Конспект лекций по Java. Занятие 16

В.Фесюнов

1. Интерфейс FileFilter

Среди методов класса , рассмотренного нами на прошлом занятии есть метод

В отличие от одноименного метода, но без параметра, этот метод отбирает не все файлы данного каталога, а только те, которые удовлетворяют определенному условию. Параметр filter предназначен для задания этого условия. При этом тип параметра ( ) — это не класс, а интерфейс.

Обратимся к документации по интерфейсу . Мы увидим, что данный интерфейс имеет всего один метод

Этот метод должен возвращать **true** , если файл нам подходит, и **false** , если нет.

С таким приемом программирования мы еще не встречались. Разберем его подробнее.

Метод будет вызывать метод для каждого файла в каталоге, и те, для которых вернет , будут включены в результирующий список. Остальные будут проигнорированы.

Для использования этих возможностей нам нужно построить класс, удовлетворяющий интерфейсу , и определить в нем соответствующий метод .

2. Практическая работа

В качестве примера использования модифицируем программу *DirClean* (рассматривали на прошлом занятии). Сейчас программа принимает только один

параметр — имя каталога и удаляет все файлы в этом каталоге.

Расширим возможности программы *DirClean* .

Программа должна принимать один или два параметра. Если параметр один, она должна работать так же, как и раньше. Если задан второй параметр, то он задает расширение удаляемых файлов, и тогда программа должна удалять только те файлы, которые имеют указанное расширение.

1. Нам нужно создать класс, удовлетворяющий интерфейсу . Определим статический вложенный класс .

2. Определим в классе метод

3. Здесь для получения расширения файла используется другой метод — , который нам тоже нужно реализовать.

В данном методе при помощи вызова мы получаем полное имя файла в виде строки. Потом, используя метод класса мы определяем позицию последней точки в строке имени файла и вырезаем из строки подстроку от последней точки до конца строки.

4. Кроме метода в методе присутствует переменная *ext* . Эта переменная должна быть полем нашего класса. Этому полю будет присваиваться

значение второго параметра вызова программы в конструкторе класса . Окончательный вид класса такой

5. Теперь осталось реализовать использование класса в нашей программе. Для этого строку

предназначенную для получения списка файлов каталога заменим на следующую

Т.е. в метод будет передаваться либо , если нет второго параметра, либо объект класса , если он есть.

Результирующая программа выглядит так

Оттранслируем и запустим данную программу.

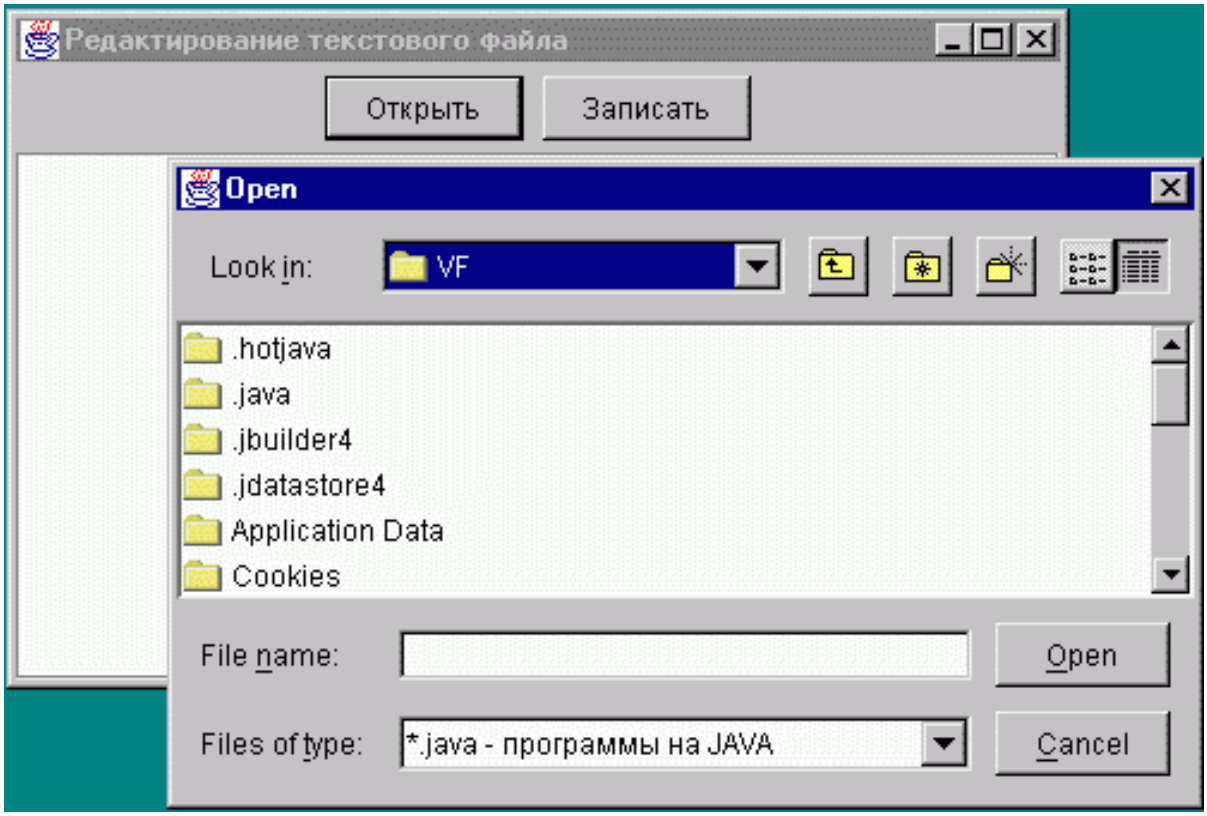
На этом рассмотрение собственно библиотеки ввода/вывода Java мы закончим.

3. Класс JFileChooser

После знакомства с вводом/выводом в Java уместно рассмотреть диалоговые возможности библиотеки языка, имеющие отношение к вводу/выводу.

Мы рассмотрим класс **JFileChooser** пакета **javax.swing** , который позволяет организовать стандартный диалог выбора файла для чтения или сохранения.

Внешний вид такого диалога примерно следующий.



Как видно из рисунка он похож на внешний вид стандартного окна открытия файла, с которым любому из пользователей приходилось сталкиваться неоднократно.

Познакомимся с классом JFileChooser более подробно.

Конструкторы класса:

Организует диалог начиная с текущего каталога.

Организует диалог начиная с указанного каталога.

То же самое, но каталог указывается объектом класса File.

Рассмотрим пример, поставляемый в составе пакета jdk1.3 (C:\jdk1.3\demo\jfc\FileChooserDemo\src\), в работе.

Этот пример демонстрирует многие возможности класса **JFileChooser** . Выбирая различные опции, мы можем увидеть, какие возможности реализует данный класс.

Разберемся, как использовать класс **JFileChooser** на примере простейшего редактора текстовых файлов.

Данный пример не закончен. В нем реализовано только чтение исходного файла. Для сохранения файла в нем предусмотрен метод , но он не реализован.

Для того чтобы разобраться в программе, рассмотрим методы класса , предназначенные для организации диалога. Это методы

Диалог открытия файла на чтение.

Диалог открытия файла на запись.

В обоих методах требуется параметр — тот компонент (окно), из которого вызван данный диалог. В нашем случае это должен быть главный программы, т.е. .

Работа с классом сосредоточена в методе :

Здесь, сначала создается объект класса , потом устанавливаются фильтры (рассмотрим ниже), после чего организуется собственно диалог методом

, которому в качестве параметра передается основной фрейм нашего приложения ( ).

Методы и организуют диалог с пользователем и в качестве результата возвращают целое значение (статус), описывающее, что сделал пользователь.

Пользователь мог:

- отказаться:

- сделать выбор:

- могла возникнуть ошибка:

Дальнейшие действия программы выполняются, только если все в порядке, т.е., если пользователь выбрал файл. В этом случае мы можем получить ссылку на этот файл в виде объекта класса с помощью метода . После этого остается открыть поток (в нашем случае), используя подходящий конструктор и ввести файл.

Класс имеет много возможностей настройки диалога. Так в нашей программе нам пришлось проверять, что пользователь выбрал именно файл, а не каталог. Это можно было бы обеспечить иначе — при помощи метода

public void setFileSelectionMode(int mode);

Здесь *mode* может принимать одно из значений:

- — только файлы
- — только каталоги
- — и те и другие

Т.е. можно было бы вставить строку

```
```

и не делать проверку " ".

Очень часто при организации такого диалога используются фильтры отбора файлов.

Для манипулирования фильтрами в имеются следующие методы:

Добавляет новый фильтр

Удаляет заданный фильтр

Удаляет все фильтры

Устанавливает/запрещает режим использования фильтра "*.* — AcceptAll " (все файлы).

По умолчанию всегда установлен фильтр *AcceptAll* . К нему можно добавить новые фильтры, используя *addChoosableFileFilter* , фильтр *AcceptAll* можно отключить при помощи вызова:

Теперь, разберемся, что такое здесь .

Это не интерфейс из пакета java.io, который мы рассматривали ранее. Это абстрактный класс из пакета **javax.swing.filechooser** . Он имеет два метода:

Метод для проверки, подходит ли данный файл под заданный фильтр.

Метод, задающий текст для диалога выбора фильтра.

Т.е. по сравнению с интерфейсом из пакета данный класс имеет дополнительный метод . Этот метод используется классом для отображения наименования фильтра в нижнем окошке. В остальном класс построен очень похоже на класс из

предыдущего примера.

Поскольку в нашей программе используется пакет , то при указании имени класса приходится указывать полное имя, т.к. иначе транслятор не может определить, какой из двух нам нужен.

4. Задание

1. Рассмотреть подробно данную программу.

2. Убрать фильтр отбора всех файлов.

3. Сделать фильтр отбора текстовых файлов фильтром по умолчанию (см.документацию).

4. Реализовать сохранение файла (реализовать метод).