**Вопросы по теме Reader:**

1. Расскажите про класс Reader и его подклассы.
2. Для чего нужен класс PipedReader?
3. Расскажите про класс InputStreamReader?
4. Для чего нужен класс FileReader?
5. Что вы знаете о классе BufferedReader?

**Reader:**

**Reader** — абстрактный класс, описывающий поток ввода, который работает с символами.

Имеет два конструктора:

*protected Object lock;*

*protected Reader() { this.lock = this; } –* синхронизация на объекте lock

*protected Reader​(Object lock) { this.lock = lock; } –* синхронизация на переданном объекте

Основные методы класса (их имеют все классы наследники):

*void* *close()* – закрывает поток и освобождает ресурсы, связанные с ним;

*int read()* – считывает текущий символ из потока ввода и возвращает его или *-1* если данных в потоке больше нет;

*int read​(byte[] cbuf)* – считывает в буфер количество символов равное *cbuf.length*, возвращает количество прочитанных символов или *-1* если данных в потоке больше нет;

*int read​(byte[]cbuf, int off, int len)* – считывает в буфер *cbuf* количество символов *len* начиная с позиции *off*, первый прочитанный символ сохранится в *cbuf [off]*, возвращает количество прочитанных символов или *-1* если данных в потоке больше нет;

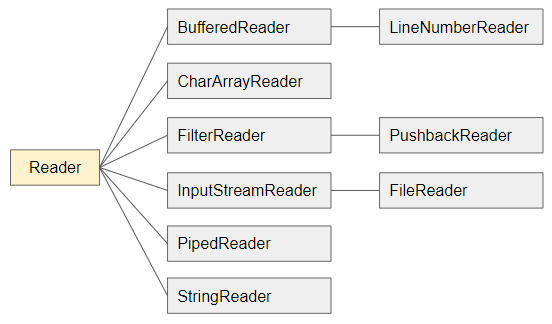
*boolean ready()* – сообщает, готов ли поток для чтения;

*long skip​(long n)* – пропускает первые *n* символов из входного потока, возвращает фактическое количество пропущенных символов;

*void mark​(int readlimit)* – отмечает текущую позицию в этом входном потоке, *readlimit* - максимальное количество символов, которое может быть прочитано до того, как позиция метки станет недействительной;

*void reset()* – перемещает этот поток в положение последнего вызова метода mark();

*boolean markSupported()* – проверяет, поддерживает ли входной поток методы mark и reset.



**Наследники класса Reader:**

**CharArrayReader** — класс, использующий в качестве источника данных массив символов.

Имеет конструкторы:

*CharArrayReader(char[] buf)*

*CharArrayReader(char[] buf, int offset, int length)*

buf – массив символов, offset – с какого символа будем считывать, length – количество считываемых символов.

Пример использования:

char[] array = new char[]{'a', 'b', 'c', '4', '5'};  
try(CharArrayReader car = new CharArrayReader(array)) {  
 int data;  
 while((data = car.read()) != -1) {  
 char ch = (char) data;  
 System.*out*.println(ch);  
 }  
}

Особенности:

Считывает символы из массива символов.

**InputStreamReader** — класс, считывающий байты и декодирующий их в символы используя указанный набор символов (кодировку).

Имеет конструкторы:

*InputStreamReader​(InputStream in)* //использует набор символов по умолчанию

*InputStreamReader​(InputStream in, String charsetName)*

*InputStreamReader​(InputStream in, Charset cs)*

*InputStreamReader​(InputStream in, CharsetDecoder dec)*

*charsetName* – именованный набор символов (должен поддерживаться абстрактным классом Charset), *cs* – набор символов, *dec* – декодер набора символов.

Пример использования:

byte[] array = new byte[]{72, 101, 108, 108, 111};  
InputStream is = new ByteArrayInputStream(array);  
try(InputStreamReader isr = new InputStreamReader(is, StandardCharsets.*UTF\_8*)) {  
 int data;  
 while ((data = isr.read()) != -1) {  
 System.*out*.print((char)data);  
 }  
}

Особенности:

Чтобы обеспечить эффективное преобразование байтов в символы при вызове одного из методов read(), из базового потока может быть прочитано больше байтов, чем необходимо для выполнения текущей операции чтения. Для максимальной эффективности InputStreamReader лучше оборачивать BufferedReader. Имеет метод getEncoding(), который возвращает имя кодировки, используемое данным потоком.

**FileReader** — класс, предназначенный для чтения символьных файлов.

Имеет конструкторы:

*FileReader(File file)*

*FileReader(FileDescriptor fd)*

*FileReader(String fileName)*

*fdObj* – экземпляр класса FileDescriptor (файловый дескриптор позволяет получить доступ к файлу даже если этот файл был переименован, удален, закрыт к нему доступ).

Пример использования:

try (FileReader fr = new FileReader("D:\\temp.txt")) {  
 int n;  
 while ((n = fr.read()) != -1) {  
 System.*out*.print((char) n);  
 }  
}

Особенности:

Конструкторы этого класса предполагают, что кодировка символов и размер символьного буфера по умолчанию являются подходящими. Чтобы указать эти значения самостоятельно, нужно создать InputStreamReader в FileInputStream. Наследуется от класса InputStreamReader.

**FilterReader** — класс, предназначенный для фильтрации, модификации или предоставления дополнительных функций для входного потока. Работает почти так же, как класс Reader. Он переопределяет все методы Reader, а эти переопределенные методы просто передают все запросы вложенному входному потоку.

protected Reader in;

protected FilterReader(Reader in) {

super(in);  
 this.in = in;  
}

*in* – входной поток для фильтрации.

public int read() throws IOException {  
 return in.read();  
}

**PushbackReader** — класс, который дает возможность «отодвинуть» или «непрочитать» символы, сохраняя вытесненные символы во внутреннем буфере с помощью метода unread(). Наследуется от FilterReader.

**BufferedReader** — читает символы из входного потока и буферизует их, чтобы обеспечить эффективное чтение символов, массивов и строк.

Имеет конструкторы:

*BufferedReader(Reader in)*

*BufferedReader(Reader in, int sz)*

*sz* — размер буфера в символах.

Пример использования:

try(BufferedReader reader = new BufferedReader(new InputStreamReader(System.*in*))) {  
 String line = reader.readLine();  
 System.*out*.println(line);  
}

Особенности:

Наследуется от Reader. Предназначен для оптимизации и ускорения процесса считывания информации за счет ее передачи порциями, равными размеру буфера. Размер буфера по умолчанию — 8192 символа.

**LineNumberReader** — буферизованный поток ввода символов, который отслеживает номера строк. Этот класс определяет методы setLineNumber(int) и getLineNumber() для установки и получения текущего номера строки соответственно. Наследуется от класса BufferedReader.

**StringReader** — класс, использующий в качестве источника данных строку.

**PipedReader** — класс, предназначенный для связи отдельных потоков друг с другом внутри одной JVM. Обычно связывается PipedReader и PipedWriter и используется при многопоточном программировании. Каждый раз, когда данные записываются в PipedWriter, они автоматически появляются в PipedReader.