Project task-jtr-practice5

Практическое занятие №5
_____

Замечания.

(1) В Jenkins время сборки проекта ограничено 2 (двумя) минутами. По истечение этого времени задача будет снята с выполнения.

```
Console Output
-------------------------------------------------------
Build timed out (after 3 minutes). Marking the build as aborted.
Build was aborted
Finished: ABORTED
-------------------------------------------------------
```

(2) В корневом пакете должен находится класс Demo, который демонстрирует функциональность всех подзадач (его будет вызывать Jenkins).

(3) Для решения каждой подзадачи X будет гарантированно использованно несколько потоков:
  * поток, который выполняет метод PartX.main (это тот же самый поток, который выполняет Demo.main);
  * некоторое количество дочерних потоков.
Завершайте выполнение дочерних потоков ДО ТОГО, как будет завершен метод PartX.main (это нужно для тестов, который запускает Jenkins).

_____

Задание 1
```
-------------------------------------------------------
Название класса: ua.nure.your_last_name.Practice5.Part1
-------------------------------------------------------
```

Создать дочерний поток, который бы в течение примерно 2 сек печатал свое имя каждые полсекунды.
Сделать это двумя способами:
  * при помощи расширения класса Thread;
  * при помощи реализации интерфейса Runnable.

Вначале должна отработать одна реализация, после ее завершения должна отработать другая реализация.

_____

Задание 2
```
-------------------------------------------------------
Имя пакета: ua.nure.your_last_name.Practice5
Названия классов: Part2, Spam
-------------------------------------------------------
```

Создать класс Spam, который получает в конструкторе массив сообщений и согласованный с ним массив интервалов времени в миллисекундах и выводит одновременно соответствующие сообщения на экран через заданные интервалы времени. По нажатию на Enter приложение должно завершать свою работу (данную функциональность поместить в метод Spam.main).

При демонстрации работы смоделировать ввод Enter через 2 сек (данную функциональность поместить в метод Part2.main).

Входные данные (массив сообщений и массив пауз) записывать в коде класса Spam. Количество элементов в каждом из массивов - минимум 2, их можно взять из примера:

Пример входной информации (класс Spam)

String[] messages = new String[] { "@@@", "bbbbbbb" };

Дополнительная информация по задаче.

(1) Не используйте потоки демоны, т.к. запуск Part2 идет в пакете вместе с другими задачами, демоны не завершат свое выполнение до тех пор пока не завершит свое выполнение Demo.main.

(2) Для того, чтобы отследить нажатие на Enter достаточно считывать консольный ввод и анализировать содержимое. Если чтение будет осуществлено с помощью классов Scanner / BufferedReader, то признаком Enter является пустая строка, которую возвращают, соответственно, методы Scanner#nextLine() / BufferedReader#readLine().

(3) Алгоритм моделирования консольного ввода (метод Part2.main):

---
a) подменить системный поток ввода на свой собственный
   System.setIn(YOUR_OWN_INPUT_STREAM);

b) в отдельном потоке вызвать Spam.main
   Thread t = new Thread() { public void run() {Spam.main(null);}};
   t.start();

c) дождаться завершения Spam.main
   t.join();

d) восстановить системный поток
   System.setIn(CACHED_VALUE_OF_SYSTEM_IN)
---

(4) Для реализации своего собственного потока ввода целесообразно создать класс, который расширяет абстрактный класс java.io.InputStream. При этом придется реализовать единственный абстрактный метод этого класса:

---
public abstract int read() throws IOException;

Все методы класса InputStream (а также всех его наследников), которые читают байты из источника информации, в конечном счете вызывают метод read. Достаточно реализовать паузу в этом методе при первом его вызове, которая заставит поток выполнения, который вызвал метод read, ожидать. Метод read должен последовательно возвращать байты, которые соответствуют ограничителю строки, после чего постоянно возвращать -1 (признак того, что информации во входном потоке больше нет).

_____

Задание 3

Название класса: ua.nure.your_last_name.Practice5.Part3

Создать класс с двумя отдельными счетчиками и объект этого класса.
Создать несколько одинаковых потоков, каждый из которых повторяет следующее:
  * сравнивает значение счетчиков и печатает результат сравнения;
  * увеличивает первый счетчик;
  * засыпает на 10 мсек;
  * увеличивает второй счетчик.

Сравнить работу программы при условии, что код синхронизирован и не синхронизирован.
Реализовать следующую схему:
  * вначале отрабатывает один вариант;
  * после его завершения отрабатывает другой вариант.
Весь вывод должен быть небольшим - всего около нескольких десятков строк.

_____

Название класса: ua.nure.your_last_name.Practice5.Part4
Входную информацию загружать из файла part4.txt

Распараллелить задачу поиска максимального значения в матрице целых чисел MxN (загружать из файла) при помощи M потоков. Дополнительно решить задачу поиска максимального значения без распараллеливания. Вывести результат и время выполнения кода (в миллисекундах) для обоих вариантов.

В обязательном порядке в обоих вариантах каждую операцию сравнения снабдить задержкой в 1 мс!

Придерживаться следующего формата вывода:

MAX
TIME
MAX2
TIME2

где MAX, TIME - максимальное значение и время поиска при многопоточном решении задачи; MAX2, TIME2 - максимальное значение и время поиска в однопоточном варианте.

Работу приложения проверить на матрице 4x100 случайных чисел.

Содержимое файла part4.txt должно представлять из себя читабельную матрицу, числа разделены пробелами, строки разделены ограничителем строки.

Замечания.

(1) Ограничитель строки должен быть платформонезависимым (учитывать это при чтении информации).

(2) Входной файл part4.txt создать любым удобным для вас способом до запуска Part4.main (например, с помощью какого-нибудь вспомогательного класса). Приложение (Part4.main) файл part4.txt не создает и содержимое его не модифицирует, размерность матрицы определять по содержимому part4.txt

Пример содержимого part4.txt для M=5, N=20
---
706 575 855 882 595 778 477 602 147 467 693 793 120 384 256 866 548 367 910 848
206 232 632 315 743 823 620 111 279 548 210 393 791 815 519 768 168 484 780 705
709 127 900 171 189 590 563 317 600 975 892 296 166 353 863 312 399 872 964 591
302 869 679 157 419 485 325 290 739 149 407 648 688 474 311 177 318 611 348 557
559 283 171 352 698 759 384 822 598 410 802 293 962 859 812 153 436 392 869 167

Пример вывода результата

975
26
975
115

Для входных данны3 M=4, N=100 время распараллеленного поиска должно быть примерно в 4 раза меньше, чем время поиска в однопоточном варианте.

_____

Задание 5

```
-------------------------------------------------------
Название класса: ua.nure.your_last_name.Practice5.Part5
Выходную информацию записывать в файл part5.txt
```



```
Создать k потоков, которые одновременно пишут в один и тот же файл символы:

-------------------------------------------------------
первый поток записывает цифру 0 ровно 20 раз на 1й строке файла;
второй поток записывает цифру 1 ровно 20 раз на 2й строке файла;
...
десятый поток записывает цифру 9 ровно 20 раз на 10й строке файла.
-------------------------------------------------------

Требования к реализации.
(1) В обязательном порядке запись каждой цифры снабдить паузой в 1 мс!
(2) Для записи использовать класс RandomAccessFile.
(3) Допускается использование не более одного объекта класса
RandomAccessFile!
(4) Перед началом работы файл в который будет происходить запись должен
быть удален, если он существует.
(5) Главный поток, после запуска дочерних потоков на выполнение, должен
дождаться их завершения, после чего вывести в консоль содержимое файла.

Замечания.
(1) Метод RandomAccessFile#seek(long) позволяет передвигать указатель
внутри файла. Каждый поток должен знать в каком месте файла ему записывать
информацию. Так как в условии фигурирует термин "строка", следует вывод
каждого потока завершать ограничителем строки, который выводить кросс-
платформенным образом.
(2) Для того, чтобы записать некоторую цифру можно использовать выражение
'0'+ n, где n - цифра от 0 до 9 включительно.
Передвижение указателя внутри файла и запись информации необходимо
синхронизировать.

Результат работы приложения
-------------------------------------------------------
00000000000000000000
11111111111111111111
22222222222222222222
33333333333333333333
44444444444444444444
55555555555555555555
66666666666666666666
77777777777777777777
88888888888888888888
99999999999999999999
-------------------------------------------------------
```

Project disk usage information + trend graph

Disk Usage: Workspace 0, Builds {all=0, locked=0}, Job directory 24441

Recent Changes

Permalinks