**Потоки ввода/вывода**

[Java Core](https://javarush.ru/quests/QUEST_JAVA_CORE)

[Уровень 8](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_CORE&level=8), Лекция 1

— Привет, Амиго! Сегодня мы будет знакомиться с**потоками ввода-вывода**. Пару дней назад мы немного цепляли данную тему, но сегодня пройдемся по ней основательно. Потоки ввода-вывода делятся на 4 категории:

1) Потоки делятся по направлению: **потоки ввода** и **потоки вывода**

2) Потоки делятся по типу данных: работают с **байтами** или работают с **символами**.

Таблица:

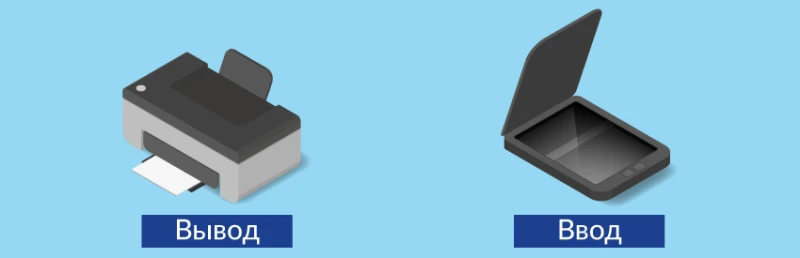
|  |  |  |
| --- | --- | --- |
|  | **Поток ввода** | **Поток вывода** |
| **Работает с байтами** | InputStream | OutputStream |
| **Работает с символами** | Reader | Writer |

Если объект реализует интерфейс **InputStream**, значит, он поддерживает последовательное чтение из него байт (byte).

Если объект реализует интерфейс **OutputStream**, значит, он поддерживает последовательную запись в него байт (byte).

Если объект реализует интерфейс **Reader**, значит, он поддерживает последовательное чтение из него символов (char).

Если объект реализует интерфейс **Writer**, значит, он поддерживает последовательную запись в него символов (char).

[](https://cdn.javarush.ru/images/article/ffb432fb-55e6-4d7f-afae-2961dec892c1/original.jpeg)

Поток вывода напоминает принтер. На принтер мы можем выводить документы. В поток вывода мы можем выводить данные.

Тогда поток ввода можно сравнить со сканером, ну или с розеткой. С помощью сканера мы можем ввести документы к себе в компьютер. Также мы можем подключится к розетке и получать из нее электричество. Из потока ввода мы можем получать данные.

— А где они используются?

— Эти классы используются в Java повсеместно. Известный нам **System.in** – это статическая переменная по имени in типа **InputStream** в классе **System**.

— Надо же! Оказывается, все это время я пользовался потоком InputStream и не знал об этом. System.out – тоже поток?

— Да, **System.out** – это статическая переменная по имени out типа **PrintStream** (наследник OutputStream) в классе **System**.

— Т.е. я все время пользовался потоками и даже не подозревал об этом?

— Да, и это говорит лишь о том, насколько такие потоки удобны. Просто берешь и пользуешься.

— Хотя этого нельзя сказать про System.in. К нему постоянно приходилось добавлять BufferedReader и InputStreamReader.

— Да, это так. Но на это тоже были свои причины.

Видишь ли, типов данных очень много, как и способов работы с ними. Поэтому количество стандартных классов ввода-вывода очень быстро росло, хоть они и делали все почти то же самое. Чтобы избежать такой сложности, разработчики Java применили принцип абстракции и разделили классы на много маленьких частей.

Зато их можно соединить последовательно и получить очень сложную функциональность, если она тебе понадобилась. Смотри пример:

Вывод строки на консоль

System.out.println("Hello");

Сохранили поток вывода на консоль в отдельную переменную.  
Выводим в поток строку.

PrintStream console = System.out;

console.println("Hello");

Создали динамический (растягивающийся) массив байт в памяти.  
Связали его с новым потоком вывода – объектов PrintStream  
Выводим в поток строку.

ByteArrayOutputStream stream = new ByteArrayOutputStream();

PrintStream console = new PrintStream(stream);

console.println("Hello");

— Действительно, чем-то похоже на конструктор Lego. Только непонятно, что весь этот код делает.

— Пусть это тебя не беспокоит сейчас. Всему свое время.

Хочу, чтобы ты запомнил вот что: если класс реализует интерфейс OutputStream – он позволяет записывать в него байты. Почти так же, как ты выводишь данные на консоль. Что он будет с этими данными делать – его задача. В «конструкторе» важно не назначение отдельного элемента, а насколько классные вещи мы можем собрать, благодаря многообразию существующих элементов.

— Хорошо. Тогда с чего мы начнем?