**Как читать с файла в Java с помощью BufferedReader Scanner Files и FileReader**

[06/10/2015](http://prologistic.com.ua/kak-chitat-s-fajla-v-java-s-pomoshh-yu-bufferedreader-scanner-files-i-filereader.html)  [prologistic](http://prologistic.com.ua/author/prologistic)  [Java](http://prologistic.com.ua/category/java),[Java IO](http://prologistic.com.ua/category/java/java-io)

При работе с файлами в Java очень часто нужно выполнять операции чтения данных. Именно поэтому сегодня мы рассмотрим различные способы прочитать файл в Java.

**Читаем информацию с файла. Теория**

1. В Java есть класс java.nio.file.Files,**который позволяет прочитать все содержимое файла в массив байтов или считать все строки в список.**Класс Filesбыл введен в Java 7 и отлично подходит, если нам нужно загрузить все содержимое файла. Следует отметить важный момент: **желательно использовать этот метод только тогда, когда работаем с небольшими файлами**, так как все содержимое это файла будет в памяти.
2. Класс java.io.FileReader. Мы можем использовать FileReader, чтобы получить BufferedReader, а затем считывать файлы построчно. FileReader **не поддерживает кодировку** и работает с кодировкой системы по умолчанию, так что это **не очень эффективный способ чтения файла в Java**, но все же имеет право на жизнь.
3. Класс java.io.BufferedReader**отлично подходит для чтения файла построчно и дальнейшей обработки считанной информации**. Он очень хорошо справляет с обработкой больших файлов и поддерживает кодировку файла. BufferedReaderявляется синхронизируемым, а поэтому значит, что операции чтения с помощью BufferedReader можно безопасно выполнять из нескольких потоков. Размер буфера BufferedReader по умолчанию составляет 8 Кбайт.
4. Класс java.util.Scanner **отлично подходит для чтения файла построчно**. Сканер разбивает входные данные на лексемы используя разделитель, который по умолчанию соответствует **пробелу**. Полученные лексемы могут быть затем преобразованы в значения разных типов с использованием различных методов. Сканер не синхронизирован, следовательно, не потокобезопасный.

Ниже представлен класс на Java для чтения содержимого файла с помощью Scanner, Files, BufferedReader с поддержкой кодировки и FileReader:

|  |
| --- |
| package ua.com.prologistic;    import java.io.BufferedReader;  import java.io.File;  import java.io.FileInputStream;  import java.io.FileReader;  import java.io.IOException;  import java.io.InputStreamReader;  import java.nio.charset.Charset;  import java.nio.charset.StandardCharsets;  import java.nio.file.Files;  import java.nio.file.Path;  import java.nio.file.Paths;  import java.util.List;  import java.util.Scanner;    public class JavaReadFile {        public static void main(String[] args) throws IOException {          String fileName = "/Users/prologistic/source.txt";            //Используем класс Files для обработки небольших файлов, получаем содержимое файла файла          readUsingFiles(fileName);            // используем класс Scanner для больших файлов, читаем построчно          readUsingScanner(fileName);            // построчно читаем с файла с помощью BufferedReader          readUsingBufferedReader(fileName);          readUsingBufferedReaderJava7(fileName, StandardCharsets.UTF\_8);          readUsingBufferedReader(fileName, StandardCharsets.UTF\_8);            // читаем с помощью FileReader без поддержки кодировки          readUsingFileReader(fileName);      }        private static void readUsingFileReader(String fileName) throws IOException {          File file = new File(fileName);          FileReader fr = new FileReader(file);          BufferedReader br = new BufferedReader(fr);          String line;          while((line = br.readLine()) != null){              //обрабатываем считанную строку - пишем ее в консоль              System.out.println(line);          }          br.close();          fr.close();        }        private static void readUsingBufferedReader(String fileName, Charset cs) throws IOException {          File file = new File(fileName);          FileInputStream fis = new FileInputStream(file);          InputStreamReader isr = new InputStreamReader(fis, cs);          BufferedReader br = new BufferedReader(isr);          String line;          while((line = br.readLine()) != null){              System.out.println(line);          }          br.close();        }        private static void readUsingBufferedReaderJava7(String fileName, Charset cs) throws IOException {          Path path = Paths.get(fileName);          BufferedReader br = Files.newBufferedReader(path, cs);          String line;          while((line = br.readLine()) != null){              System.out.println(line);          }          br.close();      }        private static void readUsingBufferedReader(String fileName) throws IOException {          File file = new File(fileName);          FileReader fr = new FileReader(file);          BufferedReader br = new BufferedReader(fr);          String line;          while((line = br.readLine()) != null){              System.out.println(line);          }          br.close();          fr.close();      }        private static void readUsingScanner(String fileName) throws IOException {          Path path = Paths.get(fileName);          Scanner scanner = new Scanner(path);          //читаем построчно          while(scanner.hasNextLine()){              String line = scanner.nextLine();          }      }        private static void readUsingFiles(String fileName) throws IOException {          Path path = Paths.get(fileName);          //считываем содержимое файла в массив байт          byte[] bytes = Files.readAllBytes(path);          //считываем содержимое файла в список строк          List<String> allLines = Files.readAllLines(path, StandardCharsets.UTF\_8);      }    } |

**Какой метод считывания с файла лучше использовать?**

Выбор между Scanner, BufferedReader или Files зависит от требований вашего проекта.

* Если вам нужно считывать с файла, опираясь на разделитель, то желательно использовать класс Scanner.
* Если файл небольшой, то используем класс Files.
* Если нужно считать информация с больших файлов, то отлично подойдетBufferedReader.