**Java работа с файлами**

Каждое приложение, программа имеют возможности записи чего-либо в файл. В данном уроке я продемонстрирую как можно записать какие-то данные в файл, а потом считать их оттуда.

[Скачать](http://devcolibri.com/download/1141/WorkWithFileInJava.zip)

**Какие проблемы мы будем решать в этом уроке?**

[1. Как записывать в файл?](http://devcolibri.com/1141#j1)

[2. Как читать файл?](http://devcolibri.com/1141#j2)

[3. Как обновить файл?](http://devcolibri.com/1141#j3)

[4. Как удалить файл?](http://devcolibri.com/1141#j4)

**Подготовительные работы**

Создадим простой проект, не обязательно Maven проект, так как нам не потребуется не каких дополнительных библиотек.

После того как вы создали проект, создайте класс **WorkInFile.java**и напишите туда стандартную конструкцию **main:**

|  |
| --- |
| public static void main(String[] args){     //тут будем вызывать наши методы  } |

Теперь создадим класс который будет иметь методы для работы с файлами, а назовем его **FileWorker.java**все методы в нем которые не есть **private**будут [статическими](http://devcolibri.com/1143) для того чтобы мы получали к ним доступ без экземпляра этого класса.

**Как записывать в файл?**

В классе**FileWorker.java**создадим [статический метод](http://devcolibri.com/1143) который будет осуществлять запись в файл и назовем этот метод**write**(String *text*; String *nameFile*):

|  |
| --- |
| public static void write(String fileName, String text) {      //Определяем файл      File file = new File(fileName);        try {          //проверяем, что если файл не существует то создаем его          if(!file.exists()){              file.createNewFile();          }            //PrintWriter обеспечит возможности записи в файл          PrintWriter out = new PrintWriter(file.getAbsoluteFile());            try {              //Записываем текст у файл              out.print(text);          } finally {              //После чего мы должны закрыть файл              //Иначе файл не запишется              out.close();          }      } catch(IOException e) {          throw new RuntimeException(e);      }  } |

Обратите особое внимание на то, что после записи каких либо данных в файл мы должны его закрыть, только после этого действия данные запишутся в файл.

|  |
| --- |
| private static String text = "This new text \nThis new text2\nThis new text3\nThis new text4\n";  private static String fileName = "[C://blog/a.txt](file:///C:\blog\a.txt)";    public static void main(String[] args) throws FileNotFoundException {        //Запись в файл      FileWorker.write(fileName, text);    } |

После чего мы получим новый файл «**a.txt**» со следующим содержимым:

|  |
| --- |
| This new text  This new text2  This new text3  This new text4 |

**2. Как читать файл?**

Теперь в классе **FileWorker** создадим метод для чтения файла, также статический:

|  |
| --- |
| public static String read(String fileName) throws FileNotFoundException {      //Этот спец. объект для построения строки      StringBuilder sb = new StringBuilder();        exists(fileName);        try {          //Объект для чтения файла в буфер          BufferedReader in = new BufferedReader(new FileReader( file.getAbsoluteFile()));          try {              //В цикле построчно считываем файл              String s;              while ((s = in.readLine()) != null) {                  sb.append(s);                  sb.append("\n");              }          } finally {              //Также не забываем закрыть файл              in.close();          }      } catch(IOException e) {          throw new RuntimeException(e);      }        //Возвращаем полученный текст с файла      return sb.toString();  } |

StringBuilder — в чем разница между обычным String? В том что когда  вы в StringBuilder добавляете текст он не пересоздается, а String пересоздает себя.

Также если файла нет то метод выкинет **Exception.**

Для проверки на существование файла создадим метод, так как нам еще потребуется эта проверка в следующих методах:

|  |
| --- |
| private static void exists(String fileName) throws FileNotFoundException {      File file = new File(fileName);      if (!file.exists()){          throw new FileNotFoundException(file.getName());      }  } |

Теперь проверим его:

|  |
| --- |
| private static String text = "This new text \nThis new text2\nThis new text3\nThis new text4\n";  private static String fileName = "[C://blog/a.txt](file:///C:\blog\a.txt)";    public static void main(String[] args) throws FileNotFoundException {      //Попытка прочитать несуществующий файл      FileWorker.read("no\_file.txt");        //Чтение файла      String textFromFile = FileWorker.read(fileName);      System.out.println(textFromFile);  } |

В первом случае когда файл не существует мы получим это:

|  |
| --- |
| Exception in thread "main" java.io.FileNotFoundException: no\_file.txt      at com.devcolibri.tools.FileWorker.read(FileWorker.java:31) |

Во втором случае, мы получим содержимое файла в виде строки. (*для этого закомментируйте первый случай*)

**3. Как обновить файл?**

Как такого **Update** для файлов нет, но способ обновить его есть, для этого можно его перезаписать.

Давайте создадим метод **update**в классе **FileWorker:**

|  |
| --- |
| public static void update(String nameFile, String newText) throws FileNotFoundException {      exists(fileName);      StringBuilder sb = new StringBuilder();      String oldFile = read(nameFile);      sb.append(oldFile);      sb.append(newText);      write(nameFile, sb.toString());  } |

Тут мы считываем старый файл в StringBuilder после чего добавляем к нему новый текст и записываем опять. Обратите внимание что для этого мы используем наши методы.

В результате обновления файла:

|  |
| --- |
| private static String text = "This new text \nThis new text2\nThis new text3\nThis new text4\n";  private static String fileName = "[C://blog/a.txt](file:///C:\blog\a.txt)";    public static void main(String[] args) throws FileNotFoundException {      //Обновление файла      FileWorker.update(fileName, "This new text");  } |

мы получим следующее содержание файла «**a.txt**«:

|  |
| --- |
| This new text  This new text2  This new text3  This new text4  This new text |

**4. Как удалить файл?**

В тот же наш утилитный класс **FileWorker**добавим метод delete, он будет очень простым так как у объекта **File**уже есть метод*delete*():

|  |
| --- |
| public static void delete(String nameFile) throws FileNotFoundException {      exists(nameFile);      new File(nameFile).delete();  } |

Проверяем:

|  |
| --- |
| private static String fileName = "[C://blog/a.txt](file:///C:\blog\a.txt)";    public static void main(String[] args) throws FileNotFoundException {      //Удаление файла      FileWorker.delete(fileName);  } |

После чего файл будет удален.