# Домашнее задание 2. Обход файловой системы.

Срок сдачи: 27 февраля, 2013

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 некоторой директории по той или иной причине запрещен (**File.canRead()** вернул **false** или 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. Для отступов используйте пробелы (заметьте, что длина отступов зависит от длин названий директорий).
4. 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.
5. В задании четко не указана архитектура приложения. Программа минимум – вывести поддерево на консоль в требуемом формате. Удачные архитектурные решения могут быть поощрены дополнительными баллами, а явные проблемы дизайна могут караться =)
6. Загрузить все дерево в память или распечатывать по мере обхода – также на ваш выбор.
7. Нелишним будет почитать немного про java security policy (в частности **SecurityManager** и **SecurityException**).
8. Ваш код будет тестироваться нами на Linux-е, папках с различными правами доступа и (вы/в)ключенной опции **-Djava.security.manager**
9. Тестируйте тщательно!