбирается наиболее подходящий метод. Расширяющее преобразование примитивов имеет приоритет над автоупоковкой/распоковкой. Сужающее преобразование примитивов автоматически не выполняется. Когда вызывается метод , количество фактических аргументов (и любых явных аргументов типа) и типы аргументов времени компиляции используются во время компиляции для определения сигнатуры метода, который будет вызван.

**6. Объясните, как вы понимаете, что такое неявная ссылка this? В каких методах эта ссылка присутствует, а в каких – нет, и почему?**

**Ответ:**

1. Объясните, как вы понимаете, что такое неявная ссылка this : это указатель на объект вызвавшему его методу.Кроме static методов.

2. В каких методах эта ссылка присутствует, а в каких – нет, и почему : в конструкторах одного типа, не присутствует в static методах.

**7. Что такое финальные поля, какие поля можно объявить со спецификатором final? Где можно инициализировать финальные поля?**

**Ответ:**

1. Что такое финальные поля, какие поля можно объявить со спецификатором final : это поля которое больше не может быть переприсвоены после инициализации. Класс,метод,переменная.

2. Где можно инициализировать финальные поля : в классе.

**8. 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?**

**Ответ:**

1. Что такое статические поля, статические финальные поля и статические методы : статические поля это общие переменные для членов класса , методы не привязаны к объекту класса и могут оперировать статическими полями,финальные поля это глобальные переменные (константы).

2. К чему имеют доступ статические методы : к статическим методам и полям.

3. Можно ли перегрузить и переопределить статические методы : нет , так как обращение к статическому методу происходит через имя класса,которому он принадлежит.

4. Наследуются ли статические методы : нет.

**9. 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?**

**Ответ:**

1. Что такое логические и статические блоки инициализации : логический это код заключённый в фигурные скобки и не принадлежащий ни одному методу в текущем классе,используются для инициализации ролей вызовов методов и обращении к полям текущего класса.

Статический это тоже самое только с приставкой static и вызывается он один раз в жизненном цикле приложения при создании объекта или при обращении к статическому методу(полю) данного класса.

2. Сколько их может быть в классе, в каком порядке они могут быть размещены и в каком порядке вызываются : блоков может быть столько сколько необходимо , порядок размещения неважен , выполняются они последовательно в порядке их размещения после выполнения последнего блока будет вызван конструктор.

**10. 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?**

**Ответ:**

1. Что представляют собой методы с переменным числом параметров, как передаются параметры в такие методы и что представляет собой такой параметр в методе : это параметр в который можно передавать не фиксированное число параметров (Type … args)(Type1 t1, Type … args ) нужен тогда когда точно не известно количество передаваемых аргументов в него можно передавать массив или массивы (Type [] … args) и методы с таким параметром могут быть перегружены.

2. Как осуществляется выбор подходящего метода, при использовании перегрузки для методов с переменным числом параметров : предположу что по типу параметров.

**11. Чем является класс Object? Перечислите известные вам методы класса Object, укажите их назначение.**

**Ответ:**

1. Чем является класс Object : суперкласс всех классов,его свойствами обладают все классы в Java.

2. Перечислите известные вам методы класса Object, укажите их назначение :

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

1. equals() : используется и переопределяется для сравнения двух объектов одинакового типа.

2. hashCode() : вычисляет и возвращает хэш-код объекта (число,

в общем случае вычисляемое на основе значений полей объекта)

3. toString() : возвращает представление объекта в виде строки.

4. clone() : создаёт и возвращает копию объекта.

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

5. notify() : пробуждает один поток, ожидающий на мониторе этого объекта.

6. notifyAll() : пробуждает все потоки.

7. wait() : заставляет поток ждать пробуждения ( notify() или notifyAll()).

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

8.finalize() : пызывается сборщиком мусора для объекта, когда сборщик мусора определяет, что больше нет ссылок на объект

**12. Что такое хэш-значение? Объясните, почему два разных объекта могут сгенерировать одинаковые хэш-коды?**

**Ответ:**

1. Что такое хэш-значение : число, являющееся уникальным идентификатором объекта.

2. Объясните, почему два разных объекта могут сгенерировать одинаковые хэш-коды : потому что хэш-код имеет диапазон int значения -2 147 483 648 до 2 147 483 647 и по этому при множестве объектов могут быть пересечение значений.

**13. Как вы думаете, для чего используется наследование классов в java-программе? Приведите пример наследования. Как вы думаете, поля и методы, помеченными модификатором доступа private, наследуются?**

**Ответ:**

1.Как вы думаете, для чего используется наследование классов в java-программе : для того чтобы наделить класс наследника доступным функционалом (методы,поля) родительского класса при условии что они не private.

2.Приведите пример наследования :

public class Cat extend Animal{

}

3. Как вы думаете, поля и методы, помеченными модификатором доступа private, наследуются : нет.

**14. Укажите, 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()?**

**Ответ:**

1. Укажите, как вызываются конструкторы при создании объекта производного класса : начиная от родительского класса и по иерархии вверх до конструктора производного класса

2. Что в конструкторе класса делает оператор super() : вызывает конструктор родителя. Должен стоять первым оператором в конструкторе производного класса.

3. Возможно ли в одном конструкторе использовать операторы super() и this() : да.

**15. Объясните, как вы понимаете 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?**

**Ответ:**

1. Ссылка базового класса может ссылаться на объекты своих производных типов :

Animal dog = new Dog();

Ссылка базового класса Animal может ссылаться на объекты своих производных типов Dog.Обратное же действие вызовет исключение ClassCastException так как Animal не является подклассом класса Dog.

Dog dog = (Dog) new Animal(); // ClassCastException

2. Объект производного класса может быть использован везде, где ожидается объект его базового типа : нет .

**16. Что такое переопределение методов? Как вы думаете, 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?**

**Ответ:**

1. Что такое переопределение методов : наделение метода родительского класса другой реализацией,при условии,что у переопределяемого метода будут те же аргументы,иначе это будет перегрузкой.

2. Как вы думаете, зачем они нужны : для наделение метода родительского класса другой реализацией.У класса Animal есть метод voice(); в классе наследнике Cat можно переопределить под нужды класса Cat и по аналогии в Dog,Rat,Duck… Это один из инструментов полиморфизма (один метод много реализаций).

3. Можно ли менять возвращаемый тип при переопределении методов : после Java 5 да.Только они должны находится ниже по иерархии суперкласса.

4. Можно ли переопределить методы в рамках одного класса : нет.

**17. Определите правило вызова переопределенных методов. Можно ли статические методы переопределить нестатическими и наоборот?**

**Ответ:**

1. Определите правило вызова переопределенных методов : метод будет вызван в ближайший к создаваемому объекту метод.

2. Можно ли статические методы переопределить нестатическими и наоборот : нельзя в обоих случаях.

**18. Какие свойства имеют финальные методы и финальные классы? Как вы думаете, зачем их использовать?**

**Ответ:**

1. Какие свойства имеют финальные методы и финальные классы : константные.Эти методы и классы нельзя наследовать,переопределять и изменять.

2. Как вы думаете, зачем их использовать : его стоит использовать в случае когда программист считает реализацию метода самодостаточной и не требующей изменений. Финальный класс нельзя будет наследовать и изменять. Использовать можно например в паттерне синглтон.

**19. Укажите правила приведения типов при наследовании. Напишите примеры явного и неявного преобразования ссылочных типов. Объясните, какие ошибки могут возникать при явном преобразовании ссылочных типов.**

**Ответ:**

1. Укажите правила приведения типов при наследовании : приведение типов можно выполнять только в иерархии наследования.

2. Напишите примеры явного и неявного преобразования ссылочных типов :

Animal animal = new Dog("DOG","Bob",2); \\Неявное преобразование (upcasting)  
Animal animal1 = new Animal("OOO");  
if(animal instanceof Dog) { \\ Проверка является ли объект animal подтипом Dog.  
 Dog dog1 = (Dog) animal; \\ Явное преобразование (downcasting)  
}  
else {  
 System.*out*.println("ClassCastException");  
}

3. Объясните, какие ошибки могут возникать при явном преобразовании ссылочных типов :

если не проводить проверку на instanceof то можно присвоить неправильный тип что приведёт к ошибке ClassCastException.

**20. Что такое объект класса Class? Чем использование метода getClass() и последующего сравнения возвращенного значения с Type.class отличается от использования оператора instanceof?**

**Ответ:**

1. Что такое объект класса Class : В Java почти все сущности являются объектами, за исключением примитивных типов. У каждого объекта есть класс. Сами классы тоже является объектами, и они принадлежат классу Class.

У класса Class нет публичных конструкторов. Class - это generic тип. Методы Class предназначены для получения информации о классе (объекте типа Class). Например, можно узнать полное имя класса, какие у него аннотации, какие конструкторы и т.п. Эти методы нужны для reflection. С помощью reflection вы можете создавать объекты, которые принадлежат этому классу, и при этом заранее класс объекта вы можете не знать.

2. Чем использование метода getClass() и последующего сравнения возвращенного значения с Type.class отличается от использования оператора instanceof :

**instanceof - проверяет, является ли ссылка на объект с левой стороны экземпляром типа с правой стороны или каким-либо подтипом .**

**getClass() == ... проверяет, идентичны ли типы.**

**21. Укажите правила переопределения методов equals(), hashCode() и toString().**

**Ответ:**

1. equals() :

1.Рефлексивность — объект равен самому себе. if(this == obj) return true;

2.Cимметричность — если x.equals(y) возвращает значение true, то и y.equals(x)

3.Транзитивность — если метод equals() возвращает значение true при срав-

нении объектов x и y, а также y и z, то и при сравнении x и z будет возвра-

щено значение true;

4.Непротиворечивость — при многократном вызове метода для двух не под-

вергшихся изменению за это время объектов возвращаемое значение всегда

должно быть одинаковым;

5.Ненулевая ссылка при сравнении с литералом null всегда возвращает значе-

ние false.

2. hashCode() : его нужно переопределять везде где переопределен метод equals.

1.Все одинаковые по содержанию объекты одного типа должны иметь одина-

ковые хэш-коды.

2.Различные по содержанию объекты одного типа могут иметь различные

хэш-коды

3.Во время работы приложения значение хэш-кода объекта не изменяется,

если объект не был изменен.

3. toString() : нужно переопределять таким образом чтобы он возвращал всю полезную информацию в нужном виде.

**22. 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?**

**Ответ:**

1. Что такое абстрактные классы и методы : абстрактция описывает поля и методы,но не реализующих их , они будут реализованы в подклассах (как каркас для будущих классов).

2. Зачем они нужны : они быть вершиной иерархии его различных реализаций.

3. Бывают ли случаи, когда абстрактные методы содержат тело : да.

4. Можно ли в абстрактных классах определять конструкторы : да.

5. Могут ли абстрактные классы содержать неабстрактные методы : да.

6. Можно ли от абстрактных классов создавать объекты и почему : объекты таких классов нельзя создать с помощью оператора new, но можно создать объекты подклассов, которые реализуют все эти методы.

**23. Что такое интерфейсы? Как определить и реализовать интерфейс в java-программе? Укажите 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?**

**Ответ:**

1. Что такое интерфейсы : описание или спецификация функциональности,

которую должен реализовывать каждый класс, его имплементирующий.(по типу шаблона)

2. Как определить и реализовать интерфейс в java-программе :

1.Создать в проекте интерфейс.

2.Определить его функционал(объявить методы,поля и т.д).

3.Имплементировать (заключить контракт с классом на реализацию всего того что написано в интерфейсе) интерфейс при помощи оператора implements в нужном классе.

3. Укажите спецификаторы, которые приобретают методы и поля, определенные в интерфейсе. Можно ли описывать в интерфейсе конструкторы и создавать объекты :

1.Для методов спецификатор public

2.Для полей public static final

3.Нет.

4. Можно ли создавать интерфейсные ссылки и если да, то на какие объекты они могут ссылаться : да,на объекты в которых реализован данный интерфейс.

**24. Для чего служит интерфейс Clonable? Как правильно переопределить метод clone() класса Object, для того, что объект мог создавать свои адекватные копии?**

**Ответ:**

1. Для чего служит интерфейс Clonable :как маркер того что данный класс переопределил метод clone() класса Object и то,что метод clone() вернёт точную копию вызвавшего его объекта.

2. Как правильно переопределить метод clone() класса Object, для того, что объект мог создавать свои адекватные копии : нужно в классе имплементировать интерфейс Clonable

потом переопределить метод clone():

public Object clone() throws CloneNotSupportedException{

return super.clone();

}

**25. Для чего служат интерфейсы Comparable и Comparator? В каких случаях предпочтительнее использовать первый, а когда – второй? Как их реализовать и использовать?**

**Ответ:**

1. Для чего служат интерфейсы Comparable и Comparator :

1. Comparable — это интерфейс даёт возможность для сравнения и сортировки объектов класса его реализовавшего,в нём же должен быть переопределен метод int compareTo(Object o) что даёт нам лишь один способ сравнения например по возрасту,если нужно чтобы было несколько моделей сравнения нужно использовать интерфейс Comparator.

2. Comparator — это интерфейс даёт возможность для сравнения и сортировки объектов но в отличии от Comparable реализуется в отдельном классе и количество его реализаций может быть сколько потребуется в программе,то есть можно написать сколько угодно моделей сравнения (по возрасту,имени,году и т.д).

2. В каких случаях предпочтительнее использовать первый, а когда – второй : интерфейс Comparable стоит использовать если в программе нужно определить одну модель сравнения (по умолчанию)

Comparator стоит использовать если исходный класс нельзя модифицировать или нужно более одной модели сравнения.

3. Как их реализовать и использовать :

**Comparable** :

public class Dog implements Comparable<Dog>{  
 private String name;  
 private int fullAge;

@Override  
public int compareTo(Dog dog) {  
 if(fullAge == dog.fullAge){  
 return 0;  
 }  
 if(fullAge < dog.fullAge){  
 return -1;  
 }  
 return 1;  
}

class Main {

public static void main(String[] args) {

List<Dog> dogs = new ArrayList<>();

dogs.add(new Dog("Bob",6));  
 dogs.add(new Dog("Tom",2));  
 dogs.add(new Dog("Ali",3));

Collections.*sort*(list); // для Comparable

Collections.*sort*(list , new SortByName()); // для Comparator от A-Z

Collections.*sort*(list , new SortByName().reversed()); // для Comparator от Z-A

for (Dog dog : list){  
 System.*out*.println(dog);  
 }

}

}

}

**Comparator** :

public class SortByName implements Comparator<Dog> {  
 @Override  
 public int compare(Dog dog1, Dog dog2) {  
 return dog1.getName().compareTo(dog2.getName());  
 }  
}

**Generic classes and Interfaces, Enums**

**1. Что такое перечисления в Java. Как объявить перечисление? Чем являются элементы перечислений? Кто и когда создает экземпляры перечислений?**

**Ответ:**

1. Что такое перечисления в Java. Как объявить перечисление : это набор логически связанных констант (дни недели,месяца года,сезоны и т.д),объявляется перечисление может как внутри класса так и вне класса,хоть перечисление относиться к типу класс но вместо слова class в нем пишется слово enum (enumeration) и он облагается рядом ограничений.Например,от него нельзя наследоваться и он не может быть наследуемым , быть

абстрактным , быть параметризированным ,создавать экземпляры, используя ключевое слово **new.** EMUM по умолчанию наследует абстрактный класс java.lang.Enum и его методы как

**static enumType[] values()** — возвращает массив, содержащий все элемен-

ты перечисления в порядке их объявления;

**static <T extends Enum<T>> T valueOf(Class<T> enumType, String arg)** —

создает элемент перечисления, соответствующий заданному типу и значению

передаваемой строки;

**static enumType valueOf(String arg)** — создает элемент перечисления, со-

ответствующий значению передаваемой строки;

**int ordinal()** — возвращает позицию элемента перечисления, начиная

с нуля, следствием чего является возможность сравнения элементов перечи-

сления между собой на больше\меньше соответсвующими операторами;

**String name()** — возвращает имя элемента, так же как и toString();

**int compareTo(T t)** — сравнивает элементы на больше, меньше либо равно.

Пример объявления перечисления :

public enum Role {

ADMIN,

USER,

UNKNOWN

}

2. Чем являются элементы перечислений : элементы перечислений это статически доступные экземпляры enum-класса.

3. Кто и когда создает экземпляры перечислений :

**2. 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?**

**Ответ:**

1. Можно ли самостоятельно создать экземпляр перечисления : нет,так как Enum не имеет публичных конструкторов только приватные.

2. А ссылку типа перечисления : можно и она будет ссылаться на экземпляр Enum.

Role userRole = Role.USER;

3. Как сравнить, что в двух переменных содержится один и тот же элемент перечисления и почему именно так : можно == и x.equals(y); В классе Enum метод equals не переопределен и также сравнивает ссылки на константы класса Enum.

**3. Что такое анонимные классы?**

**Ответ:**

1. Что такое анонимные классы : это внутренние классы даёт возможность переопределения методов в отдельно взятом экземпляре,который будет использован в месте объявления и не изменит исходный класс. Например:

public class Dog {  
 public void voice (){  
 System.out.println("Гав-гав");  
 }  
  
 public static void main(String[] args) {  
 Dog angry = new Dog(){ // Анонимный класс в котором переопределен метод voice().  
 @Override  
 public void voice (){   
 System.out.println("Pppppp");  
 }  
 };  
 Dog dog = new Dog();  
 dog.voice();  
 angry.voice();  
   
 }  
}

Свойства анонимных классов:

• расширяет другой класс или реализует интерфейс при объявлении одного

единственного объекта, остальным объектам будет соответствовать реали-

зация, определенная в самом классе;

• объявление анонимного объекта выполняется одновременно с созданием

его объекта с помощью оператора new;

• конструкторы анонимных классов ни определить, ни переопределить

нельзя;

• анонимные классы допускают вложенность друг в друга (нежелательно ис-

пользовать);

• объявление анонимного класса в перечислении отличается от простого ано-

нимного класса, поскольку инициализация всех элементов происходит при

первом обращении к типу.

Ситуации, в которых следует использовать внутренние классы:

• выделение самостоятельной логической части сложного класса;

• сокрытие реализации;

• одномоментное использование переопределенных методов;

• реализация обработчиков событий;

• запуск потоков выполнения;

• отслеживание внутреннего состояния, например, с помощью enum.

**4. 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?**

**Ответ:**

1. Что такое параметризованные классы : это классы которые принимают разные типы данных в качестве параметров .

2. Для чего они необходимы : для того чтобы использовать один код для обобщённых (разных) типов данных.

3. Приведите пример параметризованного класса и пример создания объекта параметризованного класса :

//Пример параметризованного класса ArrayList<E> где <Е> означает что этот класс может принять на вход объект любого типа (примитивные должны быть обёрнуты в соответствующую типу обёртку int → Integer и т.д )

public class ArrayList<E> extends AbstractList<E>  
 implements List<E>, RandomAccess, Cloneable, java.io.Serializable

// Пример создания объекта параметризованного класса.Класс ArrayList мы вызываем через интерфейс List<E> который реализует этот класс,передаём ему в параметры тип данных с каким мы хотим работать в данном случае это String и создаём через оператор new класс ArrayList в котором параметры будут такими же как и у интерфейса который его вызывает.

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

4. Объясните, ссылки какого типа могут ссылаться на объекты параметризованных классов : любого если это не ограничено параметрами класса.

Пример :

public class Post <K extends Number, V>

В качестве типа K допустимо использовать только подклассы класса Number,тогда как V может принять любой ссылочный тип данных.

5. Можно ли создать объект, параметризовав его примитивным типом данных : нет.

**Exceptions and Errors**

**1. Что для программы является исключительной ситуацией? Какие способы обработки ошибок в программах вы знаете?**

**Ответ:**

1. Что для программы является исключительной ситуацией : это ошибка, которая возникает во время выполнения программного кода.

2. Какие способы обработки ошибок в программах вы знаете :

• перехват и обработка исключения в блоке try-catch метода;

• объявление исключения в секции throws метода и передача вызывающему методу (в первую очередь для проверяемых исключений).

**2. Что такое исключение для Java-программы? Что значит “программа выбросила исключение”? Опишите ситуации, когда исключения выбрасываются виртуальной машиной(автоматически), и когда необходимо их выбрасывать вручную?**

**Ответ:**

1. Что такое исключение для Java-программы : это ошибка, которая возникает во время выполнения программного кода.

2. Что значит “программа выбросила исключение” : возможные исключения были обработаны и одно из них наступило, тогда в консоли выводится содержание ошибки/исключения.

3. Опишите ситуации, когда исключения выбрасываются виртуальной машиной(автоматически), и когда необходимо их выбрасывать вручную : например при попытке деления на ноль будет (автоматически) сгенерировано ArithmeticException.

Вручную генерировать исключения необходимо для указания, например, на заведомо

ошибочный результат выполнения операции, на некорректные значения парамет­

ра метода и др.

**3. Приведите иерархию классов-исключений, делящую исключения на проверяемые и непроверяемые. В чем особенности проверяемых и непроверяемых исключений?**

**Ответ:**

1. Приведите иерархию классов-исключений, делящую исключения на проверяемые и непроверяемые :

**Throwable**

**| |**

**Error Exception**

**|**

**RuntimeException**

**Error** — это непроверяемые исключения(**VirtualMachineError** — виртуальная машина Java сломана или у неё закончились ресурсы, необходимые для продолжения работы).

**RuntimeException** — это непроверяемые исключения(**NullPointerException** — попытка обращения к несуществующему(Null) объекту).

**Exception** — проверяемое исключение(**FileNotFoundException** — ошибка при обращении к файлу которого не существует или файл с ограничениями например,файл только для чтения).

2. В чем особенности проверяемых и непроверяемых исключений :

1.Проверяемое — должно быть обработано в методе которое может его генерировать, или включено в throws-список метода для дальнейшей обработки в вызывающих методах

2.Непроверяемое — можно не обрабатывать.

**4. 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 называют стеком операторов try? Как работает блок try с ресурсами.**

**Ответ:**

1. Объясните работу оператора try-catch-finally :

**1.try** — в этом блоке происходит **попытка** выполнить код помещённый в фигурных скобках.

**2.catch** — в этом блоке происходит **отлов** исключений, где в круглых скобках указывается класс исключения который ожидается, а в фигурных обработка этого исключения.

catch(IOException){ \\ объявление ожидаемого исключения

\\ обработка исключения.

}

**3. finally** — (не обязательный блок) этот блок выполнится в любом случае,может служить для корректного завершения потока или закрытие ресурса .

2. Когда данный оператор следует использовать : тогда когда нужно перехватить исключение.

3. Сколько блоков **catch** может соответствовать одному блоку **try** : столько сколько необходимо для обработки операторов в блоке **try**.

4. Можно ли вкладывать блоки **try** друг в друга, можно ли вложить блок **try** в **catch** или **finally** : да,можно.

5. Как происходит обработка исключений, выброшенных внутренним блоком **try**, если среди его блоков **catch** нет подходящего : если внутренний **try** не найдёт подходящий блок **catch** он будет искать во внешнем **catch** блоке.

6. Как работает блок **try** с ресурсами : try(with-resources) позволяет автоматически закрывать ресурсы которые объявлены в круглых скобках после оператора try, что в свою очередь позволяет избавиться он необходимости делать это вручную в блоку finally.

**5. Укажите правило расположения блоков catch в зависимости от типов перехватываемых исключений. Может ли перехваченной исключение быть 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.**

**Ответ:**

1. Укажите правило расположения блоков catch в зависимости от типов перехватываемых исключений : подклассы должны идти перед суперклассом исключения, в пративном случае суперкласс будет перехватывать это исключение.

2. Может ли перехваченной исключение быть сгенерировано снова, и, если да, то как и кто в этом случае будет обрабатывать повторно сгенерированное исключение : да , будет обработанно cледующим блоком catch.

3. Может ли блок catch выбрасывать иные исключения, и если да, то опишите ситуацию, когда это может быть необходимо : Да может. Например мы создали своё исключение и обернув ошибку пробрасываем его наверх к обработчику.

try {

\\обращение к БД.

}catch (SQLException e){

throw new DAOException («Database error»,e);

}

**6. Когда происходит вызов блока finally? Существуют ли ситуации, когда блок finally не будет вызван? Может ли блок finally выбрасывать исключений? Может ли блок finally выполнится дважды?**

**Ответ:**

1. Когда происходит вызов блока finally : после try или catch,но в любом случае он будет вызван.

2. Существуют ли ситуации, когда блок finally не будет вызван : при аварийной остановке виртуальной машины,или принудительно остановленной System.exit.

3. Может ли блок finally выполнится дважды : в случае когда метод вызывает сам себя в блоке finally, или в цикле.