**Основы работы с файлами**

Ввод и вывод данных в Java реализуется через потоки ввода-вывода. Выделяют два типа потоков: байтовые и символьные. Для работы с символьными и байтовыми потоками используются специально разработанные для этого классы. Чтобы эти классы стали доступными в программе, **необходимо импортировать пакет java.io.**

## Байтовые и символьные потоки

На вершине иерархии байтовых потоков находятся два абстрактных класса: InputStream и OutputStream. В этих классах определены методы read() и write(), предназначенные для чтения данных из потока и записи данных в поток соответственно.

Некоторые другие классы байтовых потоков перечислены в табл.1.1.

Таблица 1.1. Классы байтовых потоков

|  |  |
| --- | --- |
| **Класс байтового потока** | **Описание** |
| InputStream | Абстрактный класс, который описывает поток ввода |
| OutputStream | Абстрактный класс, который описывает поток вывода |
| FilterInputStream | Класс, который реализует абстрактный класс InputStream |
| FilterOutputStream | Класс, который реализует абстрактный класс OutputStream |
| BufferedInputStream | Класс буферизованного потока ввода |
| BufferedOutputStream | Класс буферизованного потока вывода |
| ByteArrayInputStream | Класс потока ввода для считывания из массива |
| ByteArrayOutputStream | Класс потока вывода для записи в массив |
| FileInputStream | Класс потока ввода для считывания из файла |
| FileOutputStream | Класс потока вывода для записи в файл |
| DataInputStream | Класс потока ввода с методами для считывания данных стандартных типов Java |
| DataOutputStream | Класс потока вывода с методами для записи данных стандартных типов Java |
| PrintStream | Класс потока вывода, который поддерживает методы print() и println() |

Иерархия классов для символьных потоков ввода-вывода начинается с абстрактных классов Reader и Writer. В этих классах определены методы read() для считывания символьных данных из потока и write() для записи символьных данных в поток. Некоторые из классов для символьных потоков представлены и кратко описаны в табл. 1.2.

Таблица 1.2. Классы символьных потоков Java

|  |  |
| --- | --- |
| **Класс символьного потока** | **Описание** |
| Reader | Абстрактный класс, который описывает поток ввода |
| Writer | Абстрактный класс, который описывает поток вывода Консольный ввод с использованием объекта System.in |
| FilterReader | Класс, который описывает отфильтрованный поток ввода |
| FilterWriter | Класс, который описывает отфильтрованный поток вывода |
| InputStreamReader | Класс потока ввода, который переводит байты в символы |
| OutputStreamWriter | Класс потока вывода, который переводит символы в байты |
| StringReader | Класс потока ввода для считывания из текстовой строки |
| StringWriter | Класс потока вывода для записи в текстовую строку |
| FileReader | Класс потока ввода для считывания из файла |
| FileWriter | Класс потока вывода для записи в файл |
| BufferedReader | Класс буферизованного потока ввода |
| BufferedWriter | Класс буферизованного потока вывода |
| PrintWriter | Класс потока вывода, который поддерживает методы print() и println() |
| CharArrayReader | Класс потока ввода для считывания из массива |
| CharArrarWriter | Класс потока вывода для записи в массив |
| LineNumberReader | Класс потока ввода для подсчета текстовых строк |

Часть возможностей ввода-вывода может быть реализована посредством класса System. Класс System содержит три переменных потока: in, out и err. Эти поля имеют атрибуты public и static. В частности:

* Поле System.out – поток стандартного вывода. По умолчанию он связан с консолью. Поле System.out является объектом класса PrintStream.
* Поле System.in – это поток стандартного ввода. По умолчанию он связан с клавиатурой. Поле является объектом класса InputStream.
* Поле System.err – это стандартный поток ошибок. По умолчанию поток связан с консолью. Поле является объектом класса PrintStream.

## Класс File

Некоторые методы класса File перечислены в табл. 1.3.

Таблица 1.3. Методы класса File

|  |  |
| --- | --- |
| **Название метода** | **Описание** |
| isDirectory() | проверяет, является ли данный объект каталогом, возвращает логическое значение |
| isFile() | проверяет, является ли данный объект файлом, возвращает логическое значение |
| lastModified() | возвращает дату последней модификации файла |
| canRead() | проверяет, можно ли читать данные из файла, возвращает логическое значение |
| canWrite() | проверяет, можно ли записывать данные в файл, возвращает логическое значение |
| delete() | удаляет файл |
| public boolean exists(); | проверка существование файла или каталога |
| public String getName(); | возвращает имя файла или каталога |
| public String getAbsolutePath(); | возвращает абсолютный путь к файлу или каталогу |
| public String getPath(); | позволяет определить машинно-независимый путь файла или каталога |
| public String getParent(); |  |
| public long length(); | Длина файла в байтах |
| public boolean mkdir();  public boolean mkdirs(); | Первый из этих методов создает один каталог, второй - все подкаталоги, ведущие к создаваемому каталогу (то есть полный путь). |
| public boolean renameTo(File dest); | Для переименования файла или каталога вы должны создать два объекта класса File, один из которых соответствует старому имени, а второй - новому. Затем для перовго из этих объектов нужно вызвать метод renameTo, указав ему в качестве параметра ссылку на второй объект: |
| public boolean equals(Object obj); | этот метод сравнивает пути к файлам и каталогам, но не сами файлы или каталоги. |
| public String[] list();  public String[] list(FilenameFilter filter); | Первый из этих методы возвращает массив строк с именами содержимого каталога, не включая текущий каталог и родительский каталог. Второй позволяет получить список не всех объектов, хранящихся в каталоге, а только тех, что удовлетворяют условиям, определенным в фильтре filter класса FilenameFilter. |

**Задания для самостоятельного выполнения:**

* 1. Описать класс рабочий каталог. Рабочий каталог должен быть только один, поэтому для его описания используется шаблон Singleton. Это шаблон запрещает создание объектов с помощью конструкторов. Экземпляр класса создается с помощью метода getInstance. Класс содержит два поля: рабочий каталог (тот, в котором сейчас работает пользователь) и название рабочего каталога (абсолютный путь).

public class WorkingDirectory {

private static WorkingDirectory instance;

private String directoryName;

private WorkingDirectory(String directoryName) {

this.directoryName = directoryName;

}

public static WorkingDirectory getInstance(String directoryName) {

if (instance == null) {

instance = new WorkingDirectory(directoryName);

}

return instance;

}

}

Дополните класс следующими методами:

* 1. метод вывода содержимого текущего каталога (без вложения);
  2. метод, возвращающий абсолютное имя (путь) родительского каталога;
  3. метод перехода к родительскому каталогу, при этом имя текущего каталога изменяется на имя родительского;
  4. метод проверки существования дочернего каталога с заданным именем в текущем рабочем каталоге;
  5. метод создания нового каталога в текущем каталоге;
  6. метод перехода в подкаталог текущего каталога, при этом имя текущего каталога изменяется на имя каталога, в который перешли;
  7. \*метод удаления всех подкаталогов, вложенные в данный;
  8. \*метод вывода списка файлов определенного формата (расширения). Расширение передается параметром, содержащихся в заданном каталоге;
  9. \*метод вывода иерархического списка всех подкаталогов, вложенных в данный;
  10. \*метод проверки существования подкаталога с заданным именем с любым уровнем вложенности в текущем рабочем каталоге (искомый подкаталог может быть дочерним для вложенного каталога в текущий каталог).

Напишите программу для демонстрации работы с классом. \*\*Для демонстрации работы программы используйте паттерн Команда.

* 1. Исходные данные хранятся в текстовых файлах:

1) информация об абитуриентах ВятГУ: ФИО и список предметов с баллами за ЕГЭ (хотя бы у одного абитуриента должны быть сданы ЕГЭ хотя бы по 4-м предметам, лучше больше);

2) информация о факультетах: набор предметов на эти факультеты, минимальное значение баллов за ЕГЭ, проходной балл.

Написать программу, которая будет формировать текст письма с названиями специальностей, на которые абитуриент прошел по количеству баллов. Сохранить письма в текстовые файлы с именами *ФИОабитуриента.txt*.

\*\*Для решения использовать шаблон проектирования Наблюдатель. Возможная схема использования шаблона проектирования может быть такой. Наблюдаемый объект — это класс (например, Университет или Факультет), содержащий список абитуриентов с возможностью добавить или удалить абитуриента в список поступающих на определенный факультет. Наблюдатель — абитуриент, у которого есть метод получения письма.

Реализуйте возможность считывания данных из формата xml, \* json.

\* Реализуйте возможность сохранения файла в формате pdf. Подберите подходящую библиотеку.