- Какие существуют виды потоков ввода/вывода?

Объект, из которого можно считать данные, называется **потоком ввода**, а объект, в который можно записывать данные, - **потоком вывода**.

java.io

- Назовите основные предки потоков ввода/вывода

Байтовые: java.io.InputStream, java.io.OutputStream;

Символьные: java.io.Reader, java.io.Writer;

- Что общего и чем отличаются следующие потоки: InputStream, OutputStream, Reader, Writer?

Потоки байтов

Класс InputStream является базовым для всех классов, управляющих байтовыми потоками ввода.

Класс OutputStream

Класс OutputStream является базовым классом для всех классов, которые работают с бинарными потоками записи.

Абстрактные классы Reader и Writer

Абстрактный класс **Reader** предоставляет функционал для чтения текстовой информации.

Класс **Writer** определяет функционал для всех символьных потоков вывода

RandomAccessFile

класс пакета Java IO API, он позволяет перемещаться по файлу, читать из него или писать в него. также можно заменить существующие части файла, речь идет о обновлении содержимого файла, а точней о обновлении фрагмента файла. Это невозможно сделать с помощью FileInputStream или FileOutputStream, но RandomAccessFile даст вам эту возможность.

test.seek(8) – переставляем курсор на 8 байт

строки перезаписываются!

Какие есть режимы доступа к файлу?

```
file.setReadable(true, false); - первый true для пользователя приложения, второй false для других пользователей

file.setWritable(true, false);

file.setExecutable(true, true);
```

В каких пакетах лежат классы-потоки?

java.io

java.nio

java.util - для работы с zip

Что вы знаете о классах-надстройках?

К классам-надстройкам относятся классы, наследующие свойства базовых классов и расширяющие их свойства. В качестве примеров можно привести классы BufferedOutputStream, BufferedInputStrem, BufferedWriter, BufferedReader, которые буферезируют поток, повышая, таким образом, производительность.

Какой класс-надстройка позволяет читать данные из входного байтового потока в формате примитивных типов данных?

IДля чтения байтовых данных (не строк) применяется класс DataInputStream. В этом случае необходимо использовать классы из группы InputStream.

Для преобразования строки в массив байтов, пригодный для помещения в поток ByteArrayInputStream, в классе String предусмотрен метод getBytes(). Полученный ByteArrayInputStream представляет собой поток InputStream, подходящий для передачи DataInputStream.

При побайтовом чтении символов из форматированного потока DataInputStream методом readByte() любое полученное значение будет считаться действительным, поэтому возвращаемое значение неприменимо для идентификации конца потока. Вместо этого можно использовать метод available(), который сообщает, сколько еще осталось символов.

Класс DataInputStream позволяет читать элементарные данные из потока через интерфейс DataInput, который определяет методы, преобразующие элементарные значения в форму последовательности байтов. Такие потоки облегчают сохранение в файле двоичных данных.

Какой класс-надстройка позволяет ускорить чтение/запись за счет использования буфера?

java.io.BufferedInputStream(InputStream in) || BufferedInputStream(InputStream in, int size),
java.io.BufferedOutputStream(OutputStream out) || BufferedOutputStream(OutputStream out, int size),
java.io.BufferedReader(Reader r) || BufferedReader(Reader in, int sz),
java.io.BufferedWriter(Writer out) || BufferedWriter(Writer out, int sz)

Какие классы позволяют преобразовать байтовые потоки в символьные и обратно?

OutputStreamWriter
InputStreamReader

Какой класс предназначен для работы с элементами файловой системы (ЭФС)?

Для управления информацией о файлах и каталогах используется класс java.io.File. На уровне операционной системы файлы и каталоги имеют существенные отличия, но в Java они описываются одним классом File. В Java каталог трактуется как обычный файл, но с дополнительным свойством — списком имен файлов, который можно просмотреть с помощью метода list ().

Какой символ является разделителем при указании пути к ЭФС?

Windows \\
Linux /

Что вы знаете об интерфейсе FilenameFilter?

Интерфейс `java.io.FilenameFilter` может быть реализован для фильтрации имен файлов в определенной папке. Интерфейс `FilenameFilter` содержит метод `boolean accept(File dir, String name)`. Класс должен реализовывать этот метод, а каждый тестируемый файл должен быть включен в общий список файлов.

Что такое сериализация?

Сериализация — это процесс сохранения состояния объекта в последовательность байт.

Какие условия "благополучной" сериализации объекта?

Сохраняемый объект должен реализовать интерфейс *Serializable* или унаследовать эту реализацию от вышестоящего по иерархии объекта
Сохраняемый объект должен пометить все свои несериализуемые поля как *transient*

Какие классы позволяют архивировать объекты?

Для создания архива используется класс ZipOutputStream. Для создания объекта ZipOutputStream в его конструктор передается поток вывода:

ZipOutputStream(OutputStream out)

DeflaterOutputStream, InflaterInputStream, ZipInputStream, ZipOutputStream, GZIPInputStream, GZIPOutputStream.

Основные отличия между Java IO и Java NIO

IO: Потокоориентированный, Блокирующий (синхронный) ввод/вывод

NIO: Буфер-ориентированный, Неблокирующий (асинхронный) ввод/вывод, Селекторы.

InputStreams используются для чтения байтов из потока. Поэтому они полезны для двоичных данных, таких как изображения, видео и сериализованные объекты.
Считыватели, с другой стороны, представляют собой потоки символов, поэтому их лучше всего использовать для чтения символьных данных.

В чем отличие Scanner от BufferedReader?

Scanner работает медленно, но зато предоставляет очень широкий API с кучей удобных методов, а BufferedReader работает быстрее, потому что читает часть входных данных в буфер, откуда они быстрее читаются по частям, то есть обращение к консоли происходит реже.
BufferedReader нужен, чтобы буфферизовать чтение с любого потока. Как частный случай, можно буфферизовать ввод с терминала. Иногда используется ради метода readLine, когда сложно или просто лень обрабатывать текст блоками. Для целей чтения текста из стандартного вводы или ручного ввода пользователя

вполне пригодный вариант. Но при таком подходе придётся парсить числа вручную, что не слишком большая сложность.

Что делать, если одно из полей сериализовывать не нужно.

transient

Как сериализовать объект класса?

Для сериализации объектов в поток используется класс **ObjectOutputStream**. Он записывает данные в поток.

Для создания объекта ObjectOutputStream в конструктор передается поток, в который производится запись:

```
ObjectOutputStream(OutputStream out)
```

Какие форматы сериализации существуют?

JSON
YAML
XML
BSON (Binary JSON)
Position Based Protocol

Стандартные потоки ввода/вывода?

Байтовые: java.io.InputStream, java.io.OutputStream;

Символьные: java.io.Reader, java.io.Writer;

Классы байтовых потоков ввода

Класс	***Назначение***
BufferedInputStream	Буферизированный входной поток.
BufferedOutputStream	Буферизированный выходной поток.
ByteArrayInputStream	Входной поток, читающий из массива байт.
ByteArrayOutputStream	Выходной поток, записывающий в массив байт.
DataInputStream	Входной поток, включающий методы для чтения стандартных типов данных Java.
DataOutputStream	Выходной поток, включающий методы для записи стандартных типов данных Java.
FileInputStream	Входной поток, читающий из файла.
FileOutputStream	Выходной поток, записывающий в файл.
FilterInputStream	Реализация *InputStream*.
FilterOutputStream	Реализация *OutputStream*.
ObjectInputStream	Входной поток для объектов.
ObjectOutputStream	Выходной поток для объектов.
OutputStream	Абстрактный класс, описывающий поток вывода.
PipedInputStream	Входной канал (например, межпрограммный).
PipedOutputStream	Выходной канал.

PrintStream	Выходной поток, включающий *print()* и *println()*.
PushbackInputStream	Входной поток, реализующий операцию pushback (вернуть назад).
RandomAccessFile	Позволяет перемещаться по файлу, читать из него или писать в него.
SequenceInputStream	Входной поток, представляющий собой комбинацию двух и более входных потоков, которые читаются совместно один после другого.

Классы символьных потоков

Класс	***Назначение***
BufferedReader	Буферизованный входной символьный поток.

BufferedWriter	Буферизованный выходной символьный поток.
CharArrayReader	Входной поток, который читает из символьного массива.
CharArrayWriter	Выходной поток, который пишет в символьный массив.
FileReader	Входной поток, читающий файл.
FileWriter	Выходной поток, пишущий в файл.
FilterReader	Фильтрующий читатель.
FilterWriter	Фильтрующий писатель.
InputStreamReader	Входной поток, транслирующий байты в символы.
LineNumberReader	Входной поток, подсчитывающий строки.
OutputStreamWriter	Выходной поток, транслирующий байты в символы.
PipedReader	Входной канал.
PipedWriter	Выходной канал.
PrintWriter	Выходной поток, включающий *print()* и *println()*.
PushbackReader	Входной поток, позволяющий возвращать символы обратно в поток.
Reader	Абстрактный класс, описывающий символьный ввод.
StringReader	Входной поток, читающий из строки.
StringWriter	Выходной поток, пишущий в строку.
Writer	Абстрактный класс, описывающий символьный вывод.

На каком паттерне основана иерархия потоков ввода/вывода?

Объекты классов Java, которые используются для ввода/вывода, для обеспечения необходимой функциональности наслаиваются друг на друга. Такая модель взаимодействия объектов поддерживается в паттерне «Декоратор». В этом паттерне при создании потока нужно использовать несколько объектов. Паттерн Декоратор динамически наделяет объект новыми возможностями и является альтернативой субклассированию в области расширения функциональности

Как работает метод read()?

int read() : **возвращает целочисленное представление следующего байта в потоке**

Как сериализовать статическое поле?

Создать отдельный статический метод для сериализации статического поля.

Можно ли сериализовать final поле?

Поля с модификатором *final* сериализуются как и обычные. За одним исключением – их невозможно десериализовать при использовании *Externalizable*, поскольку final-поля должны быть инициализированы в конструкторе, а после этого в *readExternal* изменить значение этого поля будет невозможно. Соответственно, если необходимо сериализовать объект с final-полем неоходимо использовать только стандартную сериализацию.

Клонирование Java

Иногда необходимо на основе существующего объекта создать второй такой же - то есть создать его клон. Это процесс в Java называется клонированием.

Для клонирования объекта в Java можно воспользоваться тремя способами:

1. Переопределение метода *clone()* и реализация интерфейса *Cloneable()*.
2. Использование конструктора копирования.
3. Использовать для клонирования механизм сериализации.

Что такое «каналы»?

Канал представляет собой открытое соединение с такими элементами, как аппаратное устройство, файл, сетевой сокет или программный компонент, и способное выполнять одну или более отдельных операций ввода/вывода, например чтение или запись.

Что делает метод read? Почему он возвращает int а не byte? Почему он не может возвращать byte?

InputStream.read () возвращает беззнаковый байт [0-255], и java не имеет этого типа, поэтому он принимается с int.

Диапазон байтов равен [-128,127], поэтому, если число, возвращаемое read (), находится в диапазоне [128,255], это означает отрицательное число, т.е

(byte)128=-128

(byte)129=-127

(byte)255=-1

Так что, если read () возвращает байт, тогда будут отрицательные числа. И «возврат -1 означает конец», эта информация не может быть выражена в байтах, поэтому вы должны использовать int.

Что вернет метод read(), если он считывает файл и ему встречается байт равный -1?

При -1 возвращает 255

Что такое клонирование? Как реализовано клонирование в Java?

Для реализации клонирования класс Person должен применить интерфейс **Cloneable**, который определяет метод clone. Реализация этого метода просто возвращает вызов метода clone для родительского класса - то есть класса Object с преобразованием к типу Person.

Кроме того, на случай если класс не поддерживает клонирование, метод должен выбрасывать исключение **CloneNotSupportedException**, что определяется с помощью оператора **throws**.

Поверхностное и глубокое клонирование.

Глубокое клонирование требует выполнения следующих правил:

1. Нет необходимости копировать отдельно примитивные данные;
2. Все классы-члены в оригинальном классе должны поддерживать клонирование. Для каждого члена класса должен вызываться super.clone() при переопределении метода clone();
3. Если какой-либо член класса не поддерживает клонирование, то в методе клонирования необходимо создать новый экземпляр этого класса

и скопировать каждый его член со всеми атрибутами в новый объект класса, по одному.

Поверхностное копирование копирует настолько малую часть информации, насколько это возможно. По умолчанию, клонирование в Java является поверхностным, т.е. Object class не знает о структуре класса, которого он копирует. При клонировании, JVM делает такие вещи:

1. Если класс имеет только члены примитивных типов, то будет создана совершенно новая копия объекта и возвращена ссылка на этот объект.
2. Если класс содержит не только члены примитивных типов, а и любого другого типа класса, тогда копируются ссылки на объекты этих классов. Следовательно, оба объекта будут иметь одинаковые ссылки.

Глубокое копирование дублирует все. Глубокое копирование — это две коллекции, в одну из которых дублируются все элементы оригинальной коллекции. Мы хотим сделать копию, при которой внесение изменений в любой элемент копии не затронет оригинальную коллекцию.

При неглубоком клонировании клонированный класс не копирует свои внутренние объекты, но при глубоком клонировании также копируются все внутренние объекты класса.

Чем отличается копирование от клонирования. Можно ли клонировать String, массив String.

- клонирование - создание чего-то нового на основе чего-то существующего.
- копирование - копирование из чего-то существующего во что-то другое (что также уже существует).

clone - это метод класса Object. Чтобы класс был "cloneable", он должен реализовать интерфейс marker Cloneable . класс String не

реализует этот интерфейс и не переопределяет метод клонирования, поэтому возникает ошибка.

Когда метод clone вызывается для массива, он возвращает ссылку на новый массив, который содержит (или ссылается) те же элементы, что и исходный массив.

Как преобразовать считанные байты в символы? Какой класс для этого используется?

```
while((i=fin.read())!=-1){
      System.out.print((char)i);
```

FileInputStream fin=new FileInputStream("C://SomeDir//notes.txt")

В чём отличие File от Path?

File — это класс, Path — это интерфейс.
Методы работы с ФС через объект Path при ошибках ввода-вывода бросают исключения; методы работы с File при ошибках возвращают false.
File — это старый способ доступа к файловой системе, Path — это новый рекомендуемый способ.
Path допускает работу с файлами на виртуальных файловых системах, а File нет.

Почему важно закрывать потоки? Какие потоки можно не закрывать (не вызывать метод close())?

При закрытии потока освобождаются все выделенные для него ресурсы, например, файл. В случае, если поток окажется не закрыт, может происходить утечка памяти.

Начиная с Java 7 можно использовать еще один способ, который автоматически вызывает метод close. Этот способ заключается в использовании конструкции **try-with-resources** (try-с-ресурсами). Данная конструкция работает с объектами, которые реализуют интерфейс AutoCloseable. Так как все классы

потоков реализуют интерфейс Closeable, который в свою очередь наследуется от AutoCloseable, то их также можно использовать в данной конструкции

Что делает метод available()?

int available() - возвращает количество байтов ввода, доступные в данный момент для чтения

Можно ли использовать flush() для небуферизированного потока и что будет. Гарантируется ли запись данных в файл при вызове flush()?

В переводе на русский означает, что flush() вызывает чистку буфера вызывая метод реализованный в самой ОС поверх которой работает JVM. При этом даже отсутствие буфера в самом потоке не означает, что не будет использоваться в качестве буфера кэш диска на уровне ОСи или даже ниже.
В общем, flush() является хорошим тоном для любого потока, поскольку, девелопер не знает есть ли буфер на уровне ОС (более того девелопер обычно даже и не знает что за ось будет использована).

вызов метода flush() гарантирует только то, что байты, ранее записанные в поток, передаются операционной системе для записи, но не гарантирует записи на диск

Что возвращает перегруженный read. Какое максимальное значение вернет?

Возвращает int. Максимальное значение - 255.

Для чего нужен Scanner? Что такое токен в Scanner? Отличие Scanner’a от BufferedReader’a? Есть ли у сканера буфер?

Класс Scanner используется для получения (считывания) данных введенных пользователем в виде String, byte, short, int, long, float, double.

Токен (или маркер) представляет собой серию цифровых или буквенно-цифровых символов, которая заканчивается разделителем. Разделителем может быть символ табуляции, возврат каретки (перевод строки или же просто 'Enter'), конец файла или пробел.
Scanner используется для разбора токенов из содержимого потока, а BufferedReader просто считывает поток и не выполняет никакого специального разбора.

Буфер 1 кбайт

Методы класса File? Как создать файл на компьютере с помощью java? Как удалить директорию с файлами. Что если в ней есть вложенные директории ? В чём отличие File от Path?

boolean createNewFile(): создает новый файл по пути, который передан в конструктор. В случае удачного создания возвращает true, иначе false

boolean delete(): удаляет каталог или файл по пути, который передан в конструктор. При удачном удалении возвращает true.

boolean exists(): проверяет, существует ли по указанному в конструкторе пути файл или каталог. И если файл или каталог существует, то возвращает true, иначе возвращает false

String getAbsolutePath(): возвращает абсолютный путь для пути, переданного в конструктор объекта

String getName(): возвращает краткое имя файла или каталога

String getParent(): возвращает имя родительского каталога

boolean isDirectory(): возвращает значение true, если по указанному пути располагается каталог

boolean isFile(): возвращает значение true, если по указанному пути находится файл

boolean isHidden(): возвращает значение true, если каталог или файл являются скрытыми

long length(): возвращает размер файла в байтах

long lastModified(): возвращает время последнего изменения файла или каталога. Значение представляет количество миллисекунд, прошедших с начала эпохи Unix

String[] list(): возвращает массив файлов и подкаталогов, которые находятся в определенном каталоге

File[] listFiles(): возвращает массив файлов и подкаталогов, которые находятся в определенном каталоге

boolean mkdir(): создает новый каталог и при удачном создании возвращает значение true

boolean renameTo(File dest): переименовывает файл или каталог

СОЗДАТЬ ФАЙЛ createNewFile()

Если хотите создать новый файл и одновременно записать какую-нибудь информацию в нем, можете использовать метод FileOutputStream.write. Этот метод автоматически создает новый файл и записывает в нем контент. Метод FileOutputStream используется для записи байтов в файл. Если хотите записать символьно-ориентированную информацию, будет лучше использовать FileWriter.

Используйте класс Java NIO Files для создания нового файла и записи информации в нем. Этот Java класс представляет метод write(Path path, byte[] bytes, OpenOption... options), который записывает байты к файлу по указанному пути.

Java не может удалять папки с данными. Перед удалением папки необходимо удалить все файлы.

java.nio.file.Path является частью более современной java.nio.file библиотеки и делает все java.io.File возможное, но в целом лучше и больше.

Что будет при сериализации объекта у которого есть поле и оно не Serializable?

1) В таком случае код скомпилируется.
2) Но в рантайме при попытке сериализации , когда мы дойдем до этого поля и объекта, мы получим NotSerializableException.

Externalizable vs Serializable?

При записи Serializable класса весь контроль над сериализацией достается JVM. С помощью определения специальных методов можно кастомизировать его части. Метод readObject при этом обычно начинается с вызова стандартной части сериализации - ObjectInputStream.defaultReadObject().Интерфейс Externalizable расширяет Serializable и добавляет методы записи и чтения writeExternal и readExternal. Входной и выходной потоки-аргументы в них представлены более абстрактно чем в специальных методах - интерфейсами ObjectInput и ObjectOutput.Этот интерфейс позволяет реализовать полностью свой механизм сериализации, стандартно запишется только идентификатор класса. Никакой автоматической работы с классом-родителем также не предусмотрено. Методы readObject и writeObject игнорируются. Ключевое слово transient эффекта на Externalizable не имеет.Externalizable объект в отличие от Serializable десерализуется не в обход конструктора, так что должен иметь конструктор без аргументов.

Какие интерфейсы реализует InputStream/ OutputStream/ Reader/ Writer?

InputStream - Closeable, AutoCloseable
OutputStream - Closeable, Flushable, AutoCloseable
Reader - Closeable, AutoCloseable, Readable
Writer - Closeable, Flushable, Appendable, AutoCloseable