**Какой пакет предоставляет средства ввода-вывода в Java?**

Базовым пакетом для обеспечения операций ввода/вывода в Java является java.io. В нём определяется несколько абстрактных классов, которые затем расширяются, и на их основе создаются некоторые полезные типы.

**Что такое поток?**

Поток — это абстрактная сущность (т. е. сущность, которая допускает

переопределение в конкретном контексте), представляющая устройство

ввода-вывода, которая выдает и получает информацию из программы.

**Что такое буферизация?**

Буфер — область памяти, выделяемая для хранения символов перед тем,

как они будут использованы программой.

Буферизация — метод организации обмена, в частности, ввода и вывода данных в [компьютерах](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) и других [вычислительных устройствах](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE), который подразумевает использование [буфера](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D1%84%D0%B5%D1%80_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) для временного хранения данных.

**На базе каких абстрактных классов построена система ввода-вывода Java?**

InputStream, OutputStream, Reader, Writer

**Как реализуется доступ к произвольной позиции в файле?**

public void seek(long pos) - Устанавливает файловый указатель в заданную позицию (в байтах). Следующий считанный или записанный байт будет иметь смещение pos.

**Для чего служит метод close() и когда его нужно использовать?**

public void close() throws IOException

Закрывает поток. Метод должен вызываться для освобождения любых ресурсов, связанных с потоком. Нужно вызывать после завершения всех работ с потоком.

**Как осуществить перекодировку содержимого файлов?**

Пакет java.io предоставляет классы, которые позволяют выполнять конвертацию между

символьными потоками Unicode и байтовыми потоками не-Unicode текста. При помощи класса

InputStreamReader существует возможность преобразовывать байтовые потоки в символьные потоки. Для перевода символьных потоков в байтовые потоки используется класс OutputStreamWriter.

При создании объектов InputStreamReader и OutputStreamWriter, задается байтовая

кодировка, в которую необходимо выполнить преобразование. Например, чтобы перевести текстовый файл в кодировке UTF-8 в Unicode, необходимо сделать следующее:

FileInputStream fis = new FileInputStream("test.txt");

InputStreamReader isr = new InputStreamReader(fis, "UTF8");

Если не указать идентификатор кодировки, InputStreamReader и OutputStreamWriter будут обращаться к кодировке по умолчанию. Вы можете определить кодировку, используемую InputStreamReader или OutputStreamWriter, вызвав метод getEncoding, таким образом:

InputStreamReader defaultReader = new InputStreamReader(fis);

String defaultEncoding = defaultReader.getEncoding();

**Как в Java осуществляется работа с конфигурационными файлами?**

В Java можно использовать как распространенный формат конфигурационного файла ini (\*.ini), так и собственный формат файла свойств properties (\*.properties).

Конфигурационные файлы это текстовые файлы, содержащие пары: ключ=значение

Для пояснения содержимого конфигурационных файлов используются комментарии. Для файлов \*.ini комментарии предваряются символами # или ;, а для файлов \*.properties - # или !.

Для работы с конфигурационными файлами обоих типов в Java можно использовать класс java.util.Properties, метод load которого загружает пары ключ=значение из конфигурационного файла в ассоциативный массив. При этом и ключи и значения являются строками. Для работы с ключами и значениями как с другими типами данных требуется использовать операции приведения типов.