Домашнее задание 2. Обход файловой системы.

Срок сдачи: 23 февраля, 2012

Прежде чем приступить к написанию, прочтите раздел **Замечания**.

1 Условие

1. Создайте класс *FilesystemWalker*. Он должен уметь обходить все поддерево файловой системы, начиная с указанного ему корня.

2. Должна быть включена возможность задавать *PatternFilter*, который в случае, если название файла или директории удовлетворяет некоторому регулярному выражению, предотвратит её обработку (и обход поддиректорий).

3. В случае если доступ к содержимому некоторой директории запрещен (произошел **SecurityException**), то она должна быть помечена как (access denied), а работа программы должна продолжиться!

4. Для каждой директории, обход ее поддиректорий (и вывод на консоль) должен проходить в лексикографическом порядке.

5. На вход *Main* дается абсолютный путь к корню интересующего поддерева.

   Программа должна распечатать это поддерево, за исключением папок и файлов, название которых начинается с символа '.' (в Unix это "скрытые" файлы).

6. Дерево должно распечатываться в консоль в следующем формате:

   folder

   |_subfolder1

   | |_subfolder1.1

   | |_subfolder1.2 (access denied)

   |_subfolder2

   |_a.txt

   |_b.txt

7. folder – имя переданного в качестве входа каталога (не путь!).

2 Замечания

1. Вам пригодятся классы **java.io.File** и **java.util.regex.Pattern**.
2. **java.io.File** используется как для представления каталогов, так и для обычных файлов.
3. Антон Михайлович попросил донести до вас следующее: если причина исключения ясна (скажем не найден файл и это FileNotFoundException), то **не нужно** выводить stack trace, а вместо него необходимо вывести осмысленное читаемое сообщение в **System.err**.
4. Для отступов используйте пробелы (заметьте, что длина отступов зависит от длин названий директорий).
5. Main не принимает на вход регулярное выражение, и использует PatternFilter только для обеспечения фильтрации скрытых файлов. Тем не менее PatternFilter, конечно, должен поддерживать произвольные регулярные выражения.
6. Не нужно в явном виде производить проверку прав доступа (и общаться с SecurityManager), используйте SecurityException как индикатор того, что прав не оказалось. Учтите, что в принципе использовать исключения для содержательного управления основным workflow **не хорошо**. Но здесь мы пока решили не заставлять вас вникать в Java sucurity policy и сделали скидку =)
7. В задании специально четко не указана архитектура приложения. Программа минимум – вывести поддерево на консоль в требуемом формате. Естественно, удачные архитектурные решения будут награждаться дополнительными баллами (только очень прошу: **не переусердствуйте**).
8. Загрузить все дерево в память или распечатывать по мере обхода – также на ваш выбор =)