Вопросы по теме Writer:

1. Расскажите про класс Writer и его подклассы.
2. Какая разница между PrintWrite и PrintStream?
3. Расскажите про класс OutputStreamWriter?
4. Для чего нужен класс FileWriter?
5. Что вы знаете о классе BufferedWriter?

Writer:

Writer — абстрактный класс, описывающий поток вывода, который работает с символами.

Имеет два конструктора:

protected Object lock;

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

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

Основные методы класса (их имеют все классы наследники):

Writer append(char c) – добавляет указанный символ к данному потоку вывода;

Writer append(CharSequence csq) – добавляет указанную последовательность символов к данному потоку вывода;

Writer append(CharSequence csq, int start, int end) – добавляет часть указанной последовательности символов к данному потоку вывода;

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

void write(int c) – записывает указанный символ в поток вывода;

void write(byte[] cbuf) – записывает количество символов равное *b.length* из указанного массива в поток вывода;

void write(byte[] cbuf, int off, int len) – записывает из массива *cbuf* в поток вывода символы, начиная с позиции *off*, количество символов *len*;

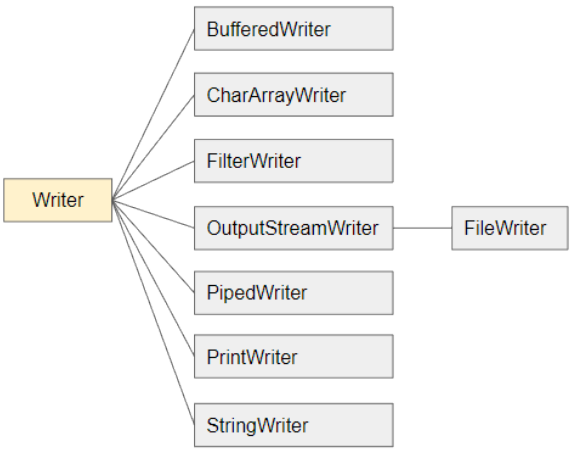
void write(String str) – записывает строку *str* в поток вывода;

void write(String str, int off, int len) – записывает часть строки *str* в поток вывода, начиная с позиции *off*, количество символов *len*;

long flush() – сбрасывает этот поток вывода и принудительно записывает любые буферизованные выходные символы;

static Writer nullWriter() – возвращает новый Writer, который отбрасывает все символы.

Виталий Жуков
unicraft1989@gmail.com
Ссылка на курс Stepik



Наследники класса Writer:

CharArrayWriter — класс, записывающий символы в массив символов.

Имеет конструкторы:

CharArrayWriter()

CharArrayWriter(int initialSize)

initialSize – емкость буфера в символах (по умолчанию size = 32).

Пример использования:

```
try (CharArrayWriter caw = new CharArrayWriter()) {
    caw.append('A').append("B");
    caw.write('C');
    caw.write("D");
    System.out.println(caw.toCharArray()); //ABCD
}
```

Особенности:

Буфер автоматически увеличивается по мере записи в него данных (создается новый массив большего размера и копируются в него текущие данные). Данные можно получить с помощью методов toByteArray() и toString(). Можно не вызывать метод close(), в этом классе он не имеет эффекта.

OutputStreamWriter — класс, который записывает символы в поток вывода предварительно кодируя их в байты, используя указанный набор символов (кодировку).

Имеет конструкторы:

OutputStreamWriter(OutputStream out) //использует набор символов по умолчанию

OutputStreamWriter(OutputStream out, String charsetName)

OutputStreamWriter(OutputStream out, Charset cs)

OutputStreamWriter(OutputStream out, CharsetEncoder enc)

Виталий Жуков
unicraft1989@gmail.com
Ссылка на курс Stepik

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

Пример использования:

```
try (OutputStreamWriter osr = new OutputStreamWriter(System.out))
{
    String s = "Ада Лавлейс - первый в мире программист";
    osr.write(s);
}
```

Особенности:

Каждый вызов метода write() приводит к вызову преобразователя кодировки для данного символа. Для максимальной эффективности OutputStreamWriter лучше оборачивать BufferedWriter. Имеет метод getEncoding(), который возвращает имя кодировки, используемое данным потоком.

FileWriter — класс, предназначенный для записи символов в файл.

Имеет конструкторы:

FileWriter(File file) //.., boolean append; .., Charset charset; .., Charset charset, boolean append

FileWriter(String fileName) //.., boolean append; .., Charset charset; .., Charset charset, boolean append

FileWriter(FileDescriptor fd)

fd – экземпляр класса FileDescriptor (файловый дескриптор позволяет получить доступ к файлу даже если этот файл был переименован, удален, закрыт к нему доступ); append – если True, то данные записываются в конец файла; *charset* – кодировка.

Пример использования:

```
try (FileWriter fr = new FileWriter("D:\\temp.txt")) {
    String s = "Ада Лавлейс - первый в мире программист";
    fr.write(s);
}
```

Особенности:

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

Доступен ли файл для записи (или есть возможность его создать) зависит от платформы на которой используется программа. Некоторые платформы позволяют одновременно открывать файл для записи только одному FileWriter (или другому объекту записи файла). В таких ситуациях конструкторы этого класса завершатся ошибкой, если соответствующий файл уже открыт.

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

```
protected Writer out;
protected FilterWriter(Writer out) {
    super(out);
    this.out = out;
}
```

out – поток вывода для фильтрации.

```
public void write(int c) throws IOException {
    out.write(c);
}
```

BufferedWriter— записывает символы в поток вывода и буферизует их, чтобы обеспечить эффективное чтение символов, массивов и строк.

Имеет конструкторы:

BufferedWriter(Writer out)

BufferedWriter(Writer out, int sz)

sz — размер буфера в символах.

Пример использования:

```
try (BufferedWriter writer =
new BufferedWriter(new OutputStreamWriter(System.out))){
    String s = "Ада Лавлейс - первый в мире программист";
    writer.write(s);
}
```

Особенности:

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

Имеет метод *newLine()*, который предназначен для разделения строк. Не все платформы используют символ новой строки *'\n'* для завершения строки. Поэтому вызов этого метода для завершения каждой строки вывода предпочтительнее, чем запись символа новой строки напрямую.

StringWriter — класс, который записывает свой вывод в строковый буфер, который затем можно использовать для создания строки. Метод *getBuffer()* возвращает строковый буфер. Метод *close()* в данном классе не имеет эффекта.

PipedWriter — 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.

PrintWriter — класс, предназначенный для вывода информации на консоль, в файл или в любой другой поток вывода. Этот класс реализует все методы печати класса PrintStream.

В отличие от PrintStream, если включена автоматическая очистка, она будет выполняться только тогда, когда вызывается один из методов println , printf или format , а не всякий раз, когда выводится символ новой строки. Эти методы используют собственное понятие разделителя строк, а не символ новой строки.

PrintStream никогда не генерирует IOException, вместо этого исключения управляются с помощью переменной *private boolean trouble = false* – внутренний флаг, сигнализирующий о наличии (true) или отсутствии (false) ошибки. Для управления этим флагом используются методы checkError() и clearError().

Виталий Жуков
unicraft1989@gmail.com
Ссылка на курс Stepik