Day 03 - Piscine Java

Threads

Резюме: Сегодня вы научитесь работать с базовыми механизмами многопоточности в Java

Contents

Chapter I
Preamble

- Любое современное клиент-серверное приложение базируется на использовании потоков (нитей, Threads).
- Потоки позволяют реализовать концепцию асинхронной работы, когда несколько различных слабосвязанных задач выполняются “параллельно”.
- Многопоточность в клиент-серверных приложениях позволяет выделить некоторые задачи в фоновый режим выполнения и не заставлять клиента ждать ответа от сервера. Например, при указании своего Email-а на сайте, вы сразу получите страницу с текстом об успешной отправке сообщения-подтверждения на почту, независимо от того, сколько реального времени займет непосредственная отправка письма на ваш Email в побочном потоке.
- Каждый ваш запрос на каком-либо сайте выполняется в отдельном независимом потоке на сервере.
- Поведение потоков регулируется операционной системой и процессором.
- Поведение потоков недетерминированно. Вы никогда не знаете, какой поток будет выполняться в тот или иной промежуток времени. Даже при повторном запуске одного и того же многопоточного кода.
- Полезные методы для работы с потоками присутствуют в классе `Object`.
- Потоки - любимая тема для собеседований Junior-ов.

Chapter II
General Rules

- Use this page as the only reference. Do not listen to any rumors and speculations about how to prepare your solution.
- Сейчас для вас существует только одна версия Java - 1.8. Убедитесь, что на вашем компьютере установлен компилятор и интерпретатор данной версии.
- Не запрещено использовать IDE для написания исходного кода и его отладки.
- Код чаще читается, чем пишется. Внимательно изучите представленный документ с правилами оформления кода. В каждом задании обязательно придерживайтесь общепринятых стандартов Oracle - https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html
- Комментарии в исходном коде вашего решения запрещены. Они мешают восприятию.
- Pay attention to the permissions of your files and directories.
- To be assessed your solution must be in your GIT repository.
- Your solutions will be evaluated by your piscine mates.
- You should not leave in your directory any other file than those explicitly specified by the exercise instructions. It is recommended that you modify your .gitignore to avoid accidents.
- When you need to get precise output in your programs, it is forbidden to display a precalculated output instead of performing the exercise correctly.
- Have a question? Ask your neighbor on the right. Otherwise, try with your neighbor on the left.
- Your reference manual: mates / Internet / Google. И еще, для любых ваших вопросов существует ответ на Stackoverflow. Научитесь правильно их задавать.
- Read the examples carefully. They may require things that are not otherwise specified in the subject.
- And may the Force be with you!
- Не откладывайте на завтра то, что можно было сделать вчера ;)

Chapter III
Exercise 00 - Egg, Hen ... or Human?

Exercise 00: Egg, Hen ... or Human?	
Turn-in directory	ex00
Files to turn-in	*.java
Разрешено:	
Типы и их методы:	Object, Thread, Runnable

Что было раньше - яйцо или курица?

В споре рождается истина - сделаем так, чтобы каждый поток говорил свой вариант ответа. Тот поток, за которым осталось последнее слово - прав.

Вам нужно реализовать работу двух потоков. Каждый из них должен вывести на экран свою версию ответа некоторое количество раз, например, 50:

```
$ java Program --count=50
Egg
Hen
Hen
Hen
...
Egg
```

В данном случае победил поток-яйцо. Но, также в программе существует главный поток - `main`. Внутри этого потока происходит выполнение метода `public static void main(String args[])`. Необходимо, чтобы именно этот поток выводил все свои ответы в конце работы программы. Таким образом, имеем следующий итоговый вариант:

```
$ java Program --count=50
Egg
Hen
Hen
...
Egg
Hen
...
Human
...
...
Human
```

Т.е. последние 50 раз программа выводит сообщение `Human`, которое печатает основной поток `main`.

Chapter IV
Exercise 01 - Egg, Hen, Egg, Hen...

Exercise 00: Egg, Hen, Egg, Hen...	
Turn-in directory	ex01
Files to turn-in	*.java
Разрешено:	
Типы и их методы:	Object, Thread, Runnable
Ключевые слова:	synchronized

Давайте сделаем спор более организованным. Теперь каждый из потоков может сказать свой вариант ответа только после того, как его озвучил предыдущий поток. Пусть первым свой вариант ответа всегда говорит поток-яйцо:

```
$ java Program --count=50
Egg
Hen
Egg
Hen
Egg
Hen
...
```

Примечание:

- Для решения данной задачи рекомендуется изучить принцип работы модели Producer-Consumer

Chapter V
Exercise 02 - Real Multithreading

Exercise 02: Real Multithreading	
Turn-in directory	ex02
Files to turn-in	*.java
Разрешено:	
Типы и их методы:	Object, Thread, Runnable
Ключевые слова:	synchronized

Попробуйте использовать многопоточность по прямому назначению - распределите вычисления внутри программы.

Пусть есть некоторый массив целых чисел, ваша задача - рассчитать сумму элементов массива несколькими потоками “суммирования”. Каждый поток считает определенный участок внутри массива. Количество элементов каждого участка постоянно, за исключением последнего (его размер может отличаться в большую или меньшую сторону).

Массив должен генерироваться каждый раз случайным образом, длина массива и количество потоков для работы передаются в качестве аргументов командной строки.

Для того, чтобы проверить, корректно ли работает программа, необходимо предварительно подсчитать сумму элементов массива стандартным способом.

Максимальное количество элементов массива - 2 000 000. Максимальное количество потоков не превышает текущее количество элементов массива. Максимальное значение каждого элемента массива по модулю - 1000. Корректность всех данных гарантируется.

Пример работы программы (каждый элемент массива равен единице):

```
$ java Program --arraySize=13 --threadsCount=3

Sum: 13
Thread 1: from 0 to 4 sum is 5
Thread 2: from 5 to 9 sum is 5
Thread 3: from 10 to 12 sum is 3
Sum by threads: 13
```

Примечание:

- В примере, указанном выше, размер последнего участка для суммирования третьим потоком меньше остальных.
- Потоки могут выводить результаты работы непоследовательно.

Chapter VI
Exercise 03 - Too many threads...

Exercise 02: Too many threads...	
Turn-in directory	ex03
Files to turn-in	*.java
Разрешено:	
Типы и их методы:	Object, Thread, Runnable
Ключевые слова:	synchronized

Пусть необходимо выполнить загрузку определенного списка файлов из сети. Некоторые файлы могут загружаться быстрее, некоторые - медленнее.

Очевидно, что для реализации такой функциональности можно использовать многопоточную загрузку, где каждый поток будет загружать конкретный файл. Но, что делать, если файлов будет очень много? Большое количество потоков не может быть выполнено одновременно, поэтому значительная часть из них будет находиться в ожидании.

Помимо этого, следует учесть, что постоянное создание и завершение потоков - чрезвычайно дорогая операция, которую хотелось бы избежать. Логичнее - запустить сразу N-ое количество потоков, и как только какой-либо из этих потоков закончил скачивание файла, он может взять следующий файл в очереди.

Необходимо создать файл `files_urls.txt` (название файла должно быть явно указано в коде программы), в котором вы укажете список URL-ов файлов для скачивания, например:

```
1 https://i.pinimg.com/originals/11/19/2e/11192eba63f6f3aa591d3263fdb66bd5.jpg
2 https://pluspng.com/img-png/balloon-hd-png-balloons-png-hd-2750.png
3 https://i.pinimg.com/originals/db/a1/62/dba162603c71cac00d3548420c52bac6.png
4 https://pngimg.com/uploads/balloon/balloon_PNG4969.png
5 http://tldp.org/LDP/intro-linux/intro-linux.pdf
```

Пример работы программы:

```
$ java Program.java --threadsCount=3
Thread-1 start download file number 1
Thread-2 start download file number 2
Thread-1 finish download file number 1
Thread-1 start download file number 3
Thread-3 start download file number 4
Thread-1 finish download file number 3
Thread-2 finish download file number 2
```

```
Thread-1 start download file number 5
Thread-3 finish download file number 4
Thread-1 finish download file number 5
```

Примечания:

- Вывод реализованной программы может отличаться от приведенного.
- Каждый файл скачивается ровно один раз ровно одним потоком.
- Программа может содержать “бесконечный цикл” без условия выхода (завершение программы в этом случае может происходить с помощью прерывания процесса).

CHECKLIST:
https://docs.google.com/document/d/1CUG8BQiQY2Dlvr-pbq86ki0DiSSPMlKAlJGtTx16TCo/edit?usp=sharing