Вопросы по теме OutputStream:

1. Расскажите про класс OutputStream и его подклассы.
2. Для чего нужен класс BufferedOutputStream?
3. Расскажите про класс FileOutputStream.
4. Для чего нужен класс ObjectOutputStream?
5. Что вы знаете о классе PrintStream?

OutputStream:

OutputStream – абстрактный класс, описывающий поток вывода, который работает с байтами.

Основные методы класса (их имеют все классы наследники):

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

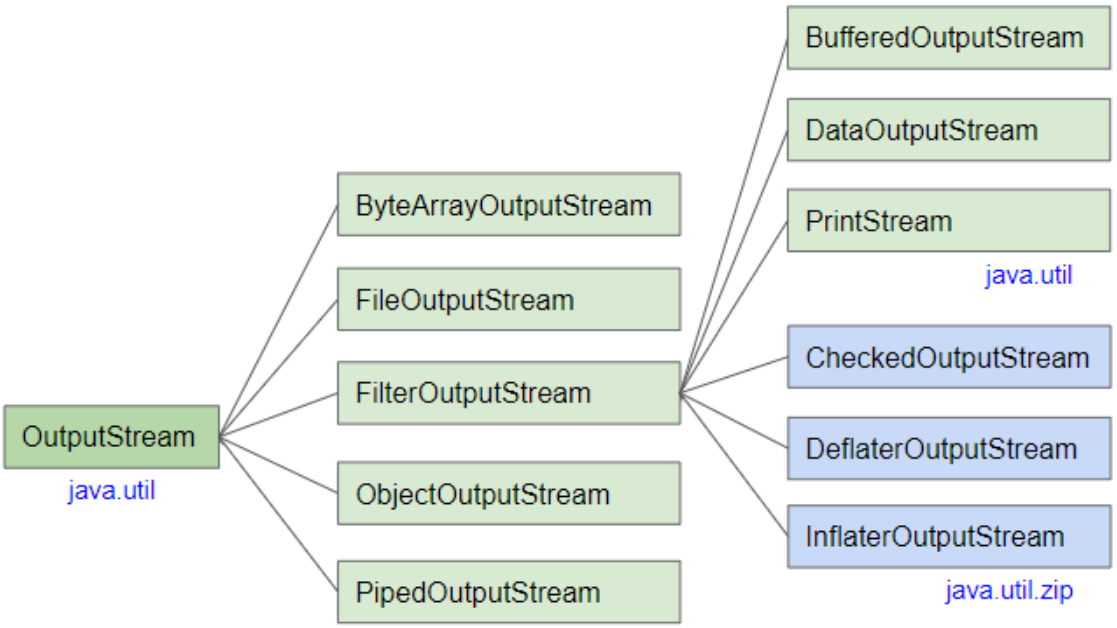
void write(int b) – записывает указанный байт в поток вывода;

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

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

void flush() – сбрасывает этот поток вывода и принудительно записывает любые буферизованные выходные байты;

static OutputStream nullOutputStream() – возвращает новый OutputStream, который отбрасывает все байты.



Виталий Жуков
unicraft1989@gmail.com
Ссылка на курс Stepik

Наследники класса OutputStream:

ByteArrayOutputStream – класс, записывающий байты в массив байтов.

Имеет конструкторы:

ByteArrayInputStream()

ByteArrayInputStream(int size)

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

Пример использования:

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
byte[] data = new byte[] {1, 2, 3, 4, 5};
baos.write(data);
for (byte b : baos.toByteArray()) System.out.print(b + " ");
```

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 toCharArray() и toString().

FileOutputStream – класс, записывающий байты в файл.

Имеет конструкторы:

FileOutputStream(File file / String name)

FileOutputStream(File file / String name, boolean append)

FileOutputStream(FileDescriptor fdObj)

append – если true, то байты будут записываться в конец файла, а не в начало, *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 (FileOutputStream fos = new FileOutputStream("D:\\temp.txt")) {
    byte[] data = new byte[] {72, 101, 108, 108, 111};
    fos.write(data);
}
```

Особенности:

Предназначен для записи необработанных байт (например, запись изображения), для записи символов лучше использовать FileWriter.

Виталий Жуков
unicraft1989@gmail.com
Ссылка на курс Stepik

FilterOutputStream – класс, предназначенный для фильтрации, модификации или предоставления дополнительных функций для выходного потока. Работает почти так же, как класс OutputStream. Он переопределяет все методы OutputStream, а эти переопределенные методы просто передают все запросы вложенному выходному потоку.

```
protected OutputStream out;

protected FilterOutputStream(OutputStream out) {
    this.out = out;
}
```

out – выходной поток для фильтрации.

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

DataOutputStream – класс, записывающий примитивные типы данных в поток вывода.

Имеет конструкторы:

DataOutputStream(OutputStream out)

Пример использования:

```
try(DataOutputStream dos = new DataOutputStream(new
FileOutputStream("D:\\temp.txt"))){
    dos.writeInt(111);
    dos.writeBoolean(true);
}
```

Особенности:

Наследуется от FilterOutputStream. Для записи каждого примитивного типа существует свой метод (writeInt(), writeChar() и т.д.).

BufferedOutputStream - накапливает выводимые данные в специальном буфере без постоянного обращения к устройству вывода. Когда буфер заполнится, происходит запись данных.

Имеет конструкторы:

BufferedOutputStream(OutputStream outputStream)

BufferedOutputStream(OutputStream outputStream, int size)

size — размер буфера в байтах.

Пример использования:

```
try(BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream("D:\\temp.txt"))){
    byte[] data = new byte[] {1, 2, 3, 4, 5};
    bos.write(data);
}
```

Особенности:

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

ObjecOutputStream — класс, предназначенный для записи сериализованных данных в указанный поток.

Имеет конструкторы:

ObjectOutputStream() - для классов, переопределяющих ObjectOutputStream

ObjectOutputStream(OutputStream out)

Пример использования:

```
try(ObjectOutputStream ois = new ObjectOutputStream(new
FileOutputStream("D:\\temp.txt")))
{
    MyObject myObject = new MyObject("type", 5);
    ois.writeObject(myObject);
}
```

Особенности:

Для записи примитивных типов данных используются методы writeInt(), writeChar() и т. д., для записи объекта используется метод writeObject().

PipedOutputStream – класс, предназначенный для связи отдельных потоков друг с другом внутри одной JVM. Обычно связывается PipedInputStream и PipedOutputStream и используются при многопоточном программировании. Каждый раз, когда данные записываются в PipedOutputStream, они автоматически появляются в PipedInputStream.

CheckedOutputStream – класс, позволяющий использовать контрольную сумму для проверки целостности записываемых данных.

PrintStream – класс, предназначенный для вывода информации на консоль. Когда мы используем *System.out.print()*, то используем класс PrintStream, т.к. переменная out класса System представляет собой объект класса PrintStream и метод этого класса print(). Также этот класс можно использовать для записи информации в потоки вывода:

PrintStream(OutputStream outputStream / File outputFile / String outputFileName)

В некоторых конструкторах PrintStream можно включить режим автоматической очистки буфера вывода каждый раз, когда вызывается метод println()или записывается символ новой строки или байт '\n' (для этого переменная autoFlush = True).

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().

Все символы, напечатанные а PrintStream, преобразуются в байты с использованием кодировки символов по умолчанию для платформы.

Виталий Жуков
unicraft1989@gmail.com
Ссылка на курс Stepik