# Программа для сканирования каталогов

## ЗАДАЧА

1. Написать программу, которая ищет файлы в каталогах, **полные** пути к которым перечисляются в параметрах программы. Каталоги могут быть как сетевыми, так и нет.

## ВЫВОД

2. Итоговым выводом является файл в кодировке **UTF-8**, в котором перечислены все найденные файлы.

3. Допустим, итоговый файл выглядит так (формат вывода **должен быть именно таким** как в этом примере):

*[*
*file = \\epbyminsd0235\Video Materials\.DS_Store*
*date = 2011.07.20*
*size = 6148][*
*file = \\epbyminsd0235\Video Materials\2008.ivc*
*date = 2008.12.12*
*size = 415892][*
*file = \\epbyminsd0235\Video Materials\CDP DAM.ivc*
*date = 2009.01.29*
*size = 3207246][*
*file = \\epbyminsd0235\Video Materials\.NET Mentoring Program\Acceptance Testing Through UI\2010-01-19 10.13 Acceptance Testing.wmv*
*date = 2010.01.19*
*size = 22904839][*
*file = \\epbyminsd0235\Video Materials\.NET Mentoring Program\Acceptance Testing Through UI\2010-01-19 10.50 Acceptance Testing.wmv*
*date = 2010.01.19*
*size = 106224657]*

4. Чтобы было понятно **как делать нельзя**, приводим разъяснения по целевому способу использования программы.

Итак, мы просканировали нашей программой все каталоги в первый раз. Далее, например, через неделю просканировали второй раз. Взяли полученные 2 файла. Сравнили их, например, с помощью WinMerge. В результате увидели:

- что добавилось,
- что убавилось,
- что изменилось.

Таким образом, если вывод от запуска к запуску будет неупорядоченным, пользы от программы не будет.

5. Для того, чтобы не было скучно глядеть в пустую консоль, нужно сопроводить процесс сканирования выводом через каждый 6 секунд «точки» (.), а через каждую минуту палочки (|). Если вы хотите что-то «сказать» **в консоль,** делать это надо **на английском языке**.

## ВВОД

6. Программа должна позволять исключать каталоги из анализа. Для этого нужно использовать ключ «минус» (-) и после него перечислить полные пути каталогов, которые анализу не подлежат.

Ниже пример параметров, при которых ничего не сканируется:

*"\\epbyminsd0235\Video Materials" "\\EPUALVISA0002.kyiv.com\Workflow\ORG\Employees\Special" "\\EPUALVISA0002.kyiv.com\Workflow\ORG\Employees\Lviv" - "\\epbyminsd0235\Video Materials" "\\EPUALVISA0002.kyiv.com\Workflow\ORG\Employees\Special" "\\EPUALVISA0002.kyiv.com\Workflow\ORG\Employees\Lviv"*

7. Должны работать разные варианты входных параметров (то есть, не должно быть такого, что программа протестирована только для одного каталога на диске C:\ с одним исключением).

8. Должно быть учтено возможное расширение набора ключей (помимо «минуса»). Например, чтобы исключить из вывода ряд файлов (таких как Thumbs.db).

## ОБЯЗАТЕЛЬНЫЕ ТРЕБОВАНИЯ

9. Функциональность программы должна быть покрыта **JUnit-тестами**. Должна быть обеспечена возможность запуска всех тестов сразу.

10. Для ускорения работы программы разработчик **должен использовать многопоточность**. На умение работать с потоками будет обращено особое внимание.

11. Файлов во всех каталогах может быть **несколько миллионов**.

12. Программа должна работать быстро и с экономным потреблением оперативной памяти.

13. Любой выбор структур данных, подходов и алгоритмов должен быть лаконично и емко обоснован комментарием. Например, если был сделан выбор в пользу ArrayList (150000), нужно пояснение почему. Если размер файлового буфера выбран равным 8192, то почему и т.д. Из кода и комментариев в нем (**на русском языке**) должен быть ясен ход мысли автора. Мы должны видеть, что выбор сделан с пониманием, а не случайно.

14. Ход программы, ее алгоритм, ее циклы должны быть простыми и без замысловатостей, чтобы не приходилось тратить нервные клетки мозга на понимание.

15. Должна использоваться Java версии 6 или больше.

16. Использовать сторонние программы и библиотеки **нельзя**. Только Java SE. Это, конечно, не касается JUnit-тестов ☺

17. Все другие вопросы по особенностям реализации приложения решаются разработчиком самостоятельно, эти решения отражают знания и опыт разработчика и учитываются при оценке созданного продукта.

18. Приложение должно решать поставленные задачи, выполнение дополнительных функций, не приведенных в задании, не влияет на оценку решения (за редким исключением).

19. При оценке решения учитываются следующие факторы:
    - применение принципов ООП, возможность расширения приложения и повторного использования кода,
    - простота и читаемость кода (в том числе комментирование),
    - архитектура решения (способы работы с данными),
    - структура кода.

20. Готовое решение должно представлять из себя архив, в котором находятся исходные тексты программы и JUnit-тестов.