



Модуль 02 - Piscine Java

ИО, Файлы

Резюме: Сегодня вы узнаете, как использовать ввод/вывод в Java и реализовывать программы для манипулирования файловой системой

Содержание

I	Предислов ие	2
II	Инструкци и	3
III	Упражнение Подписи к файлам 00 :	5
IV	Упражнение Слова 01 :	7
V	Упражнение Файловый менеджер 02 :	9

Глава I

Предисловие

Операции ввода/вывода играют важную роль в разработке корпоративных систем. Часто необходимо реализовать функциональность для загрузки и обработки пользовательских файлов, отправки различных документов по почте и т.д.

Очевидно, что ввод/вывод никогда не сводится к работе с файловой системой. Любое клиент-серверное взаимодействие между приложениями подразумевает операции ввода-вывода. Например, технология Java Servlets, используемая в веб-разработке, позволяет форматировать HTML-страницы с помощью класса `PrintWriter`.

Важно помнить, что функциональность ввода/вывода не ограничивается стеком Java IO. Существует множество библиотек, которые значительно упрощают взаимодействие с потоками данных. Apache Commons IO является одной из них.

Глава II


Инструкции

- Используйте эту страницу как единственную ссылку. Не слушайте никаких слухов и домыслов о том, как приготовить раствор.
- Теперь для вас существует только одна версия Java - 1.8. Убедитесь, что компилятор и интерпретатор этой версии установлены на вашей машине.
- Вы можете использовать IDE для написания и отладки исходного кода.
- Код чаще читают, чем пишут. Внимательно прочитайте [документ](#), в котором приведены правила форматирования кода. При выполнении каждого задания убедитесь, что вы следуете общепринятым [стандартам Oracle](#)
- Комментарии не допускаются в исходном коде вашего решения. Они затрудняют чтение кода.
- Обратите внимание на разрешения ваших файлов и каталогов.
- Чтобы ваше решение было оценено, оно должно находиться в вашем GIT-репозитории.
- Ваши решения будут оценивать ваши товарищи по аквариуму.
- Вы не должны оставлять в своем каталоге никаких других файлов, кроме тех, которые явно указаны в инструкциях к упражнению. Рекомендуется изменить свой .gitignore во избежание несчастных случаев.
- Когда вам нужно получить точный вывод в ваших программах, запрещено выводить предварительно рассчитанный вывод вместо правильного выполнения упражнения.
- У вас есть вопрос? Спросите своего соседа справа. В противном случае попробуйте поговорить с соседом слева.
- Ваше справочное пособие: товарищи / Интернет / Google. И еще кое-что. На любой ваш вопрос есть ответ на Stackoverflow. Научитесь правильно задавать вопросы.
- Внимательно прочитайте примеры. В них могут потребоваться вещи, которые не указаны в предмете.
- Используйте "System.out" для вывода

- И да пребудет с вами Сила!
- Никогда не оставляйте на завтра то, что вы можете сделать сегодня ;)

Глава III

Упражнение 00: Подписи файлов

	Упражнение 00
	Подписи к файлам
	Входящий каталог : <i>ex00/</i>
	Файлы для сдачи : *.java, подписи.txt
	Разрешенные функции : Все Рекомендуемые типы : Java Collections API (List<T>, Map<K, V> , и т.д.) InputStream, OutputStream, FileInputStream, FileOutputStream

Классы ввода/вывода в Java представлены широкой иерархией. Ключевыми классами, описывающими поведение ввода/вывода байтов, являются абстрактные классы `InputStream` и `OutputStream`. Они не реализуют конкретных механизмов обработки байтовых потоков, а делегируют их своим подклассам, таким как `FileInputStream/FileOutputStream`.

Чтобы понять, как использовать эту функциональность, необходимо реализовать приложение для анализа сигнатур произвольных файлов. Эта сигнатура позволяет определить тип содержимого файла и состоит из набора "магических чисел". Эти числа обычно располагаются в начале файла. Например, сигнатура для типа файла PNG представлена первыми восемью байтами файла, которые одинаковы для всех изображений PNG:

89 50 4E 47 0D 0A 1A 0A

Вам необходимо реализовать приложение, принимающее на вход файл `signatures.txt` (его описание вы должны сделать самостоятельно; имя файла явно указано в коде программы). Он содержит список типов файлов и соответствующих им сигнатур в формате HEX. Пример (указанный формат этого файла должен быть соблюден):

PNG, 89 50 4E 47 0D 0A 1A 0A
GIF, 47 49 46 38 37 61

Во время выполнения ваша программа должна принимать полные пути к файлам на жестком диске и сохранять тип, которому соответствует сигнатура

файла. Результат выполнения программы должен

быть записана в файл result.txt. Если сигнатура не может быть определена, результатом выполнения будет UNDEFINED (в файл не должна записываться никакая информация).

Пример работы программы:

```
Программа $java  
-> C:/Users/Admin/images.png  
PROCESSED  
-> C:/Users/Admin/Games/WoW.iso  
PROCESSED  
-> 42
```


Содержимое файла result.txt (нет необходимости загружать этот файл в качестве результата): PNG
GIF

Примечания:

- Мы можем точно определить тип содержимого, анализируя сигнатуру файла, поскольку расширение файла, содержащееся в имени (например, image.jpg), может быть изменено простым переименованием файла.
- Файл с подписями должен содержать не менее 10 различных форматов для анализа.

Глава IV

Упражнение 01: Слова

	Упражнение 01 Слова
Входящий каталог : <i>ex01/</i>	
Файлы для сдачи : *.java	
Разрешенные функции : Все Рекомендуемые типы : Java Collections API, Java IO	

В дополнение к классам, предназначенным для обработки байтовых потоков, в Java есть классы для упрощения обработки символьных потоков (char). К ним относятся абстрактные классы Reader/Writer, а также их реализации (FileReader/FileWriter и т.д.).

Особый интерес представляют классы BufferedReader/BufferedWriter, которые ускоряют обработку потоков с помощью механизмов буферизации.

Теперь необходимо реализовать приложение, которое будет определять уровень сходства между текстами. Самый простой и очевидный метод для этого - анализ частоты встречаемости одинаковых слов.

Предположим, что у нас есть два следующих текста:

- aaa bba bba bba a ccc
- bba a a a a bb xxx

Создадим словарь, содержащий все слова в этих текстах:

a, aaa, bb, bba, ccc, xxx

Теперь создадим два вектора, длина которых равна длине словаря. В i -й позиции каждого вектора отразим частоту встречаемости i -го слова из нашего словаря в первом и втором текстах:

$A = (1, 1, 0, 2, 1, 0)$

$B = (3, 0, 1, 1, 0, 1)$

Таким образом, каждый из этих векторов характеризует текст с точки зрения частоты встречаемости слов из нашего словаря. Определим сходство между векторами по следующей формуле:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

Таким образом, значение сходства для этих векторов равно:

```
Числитель A · B = (1 * 3 + 1 * 0 + 0 * 1 + 2 * 1 + 1 * 0 + 0 * 1) = 5
Знаменатель ||A|| * ||B|| = sqrt(1 * 1 + 1 * 1 + 0 * 0 + 2 * 2 + 1 * 1 + 0 * 0) * sqrt(3 * 3 + 0 * 0 + 1 * 1 + 1 * 1 + 1 * 1 + 0 * 0 + 1 * 1) = sqrt(7) * sqrt(12) = 2,64 * 3,46 = 9,1
сходство = 5 / 9,1 = 0,54
```

Ваша цель - реализовать приложение, которое принимает на вход два файла (оба файла передаются в качестве аргументов командной строки) и отображает результат сравнения их сходства (мера косинуса).

Программа также должна создать файл dictionary.txt, содержащий словарь на основе этих файлов.

Пример работы программы:


```
$ java Программа inputA.txt inputB.txt
Сходство = 0,54
```

Примечания:

1. Максимальный размер этих файлов - 10 МБ.
2. Файлы могут содержать небуквенные символы.

Глава V

Упражнение 02 : Диспетчер файлов

	Упражнение 02
	Файловый менеджер
Входящий каталог : <i>ex02/</i>	
Файлы для сдачи : *.java	
Разрешенные функции : Все	
Рекомендуемые типы : Java Collections API, Java IO, Файлы, Пути и т.д.	

Давайте реализуем утилиту для работы с файлами. Приложение должно отображать информацию о файлах, содержимом и размере папок, а также предоставлять функции перемещения/переименования. По сути, приложение эмулирует командную строку Unix-подобных систем.

Программа принимает в качестве аргумента абсолютный путь к папке, в которой мы начинаем работать, и поддерживает следующие команды:

`mv WHAT WHERE` - позволяет перенести или переименовать файл, если `WHERE` содержит имя файла без пути.

`ls` - отображает содержимое текущей папки (имена файлов и подпапок и их

размеры в KB) `cd FOLDER_NAME` - изменяет текущий каталог

Предположим, что на диске `C:/` (или в корневом каталоге, в зависимости от ОС) есть папка `MAIN` со следующей иерархией:

- ГЛАВНАЯ
 - папка1
 - * image.jpg
 - * animation.gif
 - папка2

- * text.txt
- * Program.java

Пример работы программы для каталога MAIN:

```
$ java Program --current-folder=C:/MAIN
C:/MAIN
-> ls
папка1 60 КБ
folder2 90 КБ
-> cd folder1
C:/MAIN/folder1
-> ls
image.jpg 10 КБ
animation.gif 50 КБ
-> mv image.jpg image2.jpg
-> ls
image2.jpg 10 КБ
animation.gif 50 КБ
-> mv animation.gif ../ folder2
-> ls
image2.jpg 10 КБ
-> cd ../ folder2
C:/MAIN/folder2
-> ls
text.txt 10 КБ
Program.java 80 КБ
animation.gif 50 КБ
-> ВЫХОД
```

Примечание:

Вы должны протестировать функциональность программы, используя свой собственный набор файлов/папок.