### Чтение и запись (ввод/вывод)

##### Консольная

Для чтения с консоли используется класс Scanner пакета util, импортируется

Для использования сканнера создается объект класса

Scanner scanner = new Scanner(System.in)

После завершения работы со сканером для оптимизации приложения сканнер закрывают:

scanner.close()

И записывает ввод в назначенную переменную методом (ждет ввода с консоли)

int a = scanner.nextInt()

.next() - считает символьную строку до следующего пробела

.nextInt() - чтение целого числа

.nextLine() - чтение введенной строки

.nextDouble() - чтение дроби

.nextBoolean() - чтоние условия

.hasNextInt() - проверяет входную строку на ЧИСЛА (true/false)

При считывании с консоли необходимо чистить поток, если оператор не взял значение потока

##### Файловая

Классы FileWriter и FileReader и IOException импорты паката .io, импортируются

Операции записи и чтения файлов являются потенциально опасными и требуют обязательно обрабатываются проверяемыми исключениями try/catch

Для открытия потока записи в файл или его чтения необходимо создать объект класса с именем файла:

FileWriter file = new FileWriter("text.txt") {file.write("Hello\nJava");}

FileReader file = new FileReader("text.txt") {int b; while((b=file.read()) != -1) sout(()char)b);}

После закрывается поток на чтение/запись файла - file.close();

После 7 java появилась возможность переносить открывание потока в try ресурсами, в следствие чего не закрывать поток самому т.к. по завершению блока ресурс автоматически закрывается ()автозакрытие)

Поток, открытый на запись, пересоздаст файл

.read - метод, который читает побайтно и при повторной записи файл перезаписывается

Буферизованное чтение проходит намного быстрее небуферизованного

Небуферизованное чтение: FileInputStream reader = new FileInputStream("qqq.txt")

Буферизованное чтение:

BufferedInputStream reader = new BufferedInputStream(new FileInputStream("qqq.txt"))