Оглавление

[1. Чем является класс Object? Перечислите известные вам методы класса Object, укажите их назначение. 1](#_Toc21614028)

[2. Что такое хэш-значение? Объясните, почему два разных объекта могут сгенерировать одинаковые хэш-коды? 2](#_Toc21614029)

[3. 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? 2](#_Toc21614030)

[4. 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()? 3](#_Toc21614031)

[5. 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? 3](#_Toc21614032)

[6. 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? 3](#_Toc21614033)

[7. Определите правило вызова переопределенных методов. Можно ли статические методы переопределить нестатическими и наоборот? 3](#_Toc21614034)

[8. Какие свойства имеют финальные методы и финальные классы? Как вы думаете, зачем их использовать? 3](#_Toc21614035)

[9. Укажите правила приведения типов при наследовании. Напишите примеры явного и неявного преобразования ссылочных типов. Объясните, какие ошибки могут возникать при явном преобразовании ссылочных типов. 3](#_Toc21614036)

[10. Что такое объект класса Class? Чем использование метода getClass() и последующего сравнения возвращенного значения с Type.class отличается от использования оператора instanceof? 3](#_Toc21614037)

[11. Укажите правила переопределения методов equals(), hashCode() и toString(). 3](#_Toc21614038)

[12. 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? 3](#_Toc21614039)

[13. 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? 3](#_Toc21614040)

[14. Для чего служит интерфейс Clonable? Как правильно переопределить метод clone() класса Object, для того, что объект мог создавать свои адекватные копии? 3](#_Toc21614041)

[15. Для чего служат интерфейсы Comparable и Comparator? В каких случаях предпочтительнее использовать первый, а когда – второй? Как их реализовать и использовать? 3](#_Toc21614042)

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

package java.lang;

public class Object {

private static native void registerNatives();

static {

registerNatives();

}

public final native Class<?> getClass();

public native int hashCode();

public boolean equals(Object obj) {

return (this == obj);

}

protected native Object clone() throws CloneNotSupportedException;

public String toString() {

return getClass().getName() + "@" + Integer.toHexString(hashCode());

}

public final native void notify();

public final native void notifyAll();

public final native void wait(long timeout) throws InterruptedException;

public final void wait(long timeout, int nanos) throws InterruptedException {

if (timeout < 0) {

throw new IllegalArgumentException("timeout value is negative");

}

if (nanos < 0 || nanos > 999999) {

throw new IllegalArgumentException( "nanosecond timeout value out of range");

}

if (nanos >= 500000 || (nanos != 0 && timeout == 0)) {

timeout++;

}

wait(timeout);

}

public final void wait() throws InterruptedException {

wait(0);

}

protected void finalize() throws Throwable { }

}

java.lang.Object особый класс, имеющий уникальный class file, в котором в качестве суперкласса не указан ни один класс. Создать такой же класс средствами языка Java нельзя, но также затруднительно, если вообще возможно, сделать это и манипуляциями с байтами class файла.

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

**Хеш-функция** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *hash function* от *hash* — «превращать в фарш», «мешанина»[[1]](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F#cite_note-_ff142c473b3a285a-1)), или **функция свёртки** — функция, осуществляющая преобразование [массива](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) входных данных произвольной длины в (выходную) [битовую](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82) строку установленной длины, выполняемое [определённым алгоритмом](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC). Преобразование, производимое хеш-функцией, называется **хешированием**. Исходные данные называются входным массивом, «**ключом**» или «***сообщением***». Результат преобразования (выходные данные) называется «***[хешем](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%81%D1%83%D0%BC%D0%BC%D0%B0" \o "Хеш-сумма)***», «***хеш-кодом***», «***хеш-суммой***», «сводкой [сообщения](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B5)».

Хеш-функции применяются в следующих случаях:

* при построении [ассоциативных массивов](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2);
* при поиске дубликатов в сериях наборов данных;
* при построении уникальных идентификаторов для наборов данных;
* при вычислении [контрольных сумм](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%83%D0%BC%D0%BC%D0%B0) от данных (сигнала) для последующего обнаружения в них ошибок (возникших случайно или внесённых намеренно), возникающих при хранении и/или передаче данных;
* при сохранении паролей *в системах защиты* в виде хеш-кода (для восстановления пароля по хеш-коду требуется функция, являющаяся [обратной](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) по отношению к использованной хеш-функции);
* при выработке [электронной подписи](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C) (на практике часто подписывается не само сообщение, а его «хеш-образ»);
* и др.

В общем случае (согласно [принципу Дирихле](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF_%D0%94%D0%B8%D1%80%D0%B8%D1%85%D0%BB%D0%B5_(%D0%BA%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%B8%D0%BA%D0%B0))) нет однозначного соответствия между хеш-кодом (выходными данными) и исходными (входными) данными. Возвращаемые хеш-функцией значения (выходные данные) менее разнообразны, чем значения входного массива (входные данные). Случай, при котором хеш-функция преобразует более чем один массив входных данных в одинаковые сводки, называется «[**коллизией**](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B8%D0%B7%D0%B8%D1%8F_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8)». Вероятность возникновения коллизий используется для оценки качества хеш-функций.

## 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?

С помощью наследования можно расширить функционал уже имеющихся классов за счет добавления нового функционала или изменения старого. Производный класс имеет доступ ко всем методам и полям базового класса (даже если базовый класс находится в другом пакете) кроме тех, которые определены с модификатором **private**. При этом производный класс также может добавлять свои поля и методы. Если в базовом классе определены конструкторы, то в конструкторе производного классы необходимо вызвать один из конструкторов базового класса с помощью ключевого слова **super**.

## 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()?

Сначала вызываются конструкторы базового класса, а затем конструкторы производного класса. Super() вызывает конструктор базового класса, не может одновременно использоваться вместе с this().

## 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?

## 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?

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

## Какие свойства имеют финальные методы и финальные классы? Как вы думаете, зачем их использовать?

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

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

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

## 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?

## 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?

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

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